T.C. İSTANBUL TİCARET
ÜNİVERSİTESİ

**FEN BİLİMLERİ ENSTİTÜSÜ**

# DERİN ÖĞRENME YAKLAŞIMI KULLANARAK BULUT ORTAMLARI İÇİN SALDIRI TESPİT HİZMET TASARIMI

**Wisam ELMASRY**

**Danışman**
**Prof. Dr. Abdül Halim ZAİM**

**Eş Danışman**
**Doç. Dr. Akhan AKBULUT**

**DOKTORA TEZİ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**
**İSTANBUL- 2019**

# KABUL VE ONAY SAYFASI

**Wisam ELMASRY** tarafından hazırlanan **"Derin Öğrenme Yaklaşımı Kullanarak Bulut Ortamları için Saldırı Tespit Hizmet Tasarımı"** adlı tez çalışması 25/07/2019 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **Doktora Tezi** olarak kabul edilmiştir.

| | | |
|---|---|---|
| **Danışman** | **Prof. Dr. Abdül Halim ZAİM**<br>İstanbul Ticaret Üniversitesi | |
| **Jüri Üyesi** | **Prof. Dr. Selim AKYOKUŞ**<br>İstanbul Medipol Üniversitesi | |
| **Jüri Üyesi** | **Doç. Dr. Muhammed Ali AYDIN**<br>İstanbul Üniversitesi - Cerrahpaşa | |
| **Jüri Üyesi** | **Dr. Öğr. Üyesi Metin TURAN**<br>İstanbul Ticaret Üniversitesi | |
| **Jüri Üyesi** | **Dr. Öğr. Üyesi Mustafa Cem KASAPBAŞI**<br>İstanbul Ticaret Üniversitesi | |

**Onay Tarihi :** 09/10/2019

**Prof. Dr. Necip ŞİMŞEK**
**Enstitü Müdürü**

# AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

25/07/2019

**Wisam ELMASRY**

# İÇİNDEKİLER

**Sayfa**

# ÖZET


**Doktora Tezi**

**DERİN ÖĞRENME YAKLAŞIMI KULLANARAK BULUT ORTAMLARI İÇİN SALDIRI TESPİT HİZMET TASARIMI**

**Wisam ELMASRY**

**İstanbul Ticaret Üniversitesi**
**Fen Bilimleri Enstitüsü**
**Bilgisayar Mühendisliği Anabilim Dalı**

**Danışman: Prof. Dr. Abdül Halim ZAİM**

**Eş Danışman: Doç. Dr. Akhan AKBULUT**
**2019, 137 sayfa**

Saldırı tespiti, siber güvenliğin temel taşı olarak kabul edilir. Erken ve etkili saldırı tespiti, son on yılda araştırmacılardan büyük ilgi görmüştür. Bununla birlikte, siber güvenliğe saldırı tespiti için derin öğrenme modellerinin kullanımı konusunda derin ve yeterli bir çalışmanın varlığı nadiren mümkündür. Bu tez çalışmasında kişisel bilgisayar, ağ ve bulut bilişim olmak üzere üç farklı ortamda saldırı saptama problemini araştırdık. Kişisel bilgisayar ve ağ ortamları ile ilgili olarak, sırasıyla maskeli ve ağ saldırı tespiti için bir dizi derin öğrenme modeli geliştirdik. Ayrıca, hem özellik hem de hiperparametre seçimi için yeni ve etkili bir çift Parçacık Sürü Optimizasyonu (PSO) tabanlı bir algoritma önerdik. Eski algoritmayı, derin öğrenme modelinin antrenman öncesi aşamasında, verilen antrenman setinin optimum özellik alt kümesi ve azaltılmış antrenman setinin doğruluğunu en üst düzeye çıkartan modelin optimum hipermetreleri elde edileceği şekilde kullandık. eğitim aşamasına. Ayrıca, geliştirilen derin öğrenme modellerinin saldırı tespitinde iyi bilinen bir dizi veri seti ve çeşitli analizler kullanarak etkinliğini doğruladık. Deneysel sonuçlar, çift PSO tabanlı algoritmayı kullanarak ön eğitimli derin öğrenme modellerinin performans açısından geleneksel makine öğrenme yöntemlerinden daha iyi performans gösterdiğini, tespit oranını 1% ile 10% arasında artırdığını ve yanlış alarm oranını 1% ıle 5% arasında azalttığını çoğu durumda göstermiştir.

Kişisel bilgisayar ve ağ ortamları için saldırı tespitindeki bulgularımız, dinamik, karma ve çok iş parçacıklı bir bulut tabanlı saldırı algılama sistemi tasarlamak için kullanılır. Buna ek olarak, üçüncü taraf bir bulut hizmeti, önerilen bir bulut tabanlı saldırı algılama sistemini izlemek ve yönetmek, ayrıca bir saldırı alarmı verildiğinde bulut kullanıcıları ve bulut hizmeti sağlayıcısıyla iletişim kurmak için de tasarlanmıştır.



**Anahtar Kelimeler:** Derin öğrenme, parçacık sürüsü optimizasyonu, saldırı tespiti, siber güvenlik.

# ABSTRACT

**Ph.D. Thesis**

**A DEEP LEARNING APPROACH FOR DESIGNING A CLOUD INTRUSION DETECTION SERVICE**

**Wisam ELMASRY**

**Istanbul Commerce University**
**Graduate School of Applied and Natural Sciences**
**Department of Computer Engineering**

**Supervisor: Prof. Dr. Abdul Halim ZAIM**

**Co-Supervisor: Assoc. Prof. Dr. Akhan AKBULUT**
**2019, 137 pages**

Intrusion detection is deemed to be a cornerstone of cyber security. Early and effective intrusion detection has been attracted much attention from researchers in the last decade. However, the existence of a deep and adequate study in using deep learning models for intrusion detection in cyber security is still seldom. In this thesis, we have investigated the problem of intrusion detection in three different environments, namely, personal computer, network and cloud computing. Regarding personal computer and network environments, we have developed a set of deep learning models for masquerade and intrusion detection, respectively. Furthermore, we proposed a novel and efficient double Particle Swarm Optimization (PSO)-based algorithm for both feature and hyperparameter selection. We utilized the former algorithm in the pre-training phase of the deep learning model in such a way that the optimal feature subset of the given training set and the optimal hyperparameters of the model which maximizes the accuracy over the reduced training set, are obtained prior to training phase. Moreover, we validated the effectiveness of the developed deep learning models in intrusion detection using a set of well-known datasets and various analyses. The experimental results demonstrated that the deep learning models with pre-training using the double PSO-based algorithm outperformed the traditional machine learning methods in terms of performance, increased the detection rate between 1% to 10%, and decreased the false alarm rate between 1% to 5% in most cases.

Our findings in intrusion detection for personal computer and network environments are exploited to design an integrated cloud-based intrusion detection system which is dynamic, hybrid and multithreaded. In addition to that, a third party cloud service is also designed to monitor and manage the proposed cloud-based intrusion detection system as well as communicate with cloud users and cloud service provider when an alert of attack is raised.

**Keywords:** Cyber security, deep learning, intrusion detection, particle swarm optimization.

# TEŞEKKÜR

Bu araştırma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli Danışman Hocam Prof. Dr., Abdül Halim ZAİM'e teşekkürlerimi sunarım. Araştırmanın yürütülmesinde yardımcı olan değerli Eş Danışman hocam Doç. Dr., Akhan AKBULUT'a, teşekkür ederim.

Araştırmanın yürütülmesinde maddi ve manevi yardımlarını gördüğüm olmak üzere tüm İstanbul Ticaret Üniversitesi personeline teşekkür ederim.

Tezimin her aşamasında beni yalnız bırakmayan aileme sonsuz sevgi ve saygılarımı sunarım.

Wisam ELMASRY
İSTANBUL, 2019

# ŞEKİLLER

**Sayfa**

# ÇİZELGELER

**Sayfa**

# SİMGELER VE KISALTMALAR

| | |
|---|---|
| ACIDF | Autonomic Cloud Intrusion Detection Framework |
| ACO | Ant Colony Optimization |
| AE | Autoencoder |
| AI | Artificial Intelligence |
| ANN | Artificial Neural Networks |
| AUC | Area Under Curve |
| AUPR | Area Under the PR Curve |
| AUROC | Area Under the ROC Curve |
| BDR | Bayesian Detection Rate |
| BM | Boltzmann Machine |
| BP | Back Propagation |
| BPSO | Binary Particle Swarm Optimization |
| BTNR | Bayesian True Negative Rate |
| CD | Critical Difference value |
| CIA | Confidentiality, Integrity and Availability |
| CICIDS2017 | Canadian Institute for Cybersecurity Intrusion Detection System 2017 dataset |
| CIDS | Cloud Intrusion Detection Service |
| CNN | Convolutional Neural Networks |
| CSP | Cloud Service Provider |
| CSTEM | Cloud Service Trust Evaluation Model |
| DBN | Deep Belief Networks |
| DDoS | Distibuted Denial of Service attack |
| DIDS | Distributed Intrusion Detection System |
| DNN | Deep Neural Networks |
| DoS | Denial of Service attack |
| DR | Detection Rate |
| EA | Evolutionary Algorithm |
| EC | Evolutionary Computation |
| FAR | False Alarm Rate |
| FC | Critical Friedman test value |

| | |
|---|---|
| FCPM | Finite Context Prediction Model |
| FN | False Negative |
| FNR | False Negative Rate |
| FP | False Positive |
| FPC | F-score Per Cost |
| FPR | False Positive Rate |
| FPSBPSO | Fitness Proportionate Selection Binary Particle Swarm Optimization |
| FPSBPSO-E | Fitness Proportionate Selection Binary Particle Swarm Optimization-Entropy feature selection algorithm |
| FS | Friedman test Statistic |
| FSHMPM | Finite State Hidden Markov Prediction Model |
| FTP | File Transfer Protocol |
| GA | Genetic Algorithm |
| GP | Genetic Programming |
| GRU | Gated Recurrent Unit |
| GRU-RNN | Gated Recurrent Unit Recurrent Neural Networks |
| GUI | Graphical User Interface |
| HDLTex | Hierarchal Deep Learning for Text classification |
| HIDS | Host-based Intrusion Detection System |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| HWPM | Holt-Winters Prediction Model |
| IaaS | Infrastructure as a Service |
| ICMP | Internet Control Message Protocol |
| IDPS | Intrusion Detection and Prevention System |
| IDS | Intrusion Detection System |
| IoT | Internet of Things |
| IPS | Intrusion Prevention System |
| IT | Information Technology |
| L2R | Local to Remote attack |
| LSTM | Long Short-Term Memory |
| LSTM-RNN | Long Short-Term Memory Recurrent Neural Networks |

| | |
|---|---|
| MC | Misclassification of attack Class |
| MCC | Matthews Correlation Coefficient |
| MR | Miss Rate |
| MR | Missed Rate |
| MRF | Markov Random Field |
| NIDS | Network-based Intrusion Detection System |
| OCSVM | One-Class Support Vector Machine |
| PaaS | Platform as a Service |
| PC | Personal Computer |
| PCA | Principle Component Analysis |
| PR | Precision-Recall curves |
| Probe | Probing attack |
| PSO | Particle Swarm Optimization |
| PST | Probabilistic Suffix Tree |
| PU | Purdue University dataset |
| RBM | Restricted Boltzmann Machine |
| RNN | Recurrent Neural Networks |
| ROC | Receiver Operating Characteristic curves |
| SaaS | Software as a Service |
| SEA | Schonlau Et Al. dataset |
| SGD | Stochastic Gradient Decent |
| SMTP | Simple Mail Transfer Protocol |
| SNMP | Simple Network Management Protocol |
| SVM | Support Vector Machine |
| TCP | Transmission Control Protocol |
| TN | True Negative |
| TNR | True Negative Rate |
| TP | True Positive |
| TPR | True Positive Rate |
| U2R | User to Root attack |
| UDP | User Datagram Protocol |
| VM | Virtual Machine |
| VMM | Variable Order Markov Model |

| | |
|---|---|
| WR | Wrong Rate |
| WS | Wilcoxon test Statistic |

# 1. GİRİŞ

Siber güvenlik, 1950'lerden bu yana kritik bir endişe kaynağı olmuştur. İnternetin yaygınlığına dayanarak, ağ kullanıcıları ve organizasyonlar arasında bilgi paylaşımının yaygınlığı hızla artmıştır. Bu konu, dünya çapında sayısız gizlilik veya güvenlik saldırılarını mümkün kılmıştır. Bu nedenle, işletmeler, hükümetler, yasa koyucular, akademisyenler, araştırmacılar, bilim insanları ve halk arasında siber güvenlik konusunda büyük zorluklar potansiyel bir risk olarak ortaya çıkmıştır (Kadam, 2011).

Bulut bilgi işlem dünyadaki en yeni gelişen teknolojidir. Bilgi işlem gücü, veri depolama, platform, yazılım ve bilgi gibi paylaşılan bilgisayar sistem kaynaklarının kullanılabilirliği, kullanıcının doğrudan aktif yönetimi olmadan sağlandığı, internet üzerinden isteğe bağlı bir bilgi işlem teknolojisidir. Bu dinamik olarak sanallaştırılmış ve ölçeklendirilebilir kaynaklar, üçüncü taraf bir sağlayıcıdan bir hizmet olarak kiralanır ve ardından kullanıcı yalnızca kullandığı kaynakları öder (Modi vd., 2013).

Bulutun masaüstüne bir şey yüklemeye gerek kalmadan sunduğu çeşitli özellikler nedeniyle, bulut bilgi işlem ağı ağ alanıyla ilişkili kişiler için tercih edilen bir ortam haline gelir. Ayrıca, mevcut ağ sistemi dünyasındaki kullanıcılar, geliştiriciler ve yöneticiler, bulut bilişimi farklı etkinlikler için yaygın şekilde kullanmaktadır. Bu nedenle, son zamanlarda güvenlik sağlama, bulut bilişimde en zorlu konulardan biri haline geldi. Kaynaklardan dolayı bu, bulut ortamındaki tüm sunucular, bireyler ve kullanıcılar arasında paylaşılmaktadır (Mehmood vd., 2013).

## 1.1. Araştırma Sorunu Beyanı

Günümüzde Nesnelerin İnterneti (IoT) cihazlarının gömülü olduğu, bağlı olduğu ve büyük miktarda veri ürettiği büyük bir veri dünyasıyla karşı karşıyayız. Çeşitli kötü amaçlı yazılım çeşitlerinin ve tehditlerin daha hızlı bir şekilde ortaya çıktığı güvenlik endüstrisi ve akademi için bir meydan okuma üretiyorlar, ancak

bilinen saldırı imzasına bağlı olarak çalışma zamanında bunlarla baş edemiyoruz. Buna ek olarak, bulut ortamındaki Saldırı Tespit Sistemi (IDS) için mevcut araştırma perspektifleri çoğunlukla IDS'nin bu ortak ağların ana omurgasından geçen muazzam trafik hacmini idare etme kabiliyetine ve IDS'nin yapılanmasına odaklanmıştır. Bulutun güvenlik altyapısının bir parçası olmak. Öte yandan, yakın tarihli çalışmaların birçoğu, IDS'yi buluttaki müşterilere hizmet olarak sınırlı bir şekilde sağlama yeteneğini araştırmıştır. Başka bir deyişle, IDS'nin mevcut uygulamaları bulutun müşterisinin görüşlerini dikkate almadı (Dhage vd., 2011). Bu araştırma, yukarıda belirtilen kaygının çözümüne katkıda bulunmak için bulutun üst katmanında bir Bulut İzinsiz Giriş Tespiti Hizmeti (CIDS) önermektedir.

### 1.2. Araştırma Önemi

Son günlerde, hesaplama makinelerinin gücü, özellikle Yapay Zeka (AI) alanında büyük bir gelişme gösterdi. Makine öğreniminin ileri mimarileri, özellikle derin öğrenme, güvenlik alanında kullanılmakta ve yeni sonuçlar ve sorunlar bildirilmektedir. AI ile, daha önce olduğu gibi herhangi bir güvenlik uzmanı bilgisine ihtiyaç duymadan, IDS'yi gerçekleştirmenin yanı sıra saldırı tespitinde doğruluk ve sağlamlık derecelerini önemli ölçüde artırabiliriz. Bu araştırmanın önemi, bulut ortamındaki saldırı tespit eleştirisinin farklı bir perspektiften ele alınmasından kaynaklanıyor, temel olarak IDS'nin müşterilere çeşitli optimize edilmiş derin öğrenme modellerinden oluşan bir topluluk mimarisini kullanan üçüncü taraf bir hizmet olarak sunulmasına olanak sağlama olanakları. Buna göre, bu araştırma CIDS için yeni bir derin öğrenme yaklaşımı önermek için derin öğrenmenin güçlü yeteneklerini kullanacaktır.

### 1.3. Araştırma Amacı

Bu araştırmanın amacı şu şekilde dört katıdır: i) Maskeli saldırı tespiti ve ağ saldırı tespit alanında geniş kapsamlı ampirik çalışmaların tanıtılması; ii) Derin öğrenme modelinin hiperparametre seçimi için bir Parçacık Sürü Optimizasyonu (PSO) tabanlı bir algoritma önermek, ardından hem özellik hem

de hiperparametre seçimleri için çift PSO tabanlı bir algoritma elde etmek için ikincisini geliştirmek; iii) Kötü niyetli paketleri etkili bir şekilde tespit etmek için en iyi duruma getirilmiş üç derin öğrenme modelinden oluşan bir dizi kullanan güçlü bir öngörücü bulut IDS tasarımı önermek; iv) Son olarak, bulut kullanıcıları tarafından yapılandırılabilen entegre bir CIDS tanıtılması.

### 1.4. Araştırma Kapsamı

Bu araştırma, bir bulut bilişim ortamında gelişmiş CIDS gelişimi ile ilgili zorlukları ve sorunları sunmaktadır. Son yaygın bulut hizmetlerini ve kaynaklarını güvence altına almak için en son farklı araştırma çalışmalarını bir araya getirerek CIDS'i geliştirmenin olası çözümüne dikkat çekmeyi amaçlıyor. Ayrıca, yeni bir derin öğrenme yaklaşımı kullanarak bulut bilişim için istenen CIDS'i tanımlar.

### 1.5. Hedef Kitle

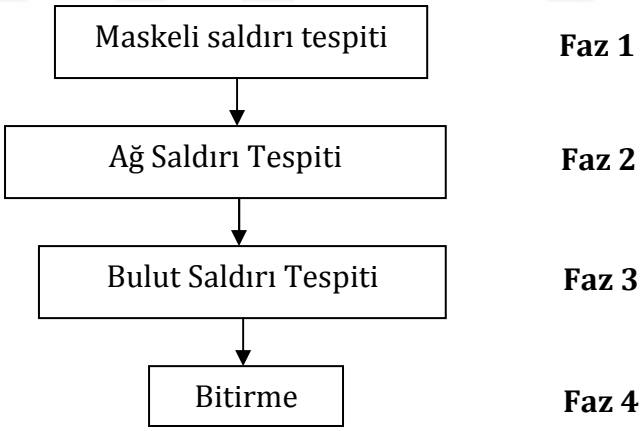
Bu tez, bulut bilişim ve saldırı algılama sistemleri hakkında temel bilgilere sahip okuyuculara yöneliktir. Ayrıca, bulut girişini tespit etmek ve önlemek için kullanılan makine öğrenme yöntemleri ve bulutun üstündeki derin öğrenme modellerinin faydalarını bir hizmet olarak kullanarak bir bulut ağının nasıl korunacağını bilmek istiyorlar. Bulut saldırı tespit sistemleri geliştiren araştırmacılar için bu çalışma daha fazla sistem gelişimi için kullanışlıdır.

### 1.6. Araştırma Metodolojisi

Mevcut bulut saldırı tespit veri setlerinin olmamasından dolayı, çalışmalarımızı bu tez boyunca üst üste dört aşamaya böldük:

- Üç ücretsiz maskeli algılama veri kümesinden yararlanarak Kişisel Bilgisayar (PC) ortamında maskeli algılama. Bu aşamanın amacı, üç derin öğrenme modelinin ön eğitim ile etkinliğini, önerilen hiperparametre seçimi için önerilen bir PSO tabanlı algoritma kullanarak araştırmaktır. Ayrıca, birçok değerlendirme ölçütünü kullanarak performanslarını karşılaştırmak.

- İki ücretsiz IDS veri setini kullanarak ağ ortamında saldırı algılama. Bu aşamanın amacı, önerilen çift PSO tabanlı algoritmayı kullanarak hesaplamaları azaltmak, kaynakları korumak ve saldırı tespitinde üç derin öğrenme modelinin performansını artırmaktır. İkili PSO tabanlı algoritma, biri özellik seçimi için iki farklı PSO değişkeni kullanır, ikincisi ise hiperparametre seçimi için önceki aşamada önerdiğimiz şey. Ayrıca, önerilen ikili PSO tabanlı algoritma ile ön eğitim kullanılırken ve ön eğitim kullanmadan derin öğrenme modellerinin performansının derin bir analizi, hem ikili hem de çok sınıflandırma görevleri için birçok değerlendirme ölçümü kullanılarak sağlanır.
- Bulut ortamında, beş modül içeren tümleşik bir IDS tasarlayarak saldırı algılama. Modüllerin çalışması önceki iki aşamadaki bulgularımızdan faydalanmaktadır. Daha sonra, bulut dışındaki kullanıcılarla ilgilenebilecek olan CIDS'in yapısını sunuyoruz.
- Bitirme tezini ve önerileri yazmak için bitirme aşaması. Şekil 1.1, tez metodolojisinin akış şemasını göstermektedir.




Şekil 1.1. Tez metodolojisinin akış şeması

### 1.7. Tez Organizasyonu

Bu tezde beş bölüm vardır. Dört bölümün geri kalanı aşağıdaki şekilde düzenlenmiştir. Bölüm 2, maskeli saldırı tespiti alanında kapsamlı bir deneysel çalışma sunmaktadır. 3. Bölümde, önerilen derin öğrenme yaklaşımının ağa

saldırı tespit alanındaki etkinliğini doğrulamak için karşılaştırmalı bir çalışma başlattık. Ardından, Bölüm 4, yeni bir bulut IDS'nin tasarımını açıklar ve bulut kullanıcıları ile ilgilenen üçüncü taraf bir CIDS'in işlevselliğini tartışır. Son olarak, 5. Bölümde gelecekteki çalışmalar için sonuçlar ve tavsiyeler verilmiştir.

# 2. LİTERATÜR ÖZETİ

Bu bölümde, siber güvenlik alanında derin öğrenmenin metodoloji ve sonuçları ile ilgili kullanımı üzerine son çalışmaları sunuyoruz.

## 2.1. Maskeli Saldırı Tespiti

Maskeli saldırı tespiti, bilgisayar güvenliği alanındaki önemi ve hassaslığı nedeniyle son on yılda aktif olarak araştırılmıştır. Kısacası ve bu çalışmanın kapsamını kısıtlaması için, temel olarak, literatürdeki makine öğrenme yaklaşımlarını ve iyi bilinen UNIX komut satırı tabanlı veri setlerini kullanarak anomaliye dayalı maskeli saldırı algılamaya odaklandık.

İlk önce Schonlau vd. (2001), SEA adlı bir UNIX komut satırı tabanlı veri seti önerdiklerinde. Ayrıca SEA veri yapılandırmasında çeşitli istatistiksel yöntemler kullanmış ve sonuçları karşılaştırmışlardır. Kısa sürede, SEA veri seti anomaliye dayalı maskeli saldırı saptama teknikleri alanında çok popüler hale gelir. Okamoto vd. (2003), SEA veri yapılandırmasında bağışıklık tabanlı bir Gizli Markov Modeli sundu ve 60% Hit ve 1% Yanlış Alarm Oranı (FAR) elde ettiler. Naive Bayes , metin sınıflandırma görevleriyle iyi çalışan ünlü bir sınıflandırıcıdır. İlk olarak SEA veri yapılandırmasına Maxion ve Townsend (2002) tarafından, biri güncelleme kullanıcı profili (Hit=61.5%, FAR=1.3%), diğeri güncelleme yapılmayan (Hit=61.5%, FAR=1.3%) olmak üzere iki modelde uygulanmıştır. Ayrıca, SEA 1v49 adlı SEA veri kümesinden yeni bir veri yapılandırması önerdiler ve aynı zamanda SEA 1v49 veri yapılandırmasını güncelleyerek Naive Bayes sınıflandırıcısını test ettiler ve 62.8% Hit ve 4.6% FAR'a sahiplerdi. Wang ve Stolfo (2003), SEA veri yapılandırmasında bir Naive Bayes sınıflandırıcısı (Hit=70%, FAR=2%) ve One-Class Destek Vektör Makinesi (OCSVM) modelini (Hit=70%, FAR=4%) uygulamıştır. Yung (2003) çalışmalarında SEA veri yapılandırmasına uygulanan güncelleme ve geri bildirimle bir Naive Bayes sınıflandırıcısı sunmuştur (Hit=76%, FAR=2%). Önceki çalışmasını geliştirdi ve SEA veri yapılandırmasını güncelleyerek

kendiliğinden tutarlı bir Naive Bayes modeli önerdi (Yung, 2004). Daha iyi sonuçlar aldı ve Hit'i 79%'a yükseltti, ancak FAR hala 2%'dir.

Destek Vektör Makinesi (SVM) aynı zamanda hem sınıflandırma hem de regresyon için kullanılan iyi bilinen bir makine öğrenme yöntemidir. Chen ve Aritsugi (2006), SEA veri yapılandırmasına uygulanan Eigen Birlikte Oluşma Matrisini kullanarak çevrimiçi güncelleme ile maskelenme tespiti için SVM tabanlı bir yöntem getirmiştir. Tek sınıflı (Hit= 62.77%, FAR =6%) ve İki Sınıflı (Hit=72.24%, FAR=3%) sınıflandırma modelleri için önerilen yöntemlerini test ettiler. Li vd. (2006), kullanıcı davranışının temel özelliklerini Prensip Bileşen Analizi (PCA) kullanarak Correlation Eigen Matrix'ten çıkardılar, daha sonra bu özellikleri SEA veri yapılandırması üzerine SVM tabanlı maskeli saldırı algılama sistemi ile beslediler. Hit=82.6% ve FAR=3% ile çok iyi sonuç aldılar. Kim ve Cha (2005), oylama motorlu SVM sınıflandırıcısını kullanarak maskeli saldırı tespiti alanında ampirik bir çalışma yaptılar. SVM sınıflandırıcılarını, iki UNIX komut satırı tabanlı veri setinde, yani SEA veri kümesi ve Greenberg veri kümesi (Greenberg, 1988) üzerinde test ettiler. SEA veri kümesi için, SVM sınıflandırıcılarını SEA veri yapılandırması (Hit=80.1%, FAR=9.7%) ve SEA 1v49 veri yapılandırması (Hit=94.8%, FAR=0.7%) olmak üzere iki farklı veri yapılandırmasına uyguladılar. Buna ek olarak, SVM sınıflandırıcılarını Greenberg veri kümesi için iki farklı veri konfigürasyonuna uyguladılar, yani Greenberg Kesilmiş ve Greenberg Zenginleştirilmiş veri konfigürasyonları (Maxion, 2003). Greenberg Kesilmiş veri yapılandırması için Hit=71.1% ve FAR=6%, bu arada Greenberg Zenginleştirilmiş veri yapılandırması için Hit=87.3% ve FAR=6.4%. Yang vd. (2007), maskeli saldırılarını tespit etmek için string çekirdek sınıflandırıcıya sahip bir OCSVM sundu. Sınıflandırıcılarını, iki adet UNIX komut satırı tabanlı veri setinde, yani SEA veri kümesi ve PU veri kümesi (Lane ve Brodley, 1997) üzerinde test ettiler. SEA veri seti için modellerini SEA veri yapılandırmasına uyguladılar (Hit=62%, FAR=1.5%) ve PU veri seti için modellerini PU Zenginleştirilmiş veri yapılandırmasına (Hit=60%, FAR=2%) uyguladılar.

Greenberg Kesilmiş ve Greenberg Zenginleştirilmiş veri yapılandırmalarında hem güncellenen kullanıcı profili olan bir Naive Bayes modeli (Maxion, 2003) tanıtıldı . Buna karşın, Greenberg Kesilmiş veri yapılandırması, bir Hit=70.9% ve FAR=4.7%, Greenberg Zenginleştirilmiş veri yapılandırması, bir Hit=82.1% ve FAR=5.7% verdi. Gebski ve Wong (2005), PU Zenginleştirilmiş veri yapılandırmasında maskeli saldırı tespiti için ağaç tabanlı bir model sundu (Hit=85%, FAR=10%). REDDY ve PUSHPALATHA (2014), maskeli saldırıları tespit etmek için şartlı bir Naive Bayes sınıflandırıcısı önermiştir. Sınıflandırıcılarını, SEA, Greenberg ve PU veri setleri olmak üzere üç farklı UNIX komut satırı tabanlı veri seti üzerinde test ettiler. SEA veri kümesi için, sınıflandırıcılarını SEA veri yapılandırması (Hit=84%, FAR=8.8%) ve SEA 1v49 veri yapılandırması (Hit=90.7%, FAR=1%) olmak üzere iki veri yapılandırmasına uyguladılar. Greenberg veri kümesi için, sınıflandırıcılarını Greenberg Zenginleştirilmiş veri yapılandırmasına uyguladılar (Hit=84.13%, FAR=9.4%). Son olarak, sınıflandırıcılarını PU Zenginleştirilmiş veri yapılandırması üzerinde test ettiler ve bir Hit=84% ve bir FAR=8% elde ettiler.

Çizelge 2.1 , yukarıdaki önceki çalışmaların en iyi sonuçlarının, her veri kümesi için Hit yüzdesi bakımından bir özetini sunmaktadır. Çizelge 2.1'den fark edebileceğimiz gibi, daha yüksek Doğruluk ve Vuruş (Hit) için bir maskeli saldırı algılama modelinin ve daha düşük FAR değerlerinin geliştirilmesi hala büyük bir zorluktur.

## 2.2. Ağ Saldırı Tespiti

Özellikle, son on yılda, ağa saldırı tespitinde derin öğrenmenin kullanılması üzerine yoğun olarak araştırılmışlar. Gerçekten de, ağ saldırı tespitinden önce sadece özellik seçiminde kullanılan makalelere odaklandık. Tang vd. (2016), yazılım tanımlanan ağ, ağ saldırı saptanması için bir DNN modeli önerilmiştir. Bu NSL-KDD veri kümesi altı özelliklere eğitilmiş. Ahmad (2015), Ana Bileşen Analizi (PCA) NSL-KDD özelliği transformasyonu için kullanılır. Sonra PCA elde edilen özellik alt küme PSO'ları algoritması kullanılarak optimize edilmiştir.

Optimize edilmiş özellikleri ağ saldırı tespiti için bir modüler Neural Network (MNN) model ile birlikte kullanılır.

Çizelge 2.1. Maskeli saldırı tespiti alanındaki ilgili çalışmaların en iyi sonuçları

| Model | Veri seti | Yapılandırma | Hit (%) | FAR (%) |
|---|---|---|---|---|
| HMM | SEA | SEA | 60 | 1 |
| Saf Bayes | SEA | SEA | 79 | 2 |
| | | SEA 1v49 | 62.8 | 4.6 |
| | Greenberg | Greenberg Kesilmiş | 70.9 | 4.7 |
| | | Greenberg Zenginleştirilmiş | 82.1 | 5.7 |
| Koşullu Naif Bayes | SEA | SEA | 84 | 8.8 |
| | | SEA 1v49 | 90.7 | 1 |
| | Greenberg | Greenberg Zenginleştirilmiş | 84.13 | 9.4 |
| | PU | PU Zenginleştirilmiş | 84 | 8 |
| SVM | SEA | SEA | 82.6 | 3 |
| | | SEA 1v49 | 94.8 | 0 |
| | Greenberg | Greenberg Kesilmiş | 71.1 | 6 |
| | | Greenberg Zenginleştirilmiş | 87.3 | 6.4 |
| | PU | PU Zenginleştirilmiş | 60 | 2 |
| Ağaç tabanlı | PU | PU Zenginleştirilmiş | 85 | 10 |

Chae vd. (2013), Bunlar Özellik Oranı (AR) kullanılarak bir özellik seçme yöntemi önerilmiştir. Sonra onlar özellik alt kümesini seçmek için NSL-KDD, önerilen yöntem uygulanır. Onlar bir karar ağacı sınıflandırıcı seçilen özellikler test etti. Wahba vd. (2015), onlar Korelasyon bazlı Özelliği Seçimi (FCS) ve Bilgi Kazancı (IG) dayalı bir hibrit özellik seçme yöntemi önerdi. Önerilen yöntem NSL-KDD uygulanır ve Naif Bayes sınıflandırıcısı (AdaBoost) tekniği Arttırılması Adaptif kullanılarak seçilen özellikler ile ilgili eğitilmiştir. Sasan ve Sharma (2016), bir yanlış algılama yaklaşımı SIRA (CART) kullanılarak sunulmuştur. Önerilen model NSL-KDD veri kümesi üzerinde uygulanır. Kurban vd. (2013), yazarlar melez İki Katmanlı davranışsal temelli özellik seçimi yaklaşımı önerdiler. Önerilen yöntem, NSL-KDD veri setinde değerlendirilir. Ambusaidi vd. (2016), karşılıklı bilgilere dayanarak bir özellik seçme yöntemi önerilmiştir ve optimal özellikler NSL-KDD üzerinde En Küçük Kareler Destek Vektör Makinesi tabanlı IDS (LSSVM-IDS) üzerinde test edilir.

Naidoo vd. (2015), yazarlar Küme Geçerlilik Endeksleri denilen iki aşamalı özelliği seçme yöntemini girmiştik. Birinci aşamada bir K-ortalama küme algoritması aday özelliği belirli alt kümelere NSL-KDD uygulanır. Daha sonra, ikinci aşamada, bir genetik algoritma (GA) uygun bir özellik alt grubu tespit etmek için kullanılır. Nkiama vd. (2016), özellik seçimi bir yaklaşım bir karar ağacı sınıflandırıcı kullanılarak özyinelemeli özelliği ortadan kaldırılması ile bağlantılı tek değişkenli özellikler seçimi kullanılarak kullanılır. Bu NSL-KDD veri seti üzerinde test edilmiştir. Pervez ve Farid (2014), onlar NSL-KDD çoklu özellik belirli alt kümelere bir SVM sınıflandırıcı kullanmıştır. Sonra onlar çok sınıflı sınıflandırma için bir SVM sınıflandırıcı bu alt kümelerini test edilmiş ve sonuçlar kaydedildi.

Çizelge 2.2, başarılarıyla ilgili çalışmaların sonuçları özetlemektedir. Çizelge 2.2'den fark edebileceğimiz gibi, yüksek Algılama Oranı (DR) ve düşük FAR ile optimal özellik alt kümesine uygulanan ağ saldırı tespiti için derin bir öğrenme yaklaşımı geliştirmek hala büyük bir zorluktur.

Çizelge 2.2. Ağ saldırı tespiti alanındaki ilgili çalışmaların sonuçları

| **Çalışma** | **Özelliklerin sayısı** | **DR (%)** | **FAR (%)** |
|---|---|---|---|
| Tang vd. (2016) | 6 | 76 | - |
| Ahmad (2015) | 8 | 99.4 | 0.6 |
| Chae vd. (2013) | 22 | - | - |
| Wahba vd. (2015) | 13 | - | 4.2 |
| Sasan ve Sharma (2016) | 29 | 88,23 | - |
| Kurban vd. (2013) | 20 | - | - |
| Ambusaidi vd. (2016) | 18 | 98.76 | 0.28 |
| Naidoo vd. (2015) | 19 | - | - |
| Nkiama vd. (2016) | 12 | 99.79 | - |
| Pervez ve Farid (2014) | 36 | 82 | 15 |

## 2.3. Bulut Saldırı Tespiti

Bulut saldırı tespiti alanında daha önce yapılmış pek çok çalışma var. Ancak bu çalışmada, kullanılan algılama yönteminden ziyade sadece bulut IDS'lerinin ve hizmetlerinin tasarımına odaklanıyoruz. Başlamak için, Yee vd. (2007), web servislerini belirli saldırılara karşı korumak için bir IDS tasarlamıştır. Önerilen

web hizmeti bulut kullanıcıları tarafından kontrol edilemez. Bosin vd. (2008) tarafından yeni nesil IDS modeli ortaya kondu. Kullanıcıların gerektiğinde izinsiz giriş saldırı tespit hizmetlerine erişmelerini sağlamak için bir web hizmeti önerdiler. Ayrıca, web servisleri bulut kullanıcılarının tüm bulut IDS'lerini kontrol etmelerine izin vermez.

Vieira vd. (2009), veri paketlerini yakalamak ve daha sonra bunları bir hibrit bulut IDS kullanarak analiz etmek için bulut katman katmanında bir CIDS önermiştir. Teklif ettikleri CIDS'i taklit ettiler ve performansını gerçek zamanlı bulut ortamı için kabul edilebilir buldular. Buna rağmen, bulut kullanıcılarına CIDDS'lerinde bir raporlama mekanizması dahil etmediler. Roschke vd. (2009), Laureano vd. (2007)'nın önerdiği VM tabanlı IDS'yi temel alan bir web servisiyle işbirliği yapan merkezi bir bulut IDS yönetimini tanıttı. Çerçeveleri, farklı bulut katmanlarından veri toplamak için birçok sensör kullandı. Daha sonra denetim verileri veri tabanında saklanır ve analiz bileşeni tarafından analiz edilir. Merkezi IDS bir saldırı tespit ederse, kullanıcıyı bilgilendirecek ve IDS'yi izlemesi ve yönetmesi için tam kontrol sağlayacaktır. Dastjerdi vd. (2009), mobil ajanları kullanarak bulutlarda izinsiz giriş saldırı tespiti için dağıtılmış bir sistem önermektedir. Bilinen ve bilinmeyen saldırıları gerçek zamanlı olarak tespit etmek için her VM'de bulundu. Buna rağmen, sistemlerinde ziyaret edilecek VM sayısı üzerinde bir zayıflık noktası vardır.

Lo vd. (2010), bulut bilişim ağları için işbirliğine dayalı bir IDS çerçevesi önermiştir. Dağıtık çerçeveleri, bir web servisi ve bir IDS olan iki ayrı bileşenden oluşuyordu. Web servisi, bulutu DDoS saldırılarından immünize etmek için tasarlanmıştır. Öte yandan, IDS, Ragsdale vd. (2000) ve Spafford ve Zamboni (2000)'nın çalışmalarına göre uygulanmaktadır. Mazzariello vd. (2010), bir NIDS'i, kullanıcı sanal makinelerini barındıran fiziksel makinenin sanal anahtarına yerleştirmiştir. NIDS, büyük hacimli verilerin yük paylaşımı açısından etkinliğini kanıtladı. Bakshi ve Dujodwala (2010), bulutu DDoS saldırılarından korumak için VM'lerde bir IDS tanıttı. Yalnızca bilinen DDoS saldırılarını etkili bir şekilde tespit edebilen bir NIDS'ti.

Patel vd. (2011), çoğu zaman saldırı türlerini gerçek zamanlı olarak tespit etmek için otonomik bir ajan tabanlı özyönetim bulutu Saldırı Tespit ve Önleme Sistemi (IDPS) önerdi. Buna rağmen, işlerinin uygulanması hakkında herhangi bir ayrıntı vermediler. Gul ve Hüseyin (2011) tarafından çoklu iş parçacığı kullanarak büyük miktarda trafik akışını gerçekleştirebildiği dağıtılmış bir bulut IDS modeli tanıtıldı. Ayrıca, bulut kullanıcısına bilgi izlemekten ve göndermekten ve CSP'ye bir uzman tavsiyesi göndermekten sorumlu üçüncü taraf bir CIDS uyguladılar. Lee vd. (2011), bulut bilişimde her konuk işletim sisteminde çok düzeyli bir HIDS ve günlük yönetimi tasarladı. Hibrit HIDS, saldırıları hızlı bir şekilde tespit edebiliyor, ancak yüksek seviye kullanıcılar için daha fazla kaynak tüketiyor.

Ayrıca, bu alandaki önemli araştırmalardan biri, bulut tabanlı bir IDS için bir çerçeve geliştirdiklerinde, Kholidy ve Baiardi'in (2012) çalışmalarıdır. Çerçeveleri, tehditleri melez bir şekilde tespit edebilen merkezi bir koordinasyon olmadan ölçeklenebilir ve esnektir. Bir saldırı tespit edilirse, sistem CSP'yi bir raporla uyarır. Alsafi vd. (2012), hem IDS hem de İzinsiz giriş saldırı Önleme Sistemini (IPS) tek bir mekanizmada birleştiren etkili ve verimli bir IDPS tanıttı. Önerilen IDPS, çeşitli saldırıları tespit edip durdurabilen bir karma algılama sistemidir. Zarrabi ve Zarrabi (2012) tarafından bulut bilişim tabanlı ölçeklenebilir bir CIDS geliştirilmiştir. Bulut bilişim özelliklerinden yararlanmak yerine, siber saldırıların üstesinden gelir. Shaikh ve Sasikumar (2012) çalışmasında, bulut ortamlarında güvenliği sağlamak için güvene dayalı bir çerçeve önerilmiştir. Önerilen çerçeve, hem CSP hem de kullanıcılar için bir güvenlik gücü değerlendirmesi olarak kullanılabilecek bir güven değeri hesaplamasına bağlıdır. Shelke vd. (2012), çok iş parçacıklı dağıtılmış bulut tabanlı bir IDS'yi tanıttı. Bulut içindeki saldırıları tespit etmek için karma algılama yöntemini kullanır ve üçüncü taraf bir bulut hizmeti tarafından kullanıcılara raporlar gönderir.

Kholidy vd. (2013), sistem olaylarını izleyebilen, analiz edebilen ve risk seviyesini değerlendirebilen hiyerarşik, özerk ve öngörmeye dayalı bir bulut IDS önermiştir. Holt-Winters algoritmasını çalıştıran bir tahmin motoruna dayanan

bir NIDS'dir. Kholidy vd. (2014), önerdikleri Autonomic Cloud Intrusion Detection Framework (ACIDF) için üç saldırı tahmin modeli sundu. Olasılıklı Son Ek Ağacı (PST) ile Değişken Sipariş Markov Modeli (VMM) kullanan Sonlu Bağlam Tahmin Modeli (FCPM), Sonlu Durum Gizli Markov Öngörü Modeli (FSHMPM) ve Holt-Winters Öngörü Modeli (HWPM) olarak üç modeli incelediler. İncelenen modellerin DARPA veri setinde doğrulandığında iyi sonuçlar elde ettiklerini bildirdiler. Bulut ortamına yasadışı erişim için bir tespit yöntemi Alguliev ve Abdullaeva (2014) tarafından önerilmektedir. Önerilen anomaliye dayalı tespit yöntemi, buluttaki herhangi bir anormal davranışı kosinüs benzerliği yöntemi kullanarak ve işbirlikçi filtreleme yöntemini uygulayarak tespit etti. Wang vd. (2019), bir Bulut Hizmeti Güven Değerlendirme Modeli (CSTEM) tasarladılar. Ağırlıkların birleştirilmesi ve gri korelasyon analizine dayanan önerilen model.

Son olarak, yukarıda değinilen literatürden, bazı çalışmaların çok iş parçacıklı bir bulut IDS önererek büyük bir bulut verisi akışını ele alma problemini çözmeye yoğunlaştığı fark edilebilir. Diğerleri, saldırı gerçekleştiğinde bulut kullanıcılarıyla ilgilenmek için üçüncü taraf bir bulut hizmeti tasarlamaya çalıştı. Bu arada geri kalan, buluta bilinen ve bilinmeyen saldırılara karşı bağışıklık kazandırmak için hibrit algılama IDS'yi tanıttı. Bu nedenle, üçüncü taraf kullanıcı dostu bir bulut hizmeti ile çok iş parçacıklı, hibrit ve dinamik olan tümleşik bulut tabanlı bir IDS'nin tasarımı hala büyük bir zorluktur.

# 3. MASKELİ SALDIRI TESPİTİ

Maskeli saldırı tespiti, özel bir saldırı tespit tipidir. Etkili ve erken saldırı tespiti, bilgisayar güvenliği için çok önemli bir faktördür. Her ne kadar kayda değer çalışmalar on yıldan fazla bir süre maskeli saldırının tespitine odaklanmış olmasına rağmen, bu alanda derin öğrenme modellerini kullanan derin bir çalışmanın varlığı nadirendir.

## 3.1. Giriş

Bilgisayar güvenlik alanında, bir maskeleyici gerçek bir müşteriyi taklit etmek isteyen bir davetsiz misafir olarak tanımlanır. Bir maskeli saldırısı, bir maskeli meşru bir kullanıcının kimlik bilgilerini kullanarak meşru bir kullanıcının bilgisine yetkisiz erişime ulaştığında gerçekleşir. Bu saldırıların bilgisayar güvenliğine yönelik en ciddi tehditler arasında olduğu düşünülmektedir. Bu tür saldırıları önlemenin en etkili yolu, tüm kullanıcılar için izleme sağlayabilen ve herhangi bir anormal davranışı arayabilen IDS kullanmaktır (Huang, 2010).

Bilgisayar güvenliği tasarımı, iki ortak IDS yaklaşımına sahiptir: imza tabanlı algılama ve anomali tabanlı algılama. İmza bazlı tespit veya yanlış tespit olarak da adlandırılan, maskeli saldırı imzası zaten bilindiğinde kullanım için değerlidir. Alternatif olarak, Anomaliye dayalı tespit, bilinen veya bilinmeyen maskeli saldırıları için kullanılabilir. Bu avantaj anomaliye dayalı algılama yaklaşımını popüler hale getirir ve son on yılda bu konuda çok sayıda önceki çalışma yayınlanmıştır (Bertachini ve Fierens, 2008). Anomaliye dayalı algılama yaklaşımının ardındaki ana fikir, kullanıcı davranışını her kullanıcı hakkında çeşitli bilgiler toplayarak profillendirmek ve daha sonra bazı özellikleri baz alarak her kullanıcı için bir profil oluşturmak için bu bilgileri kullanmaktır. Sistem kullanıldığında, kullanıcı tarafından yapılan son etkinlikleri orijinal profille karşılaştırmak için bir güvenlik kontrolü yapılır. Kullanıcı davranışı var olan normal profilden saparsa, oturum olası bir maskeli saldırı olarak sınıflandırılır. Pek çok anomaliye dayalı tespit tekniği kullanılmaktadır, ancak bunlar arasında, bilgisayardan öğrenme yöntemleri verilerden öğrenme ve

normal ve kötü niyetli kullanıcıları ayırt edebilmeleri nedeniyle en sık kullanılan yaklaşımlardır (Erbacher vd., 2007).

Sınıflandırma görevleri için geleneksel (sığ) makine öğrenme yöntemlerinin kullanılmasının popülaritesine rağmen, bunlar aşağıdaki gibi ele alınması gereken birçok eksikliğe sahiptir: tam özelliklerin gösterilmesi perspektifi, problemin karmaşıklığı ve statik sınıflandırma uygulamalarının sınırlandırılması (Deng, 2014). 2006'da Yapay Sinir Ağına dayalı, derin öğrenme olarak adlandırılan yeni bir temsil öğrenme kavramı öne sürdü. Derin öğrenme, hiyerarşik mimarilerde örüntü tanıma veya sınıflandırma için birçok bilgi işleme aşaması katmanına sahip bir makine öğrenme teknikleri sınıfı olarak kabul edilir. Sığ makine öğrenim yöntemlerinin eski eksikliklerinin üstesinden gelmek yerine; Son zamanlarda birçok araştırma alanında büyük başarılar elde ediyor. Derin öğrenmenin temel avantajları şu şekilde özetlenebilir: uygulanabilirliği, denetimsiz özellik öğrenme veya veri kümelerinden çıkarma yeteneğine sahip olma ve güçlü kendi kendine öğrenme yeteneğine sahip olma (Du vd., 2016). Autoencoder (AE), Derin İnanç Ağları (DBN), Konvolüsyonel Sinir Ağları (CNN) ve Tekrarlayan Sinir Ağları (RNN) olmak üzere dört tipik öğrenme modeli vardır . Derin öğrenme başarısı ve istikrarı sağlamak için, günümüzde şu gibi çok çeşitli uygulamalarda aktif ve sürekli olarak kullanılmaktadır: bilgisayarla görme, doğal dil işleme ve saldırı algılama sistemleri.

### 3.2. Veri Setleri ve Yapılandırmalar

Bu bölümde veri setleri, veri yapılandırmaları ve ayrıca eğitim ve test metodolojisi açıklanmaktadır. Aslında, davranışını modellemek için her bir kullanıcı hakkında bilgi toplamak ve ardından normal profilini oluşturmak için kullanılabilecek çeşitli mekanizmalar vardır: kullanıcı komut satırları geçmişi, Grafiksel Kullanıcı Arabirimi (GUI), kullanıcı dosya sistemi navigasyonu ve sistem çağrıları; işletim sistemi seviyesi. Bu yazıda, kullanıcıların UNIX komut satırı geçmişini temel alan üç veri setini seçtik: SEA, Greenberg ve PU. İnternette ücretsiz ve halka açık olmalarından ziyade, anomaliye dayalı maskeli saldırı

algılama alanında en sık kullanılan veri setleridir, bu nedenle sonuçlarımız öncekilere göre kolayca karşılaştırılır. Çizelge 3.1, veri setlerini ve özelliklerini göstermektedir.

Çizelge 3.1. Veri setleri ve özellikleri

| Veri seti | Host platformu | Kullanıcı sayısı | Denetim biçimi | Zenginleştirilmiş? | Kirlenmiş? | Oturumlar? | Gerçek maskeli saldırılar? | Yıl |
|---|---|---|---|---|---|---|---|---|
| SEA | Unix | 50 | Unix Komutları | Hayır | Evet | Hayır | Hayır | 2001 |
| Greenberg | Unix | 168 | Unix Komutları | Evet | Hayır | Evet | Hayır | 1988 |
| PU | Unix | 8 | Unix Komutları | Evet | Hayır | Evet | Hayır | 1997 |

### 3.2.1. SEA veri seti

En son yayınlanan maskeli saldırı saptama alanına odaklanmış yazılar bu veri setini kullanmıştır. SEA (Schonlau Et Al.), ücretsiz bir UNIX komut satırı tabanlı veri setidir. Birkaç ay boyunca 50 farklı kullanıcıdan gelen komutları toplamak için UNIX *acct* denetim aracını kullandılar. SEA veri seti, her kullanıcı için 15000 komut kümesi içerir ve bu komutlar yalnızca bu kullanıcı tarafından verilen komut adlarını içerir. Her kullanıcı için, 15000 komut kümesi, her biri 100 komut içeren 150 bloğa bölünmüştür. Her kullanıcı için ilk 50 blok orijinal kabul edilir ve bir eğitim seti olarak kullanılır. Her kullanıcının kalan 100 bloğu bir test seti olarak kabul edilir. Test bloklarından bazıları, diğer kullanıcıların verileriyle rasgele kirletilir, yani her kullanıcının test setinde 0 ila 24 blok arasında değişen maskeli saldırı bloklar vardır. Bu veri setiyle ilgili iki veri yapılandırması literatürde kullanılmıştır: SEA ve SEA 1v49.

#### 3.2.1.1. SEA

50 kullanıcının her biri için ayrı bir sınıflandırıcı oluşturulmuştur. Her sınıflandırıcıyı iki profil oluşturmak için eğittik: belirli bir kullanıcının ilk 50 bloğunu kullanan öz-davranış için bir profil ve diğer 49 kullanıcının (49 x 50) eğitim bloklarını kullanan öz-davranış için bir profil. Her kullanıcının test seti alt bölüm 3.2.1'de açıklandığı gibi olacaktır.

#### 3.2.1.2. SEA 1v49

Her kullanıcı için bir sınıflandırıcı oluşturulmuştur ve verilerinin yalnızca ilk 50 eğitim bloğu ile eğitilmiştir. Öte yandan, her bir kullanıcı için belirlenen test , diğer 49 kullanıcının ilk 50 antrenman bloğundan oluşur ve bu, 76 ila 100 blok arasında değişen orijinal normal bloklarına ek olarak 2450 maskeli saldırı bloğu ile sonuçlanır.

### 3.2.2. Greenberg veri seti

Bu veri seti daha önceki çalışmalarda yaygın olarak kullanılmıştır. *Csh* kabuğunu kullanan 168 UNIX kullanıcısından toplanan komutları içerir. Bu veri setinin kullanıcılarının, aşağıdaki dört gruptan birine üye olduğu düşünülmektedir: acemi programcılar, deneyimli programcılar, bilgisayar uzmanları ve programcı olmayanlar. Bu veri seti zenginleştirilmiştir, yani her kullanıcı için oturumun başlangıç ve bitiş zamanı, çalışma dizini, komut adları, komut parametreleri, komut takma adları ve bir hata bayrağı hakkında bilgiler içeren oturumlar vardır. Bu veri setiyle ilgili iki veri yapılandırması literatürde kullanılmıştır: Greenberg Kesilmiş ve Greenberg Zenginleştirilmiş.

#### 3.2.2.1. Greenberg kesilmiş

Bu yapılandırmada, öncelikle kesilmiş komut satırlarını yalnızca komut adlarını içeren Greenberg veri setinden çıkardık. Daha sonra, Greenberg veri setinde mevcut olan 168 kullanıcıdan, 2000 ila 5000 komutu normal olan 50 kullanıcı arasından rastgele seçtik. Ardından, 50 kullanıcının her birinin komutlarını her biri 10 komutlu bloklara böldük. Her kullanıcının ilk 100 bloğu eğitim seti olacaktır. Oysa sonraki 100 blok, test setinde öz-davranışın doğrulanması olarak kullanılacak. Ondan sonra, kalan 118 kullanıcıdan ek 25 kullanıcıyı rastgele seçtik. Daha sonra, 50 normal kullanıcının her biri için, maskeleyicilerinin verilerinden rastgele 30 blok seçtik ve bunları test setinde rastgele konumlara girdik; bu da test için toplam 130 blokla sonuçlandı.

#### 3.2.2.2. Greenberg zenginleştirilmiş

Greenberg kesilmiş'de açıklanan metodolojiye sahiptir, ancak yalnızca bir farkla, bu veri yapılandırması için sadece zenginleştirilmiş komut satırlarını Greenberg veri setinden çıkardık. Zenginleştirilmiş komut satırı, kullanıcı tarafından girilen tüm diğer adlarla birlikte girilen komut adı ve komut parametrelerinin birleştirilmesi anlamına gelir. Greenberg Zenginleştirilmiş veri konfigürasyonu için yukarıda açıklanan Greenberg Zenginleştirilmiş veri konfigürasyonu, 50 normal kullanıcının her birine sahiptir: eğitim için 100 blok ve test için 130 blok.

### 3.2.3. PU veri seti

Purdue Üniversitesi (PU) veri seti, 2 yıl boyunca Purdue Üniversitesi'ndeki 8 farklı kullanıcıdan toplanan temizlenmiş komutları içerir. Bu veri seti zenginleştirilmiştir, bu da komut adlarına ek olarak içerdiği anlamına gelir, ayrıca: komut parametreleri, bayraklar ve kabuk meta karakterleri. Ayrıca, bu veri setinin 8 kullanıcının her biri için oturumları vardır. Buna ek olarak, her kullanıcının verileri bir belirteç akışına işlenir. Burada belirteç, komut adı veya komut parametresi anlamına gelir. Bu veri setiyle ilgili iki veri yapılandırması literatürde kullanılmıştır: PU Kesilmiş ve PU Zenginleştirilmiş.

#### 3.2.3.1. PU kesilmiş

Bu konfigürasyon için öncelikle, kesilmiş jetonları PU veri setinden, yani sadece komut adlarını içeren jetonları çıkarttık. Daha sonra, PU veri setinde mevcut 8 kullanıcının her biri için verilerini her 10 jetondan oluşan bloklara böldük. Ardından, her kullanıcının ilk 150 bloğu eğitim seti olarak kabul edilecektir. Bundan sonra, her kullanıcı için bir sonraki 50 blok, test setinde öz-davranışın doğrulanması olarak kullanılacaktır. Maskeli saldırı faaliyetlerini simüle etmek için, her bir kullanıcı için, diğer yedi kullanıcının test verilerini (7 x 50) ekledik; bu, 8 kullanıcının her biri için toplam 400 blok test sonucu elde etti.

### 3.2.3.2. PU zenginleştirilmiş

PU kesilmiş'de açıklanan aynı metodolojiye sahiptir, ancak PU Zenginleştirilmiş veri yapılandırması için sadece bir farkla, burada yalnızca zenginleştirilmiş jetonları, yani PU veri setindeki tüm jetonları çıkardık. PU Kesilmiş veri konfigürasyonuna gelince, PU Zenginleştirilmiş veri konfigürasyonu, 8 kullanıcının her birine sahiptir: eğitim için 150 blok ve test için 400 blok. Çizelge 3.2, veri yapılandırmalarıyla ilgili tüm detayları özetlemektedir.

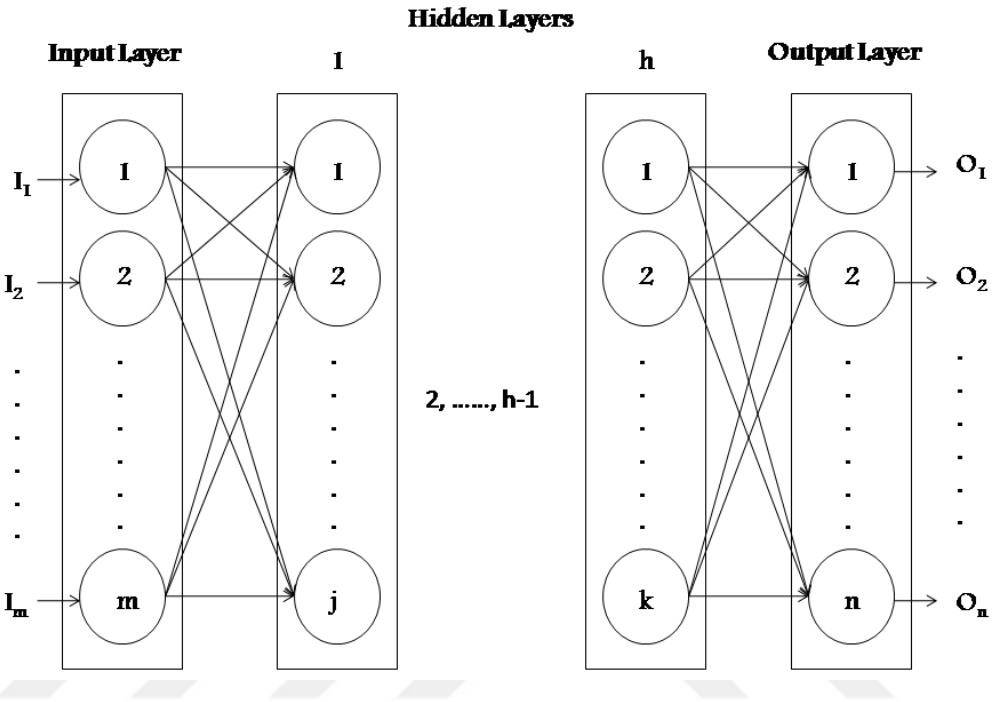
Çizelge 3.2. Kullanılan veri yapılandırmalarının yapısı

| **Özellikleri** | | **Veri yapılandırması** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **SEA** | **SEA 1v49** | **Greenberg Kesilmiş** | **Greenberg Zenginleştirilmiş** | **PU Kesilmiş** | **PU Zenginleştirilmiş** |
| kullanıcı sayısı | | 50 | 50 | 50 | 50 | 8 | 8 |
| Blok boyutu | | 100 | 100 | 10 | 10 | 10 | 10 |
| Her kullanıcı için blok sayısı | Eğitim Seti | 2500 | 50 | 100 | 100 | 150 | 150 |
| | Test seti | 100 | 2526~2550 | 130 | 130 | 400 | 400 |
| | Toplam | 2600 | 2576~2600 | 230 | 230 | 550 | 550 |
| Tüm kullanıcılar için blok sayısı | Eğitim Seti | 125000 | 2500 | 5000 | 5000 | 1200 | 1200 |
| | Test seti | 5000 | 127269 | 6500 | 6500 | 3200 | 3200 |
| | Toplam | 130000 | 129769 | 11500 | 11500 | 4400 | 4400 |
| Eğitim setinin dağıtımı | Normal | 2500 | 2500 | 5000 | 5000 | 1200 | 1200 |
| | Maskeli saldırı | 122500 | 0 | 0 | 0 | 0 | 0 |
| | Toplam | 125000 | 2500 | 5000 | 5000 | 1200 | 1200 |
| Test setinin dağıtımı | Normal | 4769 | 4769 | 5000 | 5000 | 400 | 400 |
| | Maskeli saldırı | 231 | 122500 | 1500 | 1500 | 2800 | 2800 |
| | Toplam | 5000 | 127269 | 6500 | 6500 | 3200 | 3200 |

## 3.3. DNN Hiperparametre Seçimi

Bu bölümde, Derin Sinir Ağlarının (DNN) hiperparametrelerini seçmek için PSO tabanlı bir algoritma sunacağız . Bu algoritma, maskeli saldırı tespiti için DNN inşa etme deneylerimizde ilerlememize yardımcı olacaktır. DNN, birçok gizli katmanı olan çok katmanlı bir Yapay Sinir Ağıdır. DNN'nin ağırlıkları tam olarak bağlanmıştır, yani, herhangi bir belirli katmandaki her nöron, bu belirli katmana

bitişik olarak yerleştirilmiş olan yüksek dereceli katmanın tüm nöronlarına bağlanır (Deng, 2014). DNN'deki bilgiler ileri beslemeli bir şekilde, yani girdilerden gizli katmanlar yoluyla çıktılara yayılır. Şekil 3.1 , tipik bir DNN'nin temel yapısını gösterir.




Şekil 3.1. Tipik bir DNN'nin temel yapısı

DNN'ler çeşitli makine öğrenme görevlerinde yaygın olarak kullanılır. Buna ek olarak, makine öğrenme tekniklerinin çoğunu performans açısından geçme yeteneklerini kanıtlamışlardır (Liu vd., 2015). Bununla birlikte, herhangi bir DNN'nin performansı, hiperparametrelerinin değerlerinin seçimine dayanır. DNN hiperparametreleri , temel makine öğrenme görevinde bu DNN'nin yapısını, davranışını ve performansını kontrol eden kritik parametreler kümesi olarak tanımlanır. Aslında, iki tür hiperparametre vardır : genel parametreler ve katman tabanlı parametreler. Global parametreler, DNN'nin öğrenme oranı, çağ sayısı, parti büyüklüğü, katman sayısı ve kullanılan optimize edici gibi genel davranışını tanımlayan parametrelerdir. Öte yandan, katman tabanlı parametre değerleri DNN'deki her katmana bağlıdır. Katman bazlı parametrelerin örnekleri, bunlarla sınırlı olmamak üzere, katman tipi, ağırlık başlatma yöntemi, aktivasyon işlevi ve bir dizi nörondur.

Sorun, bu hiperparametrelerin görevden göreve değişmesi ve eğitim sürecinden önce ayarlanması gerektiğidir. Bu sorunun üstesinden gelmek için bilinen bir çözüm, DNN hiperparametrelerini tam olarak ayarlamak için temel makine öğrenme görevine uygun bir uzman bulmaktır . Ne yazık ki, böyle bir uzmanın varlığı her durumda mevcut değildir. Başka bir olası çözüm, bu hiperparametrelerini deneme yanılma yöntemiyle manuel olarak ayarlamaktır. Bu, ızgara arama veya rastgele arama gerçekleştirerek hiperparametrelerin alanını arayarak ele alınabilir ( Bergstra vd., 2011; Bergstra ve Bengio, 2012). Bir ızgara araştırması, bu aralığın daha önce altta yatan görevin önceki bilgilerine bağlı olarak tanımlandığı tanımlanmış hiperparametre aralıkları üzerinde gerçekleştirilir . Bundan sonra, kullanıcı art arda önceden belirlenmiş aralıklardan gelen hiperparametre değerlerini toplar ve DNN'nin antrenman setindeki performansını test eder. Tüm olası hiperparametre değerlerinin kombinasyonu test edildiğinde, DNN'yi yapılandırmak ve test setinde test etmek için en iyi kombinasyon seçilir. Rastgele arama, grid aramaya benzer, ancak hiperparametre değerlerini metodik bir şekilde toplamak yerine, kullanıcı rastgele önceden tanımlanmış aralıklardan hiperparametre değerlerini seçer. Snoek vd. (2012), Bayesian Optimizasyonuna dayanan bir hiperparametre seçim yöntemi önermiştir. Bu yöntemde, kullanıcı, bir sonraki deney için hiperparametrelerin nasıl ayarlanacağına karar vermek için herhangi bir deneyden elde edilen bilgileri kullanarak hiperparametreleri seçme bilgisini geliştirir. Izgara ile elde edilen iyi sonuçlar olsa bile, bazı durumlarda rasgele ve Bayesian optimizasyonu aranır, ancak genel olarak, DNN hiperparametre değerlerinin karmaşıklığı ve geniş arama alanı bu tür manuel algoritmaları olanaksız ve çok yorucu bir arama işlemi yapar.

Evrimsel Algoritma (EA), doğrusal olmayan bir işlevin global optimosunu bulmak için mükemmel bir performans gösteren meta-sezgisel bir algoritmadır, özellikle çoklu yerel minima veya maksima olduğunda. EA'lar, DNN parametreleştirme problemini otomatik olarak çözmek için umut verici algoritmalar olarak düşünmektedir. Literatürde, mümkün olduğunca yüksek bir doğruluk değeri elde etmek için DNN hiperparametrelerini optimize etmede EA'ları kullanmayı amaçlayan çok sayıda çalışma önerilmiştir. En ünlü

EA'lardan biri olan Genetik Algoritma (GA), ağ parametrelerini optimize etmek için kullanılmış ve ilk ağırlıklar tanımını içeren çaprazlama ve mutasyon operatörleri arasında Taguchi metodu uygulanmıştır (Abdalla vd., 2014). GA'lar ayrıca, çok sınıflandırma görevine dayanan denetlenen adımdan önceki eğitim öncesi adımda da kullanılır (Belharbi vd., 2016). Eğitim süresini azaltmak için GA kullanan bir başka yaklaşım (Tirumala vd., 2016). GA, sinir ağının ağırlıklarını geliştirerek DNN'yi geliştirmek için kullanılır (David ve Greental, 2014). Kötü amaçlı yazılım sınıflandırma görevleri için DNN hiperparametrelerini optimize eden otomatik GA tabanlı bir yaklaşım (Martin vd., 2017).

Ayrıca, PSO aynı zamanda en tanınmış ve popüler EA'lardan biridir. Lorenzo vd. (2017a, b), PSO'yu kullandı ve iki yaklaşım önerdi; ilki sıralı ve ikincisi paraleldir, herhangi bir DNN'nin hiperparametrelerini optimize etmek için. Daha sonra Nalepa ve Lorenzo (2017), eski iki yaklaşımın yakınsama kabiliyetini resmen kanıtladı ve bunları sırasıyla tek bir iş istasyonunda ve sıralı ve paralel yaklaşımlar kümesinde ayrı ayrı test etti. Son olarak, Ye (2017), DNN hiperparametrelerini büyük ölçekli ve yüksek boyutlu verilerde seçmek için otomatik bir PSO tabanlı bir algoritma önerdi. Böylece, DNN için hiperparametreleri otomatik olarak seçmemizi sağlamak için PSO kullanmaya karar verdik. Daha sonra, bu algoritmanın bir maskeli saldırı tespiti senaryosunda kullanılan statik sınıflandırma deneyleri için nasıl uyarlanacağını açıklayacağız. Alt bölüm 3.3.1, standart PSO'nun nasıl çalıştığını gözden geçirmenin gerekli ve kısa bir önsözünü ortaya koymaktadır. Ardından, bu bölümün geri kalanı, DNN hiperparametrelerini optimize etmek için önerilen PSO tabanlı algoritmamızı sunar.

### 3.3.1. Sürekli PSO

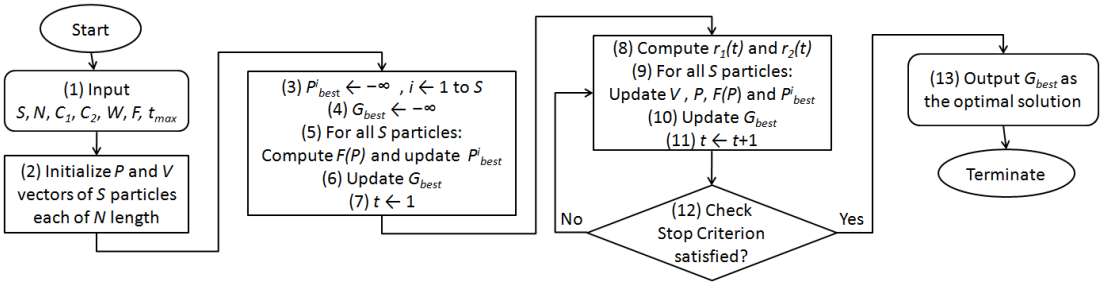
PSO, sürekli arama alanında doğrusal olmayan fonksiyonları optimize etmek için bir meta-sezgisel algoritmadır. Eberhart ve Kennedy (1995) tarafından önerilmiştir. PSO, hayvanların sosyal davranışlarını taklit etmeye çalışır. Sürü kavramı, parçacık adı verilen birçok üyeden oluşan bir kavramdır. Sürüdeki

partiküllerin sayısı *S* ile ifade edilen ve sürü büyüklüğü olarak adlandırılan bir tamsayıdır. Belirli bir kümedeki her partikül *N* uzunluğundaki iki vektöre sahiptir, burada *N*, problemin tanımlanmış değişkenlerinin (boyutların) büyüklüğüdür. İlk vektöre konum vektörü, problemin arama alanındaki o partikülün o anki pozisyonunu tanımlayan *P* ile işaret etmektedir. Her pozisyon vektörü, problemin bir aday çözümü olarak düşünülebilir. İkinci vektör, bir sonraki yinelemede problemin arama uzayındaki o partikülün hem hızını hem de yönünü belirleyen hız vektörü (*V*) olarak adlandırılır. PSO'nun yürütülmesi sırasında, her yinelemede iki vektör daha kaydedilmelidir. Birincisi, kişisel en iyi vektör olarak adlandırılan $P^i_{best}$, şimdiye dek keşfedilen sürünün içindeki en iyi pozisyonu gösteren $P^i_{best}$ ile ifade edilir. Sürüdeki her partikül, diğer partiküllerden bağımsız kişisel en iyi vektörüne sahiptir ve her yinelemede güncellenir. İkinci vektör, şimdiye kadar sürgünün üzerinde bulunan en iyi konumu gösteren $G_{best}$ 'in küresel en iyi vektörüdür. Sürüdeki tüm parçacıklar için tek bir küresel en iyi vektör vardır ve her tekrarda güncellenir. En iyi kişisel vektöre, partikülün bilişsel bilgisi olarak bakılabilirken, küresel en iyi vektör, sürünün sosyal bilgisini temsil eder. Matematiksel olarak, her bir yinelemede sürüde *S* her parçacık için *i*, *V* hızı ve *P* vektörleri, sırasıyla Denklem 3.1 ve 3.2'ye göre bir sonraki yineleme *t+1*'e güncellenir.

$$V^i_{t+1} = WV^i_t + C_1 r_1(t)\left(P^i_{best} - P^i_t\right) + C_2 r_2(t)\left(G_{best} - P^i_t\right) \quad (3.1)$$

$$P^i_{t+1} = P^i_t + V^i_{t+1} \quad (3.2)$$

*W,* bir sonraki denemede mevcut tekrarda parçacığın hızının etkisini kontrol eden sabit Atalet ağırlık, böylece partikül hızı ve yönü parçacığının arama alanının dışında olsun izin verme için ayarlanabilir yerlerde sorun. Bu arada, $C_1$ ve $C_2$ sabitleri ve ivme katsayıları, $r_1$ ve $r_2$ olarak bilinen homojen [0,1] dağıtılan, rastgele değerlerdir. Her yineleme başlangıcında, $r_1$ ve $r_{2'nin}$ yeni değerler rasgele bilgisayarlı ve bu tekrarında sürüsü tüm partiküller için sabitlerdir edilir. $C_1$, $C_2$, $r_1$ ve $r_2$ sabitleri kullanılarak amacı parçacık bilişsel bilginin her iki ölçek ve hız değişiklikleri sürüsü sosyal bilgisi. Bu nedenle, tüm parçacıkların yeni pozisyon vektörleri, problemin en uygun çözümüne yaklaşacaktır. Şekil 3.2, sürekli PSO'nun akış şemasını göstermektedir.




Şekil 3.2. PSO akış şeması

Kısacası, PSO aşağıdaki gibi çalışır: İlk önce, kullanıcı gibi bazı gerekli girişleri girer: küme büyüklüğü ($S$), Parçacıkların boyutları ($N$), hızlanma sabitleri ($C_1$, $C_2$), atalet ağırlık sabiti ($W$), uygunluk fonksiyonu $(F)$ partikül sorun alanı performansı ve yineleme maksimum sayıda ($t_{max}$) skoru. Daha sonra, PSO sürüşteki tüm parçacıklar için belirtilen boyutlarda konum ve hız vektörlerini rastgele olarak başlatır. Daha sonra, PSO sürüdeki her parçacık için kişisel en iyi vektörü belirtilen boyutlarla başlatır ve çok küçük bir değere ayarlar. Ayrıca, PSO sürünün global en iyi vektörünü belirtilen boyutlarla başlatır ve çok küçük bir değere ayarlar. PSO, fitness fonksiyonunu kullanarak her partikül için fitness skorunu hesaplar ve tüm partiküller için kişisel en iyi vektörleri ve sürünün global en iyi vektörünü günceller. Bundan sonra, PSO, sırasıyla, Denklem 3.1 ve 3.2'ye göre, her bir parçacık için $r_1$ ve $r_2$ rasgele, güncellemeler hız ve konum vektörlerinin hesaplanması ile ilk yineleme başlar. Buna ek olarak, PSO verilen fitness fonksiyonuna göre her bir partikül için fitness skorunu tekrar hesaplar ve bu partikülün bu yineleme sırasındaki fitness skoru kişisel en iyi vektörün fitness skorundan büyükse, her partikül için en iyi kişisel vektörü günceller bu parçacığın ($F(P_t^i)>F(P_{best}^i)$). Ayrıca, PSO, partiküllerin kişisel en iyi vektörünün zindelik puanlarından herhangi birinin, swarm'ın küresel en iyi vektörünün zindelik puanından daha büyük olması durumunda, kümenin global en iyi vektörünü günceller ($F(P_{best}^i)>F(G_{best})$ , $i$=1 to $S$). Ardından, PSO durma ölçütünü kontrol eder ve eğer tatmin edilirse, PSO global en iyi vektörü en uygun çözüm olarak verir ve sonlandırır. Aksi takdirde, PSO bir sonraki yinelemeye ilerler ve yukarıdaki ilk yinelemede açıklanan prosedürü, durma kriterine ulaşana kadar tekrarlar.

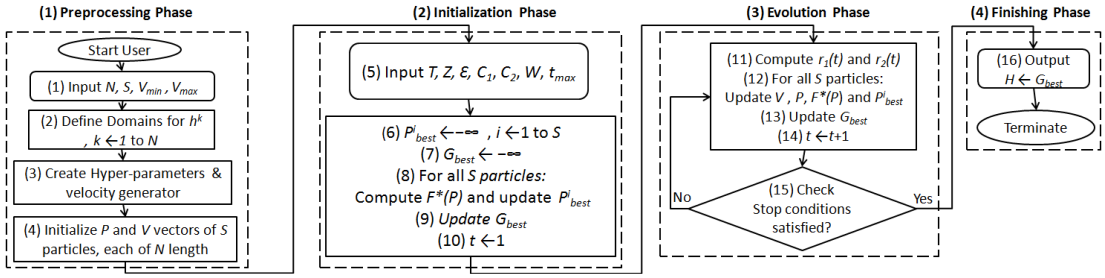
Durma kriteri, egzersiz hatası önceden tanımlanmış bir değerden ($\mathcal{E}$) küçük olduğunda veya maksimum yineleme sayısına ulaşıldığında gerçekleşir. Son olarak, PSO basitlik ve genellik bakımından GA'dan daha iyi bir performans sergiliyor (Escalante vd., 2009). PSO, GA'dan daha basittir çünkü yalnızca bir operatör içerir ve uygulaması kolaydı. Ayrıca, PSO'nun genelliği, PSO'nun herhangi bir optimizasyon problemine uygulanacak herhangi bir modifikasyona ihtiyaç duymadığı ve hesaplamaları azaltan ve kaynakları koruyan optimum çözüme yakınlaşmanın daha hızlı olduğu anlamına gelir.

### 3.3.2. Hiperparametre seçimi için önerilen PSO tabanlı algoritma

DNN'nin hiperparametrelerinin seçimi ,bir optimizasyon görevi olarak yorumlanabilir, bu nedenle ana amaç, *M'nin* DNN modeli ve *T'nin* eğitim seti olduğu kayıp fonksiyonunu $L(M,T)$ en aza indirmektir. Bu hedefe ulaşmak için, PSO'yu optimize edilmiş hiperparametrelerin vektörünü çıkaran optimizasyon algoritmamız olarak seçtik. Hiperparametreler $H$ tarafından ayarlanan DNN $M$ modelini oluşturduktan sonra $L$ fonksiyonunu en aza indiren $H$ ve eğitim seti *T'de* eğitildi. PSO tabanlı algoritmamızın fitness fonksiyonu $F^*$: $R^N \rightarrow R$, bu vektörü hyperparameters tarafından ayarlanmış ve test grubu $Z$ test eğitimli DNN bir gerçek değerli doğruluk değerine $N$ bir uzunluğa sahiptir hyperparameters bir gerçek değerli vektörü haritalar *R.* Başka bir deyişle, PSO tabanlı algoritmamız test setindeki eğitimli DNN'nin doğruluğunu en üst düzeye çıkarmak için verim veren olası tüm hiperparametre kombinasyonları arasında optimum hiperparametreler vektörünü bulur. Ayrıca, optimize edilecek ve DNN kullanarak herhangi bir sınıflandırma görevine kolayca uyarlanabilecek DNN'den bağımsız olması anlamına gelen PSO tabanlı algoritmamızın genelliğini sağlamak için, kullanıcının çalışmalarında hangi hiperparametrelerin kullanmak istediğini seçmesine izin vereceğiz. Bu nedenle, kullanıcının her bir parametrenin türünü ve alanını olduğu kadar hiperparametrelerin sayısını da tanımlaması sorumludur. Bir parametrenin alanı, bu parametrenin tüm olası değerlerinin kümesidir. Bundan sonra, PSO tabanlı algoritmamız, her bir parçanın sürüsündeki hiperparametreler vektörünü başlatmak için

tanımlanmış parametrelerin sayısına ve alanlarına bağlı olan özel bir yerleşik jeneratör kullanacaktır.

Önerilen algoritmanın yürütülmesi sırasında ve her bir yinelemede, doğrulama işlemi, önceden tanımlanmış parametrelere uygun olacak şekilde güncellenmiş konum ve hız vektörlerini doğrulamak için önerilen algoritmaya dahil edilir. Son olarak, hesaplamaları azaltmak ve daha hızlı bir şekilde birleştirmek için, her yinelemenin sonunda iki farklı durma koşulu aynı anda kontrol edilir. Birincisi, küresel en iyi vektörün zindelik puanı, kullanıcı tarafından belirtilen bir eşikten $\mathcal{E}$ daha az arttığı zaman meydana gelir. Önceki koşulun amacı, maksimum yineleme sayısına henüz ulaşılmamış olsa bile, küresel en iyi vektörün daha da geliştirilemeyeceğini garanti etmektir. İkinci koşul, maksimum yineleme sayısı gerçekleştirildiğinde gerçekleşir. Birinci veya ikinci koşul yerine getirildikten sonra önerilen algoritm en iyi çözüm olarak global en iyi $H$ vektörünü verir ve arama işlemini sonlandırır. Şekil 3.3, PSO tabanlı hiperparametreler seçim algoritmamızın akış şemasını göstermektedir.




Şekil 3.3. Önerilen PSO tabanlı algoritmanın akış şeması

### 3.3.3. PSO parametreleri

PSO parametrelerinin değerinin seçimi ($S$, $V_{max}$, $V_{min}$, $C_1$, $C_2$, $W$, $t_{max}$, $\mathcal{E}$) çok karmaşık bir süreçtir. Neyse ki, bu sorunu çözmek için birçok deneysel ve teorik önceki çalışma yayınlanmıştır (Shi ve Eberhart, 1998; 1999; Kennedy ve Mendes, 2002; Clerc ve Kennedy, 2002). Alınabilecek PSO parametrelerinin bazı önerilen değerlerini sundular. Çizelge 3.3, her PSO parametresini ve karşılık gelen önerilen değeri veya aralığı ve seçilen değeri gösterir.

Çizelge 3.3. PSO parametreleri önerilen değerler veya aralıklar

| Parametre | Değer / Aralık | Seçilmiş değer |
|---|---|---|
| $S$ | [5,40] | 20 |
| $V_{min}$ | {0,1} | 0 |
| $V_{max}$ | {0,1} | 1 |
| $C_1$ | [1,5] | 2 |
| $C_2$ | [1,5] | 2 |
| $W$ | [0.4,0.9] | 0.9 |
| $t_{max}$ | [30,50] | 30 |
| $\mathcal{E}$ | [0.001,0.0001] | 0.0001 |

### 3.4. Deneysel Kurulum ve Modeller

Bu bölüm, deneysel deneylerimizi gerçekleştirmenin metodolojisini ve maskeli saldırıları tespit etmek için kullandığımız derin öğrenme modellerinin açıklamasını açıklar. Bölüm 3.2'te belirtildiği gibi, üç UNIX komut satırı tabanlı veri seti (SEA, Greenberg, PU) seçtik. Bu veri kümelerinin her biri, her metin dosyasının bir kullanıcıyı temsil ettiği bir metin dosyaları koleksiyonudur. Belirli bir veri kümesindeki her kullanıcının metin dosyası, o kullanıcı tarafından verilen bir dizi UNIX komutu içerir. Bu veri kümelerinin gerçek maskelenmeyenleri içermediği gerçeğini yansıtmaktadır.

Bununla birlikte, maskeleyicileri simüle etmek ve bu veri setlerini maskeli saldırı algılamada kullanmak için, deneylerimize devam etmeden önce özel veri yapılandırmaları uygulanmalıdır. Bölüm 3.2'e göre ve alt bölümleri, her veri setinin iki farklı tür veri yapılandırmasına sahiptir. Bu nedenle, her birinin ayrı ayrı gözleneceği altı veri yapılandırması elde ettik. Sonuç olarak, her model için altı bağımsız deney elde edilir. Son olarak, maskeli saldırı tespiti bu veri konfigürasyonlarına statik sınıflandırma ve dinamik sınıflandırma olmak üzere iki farklı ana yaklaşımı izleyerek uygulanabilir. Takip eden iki alt bölüm, aralarındaki farkı ve her biri için derin öğrenme modellerinden yararlanıldığını gösterir.

### 3.4.1. Statik sınıflandırma yaklaşımı

Statik sınıflandırma yaklaşımında, sınıflandırma görevi, bir dizi statik özellik tarafından temsil edilen bir örnek veri seti kullanılarak gerçekleştirilir (Martin vd., 2017). Bu statik özellikler, sınıflandırmanın uygulanacağı görevin niteliğine göre tanımlanır. Buna ek olarak, veri seti örnekleri veya gözlem olarak da adlandırılanlar, bu sınıflandırma görevi alanında çalışan bazı uzmanlar tarafından elle toplanır. Bundan sonra, bu örnekler sırasıyla seçilen modeli eğitmek ve test etmek için eğitim ve test setleri olarak bilinen iki bağımsız kümeye bölünmüştür. Statik sınıflandırma yaklaşımının artıları ve eksileri de vardır. Daha hızlı ve kolay bir çözüm sunsa da, statik özelliklere sahip kullanıma hazır bir veri kümesi gerektirir. Bu veri setinin varlığı bazı karmaşık sınıflandırma görevlerinde bulunmayabilir. Bu nedenle, statik özelliklere sahip bir veri seti oluşturma denemesi zor bir görev olacaktır. Çalışmamızda, altı farklı veri yapılandırması uygulamak için üç ünlü UNIX komut satırı tabanlı veri setinin varlığını kullanmaya karar verdik. Özel veri konfigürasyonundaki her kullanıcı, bir dizi statik özellik ile temsil edilen belirli sayıda bloğa sahiptir. Aslında, bu özellikler, kullanıcının davranışını tanımlamaktan sorumlu olan ve daha sonra sınıflandırıcının maskeli saldırı algılaması için sınıflandırıcıya yardımcı olan kullanıcının UNIX komutlarıdır. Uygulanan altı veri konfigürasyonundaki statik maskeli saldırı tespit görevini gerçekleştirmek için DNN ve RNN adlı iki iyi bilinen derin öğrenme modelini kullanmaya karar verdik.

#### 3.4.1.1. DNN

İçinde Bölüm 3.3, detaylar DNN yapısını ve hiperparametreleri seçimi sorun açıkladı. Ayrıca verilen eğitim ve test setlerinde DNN'nin doğruluğunu en üst düzeye çıkaran optimum hiperparametreler vektörünü elde etmek için PSO tabanlı bir algoritma önerdik. Bu alt bölümde, önerilen PSO tabanlı algoritmayı ve DNN'yi, statik maskeli saldırı tespit görevinde, SEA, SEA 1v49, Greenberg Kesilmiş, Greenberg Zenginleştirilmiş, PU Kesilmiş ve PU Zenginleştirilmiş altı veri yapılandırması kullanarak nasıl kullandığımızı açıklıyoruz. Bunların her

veri yapılandırması, Bölüm 3.2'te açıklandığı gibi kendi yapısına ve belirli sayıda kullanıcıya sahiptir. Böylece, altı ayrı DNN deneyimiz olacak, her deney veri yapılandırmalarından birinde yapılacak.

DNN deneylerimizin metodolojisi, arka arkaya dört aşamadan oluşur, bunlar : başlatma, optimizasyon, sonuç çıkarma ve sonlandırma aşamaları. İlk aşama, gerekli tüm işletim parametrelerini başlatmak ve her bir dosyanın o veri konfigürasyonunda bir kullanıcıyı temsil ettiği belirli veri konfigürasyonunun dosyalarını hazırlamaktır. Kullanıcı dosyası, bu kullanıcının test setini takip eden eğitim setinden oluşur. Şöyle, tüm DNN-deneyler için tüm PSO parametrelerini ayarlamak : $S$=20, $V_{min}$=0, $V_{max}$= 1, $C_1$=$C_2$=2, $W$=0.9, $t_{max}$=30 ve $\mathcal{E}$=$10^{-4}$. Ardından, başlatma aşamasındaki son adım DNN'nin hiperparametrelerini ve etki alanlarını tanımlamaktır. On iki farklı DNN hiperparametresi kullandık ($N$=12). Çizelge 3.4, her DNN hiperparametresini ve ilgili tanımlanmış alanını göstermektedir.

Çizelge 3.4. Hiperparametreler ve tanımlanmış alanları

| **Hiperparametre** | **Alan** | **Tip** |
|---|---|---|
| Öğrenme oranı | [0.01,0.9] | Sürekli |
| Moment | [0.1,0.9] | Sürekli |
| Çürüme | [0.001, 0.01] | Sürekli |
| Ayrılma oranı | [0.1,0.9] | Sürekli |
| Gizli katman sayısı | [1,10] | Adım 1 ile ayrık |
| Gizli katmanların nöron sayısı | [1,100] | Adım 1 ile ayrık |
| Devir sayısı | [5,100] | Adım 5 ile ayrık |
| Parti boyutu | [100,1000] | Adım 50 ile ayrık |
| Doktoru | {Adagrad, Nadam, Adam, Adamax, RMSprop, SGD} | Adım 1 ile ayrık |
| Başlatma fonksiyonu | {Zero, Normal, Lecun uniform, Uniform, Glorot uniform, Glorot normal, He uniform, He normal} | Adım 1 ile ayrık |
| Katman tipi | {Dropout, Dense} | Adım 1 ile ayrık |
| Aktivasyon fonksiyonu | {Linear, Softmax, Relu, Sigmoid, Tanh, Hard sigmoid, Softsign, Softplus} | Adım 1 ile ayrık |

Kullanılan tüm hiperparametreler aşağıdakiler dışında nümeriktir: Doktor, Katman tipi, Başlatma fonksiyonu ve Aktivasyon fonksiyonu hiperparametreleri kategoriktir. Bu durumda, olası tüm değerlerin bir listesi 1 ile o listenin uzunluğuna sıralı numaralı bir aralığa indekslenir. Doktoru liste elemanlarını içeren: Adagrad, Nadam, Adam, Adamax, RMSprop ve SGD. Katman tipi listesinde iki öğe bulunur: Dropout ve Dense. Başlatma fonksiyon listesi öğelerini içerir: Zero, Normal, Lecun uniform, Uniform, Glorot uniform, Glorot normal, He uniform, ve He normal. Son olarak, Aktivasyon fonksiyonu listesi sekiz öğelerin olduğu: Linear, Softmax, Relu, Sigmoid, Tanh, Hard sigmoid, Softsign ve Softplus. Tüm kategorik hiperparametrelerin öğelerinin Keras uygulamasında tanımlandığı belirtilmeye değerdir (Martin vd., 2017).

Optimizasyon ve sonuç çıkarma aşamalarında belirli bir veri konfigürasyonu, her bir kullanıcı için bir kez gerçekleştirilir, yani her kullanıcı için tekrar edilecektir $U_i$ , $i$=1,2,...., $Q$, $Q$ sayıdır belirli bir veri konfigürasyonu $D$ kullanıcı. Optimizasyon aşamasında, kullanıcının $U_i$ verileri bölerek başlar iki bağımsız takımdan halinde $T_i$ ve $Z_i$ (eğitim ve test setleri), $i^{inci}$ sırasıyla kullanıcının. Bölüm 3.2'te açıklanan belirli veri yapılandırmasının yapısını takip etti. Eğitim ve test setlerinin tüm blokları metinden sayısal değerlere dönüştürülür ve sonra [0,1]'de normalleştirilir. Bundan sonra, biz optimize hiperparametreler vektörü $H$ bulmak için önerilen PSO tabanlı algoritmaya bu setleri verilir için $i$ kullanıcıya. Buna ek olarak, bir kopyasını kazandıracak $H_i$ zamandan tasarruf ve söz konusu veri yapılandırması $D$ ait RNN-deneyde onları yeniden kullanmak üzere, bir veri tabanında değerler sunulacak. Sonuç çıkarma aşaması, $H_i$ tarafından ayarlanan DNN'yi inşa ederken DNN'yi $T_i$'de eğitim ettiğinde ve $Z_i$*'de* DNN'yi test ettiğinde gerçekleşir. Sınıflandırma çıktıları değerleri: Gerçek pozitif ($TP_i$), yanlış pozitif ($FP_i$), gerçek negatif ($TN_i$) ve yanlış negatif ($FN_i$),$i^{inci}$ kullanıcı için belirli veri konfigürasyonu $D$ ekstre edilmiş ve kaydedilir daha sonraki işlemler için.

Daha sonra, bir sonraki kullanıcı gözlemlenir ve aynı optimizasyon prosedürü ve sonuç çıkarma aşamaları, özel veri konfigürasyonunda $D$ son kullanıcıya ulaşılana kadar gerçekleştirilir. Son olarak, belirli veri yapılandırmasındaki tüm

kullanıcılar tamamlandığında, son aşama (Sonlandırma Aşama) gerçekleştirilir. Sonlandırma aşama tamamlanması belirli bir veri konfigürasyonu *D* tüm kullanıcıların her edilen TP'leri toplamını hesaplar ile belirtmektedir *TP*. Aynı süreç *FP*, *TN* ve *FN* gibi diğer sonuçlara da uygulanacaktır. Denklemleri 3.3, 3.4, 3.5 ve 3.6: *TP*, *FP*, *TN* ve *FN* formüllerini ifade eder, sırasıyla.

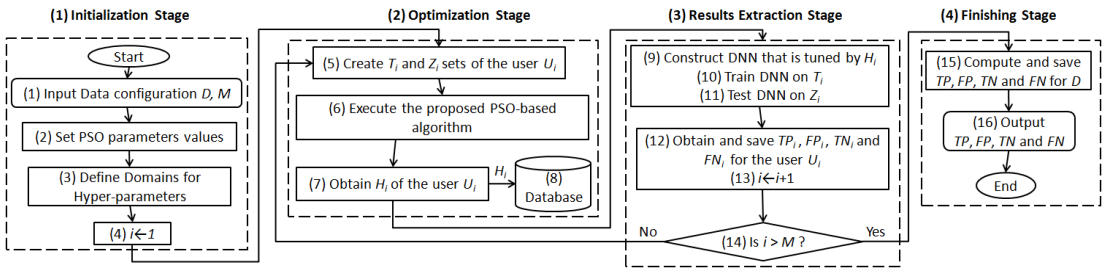
$$TP = \sum_{i=1}^{Q} TP_i \tag{3.3}$$

$$FP = \sum_{i=1}^{Q} FP_i \tag{3.4}$$

$$TN = \sum_{i=1}^{Q} TN_i \tag{3.5}$$

$$FN = \sum_{i=1}^{Q} FN_i \tag{3.6}$$

Son işlem aşaması bu sonuçları rapor edip kaydedecek ve belirli veri yapılandırması *D* için DNN deneyini sonlandıracaktır. Daha önceki sonuçlar, Bölüm 3.5'de belirtildiği gibi DNN'nin belirli bir veri yapılandırması *D* üzerindeki performansını değerlendirmek için iyi bilinen on iki değerlendirme ölçütünü hesaplamak için kullanılacaktır. Her bir veri yapılandırması için yukarıda açıklanan prosedürün uygulanacağını söylemeye değer. Şekil 3.4, DNN deneylerinin metodolojisinin akış şemasını göstermektedir.



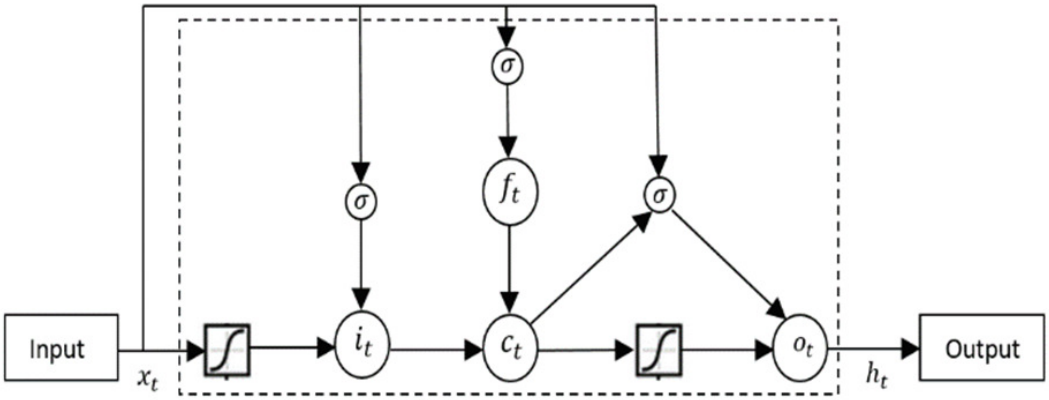

Şekil 3.4. DNN deneylerinin akış şeması

### 3.4.1.2. RNN

RNN, geleneksel ileri beslemeli Yapay Sinir Ağının (ANN) özel bir türüdür. Geleneksel ANN'den farklı olarak, RNN'de, gizli katmanların herhangi birindeki her bir nöron, aynı gizli katmanın diğer nöronlarının yanı sıra, çıktısından kendisine (kendi kendini tekrarlayan) ek bağlantılara sahiptir. Bu nedenle,

RNN'nin gizli katmanının herhangi bir *t* adımındaki çıktısı, mevcut girdiler ve gizli katmanın önceki *t-1* adımındaki çıktısı içindir. RNN'de, bu yönlendirilmiş çevrimler bilginin ağda dolaşımını sağlar ve gizli tabakaları bütün ağın depolama birimi olarak yapar (Yin vd., 2017). RNN'nin önemli özellikleri belleğe sahip olma ve periyodik diziler üretme kabiliyetidir.

Buna rağmen, yukarıda tarif edilen geleneksel RNN yapısı, özellikle RNN, geri yayılma tekniği kullanılarak eğitildiğinde ciddi bir problemi vardır. Problem, kaybolan ve patlayan olarak gradyan bilinir (Bengio vd., 1994). Gradyan kaybolma sorunu, gradyan sinyali ağ üzerinde çok küçük olduğunda, öğrenmeye yol açan ya da yavaşlayan durduğunda ortaya çıkar. Öte yandan, gradyan patlaması problemi, gradyan sinyali öğrenmenin farklılaştığı kadar büyüdüğünde ortaya çıkar. Geleneksel RNN'nin bu problemi RNN kullanımını sadece kısa süreli hafıza görevlerinde sınırladı. Bu sorunu çözmek için Hochreiter ve Schmidhuber (1997) tarafından yeni bir RNN mimarisi önerildi. Uzun Kısa Süreli Bellek (LSTM) olarak bilinenler. LSTM, oluşan ve hafıza hücresi denilen yeni bir yapı kullanır dört parça bunlar: bir giriş kapısı, kendinden tekrarlamalı bağlantıya sahip bir nöron, bir unutma kapısı ve çıkış kapısı. Bu arada, kendiliğinden tekrarlayan bağlantıya sahip bir nöron kullanmanın asıl amacı, bilgiyi kaydetmek, üç kapı kullanmanın amacı, bellek hücresinden veya içine hücre akışını kontrol etmektir. Giriş kapısı, gelen bilgilerin bellek hücresine girmesine izin verip vermemeye karar verip vermeyeceğine karar verir. Ayrıca, unutma kapısı, bellek hücresinin mevcut durumunu değiştirmek için bellek hücresinin önceki durumundan geçip geçmeyeceğini veya bunu engelleyip engellemediğini kontrol eder. Son olarak, çıkış kapısı, bellek hücresinin çıkışının geçip geçmeyeceğini belirler. Şekil 3.5, bir LSTM bellek hücresinin yapısını gösterir. LSTM modelinden ziyade, konvansiyonel RNN'nin sorunlarını aşar; Aynı zamanda, özellikle uzun süreli hafıza görevlerinde, geleneksel RNN'yi performans açısından daha iyi performans göstermektedir (Du vd., 2016). LSTM-RNN modeli, RNN'nin gizli katmanlarındaki her nöronu bir LSTM bellek hücresine değiştirerek elde edilebilir (Kim vd., 2016).

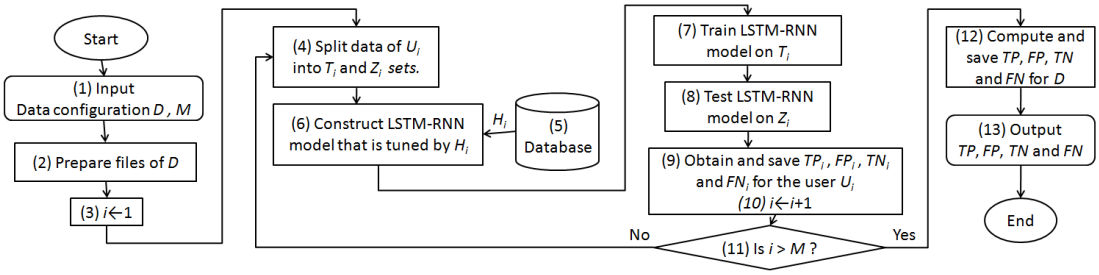



Şekil 3.5. Bir LSTM hücresinin yapısı (Kim vd., 2016)

Bu çalışmada LSTM-RNN modelini, tüm veri yapılandırmalarında statik bir maskeli saldırı tespiti görevi yapmak için kullandık. 3.4.1.1 alt bölümünde belirtildiği gibi, altı veri yapılandırması vardır ve bunların her biri ayrı deneyde kullanılacaktır. Böylece, altı ayrı LSTM-RNN deneyi yapacağız, her deney veri yapılandırmalarından birinde yapılacak. Tüm bu deneylerin metodolojisi aynıdır ve aşağıdaki gibidir: verilen veri konfigürasyonu *D* için, önce tüm veri konfigürasyon dosyalarını tüm blokları metinden sayısal değerlere dönüştürerek ve ardından [0,1] 'de normalleştirerek hazırladık. Bunun yanında, her bir $U_i$ kullanıcı için *D* içinde, *i*=1,2,...., *Q* ve *Q* kullanıcı sayısının ise *D,* aşağıdaki adımları yaptık: Biz verileri bölmek $U_i$ iki bağımsız kümeler halinde $T_i$ ve $Z_i$ (eğitim ve test setleri) vardır, $i^{inci}$ kullanıcı *D* sırasıyla. Bölünme işlemi, Bölüm 3.2'te açıklanan belirli veri yapılandırmasının yapısını takip etti.

Bundan sonra, depolanmış optimize alınan hiperparameterleri vektörünü $H_i$ , $i^{inci}$ kullanıcı için önceki DNN deneylerde oluşturulur veri tabanından. Sonra $H_i$ tarafından ayarlanan RNN modelini yaptık. LSTM-RNN modelini elde etmek için, gizli katmanların herhangi birindeki her nöron, bir LSTM bellek hücresine değiştirilir. İnşa LSTM-RNN modeli $T_i$ üzerinde eğitilmiştir $Z_i$ üzerinde test edilmiş. Test işlemi bittikten sonra, ekstre edilmiş ve sonuçlar kaydedildi :$TP_i$, $FP_i$, $TN_i$, and $FN_i$ arasında $i^{inci}$ *D* kullanıcının. Sonra, bir sonraki kullanıcıya geçin *D* son kullanıcı kadar aynı önceki adımları yapmak *D* ulaşılır. *D*'deki tüm kullanıcılardan sonra tamamlandıktan sonra, sırasıyla Denklemleri 3.3, 3.4, 3.5 ve 3.6'yı kullanarak veri yapılandırması *D'nin TP, FP, TN* ve *FN* toplam

sonuçlarını hesapladık. Şekil 3.6, LSTM-RNN deneylerinin metodolojisinin akış şemasını göstermektedir.




Şekil 3.6. LSTM-RNN deneylerinin akış şeması

### 3.4.2. Dinamik sınıflandırma yaklaşımı

Statik sınıflandırma yaklaşımının aksine, dinamik sınıflandırma yaklaşımının statik özelliklere sahip kullanıma hazır bir veri setine ihtiyacı yoktur. Doğrudan metin, resim, video, ses ve sinyal dosyaları gibi ham veri kaynakları ile uyumludur ve bunları dinamik olarak ayıklar. Bu yaklaşımı kullanan modeller, denetlenmemiş bir şekilde özellikleri öğrenmeye ve temsil etmeye çalışır. Daha sonra, bu modeller görünmeyen verileri sınıflandırabilmek için çıkarılan özellikleri kullanarak kendilerini eğitiyorlar.

Derin öğrenme modelleri bu yaklaşım için çok uygundur, çünkü derin öğrenme modellerinin temel amaçları otomatik özellik çıkarma ve kendi kendine öğrenme yeteneğidir. Bundan ziyade, dinamik sınıflandırma modelleri veri setleri gölünün probleminin üstesinden gelir; Statik sınıflandırma modellerinden daha verimli çalışır. Bu avantajlara rağmen, Dinamik sınıflandırma yaklaşımının da sakıncaları vardır. Dinamik sınıflandırma modelleri daha yavaştır ve statik sınıflandırma modelleriyle karşılaştırıldığında, bu modellerin karmaşık derin yapıları ve yapılması gereken çok sayıda hesaplama nedeniyle, eğitilmesi uzun zaman alır. Ayrıca, dinamik sınıflandırma modelleri, yüksek doğruluk değerleri elde etmek için çok büyük miktarda giriş örneği gerektirir.

Bu araştırmada, üç metin veri setinden uygulanan altı veri yapılandırması kullandık. Bu veri yapılandırmalarına dinamik maskeli saldırı tespit uygulamak için, kullanıcının komut metni dosyasından dinamik olarak özellikleri çıkarabilen ve daha sonra kullanıcıyı normal bir kullanıcı veya maskeli saldırı olarak iki sınıftan birine sınıflandırabilecek bir modele ihtiyacımız var. Bu nedenle, bir metin sınıflandırma görevi ile ilgileniyoruz. Metin sınıflandırması, içeriğine göre bir veya daha fazla sınıfa bir metin parçası (bir kelime, cümle veya hatta bir belge) atayan bir görev olarak tanımlanır. Aslında, cümle sınıflandırması, duyarlılık analizi ve belge sınıflandırması olmak üzere üç tür metin sınıflandırması vardır. Cümle sınıflandırmalarında, verilen bir cümle olası sınıflardan birine doğru bir şekilde atanmalıdır. Ayrıca, Duyarlılık analizi, verilen bir cümlenin belirli bir konuya karşı pozitif, negatif veya nötr olup olmadığını belirler. Buna karşılık, belge kategorizasyonu belgelerle ilgilenir ve verilen bir olası sınıf sınıfından hangi sınıfa ait olduğunu belirler. Dinamik sınıflamanın doğası ve metin sınıflandırmasının işlevselliğine göre, derin öğrenme modelleri, güçlü öğrenme yetenekleri nedeniyle, bu sınıflandırma türleri için diğer makine öğrenme modelleri arasında en uygun olanıdır. Dinamik sınıflamanın doğası ve metin sınıflandırmasının işlevselliğine göre, derin öğrenme modelleri, güçlü öğrenme yetenekleri nedeniyle, bu sınıflandırma türleri için diğer makine öğrenme modelleri arasında en uygun olanıdır. Dinamik sınıflamanın doğası ve metin sınıflandırmasının işlevselliğine göre, derin öğrenme modelleri, güçlü öğrenme yetenekleri nedeniyle, bu sınıflandırma türleri için diğer makine öğrenme modelleri arasında en uygun olanıdır.

Literatürde metin sınıflandırma alanında derin öğrenme modelleri kullanılarak çok çeşitli araştırmalar yapılmıştır. Bu tarafından başlatıldı LeCun vd. (1998) LeNet ailesi olarak bilinen özel bir CNN topolojisi önerdiğinde ve bunları metin sınıflandırmada verimli bir şekilde kullandı. Ardından, metin sınıflandırma algoritmalarını ve performansı etkileyen faktörleri tanıtmak için çeşitli çalışmalar yayınlanmıştır (Aggarwal ve Zhai, 2012; Zhang ve LeCun, 2015; Zhang ve Wallace, 2015). Kim (2014), CNN modeli, bir dizi metin veri kümesi kıyaslaması üzerinden cümle sınıflandırma görevi için kullanılmaktadır. Bir

bölge temelli metin gömme öğrenmek için tek bir boyutlu CNN önerilmiştir (Johnson ve Zhang, 2014). Zhang vd. (2015), metin sınıflandırma işleri için rekabetçi sonuçlara sahip yeni bir karakter tabanlı çok boyutlu CNN'yi tanıttı. Metin sınıflandırma için Hiyerarşik Derin Öğrenme (HDLTex) adı verilen yeni bir Hiyerarşik yaklaşım, Kowsari (2017) tarafından önerilmiş ve DNN, RNN ve CNN olan üç derin yapı kullanılmıştır. Tekrarlayan evrişimsel ağ modeli Metin sınıflandırma için tanıtılmış ve yüksek sonuçlar belgeler düzeyinde veri setlerinde elde edilmiştir (Lai vd., 2015). LSTM tabanlı yeni bir model tanıtıldı ve çok görevli öğrenme çerçevesine sahip metin sınıflandırması için kullanıldı (Liu vd., 2016). Yang vd. (2016), belge sınıflandırma için hiyerarşik dikkat ağı adı verilen yeni bir model önerdi ve iyi sonuçlarla altı büyük belge düzeyinde veri kümesi üzerinde test edildi. Derin CNN kullanarak metin sınıflandırma görevleri için karakter düzeyinde bir metin sunum yaklaşımı önerilmiş ve test edilmiştir (Prusa ve Khoshgoftaar, 2017). Görüldüğü gibi, CNN metin sınıflandırma görevleri için en çok kullanılan derin öğrenme modelidir. Bu yüzden, CNN'yi tüm veri yapılandırmalarında dinamik maskeli saldırı tespit için kullanmaya karar verdik. Aşağıdaki alt bölüm CNN'i gözden geçirirken, kullanılmış CNN modelinin yapısını ve CNN deneylerimizin metodolojisini açıklar.

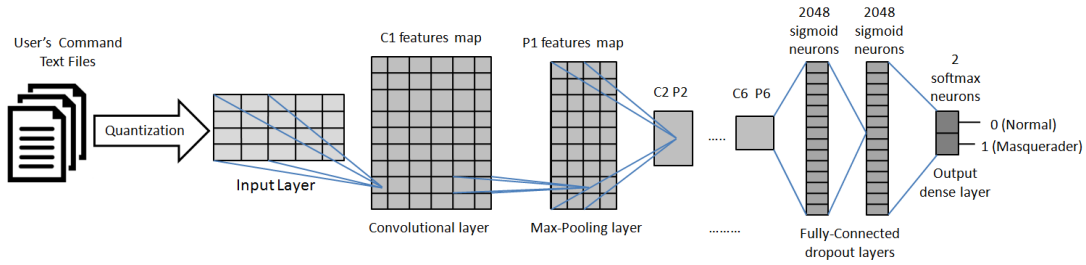
#### 3.4.2.1. CNN

CNN, hayvan görsel korteksinden biyolojik olarak ilham alan derin bir öğrenme modelidir. CNN, geleneksel ileri beslemeli ANN'nın özel bir türü olarak düşünülebilir. ANN ve CNN arasındaki en büyük fark, ANN'nin tamamen bağlı mimarisi yerine, CNN'deki bireysel nöronların, giriş alanının alt bölgelerine bağlı olmasıdır. CNN'nin nöronları, tüm giriş alanını kaplayacak şekilde düzenlenirler. Tipik CNN, bir giriş katmanı, evrişimli katman, havuzlama katmanı, tam bağlı katman ve bir çıkış katmanı olmak üzere beş ana bileşenden oluşur. Giriş katmanı, giriş verilerinin CNN'ye girildiği yerdir. İlk evrimsel CNN'deki tabaka, her birinin girdi alanının küçük bir alt kümesine bağlı olduğu ayrı ayrı nöronlardan oluşur. Bir sonraki evrişimsel katmanlardaki nöronlar yalnızca önceki havuzlama katmanı çıktısının bir alt kümesine bağlanır. Dahası,

CNN'deki evrişimli katmanlar, her bir filtrenin önceki katlarının çıktısının belirtilen alt kümesine uygulandığı bir dizi öğrenilebilir çekirdek veya filtre kullanır. Bu filtreler, her özellik haritasının aynı ağırlıkları paylaştığı özellik haritalarını hesaplar. Alt örnekleme katmanı olarak da bilinen havuz katmanı, girişinin alt kümelerini yoğunlaştıran doğrusal olmayan bir örnekleme işlevidir. CNN'de havuz katmanlarını kullanmanın temel amacı karmaşıklığı ve hesaplamaları azaltmaktır önceki katmanın çıktısının boyutunu küçülterek. Çok sayıda havuzlama doğrusal olmayan fonksiyonlar kullanılabilir, ancak bunların arasında, verilen havuzlama penceresindeki maksimum değeri seçen en çok kullanılan havuzlamadır. Tipik olarak, CNN'deki her evrişimsel katmanı bir max-biriktirme katmanı izler. CNN bir veya daha fazla yığılmış evrişimlidir, tüm girdiden özellikleri çıkarmak ve ardından bu özellikleri bir sonraki tam bağlı katmanlarına eşlemek için katman ve maks. CNN'nin üst katmanları, DNN'deki gizli katmanlara benzeyen tam bağlı katmanlardan biri veya daha fazlasıdır. Bu, tamamen bağlı katmanların nöronlarının önceki katmanın tüm nöronlarına bağlı olduğu anlamına gelir. Çıkış katmanı, CNN'deki son katmandır ve CNN'nin çıkış değerini bildirmekten sorumludur. Son olarak, geri yayılma algoritması, CNN'leri tam olarak bağlanmış katmanların ağırlıklarını ayarlamak için Stochastic Gradient Decent (SGD) aracılığıyla çalıştırmak için kullanılır (Albelwi ve Mahmood, 2017). CNN'nin çeşitli değişken yapıları literatürde önerilmektedir, ancak LeNet Yapı, birçok bilgisayar vizyonu ve metin sınıflandırma uygulamasında kullanılan en yaygın yaklaşımdır.

Metin sınıflandırmadaki kararlılığı ve yüksek etkinliği ile ilgili olarak, tüm veri yapılandırmalarında dinamik bir maskeli saldırı tespiti için (Zhang vd., 2015) tarafından önerilen CNN modelini seçtik. Kullanılan model, bir metin dosyasını girdi olarak alan ve sınıflandırma puanını veren bir karakter seviyesi CNN'dir (giriş metni dosyası normal bir kullanıcıyla ilgiliyse 0, aksi takdirde 1). Kullanılan CNN modeli LeNet ailesine ait ve bir giriş katmanı, bunu takiben altı evrişim ve maksimum havuz çifti, ardından iki tam bağlı katman ve ardından bir çıkış katmanı izliyor. Giriş katmanında, kullanılan model 70 metrelik bir alfabenin tek bir sıcak gösterimini kullanarak giriş metni dosyasındaki tüm harfleri kodladığında metin niceleme işlemi gerçekleşir. Tüm evrişimsel

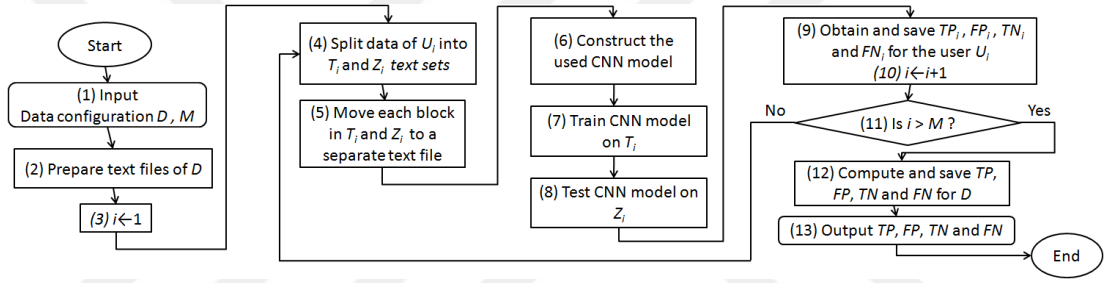
Kullanılan CNN modelindeki katmanlar, bir ReLU doğrusal olmayan aktivasyon fonksiyonuna sahiptir. Kullanılan CNN modelinde iki tam bağlı katman, düşme olasılığı 0,5'e eşit olan tip katsayılı katmandır. Buna ek olarak, kullanılan CNN modelindeki iki tam bağlı katman, Sigmoid doğrusal olmayan bir aktivasyon fonksiyonuna sahiptir ve bunların her biri aynı boyutta 2048 nörona sahiptir. Kullanılan CNN modelindeki çıkış katmanı, softmax'a aktivasyon fonksiyonu ve iki nöronun büyüklüğü sahip olmasının yanı sıra yoğun katmandandır. Kullanılan CNN modeli, SGD üzerinden geri yayılım algoritması ile eğitilmiştir. Son olarak, kullanılan CNN modeline şu parametreleri koyarız: öğrenme oranı=0.01, epochs=30 ve parti büyüklüğü=64. Bu değerler deneysel olarak bu parametrelerin mümkün olan en iyi değerlerini bulmak için bir ızgara araması yapılarak elde edilir. Şekil 3.7, kullanılan CNN modelinin mimarisini göstermektedir (Zhang vd., 2015).




Şekil 3.7. Kullanılan CNN modelinin mimarisi

Çalışmamızda, tüm veri yapılandırmalarında dinamik bir maskeli saldırı tespiti görevi gerçekleştirmek için bir CNN modeli kullandık. 3.4.1.1 alt bölümünde belirtildiği gibi, altı veri yapılandırması vardır ve bunların her biri ayrı deneyde kullanılacaktır. Bu yüzden, altı ayrı CNN deneyimiz olacak, her deney veri yapılandırmalarından birinde yapılacak. Tüm bu deneylerin metodolojisi aynıdır ve şu şekildedir: verilen veri konfigürasyonu $D$ için, öncelikle, her veri dosyasının $D$'deki bir kullanıcının eğitim ve test setlerini temsil ettiği şekilde tüm veri konfigürasyonunun metin dosyalarını hazırladık. Bunun yanında, her bir kullanıcı $U_i$ için, $D$ içinde, $i$=1,2,...., $Q$ ve $Q$ kullanıcı sayısının ise $D$, aşağıdaki adımları yaptık: Biz verileri bölmek $U_i$ için, iki bağımsız kümeler halinde $T_i$ ve $Z_i$ eğitim ve test setleri vardır, $i^{inci}$ kullanıcı $D$ sırasıyla. Bölünme işlemi, Bölüm 3.2'te açıklanan belirli veri yapılandırmasının yapısını takip etti. Ayrıca, biz de

kullanıcı eğitimi ve test setleri her blok taşındı $U_i$ ayrı bir metin dosyasına. Bu demektirki, kullanıcı eğitimi ve test setleri her $U_i$. Her bir metin dosyasının bir UNIX komut bloğu içerdiği belirli sayıda metin dosyasından oluşur. Ondan sonra kullanılmış CNN modelini yaptık. İnşa CNN modeli $T_i$ üzerinde eğitilmiştir $Z_i$ üzerinde test edilimitir. Test işlemi bittikten sonra, ekstre edilmiş ve sonuçlar kaydedildi : $TP_i$, $FP_i$, $TN_i$, ve $FN_i$ arasında $i^{inci}$ $D$ kullanıcının. Sonra, bir sonraki kullanıcıya geçin $D$ son kullanıcı kadar aynı önceki adımları yapmak $D$ ulaşılır. $D$'deki tüm kullanıcılardan sonra tamamlandıktan sonra, sırasıyla Denklemleri 3.3, 3.4, 3.5 ve 3.6'yı kullanarak veri yapılandırması *D'nin TP, FP, TN* ve *FN* toplam sonuçlarını hesapladık . Şekil 3.8, CNN deneylerinin metodolojisinin akış şemasını göstermektedir.




Şekil 3.8. CNN deneylerinin akış şeması

### 3.5. Sonuçlar ve Tartışma

DNN deneyleri, LSTM-RNN deneyleri ve CNN deneyleri olan üç ana deneysel deney yaptık. Bunların her biri, her bir alt deneyin veri konfigürasyonlarından birinde yapıldığı altı ayrı alt deneyden oluşur: SEA, SEA 1v49, Greenberg Kesilmiş, Greenberg Zenginleştirilmiş, PU Kesilmiş ve PU Zenginleştirilmiş. Temel olarak, hiperparametre seçimi için PSO tabanlı algoritmamız Python 3.6.4 (Python, 2018) ve NumPy (NumPy, 2018) ile yapıldı. Üstelik, tüm modelleri (DNN, LSTM-RNN ve CNN) eğitim ve esas alınarak test edilen ve inşa edildi ile Keras (Chollet, 2015; Keras, 2018) ve TensorFlow 1.6 (Abadi vd., 2016; TensorFlow, 2018) bu CUDA 9.0 (CUDA, 2018) ve cuDNN 7.0 (cuDDN, 2018) üzerinden geri döndü. Buna ek olarak, tüm deneyler Intel Core i7 CPU (3.8 GHz, 16 MB Önbellek), 16 GB RAM ve Windows 10 işletim sistemine sahip bir iş

istasyonunda yapıldı. Tüm deneylerde hesaplamaları hızlandırmak için, NVIDIA Tesla K20 GPU 5 GB GDDR5 ile GPU hızlandırmalı bir hesaplama yaptık. Deneysel ortam 64 bit modunda işlenir.

Herhangi bir sınıflandırma görevinde dört olası sonucu vardır: Gerçek Pozitif (TP), Gerçek Negatif (TN), Yanlış Pozitif (FP) ve Yanlış Negatif (FN). Bir maskeleyici olarak doğru bir şekilde sınıflandırıldığında bir TP alırız. İyi bir kullanıcı doğru bir kullanıcı olarak doğru bir şekilde sınıflandırıldığında, bunun bir TN olduğunu söyleriz. İyi bir kullanıcı maskeli saldırı olarak yanlış sınıflandırıldığında bir FP oluşur. Buna karşılık, FN bir maskeleyici iyi bir kullanıcı olarak yanlış sınıflandırıldığında ortaya çıkar. Çizelge 3.5, maskeli saldırı saptama sonuçlarının karışıklık matrisini göstermektedir. Her veri yapılandırması için, on iki tanınmış değerlendirme ölçütünü hesaplamak için bu veri yapılandırması için elde edilen sonuçları kullandık. Bundan sonra, bu değerlendirme ölçütlerini kullanarak, her derin öğrenme modelinin bu veri yapılandırmasındaki performansını değerlendirdik.

Çizelge 3.5. Maskeli saldırı tespiti sonuçlarının karışıklık matrisi

| **Gerçek sınıf** | **Öngörülen** | |
|---|---|---|
| | Normal kullanıcı | Maskeleyici |
| Normal kullanıcı | TN | FP |
| Maskeleyici | FN | TP |

Basit olması için bu değerlendirme metriklerini iki kategoriye ayırdık: Genel Sınıflandırma Ölçütleri ve Maskeli Saldırı Saptama Ölçütleri. Genel Sınıflandırma Ölçütleri, Doğruluk, Kesinlik, Geri Çağırma ve F1-Puanı gibi herhangi bir sınıflandırma görevi için kullanılan ölçümlerdir. Öte yandan, Maskeli Saldırı Saptama Ölçütleri, genellikle bir maskeli saldırı veya saldırı saptama görevi için kullanılan metriklerdir; bunlar: Vuruş Oranı, Kayıp Oranı, Yanlış Alarm Oranı, Maliyet, Bayes Saptama Oranı, Bayes Gerçek Olumsuz Oranı, Geometrik Ortalama ve Matthews Korelasyon Katsayısı. Kullanılan değerlendirme metrikleri tanımı ve karşılık gelen Denklemleri aşağıdaki gibidir:

- Doğruluk, tüm test setlerinde doğru tespit oranını gösterir.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.7)$$

- Kesinlik, test setindeki maskeli saldırıları olarak sınıflandırılan tüm bloklardan doğru sınıflandırılmış maskeleyici oranını gösterir.

$$Precision = \frac{TP}{TP+FP} \quad (3.8)$$

- Geri çağırma, test setindeki tüm maskeli saldırı blokları üzerinde doğru sınıflandırılmış maskeleyici oranını gösterir.

$$Recall = \frac{TP}{TP+FN} \quad (3.9)$$

- F1-Puanı, hem Precision (P) hem de Recall (R) ölçümleriyle ilgili sınıflandırıcıların doğruluğu hakkında bilgi verir.

$$F1\ Score = \frac{2}{\frac{1}{P}+\frac{1}{R}} \quad (3.10)$$

- Vuruş oranı, test setinde sunulan tüm maskeli saldırı bloklara göre doğru sınıflandırılmış maskeli saldırı blokları oranını gösterir. Aynı zamanda Hits, True Positive Rate (TPR) veya Detection Rate (DR) olarak da adlandırılır.

$$Hit\ Rate = \frac{TP}{TP+FN} \quad (3.11)$$

- Kayıp oranı (MR), Hit Rate'in (Miss=100-Hit) tamamlayıcısıdır, yani test setindeki tüm maskeli saldırı bloklardan normal bir kullanıcı olarak yanlış sınıflandırılan maskeli saldırı blokların oranını gösterir. Aynı zamanda Yanlış Negatif Oranı (FNR) veya Misses olarak da adlandırılır.

$$Miss\ Rate = \frac{FN}{FN+TP} \quad (3.12)$$

- Yanlış alarm oranı (FAR), test setinde sunulan tüm normal kullanıcı blokları üzerinden maskeli saldırı olarak sınıflandırılan normal kullanıcı bloklarının oranı hakkında bilgi verir. Aynı zamanda Yanlış Pozitif Oranı (FPR) olarak da adlandırılır.

$$FAR = \frac{FP}{FP+TN} \quad (3.13)$$

- Maliyet. hem MR hem de FAR ölçümleriyle ilgili sınıflandırıcıların etkinliğini değerlendirmek için Maxion ve Townsend (2002) tarafından önerilen bir ölçümdür.

$$Cost = MR + 6 \times FAR \quad (3.14)$$

- Bayesian sapatma oranı (BDR), Axelsson (1999) tarafından ele alınan Baz Oran Yanılgısı sorununu temel alan bir ölçümdür. Baz Oran Yanılgısı, Bayesian istatistiklerinin temellerinden biridir ve insanlar olasılıklardaki problemleri çözerken temel insidans oranını (Baz Oranı) dikkate

almadıklarında ortaya çıkar. Hit ölçümünden farklı olarak BDR, maskeleyicilerin taban oranını göz önünde bulundurarak tüm test setlerine göre doğru sınıflandırılmış maskeleyici blokları oranını gösterir. Let $I$ ve $I^*$ sırasıyla bir maskeli saldırı ve normal bir davranış göstermek. Dahası, $A$ ve $A^*$ sırasıyla belirtilen maskeli saldırı ve normal davranışı göstermesine izin verin . O zaman, BDR $P(I|A)$ olasılık olarak hesaplanabilir denklem 3.15'e göre (Axelsson, 1999).

$$BDR = P(I|A) = \frac{P(I)\times P(A|I)}{P(I)\times P(A|I)+P(I^*)\times P(A|I^*)} \quad (3.15)$$

$P(I)$ test kümesi içinde maskeleyici blokların oranıdır, $P(A|I)$ vuruş oranı ise, $P(I^*)$ test kümesi normal blokların oranıdır ve $P(A|I^*)$ FAR'dır.

- Bayesian gerçek negatif oranı (BTNR), Baz Oran Hatalılığına da dayanıyor ve öngörülen normal davranışın gerçekten normal bir kullanıcıyı gösterdiği tüm test kümeleri üzerinde gerçekten sınıflandırılmış normal blokların oranını gösteriyor (Axelsson, 1999). Let $I$ ve $I^*$ sırasıyla bir maskeli saldırı ve normal bir davranış göstermek. Dahası, $A$ ve $A^*$ sırasıyla belirtilen maskeli saldırı ve normal davranışı göstermesine izin verin. Daha sonra, BTNR, Denklem 3.16'ya göre $P(I^*|A^*)$ olasılığı olarak hesaplanabilir (Axelsson, 1999).

$$BTNR = P(I^*|A^*) = \frac{P(I^*)\times P(A^*|I^*)}{P(I^*)\times P(A^*|I^*)+P(I)\times P(A^*|I)} \quad (3.16)$$

$P(I^*)$ Test setindeki normal blokların oranı, $P(A^*|I^*)$ Gerçek Negatif Orandır (TNR) hesaplanarak kolayca elde edilen (1-FAR), $P(I)$ Test setindeki maskeli saldırı bloklar ve $P(A^*|I)$ Kayıp oranı.

- Geometrik Ortalama (g-mean), her iki hatanın da eşit olduğu düşünülen gerçek negatif oranı ve gerçek pozitif oranı belirli bir eşikte birleştiren bir performans ölçümüdür. Bu metrik dengesizlik veri setindeki sınıflandırıcıları değerlendirmek için birçok araştırmacı tarafından kullanılmıştır (Zeng and Gao, 2009). Denklem 3.17'ye göre hesaplanabilir (Kubat ve Matwin, 1997).

$$g\_mean = \sqrt{\frac{TP\times TN}{(TP+FN)\times(TN+FP)}} \quad (3.17)$$

- Matthews Korelasyon Katsayısı (MCC), doğru ve yanlış pozitifleri ve negatifleri hesaba katan bir performans ölçütüdür ve genellikle sınıflar çok farklı boyutlarda olsalar bile (dengesizlik veri kümesi) kullanılabilecek dengeli bir ölçü olarak kabul edilir (Boughorbel et al., 2017). MCC -1 ila 1

aralığına sahiptir; burada -1, tamamen yanlış bir ikili sınıflandırıcıyı gösterirken, 1, tamamen doğru bir ikili sınıflandırıcıyı belirtir. Yukarıda tartışılan diğer metriklerin aksine, MCC, denklem 3.18'e göre hesaplanabilen denklemdeki konfüzyon matrisinin tüm hücrelerini dikkate alır (Matthews, 1975).

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FN) \times (TP+FP) \times (TN+FP) \times (TN+FN)}} \quad (3.18)$$

Aşağıdaki iki alt bölümde, deneysel sonuçlarımızı sunacağız ve bunları iki tür analiz kullanarak açıklayacağız: Performans Analizi ve ROC Eğrileri Analizi.

### 3.5.1. Performans analizi

Herhangi bir modelin maskeli saldırı tespit etmedeki etkinliği, değerlendirme ölçütlerinin değerlerine bağlıdır. Doğruluk, Kesinlik, Geri Çağırma, F1-Puanı, Vuruş Oranı, BDR, BTNR, Geometrik Ortalama ve MCC değerlerinin yanı sıra Kayıp Oranı, FAR ve Maliyet değerlerinin düşük değerleri de verimli bir sınıflandırıcı olduğunu gösterir. İdeal sınıflandırıcı, Doğruluk ve Vuruş Oranı değerlerinin 1'e ve Kayıp Oranı ve FAR değerlerinin 0'a ulaşmasına sahiptir. Çizelge 3.6, DNN deneyleri, LSTM-RNN deneyleri ve CNN deneyleri için kullanılan değerlendirme ölçütlerinin yüzdelerini göstermektedir. Aslında, Çizelge 3.6'da DNN ve LSTM-RNN tarafından etiketlenen satırlar, sırasıyla DNN ve LSTM-RNN modelleri kullanılarak statik maskeli saldırı tespitinin sonuçlarını göstermektedir. Oysa, Çizelge 3.6'da CNN ile etiketlenen satırlar, CNN modelini kullanarak dinamik maskeli saldırı tespitinin sonuçlarını göstermektedir. Ayrıca, kalın satırlar aynı veri yapılandırması arasında en iyi sonuçları temsil eder. Oysa, açık gri ile vurgulanan değerler, tüm veri yapılandırmaları için en iyisidir.

Öncelikle, PSO tabanlı algoritmamızın hem DNN hem de LSTM-RNN modellerinin elde edilen sonuçlarında kullanılmasının etkisinin olduğu görülmektedir. PSO tabanlı algoritma, hiperparametre seçimini optimize etmek için kullanılır. Bu, doğruluğu en üst düzeye çıkardı; bu, TP ve TN sonuçlarının toplamının önemli ölçüde artırılacağı anlamına geliyor. Bu nedenle, Denklem

3.11 ve 3.13'e göre, TP ve TN'nin toplamının arttırılması kesinlikle vuruş oranının değerini artırmanın yanı sıra FAR'ın değerini de düşürecektir.

Çizelge 3.6. Deneylerimizin sonuçları

| Veri seti | Veri yapılandırması | Model | Değerlendirme ölçütleri (%) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Accuracy | Precision | Recall | F1-Score | Hit | Miss | FAR | Cost | BDR | BTNR | g-mean | MCC |
| SEA | SEA | DNN | 98.08 | 76.26 | 84.85 | 80.33 | 84.85 | 15.15 | 1.28 | 22.83 | 76.25 | 99.26 | 91.52 | 79.45 |
| | | LSTM-RNN | 98.52 | 82.30 | 86.58 | 84.39 | 86.58 | 13.42 | 0.90 | 18.83 | 82.33 | 99.34 | 92.63 | 83.64 |
| | | CNN | **98.84** | **87.77** | **87.01** | **87.39** | **87.01** | **12.99** | **0.59** | **16.51** | **87.72** | **99.37** | **93** | **86.78** |
| | SEA 1v49 | DNN | 96.54 | 99.98 | 96.43 | 98.17 | 96.43 | 3.57 | 0.48 | 6.47 | 99.98 | 52.04 | 97.96 | 70.64 |
| | | LSTM-RNN | 97.86 | 99.98 | 97.79 | 98.87 | 97.79 | 2.21 | 0.38 | 4.48 | 99.98 | 63.70 | 98.7 | 78.74 |
| | | CNN | **98.78** | **99.99** | **98.74** | **99.36** | **98.74** | **1.26** | **0.19** | **2.40** | **99.99** | **75.51** | **99.27** | **86.22** |
| Greenberg | Greenberg Kesilmiş | DNN | 93.97 | 92.23 | 80.67 | 86.06 | 80.67 | 19.33 | 2.04 | 31.57 | 92.22 | 94.41 | 88.89 | 82.53 |
| | | LSTM-RNN | 94.72 | 94.88 | 81.53 | 87.70 | 81.53 | 18.47 | 1.32 | 26.39 | 94.87 | 94.68 | 89.7 | 84.76 |
| | | CNN | **95.43** | **96.16** | **83.53** | **89.40** | **83.53** | **16.47** | **1.0** | **22.47** | **96.16** | **95.24** | **90.94** | **86.86** |
| | Greenberg Zenginleştirilmiş | DNN | 97.57 | 96.92 | 92.40 | 94.61 | 92.40 | 7.60 | 0.88 | 12.88 | 96.92 | 97.75 | 95.7 | 93.08 |
| | | LSTM-RNN | 97.98 | 97.57 | 93.60 | 95.54 | 93.60 | 6.40 | 0.70 | 10.60 | 97.56 | 98.10 | 96.41 | 94.28 |
| | | CNN | **98.60** | **98.55** | **95.33** | **96.92** | **95.33** | **4.67** | **0.42** | **7.19** | **98.55** | **98.61** | **97.43** | **96.03** |
| PU | PU Kesilmiş | DNN | 81.0 | 99.59 | 78.61 | 87.86 | 78.61 | 21.39 | 2.25 | 34.89 | 99.59 | 39.49 | 87.66 | 54.63 |
| | | LSTM-RNN | 82.19 | 99.69 | 79.89 | 88.70 | 79.89 | 20.11 | 1.75 | 30.61 | 99.68 | 41.10 | 88.6 | 56.46 |
| | | CNN | **83.75** | **99.74** | **81.64** | **89.79** | **81.64** | **18.36** | **1.50** | **27.36** | **99.73** | **43.38** | **89.68** | **58.79** |
| | PU Zenginleştirilmiş | DNN | 90.44 | 99.84 | 89.21 | 94.23 | 89.21 | 10.79 | 1.0 | 16.79 | 99.84 | 56.72 | 93.98 | 70.64 |
| | | LSTM-RNN | 91.31 | 99.88 | 90.18 | 94.78 | 90.18 | 9.82 | 0.75 | 14.32 | 99.88 | 59.08 | 94.61 | 72.61 |
| | | CNN | **93.75** | **99.92** | **92.93** | **96.30** | **92.93** | **7.07** | **0.50** | **10.07** | **99.92** | **66.78** | **96.16** | **78.52** |

Her model için SEA 1v49 veri konfigürasyonunun doğruluk değerleri, SEA veri konfigürasyonunun ilgili değerlerinden biraz daha düşük olmakla birlikte, aynı zamanda, tüm modellerde SEA 1v49'da hit değerler çarpıcı olarak, (10%-14%), SEA veri yapılandırması. Bunun nedeni, SEA 1v49 test setinde, SEA veri konfigürasyonundaki sadece 231 bloğa kıyasla, 122500 maskeliyici bloğunun

bulunduğu SEA 1v49 veri konfigürasyonunun yapısıdır. Dahası, Tüm modeller için SEA 1v49'un FAR değerleri, SEA veri yapılandırmasının karşılık gelen değerlerinden önemli ölçüde düşüktür. Bu nedenle, SEA veri setiyle ilgili olarak, SEA 1v49, maskeli saldırı algılamada SEA veri yapılandırmasından daha iyidir.

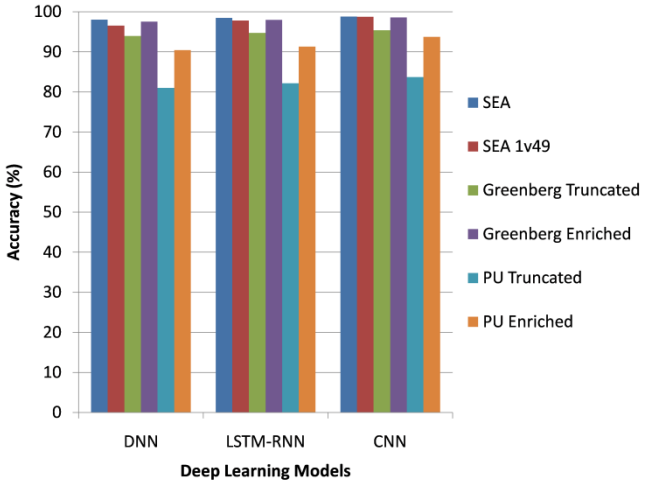
Diğer taraftan, Beklediğimiz gibi, Greenberg Zenginleştirilmiş, Greenberg Kesilmiş veri yapılandırmasının karşılık gelen değerlerinden kullanılan tüm değerlendirme ölçümleri açısından tüm modellerin performansını gözle görülür biçimde arttırdı. Bu, Greenberg Zenginleştirilmiş veri yapılandırmasının, Greenberg Kesilmiş'de yalnızca komut adıyla karşılaştırıldığında komut adı , parametreler, takma adlar ve bayraklar dahil olmak üzere kullanıcı davranışı hakkında daha fazla bilgiye sahip olması ile açıklanabilir. Bu nedenle, Greenberg veri seti ile ilgili olarak, Greenberg Zenginleştirilmiş veri yapılandırması daha iyidir. Greenberg Kesilmiş'den daha maskeli saldırı saptamasında kullanın. Aynı şey, PU Zenginleştirilmiş veri yapılandırmasının, tüm modellerde PU Kesilmiş 'den daha iyi sonuçlara sahip olduğu PU veri setinde de oldu. Bu nedenle, PU veri seti ile ilgili olarak, PU Zenginleştirilmiş, maskeli saldırı tespitinde PU Kesilmiş veri konfigürasyonundan daha iyidir.

Aslında, PU Kesilmiş ve Greenberg Kesilmiş veri yapılandırmaları, yalnızca komut adının dikkate alındığı SEA ve SEA 1v49 veri yapılandırmalarını simüle eder. Buna rağmen, kullanılmış tüm modellerde, SEA 1v49, diğer kesilmiş veri yapılandırmaları arasında en iyi sonuçları kaydetti. Diğer taraftan, PU Zenginleştirilmiş ve Greenberg Zenginleştirilmiş, kullanıcılar hakkında ek bilgilerin dikkate alındığı zenginleştirilmiş veri yapılandırmaları olarak kabul edilir. Bu nedenle, zenginleştirilmiş veri yapılandırmaları, modellerin, kullanıcının davranış profilini kesilmiş veri yapılandırmalarından daha doğru şekilde oluşturmasına yardımcı olur. Tüm modellerle ilgili olarak, Greenberg Zenginleştirilmiş ile özellikle doğruluk, vuruş oranı ve FAR değerleri açısından ilişkili sonuçlar, PU Zenginleştirilmiş veri yapılandırmasının karşılık gelen değerlerinden daha iyidir. Çünkü PU veri seti nispeten az sayıda kullanıcıyla (sadece 8 kullanıcı) çok küçük bir maskeli saldırı algılama veri setidir. Ayrıca, bu neden birkaç önceki çalışmaların maskeli saldırı saptamasında PU veri setini

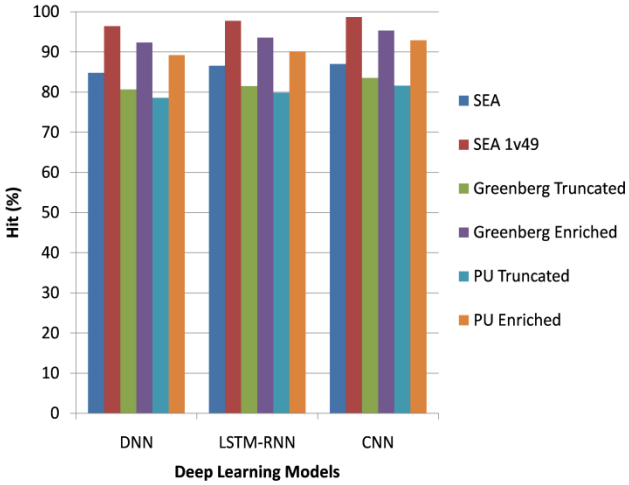
neden kullandığını açıklayabilir. Bununla birlikte, veri konfigürasyonları, elde edilen sonuçlara göre, tüm kullanılmış modeller için yukarıdan aşağıya doğru sıralanabilir: SEA1 v49, Greenberg Zenginleştirilmiş, PU Zenginleştirilmiş, SEA, Greenberg Kesilmiş ve PU Kesilmiş.

Kısalıklık ve alan sınırlaması adına, Şekil 3.9 ve 3.10'da görsel olarak gösterilmek üzere Çizelge 3.6'deki kullanılmış performans ölçütlerinin bir alt kümesini seçtik. Şekil 3.9 (a), 3.9 (b), 3.9 (c), 3.9 (ç), 3.9 (d), 3.9 (e), 3.9 (f) ve 3.9 (g) kullanılan modellerin Doğruluk, Vuruş oranı, Kayıp oranı, FAR, Maliyet, BDR, F1-Puanı ve MCC yüzdeleri sırasıyla her bir veri yapılandırmasında. Şekil 3.10 (a), 3.10 (b), 3.10 (c), 3.10 (ç), 3.10 (d) ve 3.10 (e), ortalama performans için Doğruluk, Vuruş oranı, FAR, BDR, F1-Puanı ve MCC yüzdelerini veri setlerinde kullanılan modellerin sırasıyla gösterir. Şekil 3.9 ve 3.10, her veri yapılandırması ve veri seti için olduğu kadar tüm veri setlerinde kullanılan derin öğrenme modellerinin performansının görsel bir karşılaştırmasını verebilir.

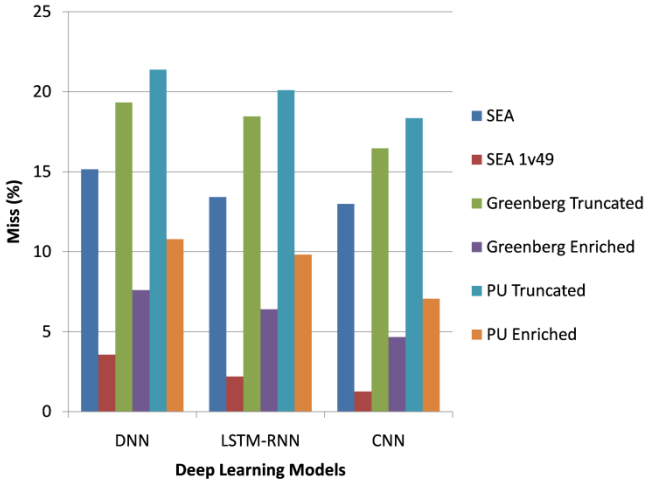
Şekil 3.9 ve 3.10'a bakıldığında, derin öğrenme modellerinin kararlılığını, bir veri konfigürasyonundan diğerine tutarlı bir şekilde maskeli saldırı algılamayı tespit edecek şekilde fark edebiliyoruz. Bunu açıklamak için, elde edilen sonuçları statik ve dinamik maskeli saldırı algılama teknikleri açısından tartışacağız. Statik sayısal özelliklere sahip veri konfigürasyonlarında statik maskeli saldırı algılama tespit görevi yapmak için DNN ve LSTM-RNN modellerini kullandık. DNN ve LSTM-RNN, hiperparametrelerini optimize eden PSO tabanlı bir algoritma ile desteklenir. Bir kullanıcının verilen eğitim ve test setlerinde doğruluğu en üst düzeye çıkarmak için. Eski gerçeğe önem veren DNN ve LSTM-RNN modellerimiz, belirli veri konfigürasyonunda her kullanıcı için ulaşabilecekleri kadar iyi maskeli saldırı algılama çıktıları vermektedir. Buna göre, sonuçta, performansları bu belirli veri konfigürasyonunda önemli ölçüde geliştirilecektir. Ayrıca, performanslarının bu şekilde artması, birinden diğerine farklılık gösteren veri konfigürasyon yapısından etkilenecektir.


Accuracy (%)
100
90
80
70
60
50
40
30
20
10
0
DNN
LSTM-RNN
CNN
Deep Learning Models
SEA
SEA 1v49
Greenberg Truncated
Greenberg Enriched
PU Truncated
PU Enriched

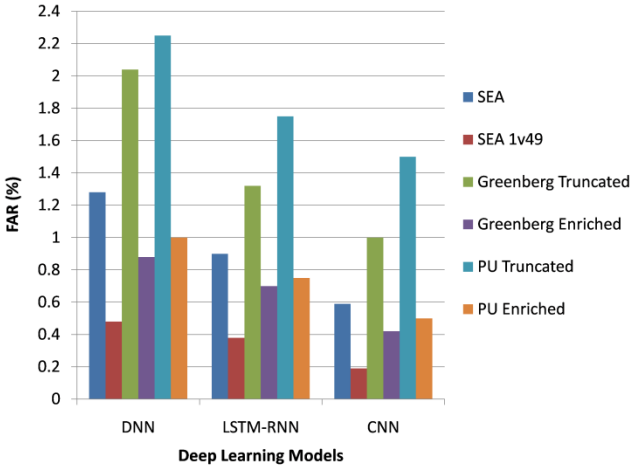
(a)


Hit (%)
100
90
80
70
60
50
40
30
20
10
0
DNN
LSTM-RNN
CNN
Deep Learning Models
SEA
SEA 1v49
Greenberg Truncated
Greenberg Enriched
PU Truncated
PU Enriched

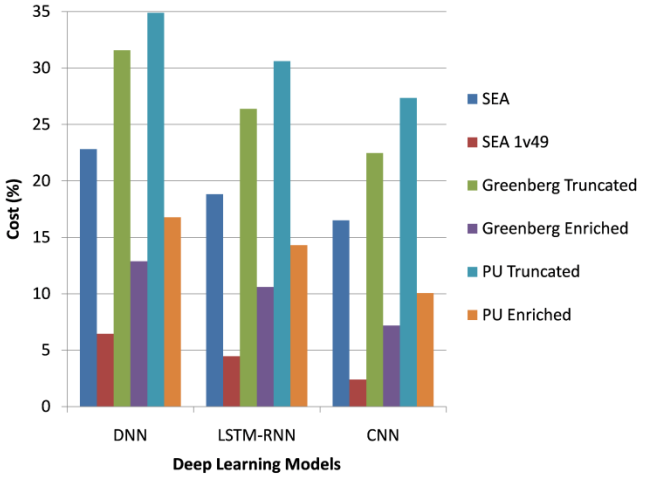
(b)


Miss (%)
25
20
15
10
5
0
DNN
LSTM-RNN
CNN
Deep Learning Models
SEA
SEA 1v49
Greenberg Truncated
Greenberg Enriched
PU Truncated
PU Enriched

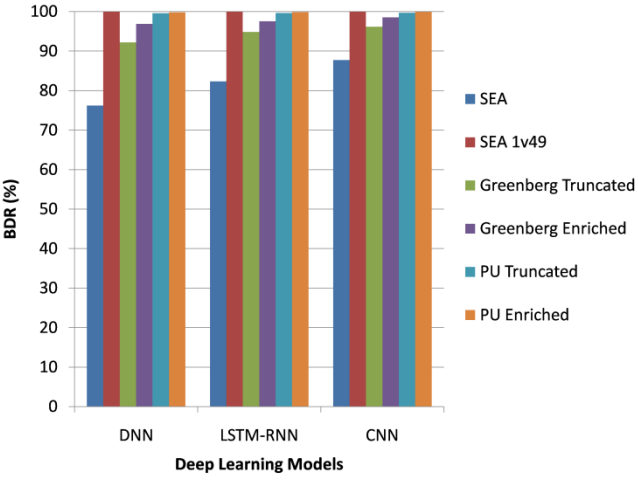
(c)


FAR (%)
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
DNN
LSTM-RNN
CNN
Deep Learning Models
SEA
SEA 1v49
Greenberg Truncated
Greenberg Enriched
PU Truncated
PU Enriched

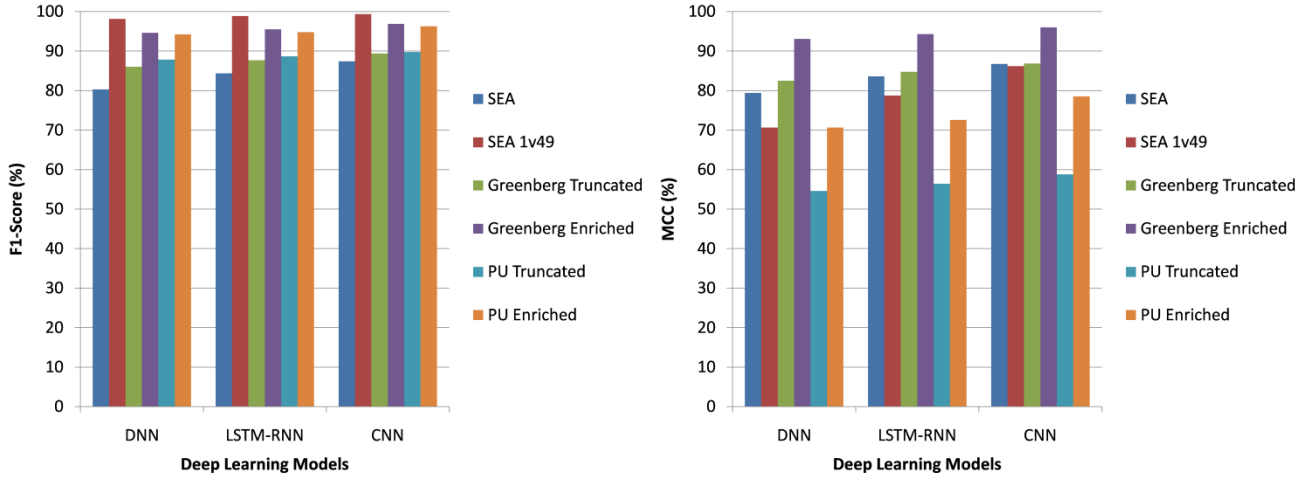
(ç)


Cost (%)
35
30
25
20
15
10
5
0
DNN
LSTM-RNN
CNN
Deep Learning Models
SEA
SEA 1v49
Greenberg Truncated
Greenberg Enriched
PU Truncated
PU Enriched

(d)

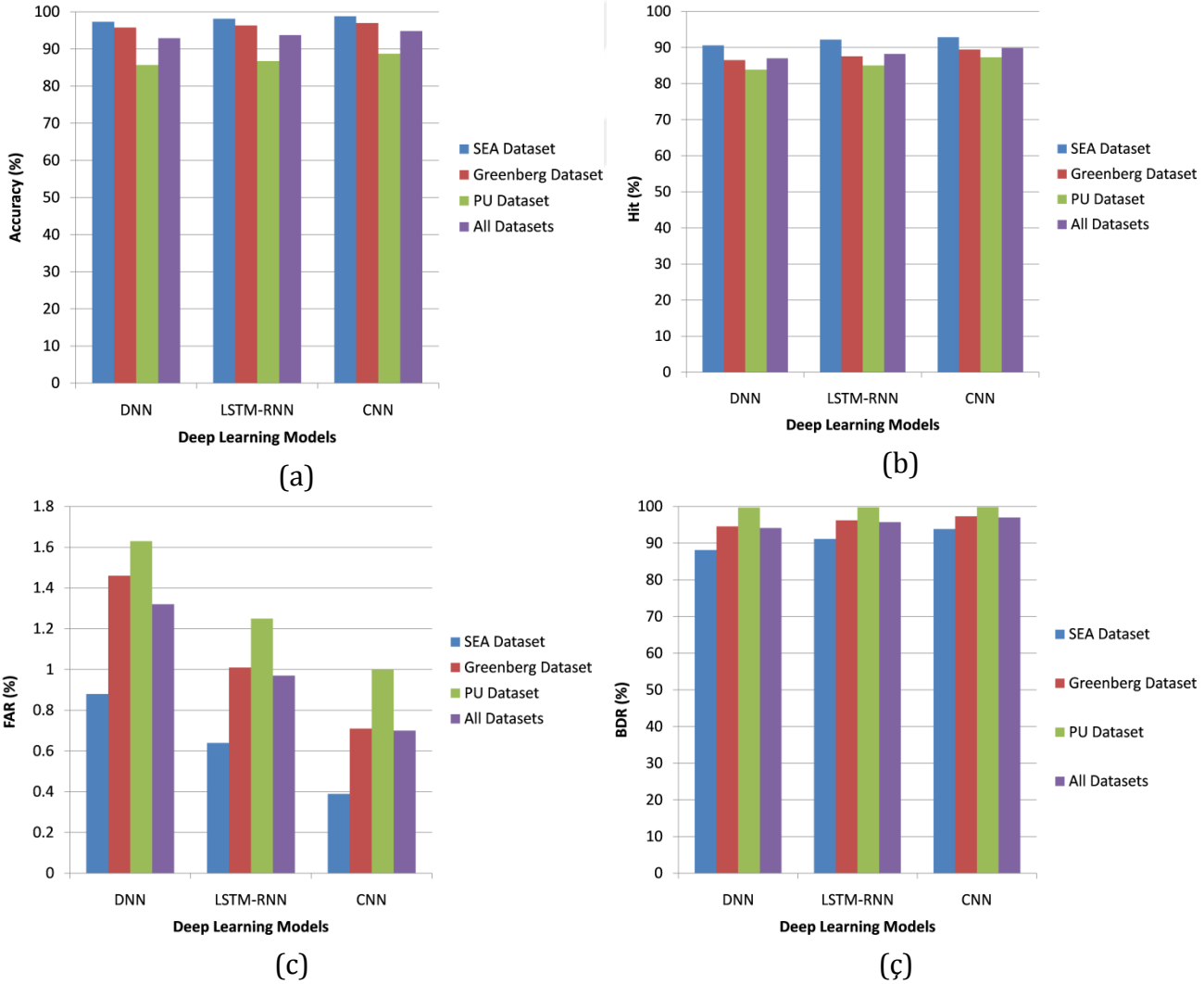

BDR (%)
100
90
80
70
60
50
40
30
20
10
0
DNN
LSTM-RNN
CNN
Deep Learning Models
SEA
SEA 1v49
Greenberg Truncated
Greenberg Enriched
PU Truncated
PU Enriched

(e)



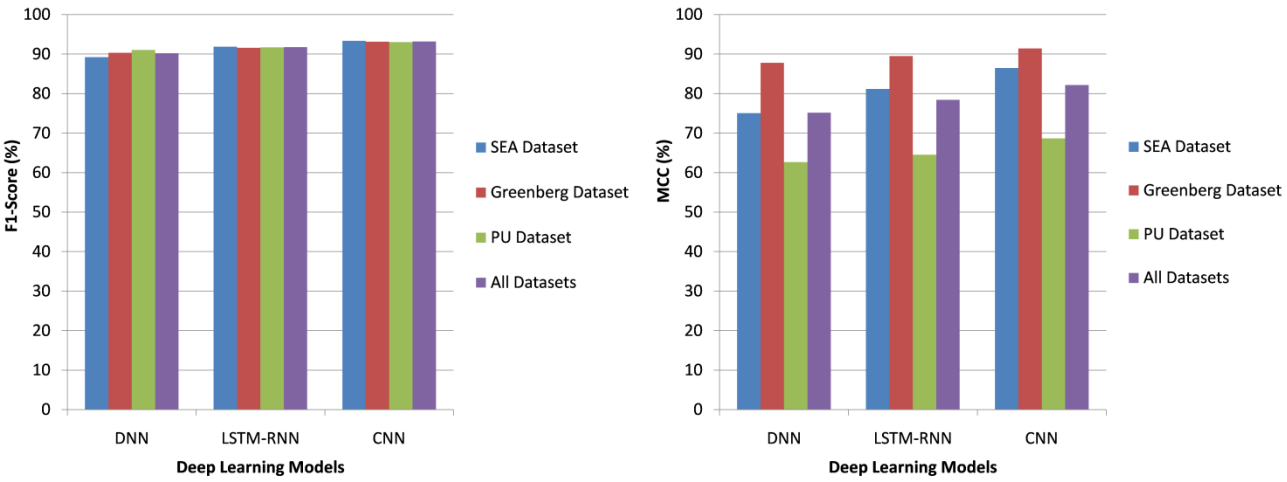

(f) (g)

Şekil 3.9. Veri yapılandırmalarındaki modeller arasında değerlendirme metrikleri karşılaştırması a) Doğruluk, b) Vuruş Oranı, c) Kayıp Oranı, ç) FAR, d) Maliyet, e) BDR, f) F1-Puanı, g) MCC




(a) (b)

(c) (ç)




(d) (e)

Şekil 3.10. Veri setlerinde modellerin ortalama performansları için değerlendirme ölçütleri a) Doğruluk, b) Vuruş Oranı, c) FAR, ç) BDR, d) F1-Puan, e) MCC

Her neyse, LSTM-RNN, tüm veri konfigürasyonları ve veri setleri ile ilgili kullanılan tüm değerlendirme ölçümleri açısından DNN'den daha iyi performans gösterdi. Bu, LSTM-RNN modelinin tüm gizli katmanlarda yapay nöronlar yerine LSTM bellek hücrelerini kullanması nedeniyledir. Ayrıca, LSTM-RNN modeli, aynı gizli katmandaki bellek hücreleri arasındaki bağlantıların yanı sıra kendi kendine tekrarlayan bağlantılara sahiptir. DNN'de bulunmayan LSTM-RNN'nin bu özellikleri, LSTM-RNN'nin önceki durumları ezberlemesine, aralarındaki bağımlılıkları keşfetmesine ve nihayetinde çıktıyı tahmin etmek için mevcut girdilerle birlikte kullanmasına izin verir. Bununla birlikte, LSTM-RNN ve DNN modellerinin tüm veri konfigürasyonlarındaki performansı arasındaki fark, göreceli olarak düşüktür; bu, vuruş ve doğruluk için (1%-3%) arasında ve tüm durumlarda FAR için (0.2%-0.8%) arasındadır .

Statik maskeli saldırı algılama tekniğinin yanı sıra, veri yapılandırmalarında dinamik maskeli saldırı algılama görevini gerçekleştirmek için CNN modelini kullandık. Aslında, CNN, belirli bir veri konfigürasyonunda girişin her kullanıcı için komut metni dosyaları olduğu metin sınıflandırma görevinde kullanılır. Elde edilen sonuçlar, CNN'nin tüm DNN ve LSTM-RNN modellerini, tüm veri yapılandırmalarında kullanılan tüm değerlendirme ölçümleri açısından daha iyi performans gösterdiğini açıkça göstermektedir. Bunun nedeni, giriş metin dosyalarından dinamik olarak kullanıcının bireysel komutları arasındaki

ilişkinin tanınabileceği şekilde çıkarılan ve öğrenilen derin yapı karakter düzeyinde bir CNN modeli kullanılmasıdır. Daha sonra, çıkarılan özellikler, daha sonra maskeli saldırılarını etkin bir şekilde saptamak için kullanılacak kullanıcının normal profilini oluşturmak için kendini geliştirmek için tamamen bağlı katmanlarıyla temsil edilir. Bu dinamik süreç ve kendi kendine öğrenme yetenekleri, bu tür derin öğrenme modellerinin ana hedeflerini ve güçlü yönlerini oluşturur. Kullanılan CNN modeli, tüm veri konfigürasyonlarında çok iyi sonuçlar vermiştir, doğruluk (83.75%-98.84%), vuruş oranı (81.64%-98.74%) ve FAR (% 0.19-1.5) arasındadır. Bu nedenle çalışmamızda dinamik maskeli saldırı algılama, statik maskeli saldırı algılama tekniğinden daha iyidir. Bu, dinamik maskeli saldırı saptama tekniğinin, UNIX komut satırı tabanlı veri kümeleriyle ilgili olarak maskeli saldırı saptaması için en iyi seçenek olduğu izlenimini verir, çünkü bu veri setleri başlangıçta metinsel veri setleridir ve bunları statik sayısal veri setlerine dönüştürürler, onları yeterli miktarda bilgi kaybedebilir. Buna rağmen, DNN ve LSTM-RNN, veri yapılandırmalarında maskeli saldırı tespitinde de çok iyi performans gösterdi .

BDR ve BTNR ölçümleri ile ilgili olarak, tüm kullanılmış modeller çoğu durumda yüksek değerlere sahiptir, bu modellerin öngörülen davranışlarının güvenliğinin çok yüksek olduğu anlamına gelir. Aslında, bu incelenen veri yapılandırmasının yapısına bağlıdır, yani BDR, incelenen veri yapılandırmasının test setindeki maskeleyici bloklarının sayısı kadar artacak ve vuruş oranı değeri daha büyük olacaktır. Buna karşılık, BTNR, incelenen veri yapılandırmasının test setindeki normal blokların sayısı arttıkça FAR değeri daha küçük artacaktır. Tüm kullanılan veri yapılandırmaları dengesiz olmasına rağmen, kullanılan tüm derin öğrenme modelleri, tüm veri yapılandırmaları için yüksek g-ortalama yüzdelere sahipti. Aynı şey, kullanılan tüm derin öğrenme modellerinin PU Kesilmiş hariç tüm veri yapılandırmaları için yüksek yüzdeleri kaydettiği MCC ölçümünde de oldu .

Çizelge 3.6'daki sonuçların daha ayrıntılı bir incelemesini yapmak için, Friedman ve Wilcoxon testleri olmak üzere iki iyi bilinen istatistiksel test yaptık. Friedman testi, üç veya daha fazla tekrarlanan numune (veya tedavi arasındaki)

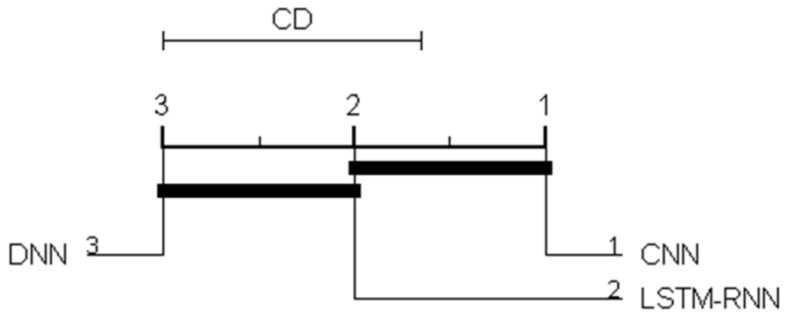
farkları bulmak için parametrik olmayan bir testtir (Daniel, 1990). Parametrik olmayan test, testin verilerinizin belirli bir dağıtımdan geldiğini varsaymadığı anlamına gelir. Bizim olgumuzda, kullanılan derin öğrenme modellerinden biri ve altı denekten ($R$=6) her biri için üç tekrarlı tedavimiz var ($k$=6). Her muamelede, her bir konunun, kullanılan veri yapılandırmalarından biriyle ilgili olduğu. Friedman testinin boş hipotezi, tedavilerin hepsinin aynı etkilere sahip olmasıdır. Matematiksel olarak, eğer hesaplanan Friedman testi İstatistiği (FS), Kritik Friedman testi değerinden (FC) daha büyükse ve boş hipotezi reddedebiliriz .

Öte yandan, Sıralama Toplamı testini veya İşaretli Sıra testini ifade eden Wilcoxon testi, iki eşleştirilmiş grubu ($k$=2) karşılaştıran parametrik olmayan bir testtir (Wilcoxon, 1945). Test, esas olarak her bir çift seti arasındaki farkı hesaplar ve bu farklılıkları analiz eder. Bizim olgumuzda her tedavide altı denek ($R$=6) ve üç eşleştirilmiş grup var, yani *p1*=(DNN,LSTM-RNN), *p2*=(DNN,CNN), and *p3*=(LSTM-RNN,CNN). Wilcoxon testinin boş hipotezi medyan farkın sıfır olmasıdır. Matematiksel olarak boş hipotezi reddedebiliriz, eğer ve sadece kullanarak hesaplanan olasılık (*P-değer*) Wilcoxon testi İstatistik (WS), belirli bir anlamlılık seviyesinden ($\alpha$) daha küçüktür. Seçilmiş $\alpha$ oldukça yaygın olduğu için $\alpha$=0.05. Çizelge 3.7, TP, FP, TN ve FN ölçümleri için Friedman ve Wilcoxon testlerinin sonuçlarını göstermektedir.

Çizelge 3.7. İstatistiksel testlerin sonuçları

| **Ölçümler** | **Friedman testi** | | **Wilcoxon testi** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | p1 | | p2 | | p3 | |
| | FS | FC | WS | P-değeri | WS | P-değeri | WS | P-değeri |
| TP | 12 | 7 | 0 | 0.0025 | 0 | 0.0025 | 0 | 0.0025 |
| FP | 12 | 7 | 0 | 0.0025 | 0 | 0.0025 | 0 | 0.0025 |
| TN | 12 | 7 | 0 | 0.0025 | 0 | 0.0025 | 0 | 0.0025 |
| FN | 12 | 7 | 0 | 0.0025 | 0 | 0.0025 | 0 | 0.0025 |

Çizelge 3.7'den, Friedman testinin boş hipotezini tüm durumlarda reddedebileceğimiz anlaşılabilir çünkü (FS>FC). Bu, kullanılan her öğrenme için kullanılan derin öğrenme modellerinin puanlarının farklı olduğu anlamına gelir. Friedman testinin sonuçlarını görsel olarak yorumlamanın bir yolu, Kritik Fark Diyagramı'nı çizmektir (Demsar, 2006). Şekil 3.11, kullanılan derin öğrenme modellerinin Kritik Fark Diyagramını göstermektedir. Çalışmamızda Kritik Fark (CD) değeri 1.3533'e eşittir.




Şekil 3.11. Tüm veri yapılandırmalarında kullanılan derin öğrenme modellerinin Kritik Fark Diyagramı

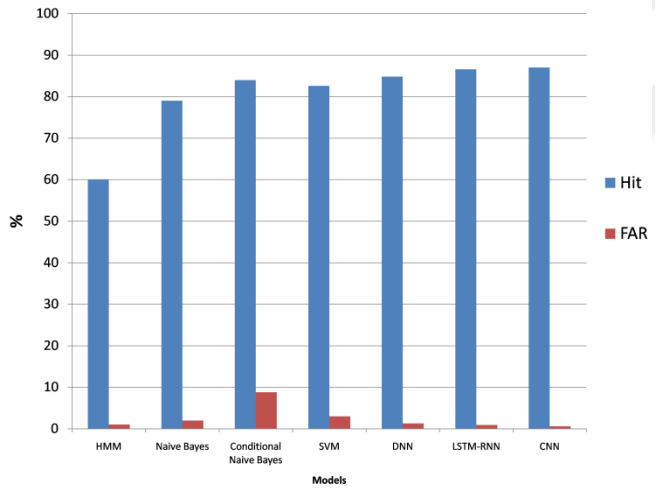
Ayrıca, Çizelge 3.7'den biz hipotezini reddedebilir Wilcoxon testi için *P-değeri* daha küçük olduğu $\alpha$ tüm durumlarda (0.0025<0.05). Böylece, her bir eşleştirilmiş grubun ortancalarının farklı olduğuna dair istatistiksel olarak anlamlı kanıtlara sahip olduğumuzu söyleyebiliriz. Son olarak, tüm ölçümlerin aynı sonuçlarının nedeni, sıralı modellerin (CNN, LSTM-RNN, DNN) TP ve TN'de daha yüksek puanlara sahip olmasının yanı sıra, tüm veri yapılandırmalarında FP ve FN'de daha küçük puanlara sahip olmasıdır.

Şekil 3.12(a), 3.12(b), 3.12(c), 3.12(ç) and 3.12(d) geleneksel makine öğrenme modellerinin performansı ile kullanılan derin öğrenme modellerinin için Hit ve FAR yüzdeleri açısından karşılaştırılmasını, sırasıyla, SEA, SEA 1v49, Greenberg Kesilmiş, Greenberg Zenginleştirilmiş ve PU Zenginleştirilmiş gösterir. Geleneksel makina öğrenme modelleri için Hit ve FAR yüzdelerini Çizelge 2.1'den literatürdeki en iyi sonuç olarak aldık. Geleneksel makine öğreniminin performansı ile kullanılan derin öğrenme modelleri arasındaki fark açıkça algılanabilir. DNN, LSTM-RNN ve CNN, hiperparametreler için PSO tabanlı bir

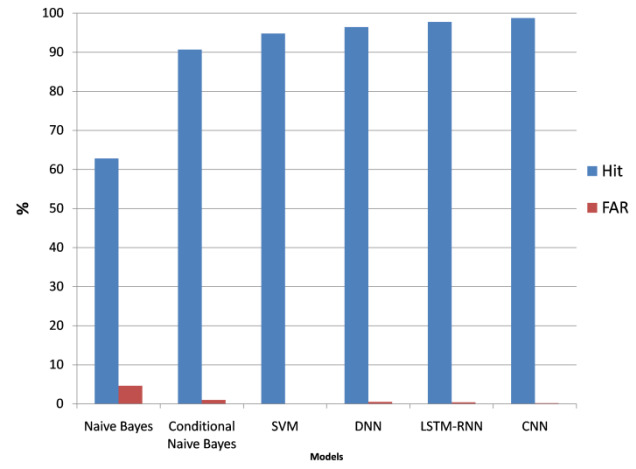
algoritma nedeniyle tüm geleneksel makine öğrenme modellerinden daha iyi performans gösterdi. DNN ve LSTM-RNN ile kullanılan ve CNN ile kullanılan özellikli öğrenme mekanizmasının seçimi. Buna ek olarak, derin öğrenme modelleri geleneksel makine öğrenme modellerinden daha derin yapılara sahiptir. Kullanılan derin öğrenme modelleri, çoğu durumda geleneksel makine öğrenimi modellerine göre vuruş oranının yüzdelerini (2%-10%) arttı ve FAR yüzdelerini (1%-10%) azalttı.

### 3.5.2. ROC eğrileri analizi

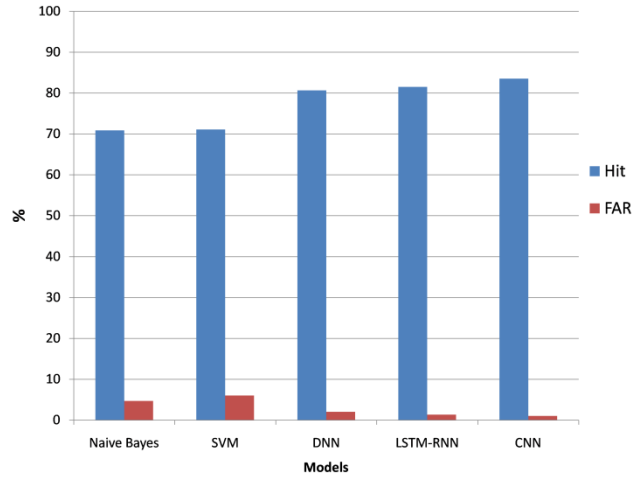
Alıcı Çalışma Karakteristiği (ROC) eğrisi, Y eksenindeki Gerçek Pozitif Oranın (veya Vuruş) X eksenindeki Yanlış Pozitif Orana (veya FAR) karşı değerlerinin bir grafiğidir.



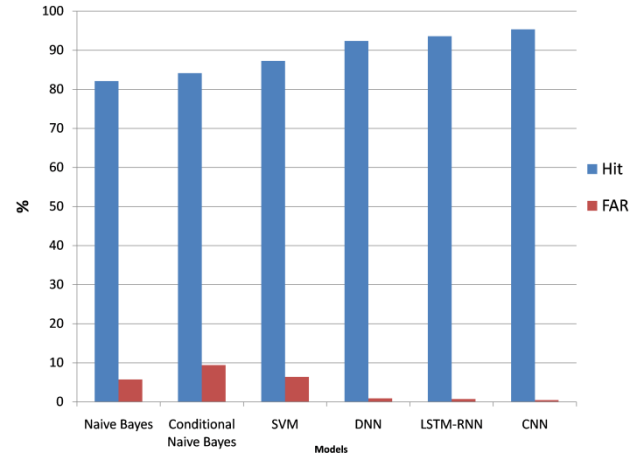

(a)



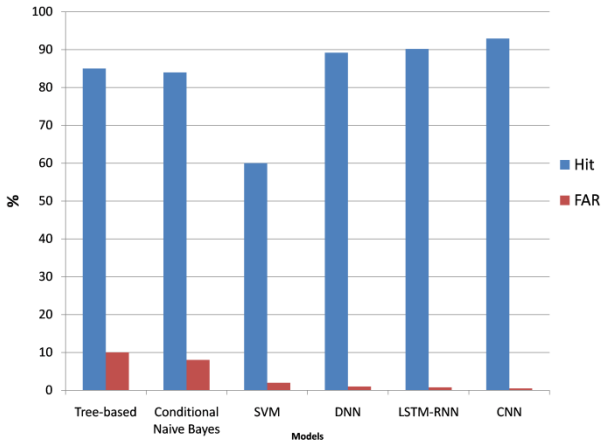

(b)




(c)




(ç)

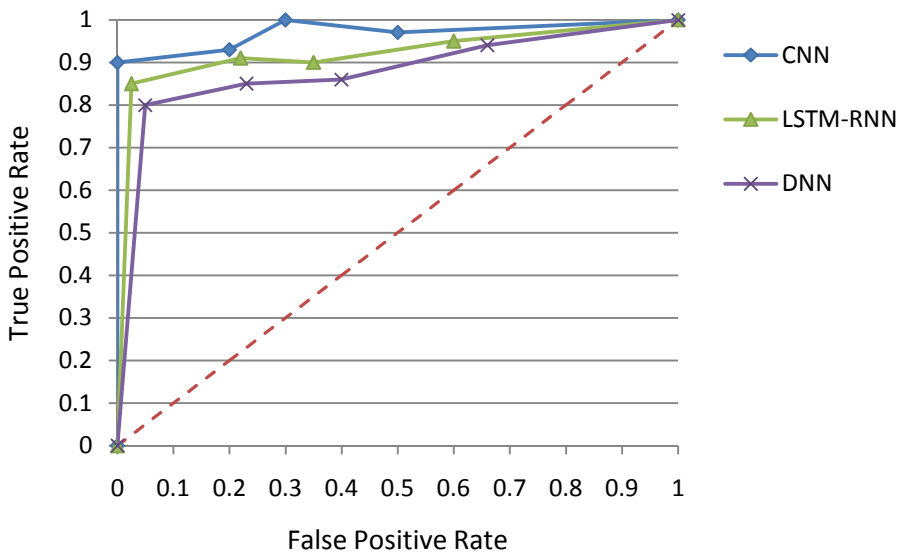



(d)

Şekil 3.12. Her veri yapılandırması için modeller performans karşılaştırması a) SEA, b) SEA 1v49, c) Greenberg Kesilmiş, ç) Greenberg Zenginleştirilmiş, d) PU Zenginleştirilmiş

Farklı makine öğrenme algoritmalarının performansını değerlendirmek ve en uygun sınıflandırıcıyı seçmek için aralarındaki dengeyi göstermek için yaygın olarak kullanılır. ROC'nin diyagonal çizgisi referans çizgisidir ve bu, performansın 50%'sinin elde edildiği anlamına gelir. ROC'nin sol üst köşesi, 100% ile en iyi performans anlamına gelir. Şekil 3.13, tüm derin öğrenme modellerinin her birinin ortalama performansının ROC eğrilerini tüm veri yapılandırmaları üzerinden gösterir. ROC eğrileri, sırayla modellerin: CNN, LSTM-RNN ve DNN'nin tüm veri yapılandırmaları üzerinde etkili maskeli saldırı tespiti için performansına sahip olduğunu göstermektedir. Ancak, bu üç derin öğrenme modelinin tümü hala oldukça iyi bir uyum sergiliyor.

Eğri Altındaki Alan (AUC) ayrıca çeşitli ROC eğrileri arasında kantitatif olarak karşılaştırma yapmak için iyi bilinen bir ölçü olarak kabul edilir (Cortes ve Mohri, 2004). Bir ROC eğrisinin AUC değeri, 0 ile 1 arasında olmalıdır. İdeal sınıflandırıcı, AUC değerinin 1'e eşit olmasını sağlar. Çizelge 3.8, kullanılan üç derin öğrenme modelinin Şekil 3.13'te çizilen ROC eğrilerinin AUC değerlerini göstermektedir. Tüm bu modellerin neredeyse 1'e ulaşan çok yüksek AUC değerlerine sahip olduğunu açıkça görebiliyoruz, bu da UNIX komut satırı tabanlı veri kümelerinde maskeli saldırı algılamakta etkin olmalarının oldukça kabul edilebilir olduğu anlamına geliyor.




Şekil 3.13. Tüm veri yapılandırmalarında kullanılan modellerin ortalama performansının ROC eğrileri

Çizelge 3.8. Kullanılan modellerin ROC eğrilerinin AUC değerleri

| **Model** | **AUC** |
|---|---|
| DNN | 0.9246 |
| LSTM-RNN | 0.9385 |
| CNN | 0.9617 |

# 4. AĞ SALDIRI TESPİTİ

Saldırının önlenmesi, ağ güvenliğinin temel taşı olarak kabul edilir. Son on yılda ağ saldırı tespitinde aşırı bir çalışma yapılmış olmasına rağmen, güçlü saldırı tespit mekanizmasına sahip saldırı tespit sisteminin bulunması hala oldukça istenmektedir. Çok sayıda yanlış alarmın ve düşük bir algılama oranının önde gelen nedenlerinden biri, veri kümelerinde yedekli ve alakasız özelliklerin varlığıdır. Bu sorunu önlemek için, tek bir işlemde hem özellik alt kümesini hem de hiperparametreleri seçmek için çift Parçacık Sürüsü Optimizasyonu tabanlı bir algoritma önerdik. Söz konusu algoritma, eğitim öncesi aşamada optimize edilmiş özellikleri ve modelin hiperparametrelerini otomatik olarak seçmek için kullanılır.

## 4.1. Giriş

Hacim, Hız ve Çeşitlilik: Bugünlerde verinin üç Vs büyümesi artan "Büyük Verilerden" çağında yaşıyoruz. Hacim oluşturulan veri boyutu ve GB, terabyte'tan, hatta petabayttan ölçülebilir. Hız verileri belirli bir zaman birimi başına elde edildiği oranı anlamına gelir ise, çeşitli bu verilerin türü vardır. Veri depolama, işleme, satın alma, ve geçiş bu sayısız değişiklikleri takip amacıyla, bilgi sistemleri üzerinde bir devrim talep edilir ve BT endüstrisi tarafından ortaya koydu. Aynı derecede önemli, zorluklar ve tehditler güvenlik perspektifinde kadar monte edilmiştir olduğunu (Mahmood ve Afzal, 2013).

Siber güvenlik olarak, bir yabancının bazı anonim parti sistemi güvenlik açıklarını kullanarak gerçek bir istemci özelliklerini amaçlamaktadır olduğunu. Öncelikle, bilgisayar ağ saldırı başlatılması saldırılara elde edilebilir. Bir siber saldırgan meşru bir kullanıcının bilgilerine yetkisiz erişim elde etmek için ya da sistemi aşağı yapmak ya şüpheli faaliyetleri başlattığında bir bilgisayar ağı saldırısı gerçekleşir. Hizmet Reddi (DoS), Problama (Prob), Yerelden Uzaka (R2L), Kökten Kullanıcıya (U2R), Brute Force, vb. Bilinen pek çok bilgisayar ağı saldırısı vardır. Kullanıcı akılda tutarak bu saldırıların olabileceğini öyle ki bir sonucu olarak vb HTTP, FTP, ICMP, TCP, UDP, SMTP gibi ağ protokolleri ve

hizmetlerden herhangi birini, bu saldırılar daha yaygın hale geliyor ve ağ güvenliği en ciddi tehdit olarak kabul edilir kullanılarak yürütülür. Özellikle, bu tür saldırıları önlemenin en etkili yolu, sistemin fiziksel ve yazılım altyapısını izleyebilen ve herhangi bir anormal davranışı arayabilen Ağ Tabanlı Saldırı Tespit Sistemi (NIDS) kullanmaktır (Marir vd., 2018).

Ağ güvenlik literatüründe, NIDSs iki genel yaklaşım vardır: imza tabanlı algılama ve anomali tabanlı algılama. İmza tabanlı algılama ya da adlandırılan kötüye kullanma tespiti saldırı imzası zaten bilindiği kullanılacak değer. Öte yandan, bilinen veya bilinmeyen ataklar için anomaliye dayalı tespit kullanılabilir. NIDS kullanmanın arkasındaki ana fikir, arka arkaya üç adımda trafik tanımlamasıdır. İlk olarak, NIDS ağdan geçen trafiği yakalar. Daha sonra, yakalanan trafiği trafik kayıtları oluşturacak şekilde işler. Bu kayıtların her biri, doğrudan yakalanan trafikten elde edilen birçok kullanışlı özellikten oluşur. Ardından, önceden eğitilmiş veri madenciliği algoritmalarından birini kullanarak, test edilen trafik kaydını normal veya saldırı faaliyetlere göre sınıflandırır. Pratik olarak, bir trafik kaydı anormal olarak sınıflandırıldığında, NIDS kırmızı bayrağı yükseltir ve sistemin olası bir saldırıya uğradığı alarmını verir. Birçok ağ saldırı tespit algoritması kullanılmış, ancak aralarında makine öğrenme modelleri veriden öğrenme ve normal ve kötü niyetli kullanıcıları ayırt edebilmeleri nedeniyle en sık kullanılan yaklaşımlardır (Dong ve Wang, 2016).

### 4.2. Veri Setleri

Önemli olan, bir NIDS'in etkinliğini ölçmek için etkili bir veri setine ihtiyaç vardır. Etkili bir veri kümesi, bir veri kümesinin, gerçek dünyadaki ağları yansıtan yeterli miktarda güvenilir veriden oluştuğu anlamına gelir. Böylece, 1998'den bu yana çok sayıda IDS veri seti oluşturulmuştur. Bu çalışmada NSL-KDD ve CICIDS2017 olarak iki farklı IDS veri seti seçtik. Her ne kadar NSL-KDD veri seti nispeten eski olsa da, hala iyi bilinen bir kriter olarak kabul edilir ve yaygın olarak ağa saldırı tespit alanında kullanılır. Çeşitlilik ve güncel IDS veri setinin araştırılması için CICIDS2017 veri seti bu çalışmaya dahil edilmiştir. Gelecek alt bölümler, her veri kümesinin yapısını açıklar. Çizelge 4.1, kullanılan

veri setlerinin temel özelliklerini göstermektedir. Çizelge 4.1'deki CICIDS2017 veri setinin hem eğitiminde hem de test setinde bulunan örneklerin sayısının sadece 10% olduğunu söylemek gerekir.

Çizelge 4.1. Kullanılan veri setleri ana özellikleri

| **Özellikleri** | | **NSL-KDD** | **CICIDS2017** |
|---|---|---|---|
| Yıl | | 2009 | 2017 |
| Denetim biçimi | | tcpdump | pcap |
| Özelliklerin sayısı | | 41 | 80 |
| Protokollerin sayısı | | 6 | 5 |
| Saldırıların sayısı | | 38 | 20 |
| Saldırı kategorin sayısı | | 4 | 7 |
| Etiketli sınıfların sayısı | | 5 | 8 |
| Eğitim setinin Dağılımı | Normal | 67343 | 18750 |
| | Saldırılar | 58630 | 131250 |
| | Toplam | 125973 | 150000 |
| Test setinin Dağılımı | Normal | 9711 | 18750 |
| | Saldırılar | 12833 | 131250 |
| | Toplam | 22544 | 150000 |

### 4.2.1. NSL-KDD veri seti

İlk olarak MIT Lincoln Laboratory DARPA veri setini oluşturduğu zaman başlamıştır (MIT Lincoln Laboratory, 1998). Kısa sürede, gerçek dünyadaki ağ trafiğini temsil etme sınırlaması nedeniyle DARPA'nın ağ saldırı tespitinde yetersiz olduğu belirtildi (McHugh, 2000). Bu nedenle, DARPA'nın güncellenmiş bir versiyonu, yani KDD CUP 99, 1999'da *tcpdump* kısmının işlenmesiyle yaratılır (Stolfo vd., 2000). KDD CUP 99'un 10%'u, ağa saldırı tespiti alanında yaygın olarak kullanılmasına rağmen, yeni bir IDS veri setine ihtiyaç duyulmasını gerektiren doğal problemlerden ciddi şekilde zarar görmektedir. Dolayısıyla, Tavallaee vd. (2009), KDD CUP 99 veri setinin 10%'unun, NSL-KDD'nin geliştirilmiş ve azaltılmış bir versiyonunu önermiştir. KDD CUP 99'un 10%'unun tüm dezavantajlarını iki şekilde çözdüler. Öncelikle, tüm gereksiz kayıtları hem eğitim hem de test setlerinden elimine ettiler. İkinci olarak, kayıtları çeşitli zorluk seviyelerine ayırdılar, ardından KDD CUP 99 veri setinin orijinal 10%'undaki kayıtların yüzdesiyle ters orantılı olarak her zorluk seviyesinden kayıt seçtiler. Sonuç olarak, NSL-KDD hem eğitim hem de test

setlerinde makul sayıda kayda sahiptir ve deneyleri tüm sette yürütmeyi uygun kılar. Ayrıca, NSL-KDD veri kümesi İnternet üzerinde kamuya açıktır (NSL-KDD Dataset, 2009).

NSL-KDD'nin yapısı ile ilgili olarak, her numune normal veya bir saldırı kaydında etiketlenir. Temel olarak, NSL-KDD'ye dahil olan toplam 38 tip saldırı vardır. NSL-KDD'nin eğitim setinde, yalnızca 24 saldırı türünden örnekler ortaya çıktı. Buna karşılık, test setinde her tür saldırıdan örnekler ortaya çıktı. Bu, test setinde NIDS'in eğitim setinde görünmeyen yeni saldırıları tespit etmedeki etkinliğini değerlendirmek için kullanılır. Buna ek olarak, algılama oranını iyileştirmek için, benzer saldırılar, DoS, Probe, R2L ve U2R olmak üzere dört ana saldırı kategorisi oluşturan tek bir kategoride bir araya getirilir. Buna göre, NSL-KDD veri setinde Normal, DoS, Probe, R2L ve U2R olan beş sınıf vardır. NSL-KDD'nin eğitim kümesinde toplam 125973 trafik örneği varken, test kümesinde 22544 örneği bulunuyor. Eğitim kümesinde örneklerin her sınıfa dağılımı şöyledir: Normal (67343), DoS (45927), Prob (11656), R2L (995) ve U2R (52). Diğer taraftan, test kümesindeki örneklerin dağılımı aşağıdaki gibidir: Normal (9711), DoS (7458), Prob (2421), R2L (2887) ve U2R (67).

NSL-KDD, SMTP, HTTP, FTP, Telnet, ICMP ve SNMP gibi altı farklı ağ protokolüne ve servisine sahiptir. Son olarak, temel özellikler, trafik temelli özellikler ve içerik temelli özellikler olmak üzere üç türe ayrılabilen 41 özellik içerir. Bu özelliklerin veri tipleri nominal (3), ikili (4) ve süreklidir (34).

### 4.2.2. CICIDS2017 veri seti

Kanadalı Siber Güvenlik Saldırı Tespit Sistemi (CICIDS2017), 2017 yılında teklif edilen ve sahiplerinin talebi üzerine İnternet üzerinden kamuya açık bir şekilde sunulan modern anomali tabanlı IDS veri setidir (Kanada Siber Güvenlik Enstitüsü - IDS, 2017). Bu veri setinin oluşturulmasının arkasındaki amaç, araştırmacıların modellerini doğru bir şekilde değerlendirmelerine yardımcı olmak için gerçekçi bir veri içeren güvenilir ve güncel bir veri seti sunmaktır. Ayrıca, bu veri setinin, mevcut diğer IDS veri setlerinin tüm eksikliklerinin

üstesinden geldiği bildirilmektedir (Sharafaldin vd., 2018). Saldırı Ağı ve Kurban Ağı olmak üzere iki ayrı ağdan oluşan bir çevre altyapısı kullandılar. Kurban Ağı, B-Profile sistemini kullanarak iyi huylu davranışı sağlamak için kullanılır (Sharafaldin vd., 2017). Öte yandan, Saldırı Ağı, saldırı akışlarını gerçekleştirmek için kullanılır. Her ikisi de gerekli ağ aygıtları ve farklı işletim sistemleri çalıştıran bilgisayarlar ile donatılmıştır. Ayrıca, yakalanan *pcap* verilerini beş iş günü boyunca analiz etmek için CICFlowMeter'i kullandılar. Bu veri setindeki ağ bağlantısı kayıtları, HTTP, HTTPS, FTP, SSH ve e-posta protokollerini temel almaktadır. Buna ek olarak, saldırı akışları toplam 20 saldırıdan oluşur ve Brute Force, Heartbleed, Botnet, DoS, DDoS, Web saldırısı ve Sızma olmak üzere yedi ana kategoriye ayrılır. Son olarak, CICIDS2017, belirli trafik kaydını sekiz olası sınıftan birine tanımlamak için bir sınıf etiketinin yanı sıra 80 farklı özellik içerir.

Hem eğitim hem de test için belirli sayıda örneğin bulunduğu NSL-KDD veri setinden farklı olarak, CICIDS2017, farklı dosyalarda yaklaşık 3 milyon ağ akışına sahip çok büyük bir veri kümesidir (Kanada Siber Güvenlik Enstitüsü - IDS, 2017). Ayrıca, CICIDS2017 veri setinde, deneylerde kullanılacak belirli bir eğitim veya test kümesi yoktur. Bu nedenle, eğitim ve test süresini makul bir şekilde azaltmak için CICIDS2017'nin 10%'unu eğitim ve test için seçtik. Çünkü CICIDS2017'nin tam boyunu kullanırken, eğitim ve test süreleri çok uzun olacaktır. CICIDS2017'nin 10%'u, bir nesne seçildikten sonra popülasyondan çıkarılmasını sağlamak için yerine koyma tekniği olmadan örnekleme kullanılarak rastgele seçilir. Trafik kayıtlarının çeşitliliğini sağlamak ve fazla uydurmamaktan kaçınmak için, dengeli eğitim ve test setleri uyguladık, yani bunlar eşdeğerdir (her birinde 150 bin örnek). Eğitim kümesindeki örnekler aşağıdaki gibi eşit şekilde dağılmıştır: Normal (18750), Brute Force (18750), Heartbleed (18750), Botnet (18750), DoS (18750), DDoS (18750), Web saldırısı (18750) ve Sızma (18750). Ardından, test setini oluştururken aynı işlem tekrarlanır, ancak eğitim verilerinde bulunmayan ve eğitim kümesinde olduğu gibi eşit olarak dağıtılan örneklerle tekrarlanır. Ayrıca, bazı saldırı türleri, kullanılmış NIDS'in bunları doğru şekilde sınıflandırabildiğini incelemek için verilen eğitim setinden ziyade yalnızca test setine dahil edilir. Bu prosedüre

göre hem eğitim hem de test kümelerinin dengeli olduğuna ve kullanılan modellerin eğitim aşamasında herhangi bir sınıfa eğilimli olmayacağına inanıyoruz.

### 4.3. Önerilen Çift PSO Tabanlı Algoritma

Bu bölümde, hem özellik hem de hiperparametre seçimi için çift PSO tabanlı algoritmamızı tanıtıyoruz. Önerilen algoritma, eğitim öncesi aşamada, ağa saldırı tespitinde derin öğrenme modellerinin performansını artırmak için kullanılacaktır.

#### 4.3.1. Özellik seçimi

Öncelikle, herhangi bir sınıflandırma görevinde, özellik alanı, bir sınıflandırıcının performansını etkileyen önemli bir faktördür. Eldeki sınıflandırma görevi için hangi özelliklerin önemli olduğunu belirlemek zor bir süreçtir. Bu sorunu çözmek için, önemsiz özellikleri kaldırma, örneğin; gereksiz ve alakasız özellikler. Gereksiz özellikler, bir veya daha fazla başka nitelikte yer alan bilgilerin çoğunu veya tamamını çoğaltmak anlamına gelirken, alakasız özellikler, belirli veri madenciliği görevi için yararlı olan hiçbir bilgi içermez. Özellik seçiminin faydaları, önemsiz özelliklerin ortadan kaldırılması ile sınırlı değildir, aynı zamanda boyutsallığın lanetini önlemek, gürültüyü azaltmak, veri madenciliğinde gereken zamanı ve alanı azaltmak ve daha kolay görselleştirmeyi sağlamak için de geçerlidir. Özellikle, özellik seçimi saldırı tespitinde sınıflandırıcının performansını önemli ölçüde artırır, çünkü gereksiz ve alakasız özellikler sınıflandırıcıyı karıştırır ve yanlış sınıflandırma sayısını artırabilir. Ayrıca, çalışma süresini kısaltarak ve modelin yapısını basitleştirerek hesaplama verimliliğini artırabilir (Dash ve Liu, 1997).

Geleneksel özellik seçim yöntemlerinden biri, çok uzun zaman alabilecek en iyi özellik altkümesini bulmak için kapsamlı bir arama yapmaktır (Yang vd., 2008). Bu nedenle, en uygun çözümden ziyade makul bir süre içinde iyi bir çözüm bulmak gerçek dünyadaki uygulamalarla daha fazla ilgilenmektedir. Son

zamanlarda, özellik seçimi probleminin optimal veya optimal çözümünü elde etmek için Evrimsel Hesaplama (EC) teknikleri uygulanmıştır. Örneğin, GA (Oliveira vd., 2002; Waqas vd., 2009), PSO (Azevedo vd., 2007; Lin vd., 2008; Mohemmed vd., 2009), Genetik Programlama (GP) (Purohit vd., 2010; Neshatian vd., 2012) ve Karınca Koloni Optimizasyonu (ACO) (Sivagaminathan ve Ramakrishnan, 2007; Ke vd., 2010). Diğer EC tekniklerine kıyasla, PSO'nun özellik seçimi problemleri için etkili bir algoritma olduğu gösterilmiştir (Azevedo vd., 2007; Lin vd., 2008; Huang ve Dun, 2008; Mohemmed vd., 2009). Uygulaması daha kolay ve daha hızlı bir şekilde birleşmesi daha kolaydır (Kennedy ve Spears, 1998). PSO'nun teorik arka planını ve varyasyonlarını ve özellik seçiminde kullanımını açıkladıktan sonra, özellik seçimi için deneylerimizde kullandığımız algoritma olan 4.3.1.3 alt bölümünde sunuyoruz.

#### 4.3.1.1. İkili PSO

Geleneksel PSO, sürekli alanlar için iyi çalışır, ancak ayrı alanlarla ilgilenirken sonuçlar üzerinde olumsuz etkiler yaratabilir. Bu nedenle, Kennedy ve Eberhart (1997), bu sorunun üstesinden gelmek için İkili PSO (BPSO) algoritmasını tanıttı. PSO'dan farklı olarak, BPSO'da, pozisyon, kişisel en iyi ve global en iyi vektörler ikili dizelerle temsil edilir, yani tüm vektörlerin elemanları 0 veya 1 ile sınırlandırılır. Konum vektöründe 1 değerini alır. Matematiksel olarak, her yinelemede hız vektörünü güncellemek için Denklem 3.1 hala uygulanır. Daha sonra, Denklem 4.1'deki sigmoid işlevi $V_{t+1}^{i}$ i'yi [0,1] aralığına dönüştürmek için kullanılır. Daha sonra, BPSO her parçacık için konum vektörünü Denklem 4.2 kullanarak günceller; buradaki *rand ()*, [0,1] 'de düzgün bir dağılımdan seçilen rastgele bir sayıdır.

$$S\left(V_{t+1}^{i}\right) = \frac{1}{1+e^{-V_{t+1}^{i}}} \tag{4.1}$$

$$P_{t+1}^{i} = \begin{cases} 1, \text{eğer } rand() < S\left(V_{t+1}^{i}\right) \\ 0, \qquad \text{aksi takdirde} \end{cases} \tag{4.2}$$

Geleneksel BPSO algoritmasının iki ana dezavantajı olduğu bildirilmiştir (Zhou vd., 2014). Birincisi, bir sonraki yinelemede parçacığın konumu yalnızca hız

vektörüne bağlıdır. Bu nedenle, mevcut partikül pozisyonunun etkisini hesaba katarak yeni partikül pozisyonunu hesaplamak için yeni bir yola ihtiyaç vardır. İkincisi, BPSO'nun genel çeşitliliği koruyarak erken bir yakınsamaya sahip olma ihtimalinin büyük olması. Bu nedenle, parçacığın sürekli olarak en iyi çözüme doğru hareket etmesini sağlamak için hız güncelleme formülünü değiştirmeye de ihtiyaç vardır. Sonuç olarak, yukarıda bahsedilen iki sorunu çözmek için Fitness Proportionate Selection İkili Parçacık Sürü Optimizasyonu (FPSBPSO) adlı yeni bir ikili PSO algoritması önerdiler. FPSBPSO, partikülün hızını ve konumunu, sırasıyla Denklem 4.3 ve 4.4'e göre bir sonraki yinelemede günceller (Zhou vd., 2014).

$$V_{t+1}^{i} = \begin{cases} mr, \ eğer\ n_0 = 0 \\ 1 - mr, eğer\ n_1 = 0 \\ \frac{n_1}{n_0 + n_1}, aksi\ takdirde \end{cases} \tag{4.3}$$

$$P_{t+1}^{i} = \begin{cases} 1, eğer\ rand() < V_{t+1}^{i} \\ 0, aksi\ takdirde \end{cases} \tag{4.4}$$

*mr*, algoritmanın serbest parametresiyse, $n_0$, partikülün mevcut pozisyonunun karşılık gelen bitlerinin, partikülün kişisel en iyi ve global en iyi vektörlerin karşılık gelen bitlerinin sıfır değeridir ve $n_1$, $n_0$'ın tersidir ve (3- $n_0$) eşitir. FPSBPSO algoritması BPSO'nun dezavantajlarını çözmekten ziyade, FPSBPSO'nun, özellikle özellik seçim sürecinde, optimizasyon problemlerinin sonuçlarını iyileştirdiği de gösterilmiştir (Zhou vd., 2014). Ayrıca, FPSBPSO'nun ayarlanması BPSO'dan daha kolaydır, çünkü sadece bir parametreye sahiptir. Çoğu durumda, 0.01 değerinin *mr* parametresi için iyi bir seçim olduğu sonucuna vardılar. Son olarak, ikili PSO genellikle, özellik seçimi probleminin ayrı bir arama alanında meydana gelmesi nedeniyle, özellik seçim probleminde sürekli PSO'dan daha iyi performans gösterir (Cervante vd., 2012). Bu nedenle, özellik seçimi yönteminin tasarımında ikili PSO'dan yararlanacağız.

#### 4.3.1.2. Özellik seçimindeki BPSO

Genel olarak, özellik seçimi yöntemlerinden herhangi birinin, aday özellik altkümelerinin iyiliğini ölçmek için bir değerlendirme sürecine ihtiyacı vardır.

Açıkçası, BPSO algoritması sürüdeki her partikül için fitness skorunu hesaplayarak bu görevi yerine getirmek için önceden tanımlanmış bir fitness fonksiyonunu kullanır. Bu nedenle, uygunluk fonksiyonunun bir öğrenme algoritması içerip içermediğine bağlı olarak, Langley (1994) bunları iki geniş kategoride gruplandırmıştır: filtre tabanlı yaklaşımlar ve sarıcı bazlı yaklaşımlar. Filtre tabanlı yaklaşımlar, bir değerlendirme kriteri olarak bir öğrenme algoritması kullanmadan özellikleri seçer. Öte yandan, sarıcı bazlı yaklaşımlar, değerlendirme prosedüründe görünmeyen veriler üzerinde aday özellik alt kümesini test etmek için bir sınıflandırıcı kurmaktadır. sarıcı bazlı yaklaşımlara rağmen, genellikle özellik seçim problemindeki filtre temelli yaklaşımlardan daha iyi sonuçlar elde edilir (Tran vd., 2014), ancak özellikle özellik sayısı yüksek olduğunda, bunlar hesaplamalı olarak pahalıdır (Purohit vd., 2010). Bununla birlikte, filtre temelli yaklaşımlar, özellik seçimi görevlerinde en çok tercih edilenlerdir, çünkü hesaplama açısından daha ucuz ve daha genel olduklarını iddia etmişlerdir (Chakraborty, 2008; Yang vd., 2008). Bu nedenle çalışmamızda filtre tabanlı yaklaşımlar üzerinde durduk.

Ayrıca, istenen özellik seçim yöntemlerinin amacı, en düşük sınıflandırma kesinliğini elde edebileceği gibi en küçük olan optimum özellik alt kümesini bulmaktır. Bu perspektife göre, özellik seçimi temel olarak çok amaçlı bir optimizasyon problemidir ve çeşitli takas çözümleri (özellik alt kümeleri) üretir (Subbotin ve Oleynik, 2008). Tek hedefli özellik seçim yöntemi, birincinin yalnızca bir tane optimal özellik alt kümesi oluşturabildiği çok hedefli ile tamamen çelişkilidir (Tran vd., 2014). Çalışmamızda, sadece tek objektif özellik seçim yöntemlerine odaklandık çünkü kullanıcının herhangi bir müdahalesi olmadan tek bir optimal özellik alt kümesi üretmeyi gerektiren yaklaşımımızın doğası.

Literatürde önerilmiş olan birkaç tane tek objektif filtre temelli özellik seçme yöntemi vardır. Cervante vd. (2012) tarafından önerilen iki yöntem olduğu bildirildi, diğerlerine göre etkinlik ve üstünlüklerini kanıtlamıştır (Tran vd., 2014). BPSO ve bilgi teorisi kullanarak iki adet tek objektif filtre temelli özellik seçim yöntemi geliştirdiler. Birincisi, her bir özellik çiftinin karşılıklı bilgilerini

ölçerek, seçilen özellik alt kümesinin alaka düzeyini ve fazlalığını değerlendirir. Matematiksel olarak, ilk yöntemin uygunluk fonksiyonu Denklem 4.5 kullanılarak elde edilebilir (Cervante vd., 2012).

$$Fitness_1 = \alpha_1 \times D_1 - (1 - \alpha_1) \times R_1 \quad (4.5)$$

where, $D_1 = \sum_{x \in X, c \in C} I(x; c)$ ,

$R_1 = \sum_{x_i, x_j \in X} I(x_i; x_j)$.

*X*, seçilen özelliklerin kümesidir ve *C*, sınıf etiketlerinin kümesidir. $D_1$, her bir özellik ile sınıf etiketleri arasındaki karşılıklı bilgileri belirleyerek, seçilen özellik alt kümesinin sınıf etiketlerine uygunluğunu hesaplar. Öte yandan, $R_1$, seçilen her bir özellik çifti tarafından paylaşılan karşılıklı bilgileri göstererek, özellik alt kümesinde bulunan fazlalığı değerlendirir. $Fitness_1$'i kullanmanın amacı, sınıf etiketleriyle en fazla alaka düzeyine sahip ve aynı anda birbirleriyle minimum yedeklilik sağlayan bir özellik alt kümesi seçmektir.

Bu arada, ikinci yöntem, her bir özellik grubunun entropisini ölçerek, seçilen özellik alt kümesinin alaka düzeyini ve fazlalığını belirler. Matematiksel olarak, ikinci yöntemin uygunluk fonksiyonu Denklem 4.6 kullanılarak elde edilebilir (Cervante vd., 2012).

$$Fitness_2 = \alpha_2 \times D_2 - (1 - \alpha_2) \times R_2 \quad (4.6)$$

where, $D_2 = \sum_{c \in C} IG(c|X)$ ,

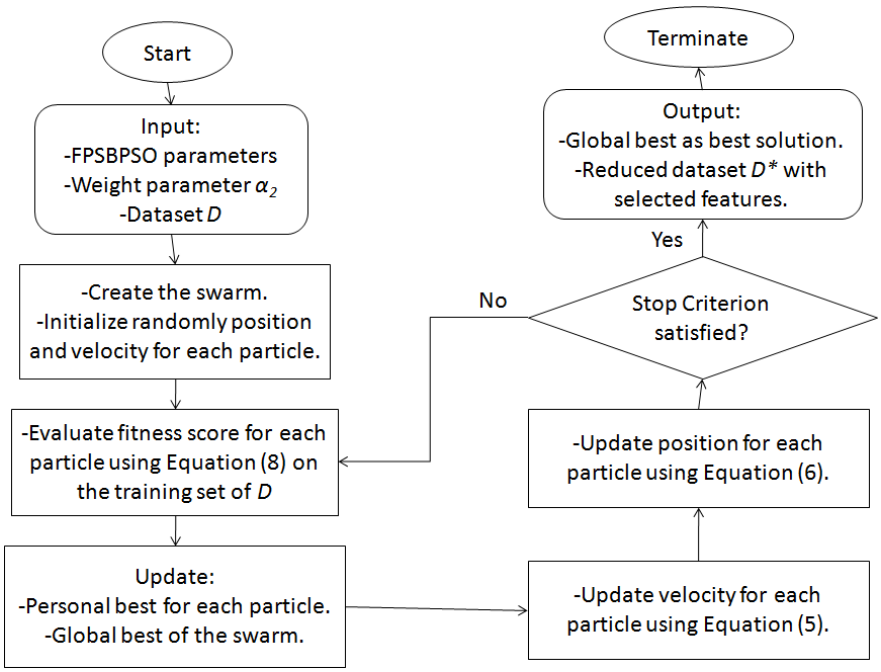
$R_2 = \frac{1}{|S|} \sum_{x \in X} IG(x|\{X/x\})$ .

*X* ve *C*, Denklem 4.5'te tanımlandığı gibidir. $D_2$, seçilen özelliklerle ilgili bilgi verilen sınıf etiketlerindeki bilgi kazancını (entropi) hesaplayarak, seçilen özelliklerle sınıf etiketleri arasındaki ilişkiyi gösterir. $R_2$, seçilen tüm özelliklerin eklem entropisini ölçerek, seçilen özellik alt kümesinde bulunan fazlalığı değerlendirir. $Fitness_2$, fazlalığı en aza indiren ($R_2$) ve eşzamanlı olarak alaka düzeyini ($D_2$) en üst seviyeye çıkartan bir maksimizasyon fitness işlevidir. Buna ek olarak, $\alpha_1$ ve $\alpha_2$, [0,1] arasında değişen sabit değerler olan ağırlık

parametreleridir. Bu parametreler, önerilen yöntemlerin performansını iyileştirmek için seçilen özelliklerin alaka düzeyinin ve fazlalığının önemini kontrol etmek için kullanılır. Deneysel sonuçlar bu parametrelerin uygun değerlerinin 0.8 veya 0.9 olduğunu göstermiştir. İlk yöntem daha küçük bir özellik alt kümesi üretti, ikinci yöntem de özelliklerin sayısını önemli ölçüde düşürdü ve daha yüksek sınıflandırma doğruluğu elde etti (Cervante vd., 2012). Bu nedenle, ikinci yöntemi özellik seçim algoritmamızın alt bölümde 4.3.1.3 temelini seçtik.

#### 4.3.1.3. FPSBPSO-E algoritması

FPSBPSO ve Entropy (FPSBPSO-E) kullanarak tek objektif bir filtre tabanlı özellik seçim algoritması sunuyoruz. Önceki alt bölümde açıklanan ikinci yönteme benzer şekilde çalışır, ancak büyük bir farkla. Geleneksel BPSO kullanmak yerine, parçacığın konum ve hız vektörlerini güncellemek için FPSBPSO algoritmasını kullandık. Bu önemli değişiklik erken yakınsamayı önler ve özellik seçim sürecinde genel performansı geliştirir. Şekil 4.1, FPSBPSO-E algoritmasının akış şemasını göstermektedir.



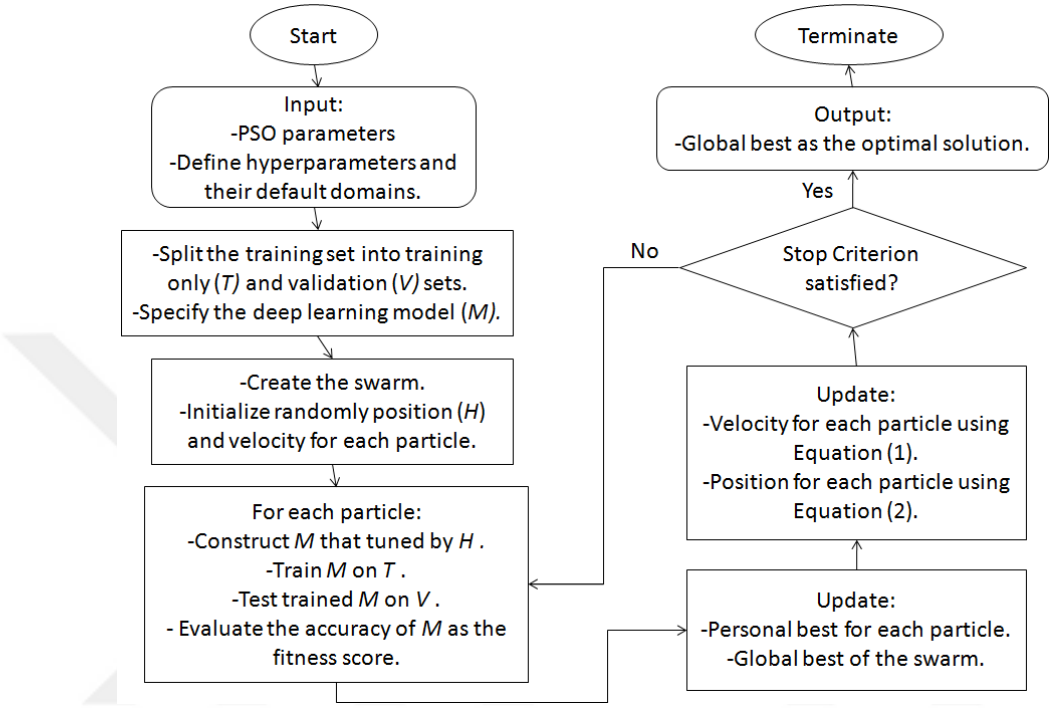

Şekil 4.1. FPSBPSO-E algoritmasının akış şemasıdır

İlk olarak, algoritma belirli sayıda partikül içeren bir sürüyü oluşturur. Her parçacık, uzunluğunun veri kümesindeki tüm kullanılabilir özelliklerin boyutuna eşit olduğu bir ikili dize ile temsil edilir. İkilik dizgide, her bitin değeri karşılık gelen özelliğin seçilip seçilmediğini gösterir, yani "1" değeri, özelliğin seçildiğini ve aksi takdirde "0" anlamına gelir. Daha sonra, algoritma her parçacığın konumunu ve hızını rastgele başlatır. Bundan sonra, algoritma aşağıdaki adımları uygular: verilen veri setinin eğitim setinde Denklem 4.6 kullanarak her bir parçacık için uygunluk puanını hesaplar. Ayrıca, her parçacık için kişisel en iyi vektörü ve ayrıca sürünün global en iyi vektörünü günceller. Daha sonra, her parçacık için, parçacıkların konum vektörünün her bir bitini Denklem 4.4'e göre güncellemekle birlikte, parçacıkların hız vektörünün her bitini Denklem 4.3'e göre günceller. Ayrıca, durma kriterini karşılayıp karşılamadığını kontrol eder. İki farklı durma kriteri belirledik, ilki önceden tanımlanmış yineleme sayısına ulaşıyor, ikincisi ise küresel en iyi vektörün zindelik puanının gelişimi önceden belirlenmiş eşikten daha az ($\mathcal{E}$). Bu koşullardan herhangi biri karşılanırsa, algoritma en iyi çözüm olarak global en iyi vektörü döndürür, yalnızca seçilen özellikleri koruyarak ve diğerlerini kaldırarak orijinal veri kümesini azaltır ve sonlandırır. Aksi takdirde, algoritma bir sonraki yinelemeye başlar ve durma kriterine ulaşılana kadar aynı adımları tekrarlar.

### 4.3.2. Hiperparametere seçimi

Her ne kadar derin öğrenme modelleri karmaşıklıklarından dolayı eğitim süresini uzatabilse de, performans açısından geleneksel makine öğrenme algoritmalarından daha iyi performans gösterirler. Bununla birlikte, performansları, hiperparametrelerin seçilen değerlerine son derece dayanmaktadır. Bu nedenle, bu derin öğrenme modellerinden birini kullanırken kritik olan endişe, hiperparametrelerini nasıl doğru şekilde ayarlayacağıdır. Son zamanlarda, Elmasry vd. (2018), hiperparametre seçimi için yeni bir PSO tabanlı algoritma önermiştir. Bu çalışma, önerilen algoritmanın verilen herhangi bir derin öğrenme modeli için tutarlı ve esnek olması nedeniyle dikkat çekmiştir. Önceki PSO tabanlı algoritma alt bölüm 3.3.2'de önerdiğimiz şeydir.

Şekil 4.2, hiperparametre seçimi için PSO tabanlı algoritma sürecini özetlemektedir. Belirli bir veri setinin eğitim setinin, sadece eğitim için 66% ve doğrulama için 34% oranında bir tutma örneklemesi kullanılarak iki ayrı bölüme ayrıldığını söylemek gerekir.



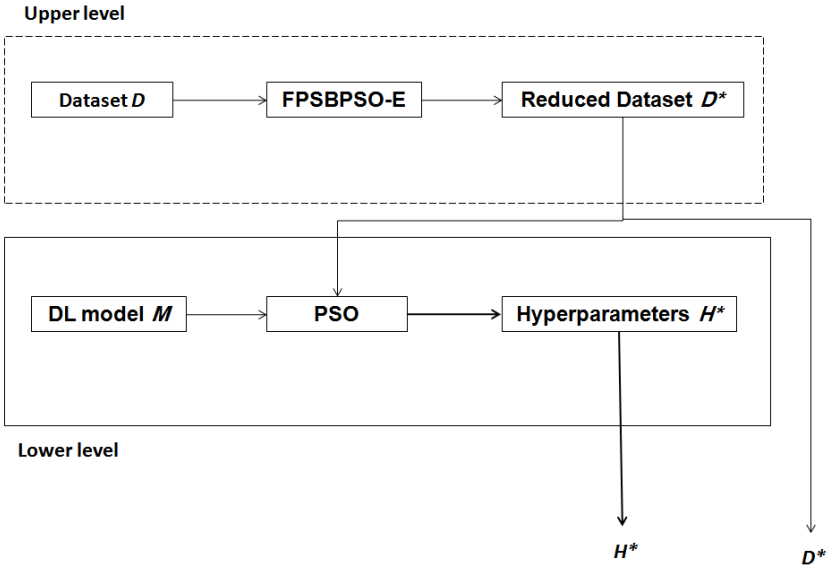

Şekil 4.2. Hiperparametere seçimi için PSO dayalı algoritmanın akış şemasıdır

### 4.3.3. Çift PSO tabanlı algoritma

Çift PSO tabanlı algoritma, hiyerarşik çok amaçlı bir optimizasyon algoritmasıdır. Yukarıdan aşağıya doğru bir algoritma iki seviyeden oluşur. FPSBPSO-E algoritmasını kullanan özellik seçimi için üst seviye, alt bölüm 4.3.2'de açıklanan PSO tabanlı algoritmayı kullanan hiperparametre seçimi için alt seviye. Kullanıcı, gerekli tüm işletim parametrelerini iki seviyeye ayrı ayrı girmekten sorumludur.

Daha sonra, çift PSO tabanlı üst seviyede başlar ve bir dizi özellik (*D*) 'nin tamamını içeren orijinal veri setini bir girdi olarak alır. FPSBPSO-E algoritması çıktı olarak yalnızca seçilen özellik alt kümesine (*D**) sahip olan azaltılmış veri kümesini üretir. Ardından, çift PSO tabanlı algoritma daha düşük seviyeye iner

ve girdi olarak *D** kopyasıyla birlikte derin öğrenme modeli (*M*) türünü alır. PSO tabanlı algoritma, *M* için en uygun hiperparametre vektörünü (*H**) bir çıkış olarak bulur. Son olarak, çift PSO tabanlı algoritma hem *D** hem de *H** çıkışlarını sonra sonlandırır. Şekil 4.3, önerilen algoritmanın mekanizmasını göstermektedir.




Şekil 4.3. Çift PSO tabanlı algoritma mekanizması

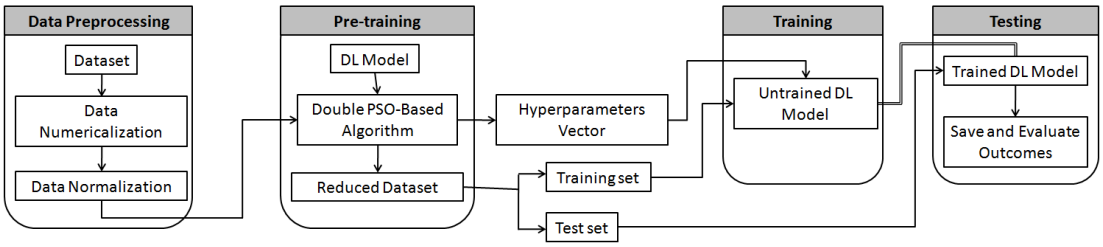
## 4.4. Deneysel Kurulum

Her biri Bölüm 4.2'te açıklanan veri setlerinden birinde iki temel deneysel deney yaptık. Aslında, her bir deney iki defa, önce ikili bir sınıflandırma için, sonra çok sınıflı bir sınıflandırma görevi için yapılır. Üstelik, her deneyde, üç derin öğrenme modelinin ağa saldırı tespitindeki performansı belirli veri setinde doğrulanmıştır. Gelecek alt bölümlerde, deneysel deneylerimizi yürütme metodolojisini ve her modelin tanımını açıklıyoruz.

### 4.4.1. Metodoloji

Metodolojimiz açık ve anlaşılır olacak şekilde tasarlanmıştır. Ön işleme, ön eğitim, eğitim ve test aşamalarından oluşan arka arkaya dört aşamadan oluşur.

Bu aşamaların detayları aşağıdaki alt bölümlerde sunulmuştur. Şekil 4.4, deney metodolojimizin diyagramını göstermektedir.




Şekil 4.4. Deneylerin yöntem şeması

#### 4.4.1.1. Veri ön işleme

İlk olarak, veri sayısallaştırma ve normalizasyon işlemlerini uygulayarak belirli veri setinde veri ön işleme uyguladık. Deneylerimizin ilk kısıtı; Makine öğrenim modellerinin çoğu, eğitim ve test için sadece sayısal değerlerle çalışabilir. Bu nedenle, sayısal olmayan tüm değerleri bir veri sayısallaştırma gerçekleştirerek sayısal değerlere dönüştürmek gerekir. Aslında, literatürde veri sayısallaştırmasını gerçekleştirmenin iki yöntemi vardır. Birincisi, nominal özelliğin her türüne farklı bir ikili vektör veren "tek-sıcak kodlama" olarak adlandırılır. Örneğin, NSL-KDD veri kümesinde, 'protokol_tipi', 'hizmet' ve 'bayrak' özellikleri olmak üzere üç nominal özellik vardır. 'protokol_tipi' özelliği için, üç tür nitelik vardır: 'tcp', 'udp' ve 'icmp' ve sayısal değerleri ikili vektörler olarak kodlanır (1,0,0), (0,1,0) ve (0,0,1). Benzer şekilde, 'hizmet' özelliği 70 tür özniteliğe sahiptir ve 'bayrak' özelliği 11 tür özniteliğe sahiptir. Bu şekilde devam ederek, 41 boyutlu özellikler dönüşümden sonra 122 boyutlu özelliklerle eşleşir. İkinci yöntem bu çalışmada kullandığımız yöntemdir, her nominal özellik için değerleri alfabetik olarak sıralanır. Bundan sonra, sıralanan nominal değerler, [1, listenin uzunluğu] arasında değişen her değişkene özel değerler atanarak sayısal değerlere dönüştürülür (örneğin, 'icmp' = 1, 'tcp' = 2 ve 'udp' = 3).

Bir-sıcak kodlama yöntemiyle karşılaştırıldığında, ikinci yöntemi kullanmayı tercih ediyoruz, çünkü birçok avantaja sahip. İkinci yöntem, özellik sayısını

artırmaz, çünkü dönüştürülen her nominal özellik, bir değer ile temsil edilir. Buna karşılık, bir sıcak kodlama yöntemi, özelliklerin sayısını arttırmaktadır, çünkü dönüştürülen her nominal özellik, uzunluğu, nominal özellik değerlerinin sayısına bağlı olan bir ikili vektör ile temsil edilmektedir. Sonuç olarak, ikinci yöntemi kullanırken modellerin mimarisi, bir-sıcak kodlama yöntemini kullanmaktan daha basit olacaktır çünkü modelin girişleri daha az olacaktır. Bu nedenle, ikinci yöntem modeli eğitmek ve test etmek için gereken zamanı azaltır.

Daha sonra, veri setindeki tüm sayısal özellikler (dönüştürülmüş nominal özellikler dahil), denklem 4.7'ye sahip Min-Max dönüşümü kullanılarak doğrusal olarak [0,1] olarak eşlendiğinde, veri normalleştirme işlemi gerçekleştirilir.

$$x_i = \frac{x_i - Min}{Max - Min} \tag{4.7}$$

$x_i$, örneklemenin sayısal özellik değeri ise, Min ve Max, her sayısal özelliğin sırasıyla minimum ve maksimum değerleridir.

#### 4.4.1.2. Ön eğitim

Ön eğitim aşamada, önerilen çift PSO tabanlı algoritmayı kullanmak için zorunludur. Bu nedenle, önceki aşamadan elde edilen önceden işlenmiş veri seti, önerilen algoritmaya girdi olarak derin öğrenme modeli ile birlikte geçirilir. Çift PSO tabanlı algoritma bittiğinde, kullanılan modelin optimal hiperparametre değerlerini ve ayrıca belirli veri setinin azaltılmış versiyonunu verir. Bu çıktılar daha ileri işlemler için bir sonraki aşamaya iletilir. Çizelge 4.2, önerilen algoritmanın ana çalışma parametrelerinin değerlerini göstermektedir. Seçilen değerler, önceden tanımlanmış etki alanındaki her parametre için bir ızgara araması gerçekleştirilerek elde edilir. Buna ek olarak, birçok teorik ve ampirik önceki çalışmada etki alanları önerilmektedir (Shi ve Eberhart, 1998; Clerc ve Kennedy, 2002).

Çizelge 4.2. Çift PSO tabanlı algoritmanın ana çalışma parametreleri

| **Parametre** | **Alan** | **Seçilen değer** | |
|---|---|---|---|
| | | Özellik seçimi | Hiperparametere seçimi |
| | | FPSBPSO-E | Sürekli PSO |
| Sürü büyüklüğü | [5,40] | 20 | 40 |
| En az hızı ($V_{min}$) | {0,1} | - | 0 |
| Maksimum hızı ($V_{max}$) | {0,1} | - | 1 |
| Hızlanma katsayıları ($C_1$, $C_2$) | [1,5] | - | 1.43 |
| Atalet ağırlık sabit ($W$) | [0.4,0.9] | - | 0.69 |
| Tekrarlamalar maksimum sayısı | [30,50] | 30 | 50 |
| Eşik Durdurma ($\varepsilon$) | [0.001,0.0001] | 0.0001 | 0.001 |
| Ücretsiz parametresi ($mr$) | [0,1] | 0.01 | - |
| Ağırlık parametresi ($\alpha_2$) | [0,1] | 0.9 | - |

Önerilen çift PSO tabanlı algoritmanın üst seviyesindeki FPSBPSO-E algoritmasını kullanan özellik seçim süreci ile ilgili olarak, her bir veri seti için seçilen özellik alt kümesi ve özellik azaltma oranı Çizelge 4.3'te listelenmiştir. Özellik azaltma oranı, tüm özellik kümesinden kaldırılan özelliklerin sayısı hakkında bilgi verir. Denklem 4.8'e göre hesaplanabilir.

$$Feature\ Reduction\ Rate = 1 - \frac{number\ of\ selected\ features}{number\ of\ all\ features} \tag{4.8}$$

Çizelge 4.3. Her veri seti için özellik seçimi sonuçları

| **Veri seti** | **Özelliklerin sayısı** | **Seçilen özellikler** | **Seçilen özelliklerin sayısı** | **Seçilen özelliklerin yüzdesi (%)** | **Özellik azaltma oranı (%)** |
|---|---|---|---|---|---|
| NSL-KDD | 41 | 1,2,3,5,6,23,25,30,32,37 | 10 | 24.39 | 75.61 |
| CICIDS2017 | 80 | 8,11,15,18,20,23,24,26,28,31,33,37,43 ,44,51,53,54,59,70,73,74,77,80 | 23 | 28.75 | 71.25 |

Çizelge 4.4, ilgili veri setlerinde derin öğrenme modelleri için ilişkili olan küresel hiperparametrelerin değerlerini göstermektedir. Bu bulgular, önerilen çift PSO tabanlı algoritmanın alt seviyesindeki PSO tabanlı algoritma kullanılarak hiperparametre seçim işleminin bitmesinden sonra elde edilir. Tabaka bazlı hiperparametreler ve her derin öğrenme modelinin kısa bir açıklaması alt bölüm 4.4.2'de sunulmaktadır.

Çizelge 4.4. Ortaya çıkan küresel hiperparameterelerin değerleri

| Hiperparametere | NSL-KDD | | | CICIDS2017 | | |
|---|---|---|---|---|---|---|
| | DNN | LSTM-RNN | DBN | DNN | LSTM-RNN | DBN |
| Öğrenme oranı | 0.4 | 0.2 | 0.09 | 0.1 | 0.06 | 0.01 |
| Çürüme | 0.01 | 0.01 | 0.008 | 0.005 | 0.003 | 0.001 |
| Moment | 0.71 | 0.55 | 0.2 | 0.3 | 0.14 | 0.1 |
| Epok sayısı | 55 | 30 | 20 | 15 | 10 | 5 |
| Parti boyutu | 200 | 350 | 400 | 450 | 550 | 550 |
| Doktoru | SGD | Adagrad | Adamax | RMS-prop | Nadam | Adam |
| Başlatma işlevi | Normal | Lecun Uniform | He normal | Uniform | Zero | He uniform |
| Bırakma oranı | 0.5 | 0.4 | - | 0.25 | 0.1 | - |

#### 4.4.1.3. Eğitim ve test

Derin öğrenme modelini yaptık ve en uygun hipermetrelerle ayarladık. Ortaya çıkan model, azaltılmış veri setinin tam eğitim seti üzerinde eğitilmiştir. Daha sonra, eğitimli model azaltılmış veri setinin test setinde test edilir. Son olarak, sınıflandırma sonuçları daha sonra tekrar işlenmek üzere saklanır.

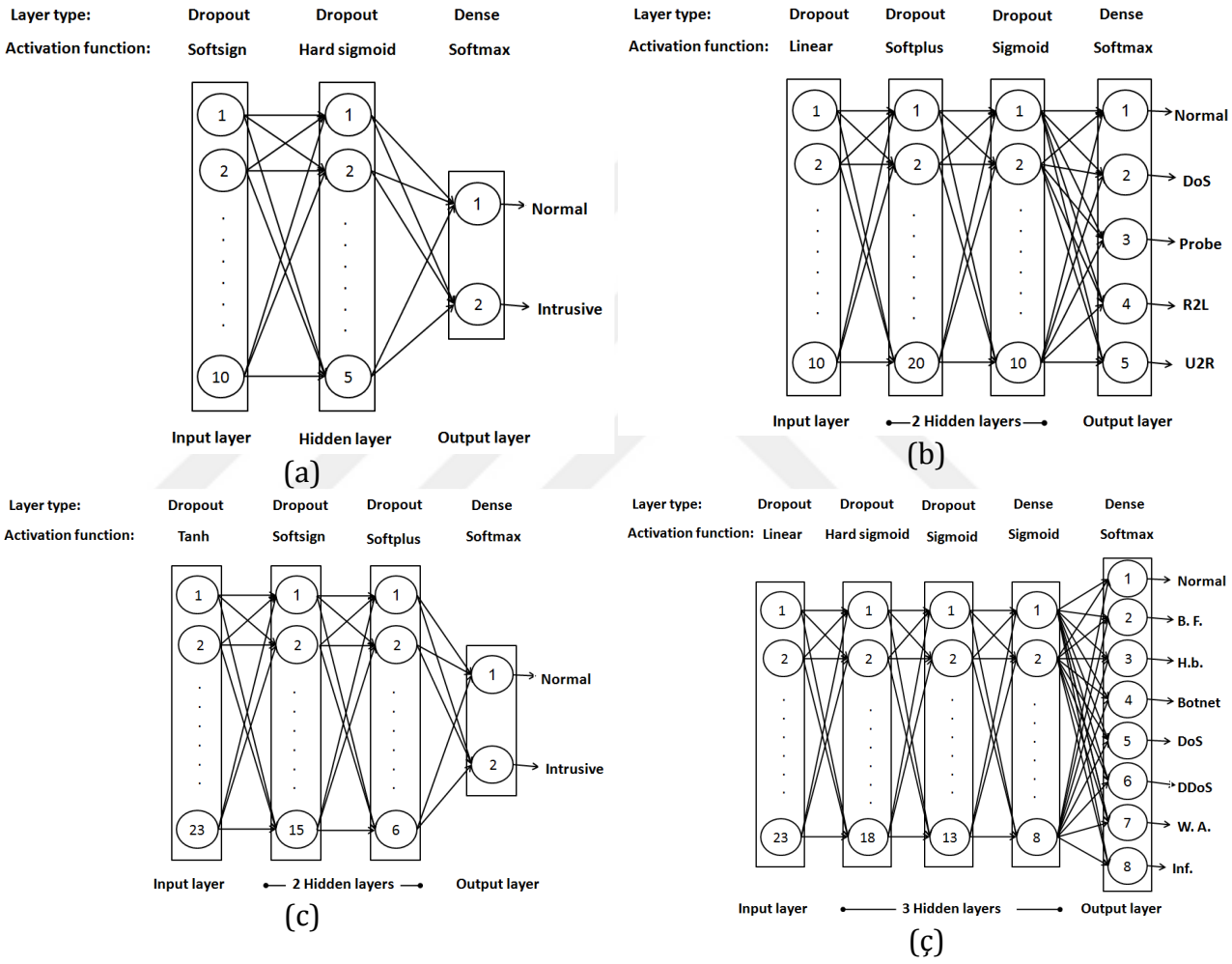
### 4.4.2. Modeller

Her veri setinde ağa saldırı tespitini gerçekleştirmek için, DNN, LSTM-RNN ve DBN olmak üzere üç iyi bilinen derin öğrenme modelini kullandık. Aslında, bu modellerin seçilmesi için diğer derin öğrenme tekniklerinden çok nedenler var. Birincisi, birçok inceleme makalesi ağa saldırı tespit problemini çözmedeki başarılarına işaret etmektedir (Aminanto ve Kim, 2016; Vani, 2017). Ayrıca, saldırı tespiti gibi statik sınıflandırma görevlerinde de yaygındır. Son olarak, yukarıda bahsedilen modeller literatürde yaygın olarak kullanılmaktadır, bu nedenle sonuçlarımız kolayca karşılaştırılabilir.

#### 4.4.2.1. DNN

DNN yalnızca ANN'i değil, birçok derin gizli katmana sahiptir (Hinton vd., 2012). Genellikle, DNN bir giriş katmanı, bir veya daha fazla gizli katman ve bir çıkış

katmanından oluşur. Gizli katmanlar, DNN'deki en önemli işi yapmış sayılıyordu. DNN'deki her katman, tez nöronlarının katmandan katmana tam olarak bağlanacağı şekilde bir veya daha fazla yapay nörondan oluşur. Ayrıca, bilgi DNN yoluyla ileri beslenerek işlenir ve yayılır, yani giriş katmanından çıkış katmanlarına gizli katmanlar aracılığıyla iletilir. Şekil 4.5, ikili ve çok sınıflı sınıflandırmalarla ilgili her veri seti için ortaya çıkan DNN mimarisini sunmaktadır.



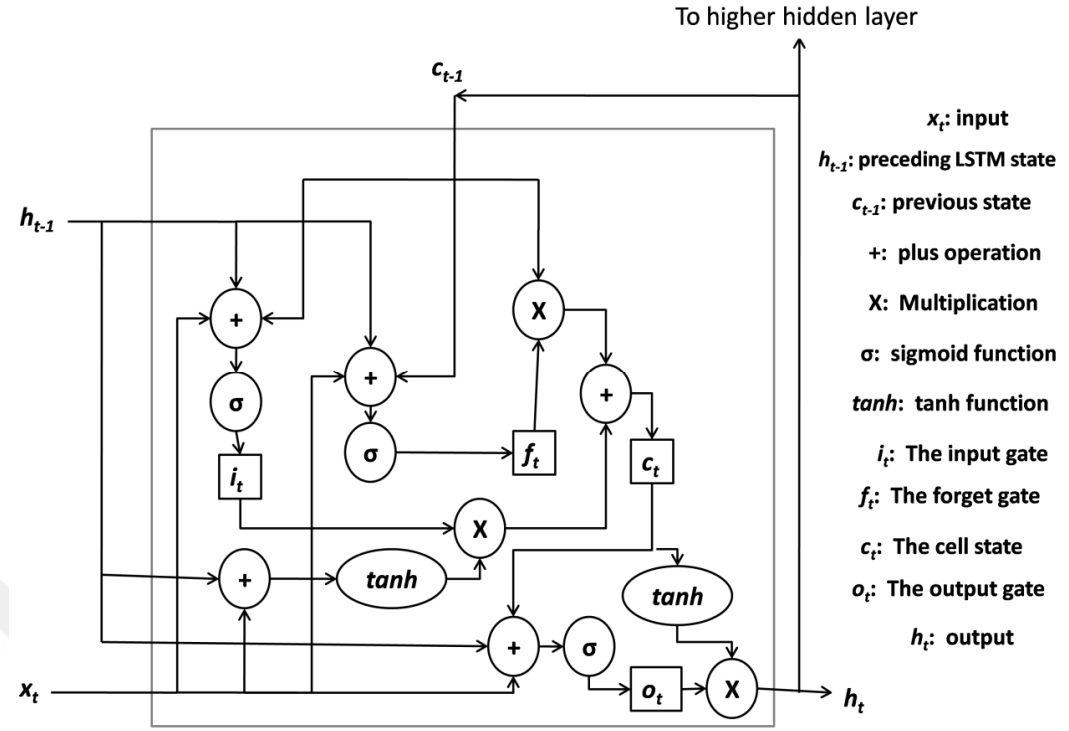

Şekil 4.5. İlgili olarak DNN mimari a) NSL-KDD (ikili), b) NSL-KDD (çok sınıflı), c) CICIDS2017 (ikili), ç) CICIDS2017 (çok sınıflı)

#### 4.4.2.2. RNN

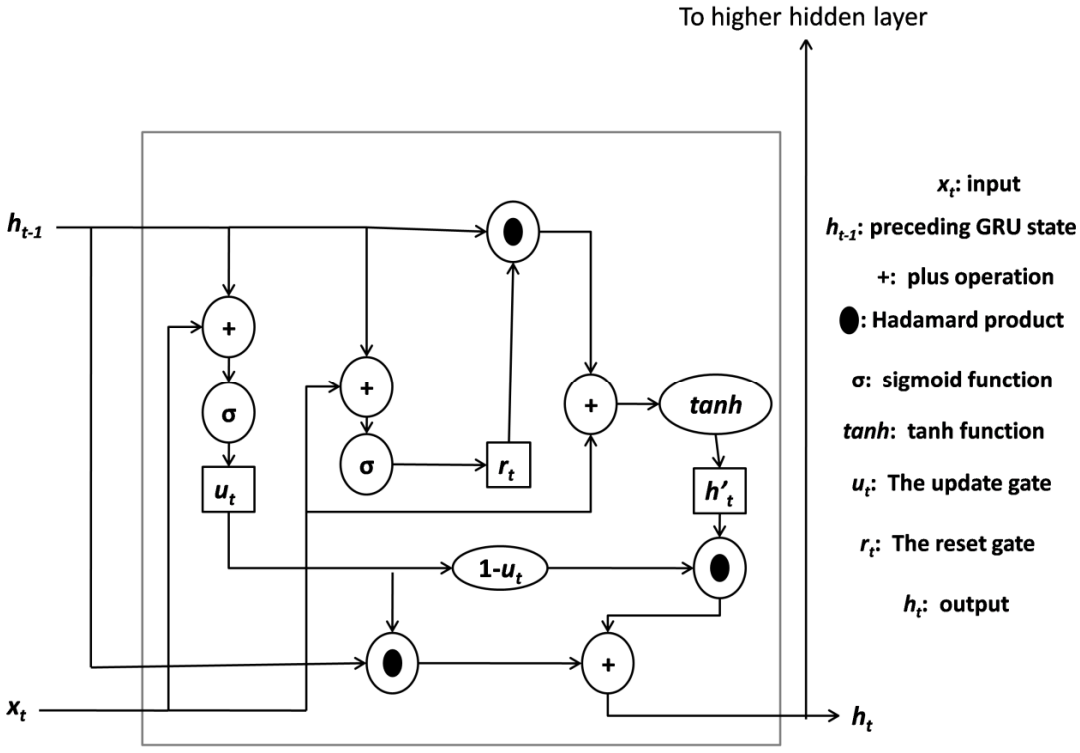
Geleneksel ANN'den farklı olarak, RNN'nin gizli katmanlarının herhangi birindeki her bir nöron, çıktısından kendisine, kendisinin tekrarlayan ve aynı

gizli katmandaki bitişik nöronuna ek bağlantılara sahiptir. Bu nedenle, ağda, gizli katmanları pratik olarak bütün ağın bir depolama birimi olarak yapan bilgiler dolaşmaktadır. Bununla birlikte, geleneksel RNN'nin yapısı, gradyan kayması ve patlaması olarak bilinen içsel dezavantajdan muzdariptir (Bengio vd., 1994). Bu ciddi problem, sıklıkla RNN, geri yayılım metodu kullanılarak eğitildiğinde ortaya çıkar. RNN kullanımını yalnızca kısa süreli hafıza görevlerinde sınırlar. Bu sorunu çözmek için, Hochreiter ve Schmidhuber (1997) tarafından bir LSTM yapısı tanıtılmıştır. LSTM; giriş kapısı, kendinden tekrarlamalı bağlantıya sahip nöron, unutma kapısı ve çıkış kapısı olmak üzere dört bölümden oluşan bir bellek hücresi kullanır. Kendini yineleyen bir nöronu kullanmanın temel amacı bilgiyi kaydetmek olsa da, üç kapı kullanmanın amacı, bellek hücresinden ya da içine hücre akışını kontrol etmektir. Şekil 4.6 (a), bir LSTM bellek hücresinin yapısını gösterir. LSTM-RNN modelinin, RNN'nin gizli katmanlarındaki her bir nöronu bir LSTM bellek hücresine değiştirerek kolayca elde edilebileceği rapor edilmiştir (Kim vd., 2016).

Alternatif olarak, geleneksel RNN'de gradyanın ortadan kalkması ve patlaması, Cho vd. (2014), Geçitli Tekrarlamalı Ünite (GRU) olarak adlandırılan yeni bir karmaşık geçiş mekanizması önermiştir. LSTM ve GRU arasındaki ana ayrım, GRU'nun LSTM'de üç yerine sadece iki kapısına (güncelleme ve sıfırlama) sahip olmasıdır. Bu nedenle, GRU gizli içeriği yalnızca kontrol etmeden gösterir. Şekil 4.6 (b), bir GRU biriminin yapısını göstermektedir. Sezgisel olarak, güncelleme kapısının GRU'daki işlevi, önceki bilgilerin ne kadarının saklanması gerektiğini belirlemektir. Öte yandan, sıfırlama kapısı önceki bilgilerin ne kadarının unutulacağına karar vermekten sorumludur. Aslında, GRU'nun performansı genel olarak LSTM ile aynıdır, ancak LSTM'ler büyük veri setlerinde GRU'dan daha iyi performans gösterir (Chung vd., 2014). Şekil 4.7, ikili ve çok sınıflı sınıflandırmalarla ilgili her veri seti için ortaya çıkan LSTM-RNN mimarisini sunmaktadır.

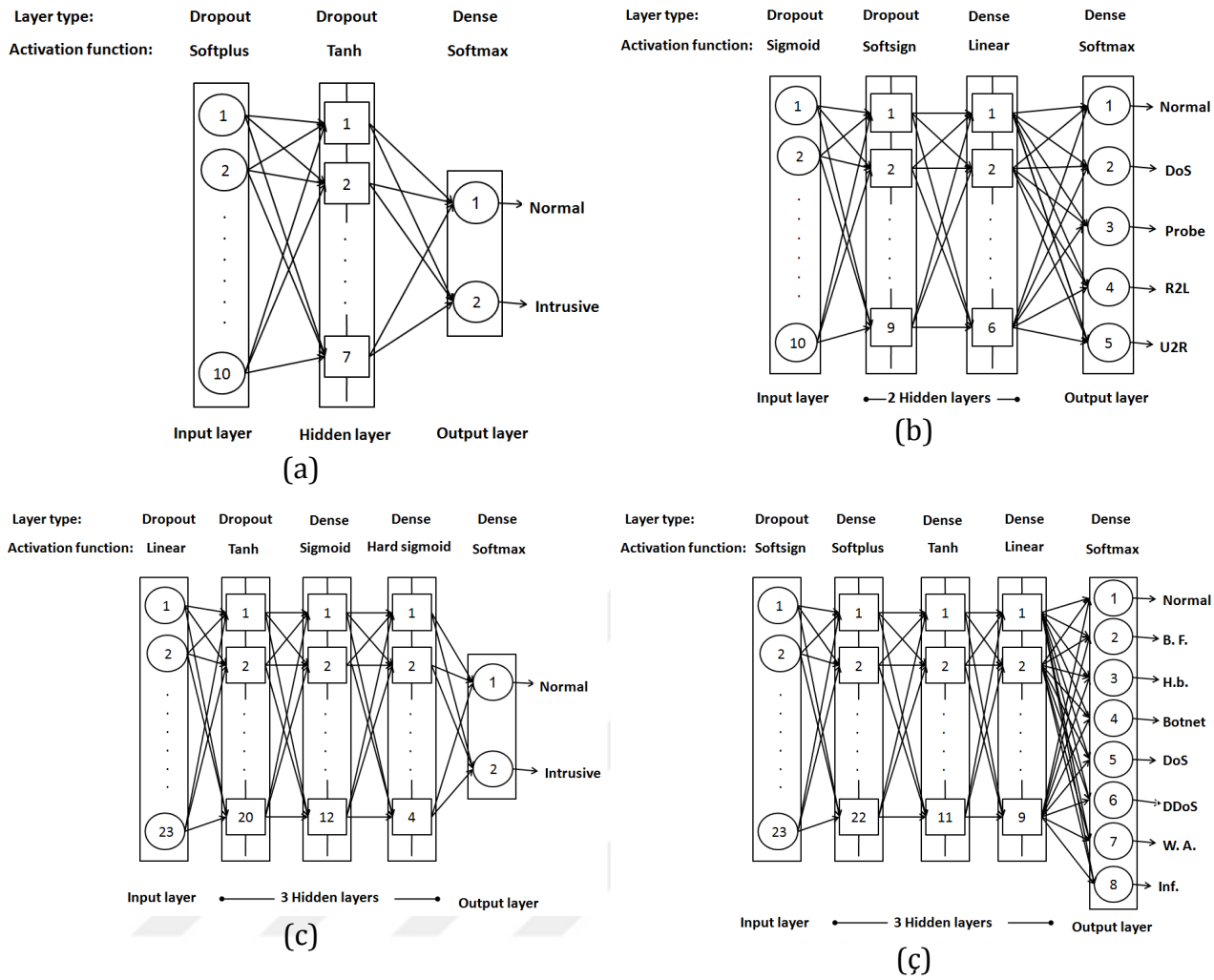



(a)




(b)

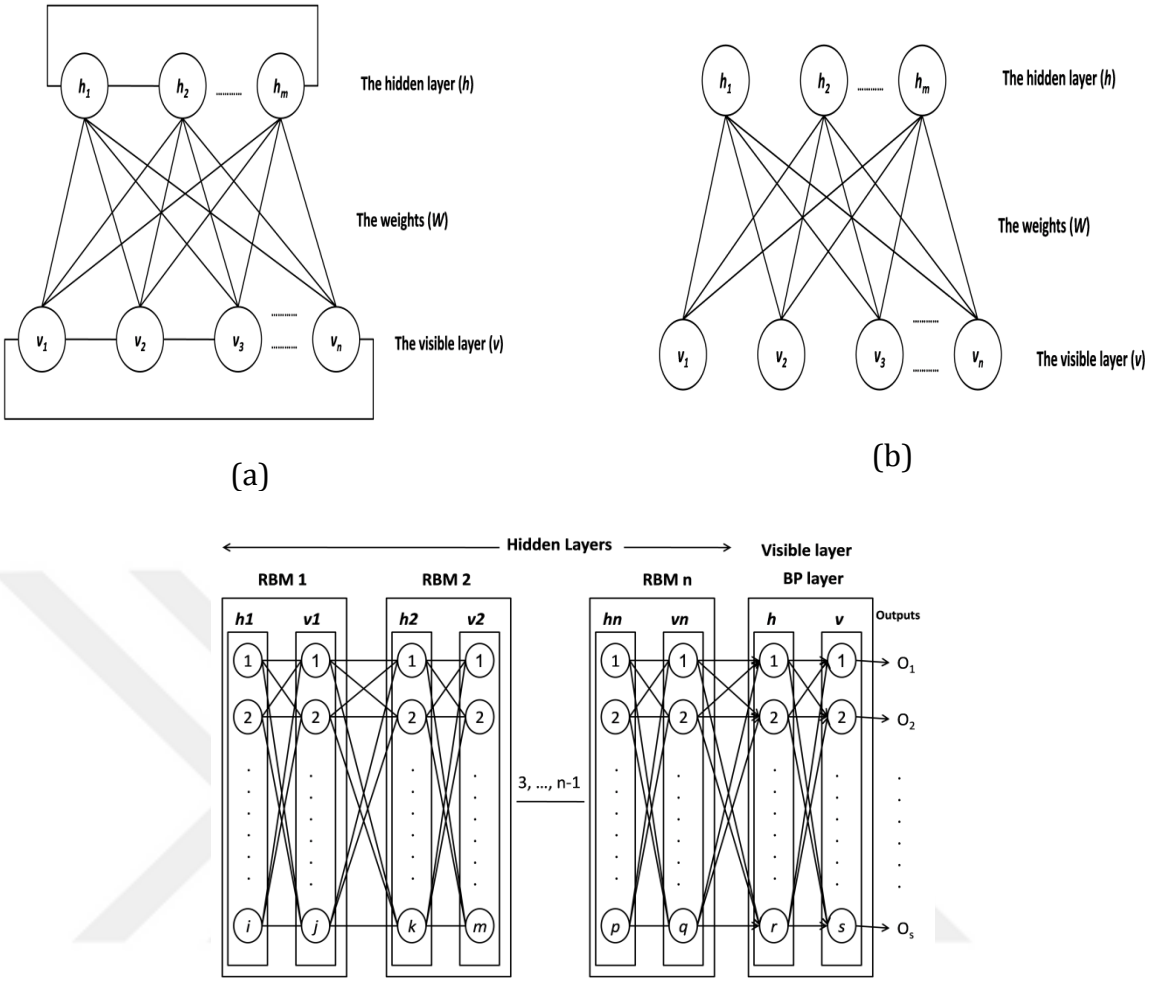
Şekil 4.6. Kapılı RNN temel yapısı a) LSTM hücresi, b) GRU birimi




Şekil 4.7. İlgili olarak LSTM-RNN mimari a) NSL-KDD (ikili), b) NSL-KDD (çok sınıflı), c) CICIDS2017 (ikili), ç) CICIDS2017 (çok sınıflı)

#### 4.4.2.3. DBN

Boltzmann Makinesi (BM) sadece iki basamaklı katmandan oluşan, görünür katman (*v*) ve gizli katmandan (*h*) oluşan, enerji bazlı bir sinir ağı modelidir (Aminanto ve Kim, 2016). Şekil 4.8 (a), bir BM'nin temel yapısını gösterir. BM, log-linear Markov Rastgele Alanının (MRF) belirli bir şeklidir ve tüm nöronları ikilidir, yani ya 0 ya da 1'i verir. BM'nin yapısına göre, hem aynı katmandaki nöronlar arasında hem de nöronlar görünür ve gizli katmanlar arasındaki yönlendirilmemiş bağlantılarla bağlanır.




Şekil 4.8. Tipik yapısı a) BM, b) RBM, c) DBN

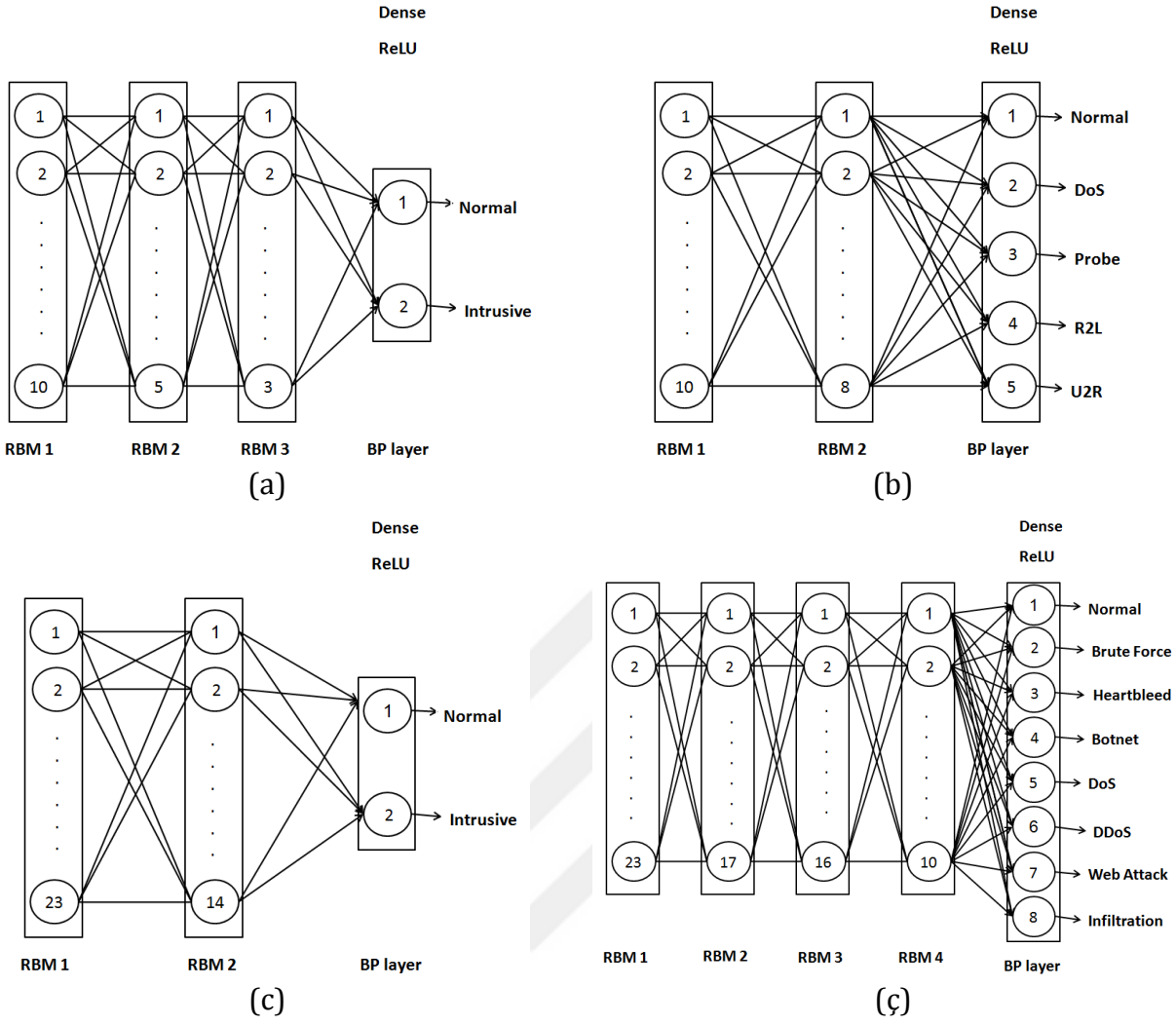
Son zamanlarda, BM'nin özelleştirilmiş bir sürümü Kısıtlanmış Boltzmann Makinesi (RBM) olarak tanıtılmaktadır (Salakhutdinov ve Hinton, 2009). RBM, bir tür stokastik üretken öğrenme modeli olarak kabul edilir. Görünür-görünür ve gizli-gizli bağlantıları olmayan bir BM'den başka bir şey değildir, yani tüm bağlantı yalnızca görünür katman ile gizli katman arasındadır. Bir RBM'nin temel yapısı Şekil 4.8 (b) 'de gösterilmektedir.

Hinton vd. (2006), DBN adı verilen bir üretici olasılıksal sinir ağı önermiştir. Şekil 4.8 (c) DBN'nin tipik yapısını göstermektedir. Yapısal açıdan bakıldığında, DBN birkaç istiflenmiş RBM'yi bir Arka Yayılma (BP) sinir ağı katmanına birleştiren derin bir sınıflandırıcıdır (Rumelhart vd., 1986). Bu sırada, yığılmış

RBM'ler DBN'nin gizli katmanları olarak kabul edilir, BP katmanı görünür katmandır. Buna ek olarak, DBN'deki tüm gizli katmanlar arasındaki bağlantı, yönlendirilmemiş bağlantılardır. Buna karşılık, son RBM, yönlendirilmiş ağırlıklar ile görünür tabakaya bağlanır.

Açık literatürde, konvansiyonel DBN, antrenman öncesi ve ince ayar yapan iki sıralı prosedüre sahiptir. Antrenman öncesi prosedür, tüm gizli katmanları (RBM'ler) katman bazında eğiterek gerçekleştirilir, yani, alt katmanın girişi olarak kullanılan yüksek katmanın çıktısı ile bir seferde yalnızca bir kat çalışır. Bunu başarmak için, etiketlenmemiş bir eğitim verisi ile birlikte açgözlü bir katman şeklinde denetimsiz bir eğitim algoritması kullanılır (Hinton, 2002). Daha sonra, tüm DBN'nin parametreleri, etiketli eğitim verileriyle birlikte BP öğrenme algoritması kullanılarak ince ayarlanmıştır. Son zamanlarda, DBN büyük ilgi gördü ve birçok veri madenciliği uygulamasında kullandı. Bu uygulamalardan bazıları denetlenmemiş bir özellik çıkarma ve özellik azaltma modeli olarak DBN kullanıyor. Bu durumda, DBN yalnızca herhangi bir BP katmanı olmayan istiflenmiş RBM'lere sahiptir. Öte yandan, bazı uygulamalar sınıflandırma görevleri için DBN kullanmaktadır ve bu durumda, DBN bir BP katmanı ile birlikte birkaç istiflenmiş RBM'den oluşmaktadır (Salama vd., 2011). Bu çalışmada, DBN'yi her veri setinde bir sınıflandırıcı olarak kullandık. Şekil 4.9, ikili ve çok sınıflı sınıflandırmalarla ilgili her veri seti için DBN'nin sonuç mimarilerini göstermektedir.

Dahası, *e*'nin RBM katmanlarının sayısını ifade ettiği DBN'e terimi, bir DBN modelinin yapısını tarif etmek için kullanılmaktadır. Elde ettiğimiz sonuçlara göre; $DBN^3$ (10-5-3-2), $DBN^2$ (10-8-5), $DBN^2$ (23-14-2) ve $DBN^4$ (23-17-16-10-8) NSL-KDD (ikili), NSL-KDD (çok sınıflı), CICIDS2017 (ikili) ve CICIDS2017 (çok sınıflı), sırasıyla. İterasyon sayısıyla ilgili olarak, NSL-KDD (ikili), NSL-KDD (çok sınıflı), CICIDS2017 (ikili) ve CICIDS2017 (çok sınıflı) DBN modelleri için 250, 350, 450 ve 500 iterasyon sayısı, sırasıyla da vardır.




Şekil 4.9. İlgili olarak DBN mimari a) NSL-KDD (ikili), b) NSL-KDD (çok sınıflı), c) CICIDS2017 (ikili), ç) CICIDS2017 (çok sınıflı)

## 4.5. Değerlendirme Ölçütleri

Bu çalışmada, hem ikili hem de çok sınıflı sınıflandırma görevleri için deneylerimizi gerçekleştirdik. Bölüm 4.2'te belirtildiği gibi, her veri setinde normal (negatif) ve çeşitli saldırı (pozitif) örneklerinin bir karışımı vardır. İkili sınıflandırmada, normal ve saldırı olmak üzere sadece iki etiketli sınıf vardır. Saldırı türüne bakılmaksızın, saldırı sınıfı, bir sınıftaki tüm saldırı örneklerini içerir. Öte yandan, çok sınıflı sınıflandırma sadece kötü amaçlı bir bağlantıyı tespit etmekle kalmayıp, aynı zamanda doğru türü de tayin etmeyi amaçlamaktadır. Sonuç olarak, etiketli sınıfların sayısı veri kümesinden diğerine (normal sınıf + n saldırı sınıfı) değişir. Aşağıdaki alt bölümlerde, ikili ve çok

sınıflı sınıflandırmaları ve bunlarla birlikte hangi değerlendirme ölçümlerinin kullanıldığını tanıtıyoruz.

### 4.5.1. İkili sınıflandırma

Özellikle, herhangi bir sınıflandırma görevinden (TP, TN, FP ve FN) elde edilebilecek dört ana sonuç vardır. Çizelge 4.5, ağ saldırı tespitini ikili sınıflandırma olarak uygularken olası karışıklık matrisini göstermektedir. Daha sonra, dört sonuç, derin öğrenme modelinin belirli bir veri setindeki performansını değerlendirmek için iyi bilinen on beş değerlendirme ölçütünü hesaplamak için kullanılır. Kullanılan metriklerin bazıları, ağ saldırı tespit konusuna odaklanan önceki çalışmalarda yaygın olarak kullanılmaktadır, bu nedenle okuyucuların sonuçları karşılaştırması kolay olacaktır.

Çizelge 4.5. Ağ saldırı tespit ikili sınıflandırma karışıklık matrisi

| **Gerçek sınıf** | **Öngörülen sınıf** | |
|---|---|---|
| | Normal | Saldırı |
| Normal | TN | FP |
| Saldırı | FN | TP |

Değerlendirme ölçütlerinin tanımı ve bunlara karşılık gelen denklemlerin listesi aşağıdaki gibidir:

- Doğruluk, tüm test seti için gerçek sınıflandırmaların oranıdır.
  $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.9)$$
- Kesinlik, sınıflandırıcının kesinliğini, yani, bir saldırı olarak sınıflandırılan test setindeki tüm numunelerden bir saldırı olarak doğru şekilde etiketlenen numunelerin oranını gösterir.
  $$Precision = \frac{TP}{TP+FP} \quad (4.10)$$
- Geri çağırma, sınıflandırıcının eksiksizliğini, yani test setinde bulunan tüm saldırı örnekleri için doğru bir saldırı olarak sınıflandırılan örneklerin oranını gösterir. Aynı zamanda Hit, TPR, DR veya Hassasiyet olarak da adlandırılır.
  $$Recall = \frac{TP}{TP+FN} \quad (4.11)$$

- F1-Puanı, hem Precision ($P$) hem de Recall ($R$) metriklerinin harmonik ortalaması olarak kabul edilir. Aynı zamanda F1 metrik olarak da bilinir.

$$F1\_Score = \frac{2 \times P \times R}{P+R} \quad (4.12)$$

- FAR, test setinde bulunan tüm normal numuneler için bir saldırıya sınıflandırılmış normal numunelerin oranıdır. Ayrıca FPR olarak da adlandırılır.

$$FAR = \frac{FP}{TN+FP} \quad (4.13)$$

- Spesifiklik, test setinde bulunan tüm normal numuneler için doğru tahmin edilen normal numunelerin oranını gösterir. TNR olarak da bilinir.

$$Specificity = \frac{TN}{TN+FP} \quad (4.14)$$

- FNR, geri çağırma işleminin tamamlayıcısıdır, yani test setindeki tüm saldırı örneklerinden hatalı olarak normal bir sınıf olarak sınıflandırılan saldırı örneklerinin oranı hakkında bilgi verir. Aynı zamanda Kayıp oranı denir.

$$FNR = \frac{FN}{TP+FN} \quad (4.15)$$

- Negatif Kesinlik, test setinde normal sınıf olarak sınıflandırılan tüm numuneler üzerinde doğru sınıflandırılmış normal numunelerin oranını gösterir.

$$Negative\ Precision = \frac{TN}{TN+FN} \quad (4.16)$$

- Hata Oranı, tüm test setii için yanlış tahminlerin oranı hakkında bilgi verir.

$$Error\ Rate = \frac{FP+FN}{TP+TN+FP+FN} \quad (4.17)$$

- BDR, ilk olarak Axelsson (1999) tarafından ele alınan Baz Oran Yanılgısı problemine dayanmaktadır. Baz Oran Yanılgısı, Bayesian istatistiklerinin temellerinden biridir. Olasılıklardaki problemleri çözerken, insanlar temel insidans oranını (Baz Oranı) dikkate almazlarsa ortaya çıkar. Geri çağırma metriğinin aksine, BDR, saldırı sınıfının taban oranını göz önünde bulundurarak tüm test kümeleri için doğru sınıflandırılmış saldırı örneklerinin oranını gösterir. Matematiksel olarak, $I$ ve $I^*$ saldırı ve normal davranışları, sırasıyla gösterelim. Ayrıca, $A$ ve $A^*$ 'nın sırasıyla öngörülen saldırı ve normal davranışı göstermesine izin verin. Daha sonra BDR, denklem 4.18'e göre $P(I|A)$ olasılığı olarak hesaplanabilir (Axelsson, 1999).

$$BDR = P(I|A) = \frac{P(I) \times P(A|I)}{P(I) \times P(A|I) + P(I^*) \times P(A|I^*)} \quad (4.18)$$

*P(I)* test setindeki saldırı örneklerinin oranı ise, *P(A|I)* Geri Çağırma, *P(I*)* test setindeki normal örneklerin oranıdır ve *P(A|I*)* FAR'dır.

- BTNR, Baz Oran Hatalılığı sorununa da dayanmaktadır. Öngörülen normal davranışın gerçekten normal bir bağlantıyı göstereceği şekilde tüm test seti için doğru sınıflandırılmış normal örneklerin oranı hakkında bilgi verir (Axelsson, 1999). Matematiksel olarak, $I$ ve $I^*$, sırasıyla saldırı ve normal davranışları gösterelim. Dahası, $A$ ve $A^*$ 'nın sırasıyla öngörülen saldırı ve normal davranışı göstermesine izin verin. Daha sonra BTNR, denklem 4.19'a göre $P(I^*|A^*)$ olasılığı olarak hesaplanabilir (Axelsson, 1999).

$$BTNR = P(I^*|A^*) = \frac{P(I^*)\times P(A^*|I^*)}{P(I^*)\times P(A^*|I^*)+P(I)\times P(A^*|I)} \quad (4.19)$$

  $P(I^*)$ test setindeki normal örneklerin oranı ise, $P(A^*|I^*)$ Spesifiklik, $P(I)$ test setindeki saldırı örneklerinin oranı ve $P(A^*|I)$ FNR'dir.

- Geometrik Ortalama (g-mean), her iki hatanın eşit olduğu kabul edilen Spesifiklik ve Geri Çağırma metriklerini belirli bir eşikte birleştirir. Sınıflandırıcıların dengesizlik veri setindeki performansını değerlendirmek için son derece kullanılmıştır (Zeng ve Gao, 2009). Aslında, iki farklı denklem sahiptir. $G\_mean_1$ hem olumlu hem de olumsuz sınıflara odaklanırken (Kubat ve Matwin, 1997), $g\_mean_2$ yalnızca olumlu sınıfa odaklanmaktadır (He ve Ma, 2013).

$$g\_mean_1 = \sqrt{Recall \times Specificity} \quad (4.20)$$

$$g\_mean_2 = \sqrt{Recall \times Precision} \quad (4.21)$$

- MCC, konfüzyon matrisinin tüm hücrelerini denklemindeki hesaba katan bir ölçüdür. Sınıflar çok farklı boyutlarda olsalar bile dengesizlik veri setleriyle kullanılabilecek dengeli bir ölçü olarak kabul edilir (Boughorbel vd., 2017). MCC, -1 ila 1 aralığına sahiptir; burada -1, tamamen yanlış bir sınıflandırıcıyı belirtirken 1, tamamen doğru bir sınıflandırıcıyı belirtir. Denklem 4.22 kullanılarak hesaplandığı gibi (Matthews, 1975).

$$MCC = \frac{(TP\times TN)-(FP\times FN)}{\sqrt{(TP+FN)\times(TP+FP)\times(TN+FP)\times(TN+FN)}} \quad (4.22)$$

- Eğitim süresi, modelin eğitim aşamanın tamamlanması için geçen zamandır.
- Test süresi, modelinin test aşamanın tamamlanması için geçen zamandır.

### 4.5.2. Çok sınıflı sınıflandırma

Çok sınıflı bir sınıflandırma görevinin karışıklık matrisi, öngörülen sınıflar yerine gerçek sınıflar listesinden oluşturulmuştur (Swets, 1988). Bu önlemin faydasının yorumlanabilirliğinde olduğu çok kullanışlı ve sezgisel bir önlemdir. İkili sınıflandırmadan farklı olarak, dört ana sonucun çok sınıflı sınıflandırma görevinde biraz farklı anlamı vardır. Başlamak için, TN normal numunelerin doğru sınıflandırma numarasıdır. Bunun aksine, FP, saldırı sınıflarının herhangi birine göre sınıflandırılmamış tüm iyi huylu trafik durumlarının toplamıdır. FP denklem 4.23'e göre hesaplanabilir, burada *B* saldırı sınıflarının sayısıdır ve $FP_i$, saldırı sınıfına yanlış şekilde sınıflandırılmış iyi huylu trafik durumlarının sayısıdır. TP, 4.24 denklemiyle hesaplanan, uygun saldırı sınıfına gerçekten sınıflandırılmış tüm saldırı örneklerinin toplamıdır; burada $TP_i$, saldırı sınıfının doğru tahminlerinin sayısıdır. Son olarak, FN, hatalı olarak normal sınıfa sınıflandırılan tüm saldırı örneklerinin toplamıdır. FN, denklem 4.25'e göre hesaplanabilir, burada $FN_i$, normal sınıfa yanlış sınıflandırılmış saldırı sınıfı örneklerinin sayısıdır.

$$FP = \sum_{i=1}^{B} FP_i \tag{4.23}$$

$$TP = \sum_{i=1}^{B} TP_i \tag{4.24}$$

$$FN = \sum_{i=1}^{B} FN_i \tag{4.25}$$

Ardından, dört sonuç, çok sınıflı bir sınıflandırma görevi gerçekleştirirken ortak önlemler olan on beş değerlendirme ölçütünü hesaplamak için kullanılır. Bazı denklemlerin yukarıda açıklanan çok sınıflı sınıflamanın terminoloji tanımına adapte olacak şekilde modüle edildiğinden bahsetmekte fayda vardır. Değerlendirme metriklerinin tanımları ve karşılık gelen denklemlerinin listesi aşağıdaki gibidir:

- Genel Doğruluk, tüm test seti için genel gerçek tahminlerin oranıdır. $Overall\ Accuracy = \frac{TP+TN}{Test\ set\ size}$ (4.26)
- Ortalama Doğruluk, bir sınıflandırıcının ortalama sınıf başına etkinliğidir (Sokolova ve Lapalme, 2009).

$$Avearge\ Accuracy = \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \quad (4.27)$$

$tp_i$, $fp_i$, $tn_i$ ve $fn_i$, sırasıyla $i^{inci}$ sınıfı için gerçek pozitif, yanlış pozitif, gerçek negatif ve yanlış negatif olduğunda ve $l$, veri setindeki kullanılabilir sınıfların sayısıdır.

- Genel Hata Oranı, tüm test kümeleri için genel yanlış tahminlerin oranı hakkında bilgi verir.

$$Overall\ Error\ Rate = \frac{FP + FN}{Test\ set\ size} \quad (4.28)$$

- Ortalama Hata Oranı, sınıf başına ortalama sınıflandırma hatasıdır (Sokolova ve Lapalme, 2009).

$$Avearge\ Error\ Rate = \frac{\sum_{i=1}^{l} \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{l} \quad (4.29)$$

- Makro Ortalamalı Kesinlik, her bir sınıfın kesinliğinin ortalamasıdır.

$$Precision_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \quad (4.30)$$

$M$ indeksi, Makro Ortalama'yı sunar.

- Makro Ortalamalı Geri Çağırma, her bir sınıfın geri çağırmanın ortalamasıdır.

$$Recall_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \quad (4.31)$$

- Makro Ortalamalı F1-Puanı, sınıf başına F1-Ölçü ortalamasıdır (Hossin ve Süleyman, 2015).

$$F1\_Score_M = \frac{2 \times Precision_M \times Recall_M}{Precision_M + Recall_M} \quad (4.32)$$

- Mikro Ortalamalı Kesinlik, pay ve payda toplamından hesaplanan kesinliktir.

$$Precision_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fp_i)} \quad (4.33)$$

$\mu$ indeksi, Mikro Ortalama'yı sunar.

- Mikro Ortalamalı Geri Çağırma, genel bir bölümü hesaplamak için sınıf başına geri çağırma metriğini oluşturan temettüler ve bölenlerin toplamıdır.

$$Recall_{\mu} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l} (tp_i + fn_i)} \quad (4.34)$$

- Mikro Ortalamalı F1-Puanı, her bir sınıfı mikro ortalamalı kesinlik ve geri çağırmadan hesaplanan F1 ölçümüdür.

$$F1\_Score_{\mu} = \frac{2 \times Precision_{\mu} \times Recall_{\mu}}{Precision_{\mu} + Recall_{\mu}} \quad (4.35)$$

- Kayıp Oranı (MR), Elhamahmy vd. (2010), tarafından önerilen çok sınıflı bir sınıflandırıcı için bir performans ölçütüdür. Saldırı sınıfının yanlış sınıflandırılması (MC) gibi çok sınıflı bir karmaşa matrisinden elde edilebilecek yeni bir sonuç tanımladılar. MC, başka bir saldırı sınıfına yanlış şekilde sınıflandırılan belirli saldırı sınıfı örneklerinin sayısını belirler. Bu durumda, bu yanlış etiketlenmiş örnekler dört ana sonucun hiçbirine ait olamaz. MR, denklem 4.36 kullanılarak hesaplanabilir.

  $$MR = \frac{FN+\sum_{i=1}^{l} MC_i}{Actual\ attacks\ size} \quad (4.36)$$

  $MC_i$, $i^{inci}$ saldırı sınıfının MC'sidir.
- Yanlış Oranı (WR) aynı zamanda çok sınıflı bir sınıflandırıcı için ve MC sonucuna dayanan bir performans ölçütüdür (Elhamahmy vd., 2010). Yanlış etiketlenmiş saldırı örneklerinin test setinde saldırı olarak sınıflandırılan ve denklem 4.37'ye göre hesaplanabilen tüm örneklerle orantılı bölümüdür.

  $$WR = \frac{FP+\sum_{i=1}^{l} MC_i}{TP+FP+\sum_{i=1}^{l} MC_i} \quad (4.37)$$
- Maliyet Başına F-puanı (FPC), çok sınıflı bir sınıflandırıcı için yeni bir ölçümdür ve F1-Puanı, MR ve WR ölçümlerine dayanır (Elhamahmy vd., 2010). FPC değeri 0 ile 1 arasında değişmektedir, burada 0 tamamen yanlış bir sınıflandırıcıyı ifade eder. Aksi takdirde, 1'e eşit olduğunda ideal bir sınıflandırıcıya atıfta bulunur.

  $$FPC = \frac{F1\_Score}{\sqrt{(F1\_Score)^2+(Cost)^2}} \quad (4.38)$$

  Where, $Cost = \sqrt{(MR)^2 + (WR)^2}$ (4.39)
- Eğitim süresi ve Test süresi metrikleri alt bölüm 4.5.1'de tanımlandığı gibidir.

### 4.6. Sonuçlar ve Tartışma

Çift PSO tabanlı algoritma, NumPy kütüphanesi (NumPy, 2018) ile Python 3.6.4 (Python, 2018) programlama dili sürümü kullanılarak gerçekleştirildi. Ayrıca, tüm derin öğrenme modellerini, Keras açık kaynaklı sinir ağı kütüphanesini (Chollet, 2015; Keras, 2018) TensorFlow 1.6 makine öğrenme çerçevesi sürüm (Abadi vd., 2016; TensorFlow, 2018) üzerinden CUDA 9.0 (CUDA, 2018) entegre arka sürüm ve cuDNN versiyon 7.0 (cuDDN, 2018) kullanarak dağıttık.

Donanım ortamı aşağıdaki gibidir: Intel Core i7 CPU (3.8 GHz, 16 MB Önbellek), 16 GB RAM ve Windows 10 işletim sistemi (64 bit modu). Hesaplamaları hızlandırmak ve verimliliği artırmak için, NVIDIA Tesla K20 GPU 5 GB GDDR5 ile GPU ile hızlandırılmış hesaplama da kullanılmaktadır. Aşağıdaki alt bölümlerde, deneysel sonuçlarımızı sunuyoruz ve bunları performans analizi, Friedman istatistiksel testi ve sıralama yöntemleri gibi üç farklı analizle tartışıyoruz.

### 4.6.1. Performans analizi

Bir modelin değerlendirme ölçütlerinin puanına dayanarak ağ saldırı tespitindeki etkinliğinin kilit bileşeni. Doğruluk, Kesinlik, Geri Çağırma ve F1-Puanın'daki yüksek değerler ile FNR, Hata Oranı ve FAR'daki düşük değerler etkin bir sınıflandırıcıyı belirtir. İdeal sınıflandırıcı, Doğruluk ve Geri Çağırma değerlerinin 1'e, FNR ve FAR değerlerinin ise 0'a ulaştığını gösterir.

#### 4.6.1.1. İkili sınıflandırma sonuçları

Çizelge 4.6 ve 4.7 sırasıyla NSL-KDD deneyi ve CICIDS2017 deneyi için değerlendirme ölçütlerinin değerlerini sunmaktadır. Aslında, Saniye cinsinden Eğitim süresi ve Test süresi hariç tüm değerler yüzde biçiminde sunulur. Ayrıca, kalın değerler aynı veri setinde en iyi sonuçları temsil eder. Performans farklılıklarını göstermek için, Çizelge 4.6 ve 4.7'deki eksi işaretli sütunlar (-), yaklaşımımızı kullanmadan modelin sonuçlarını (tam özellik kümesi) belirtirken, artı işaretli sütunların (+) önerilen çift PSO tabanlı algoritmayı kullanarak bir eğitim öncesi aşama uygularken modelin sonucunu belirtir.

Veri setleri ile ilgili olarak, tüm kullanılmış modeller CICIDS2017'de NSL-KDD'den daha yüksek performans kazandı. Bunun nedeni, CICIDS2017 veri setinin, NSL-KDD veri setinden daha güvenilir olan modern IDS veri setleri biçimi olmasıdır. Bununla birlikte, NSL-KDD için kaydedilen tüm modellerin eğitim ve test süreleri, CICIDS2017 için karşılık gelen değerlerden daha azdır. Bu, NSL-KDD veri setinin, CICIDS2017'ye (tam özellikli için 80, özellik alt kümesi için 23), kıyasla az sayıda özelliğe (tam özellik için 41, özellik alt kümesi

için 10) sahip olması ile açıklanabilir. Ayrıca, CICIDS2017'nin eğitim ve test setlerindeki örnek sayısı, NSL-KDD veri setindeki örneklerden daha fazla.

Çizelge 4.6. NSL-KDD deneyinin ikili sınıflandırma sonuçları

| **Metrik (%)** | **DNN** | | **LSTM-RNN** | | **DBN** | |
|---|---|---|---|---|---|---|
| | **-** | **+** | **-** | **+** | **-** | **+** |
| Doğruluk | 92.18 | 97.72 | 94.07 | 98.8 | 95.72 | **99.79** |
| Kesinlik | 95.77 | 99.6 | 97.23 | 99.7 | 98.37 | **99.83** |
| Geri çağırma | 90.25 | 96.38 | 92.21 | 98.18 | 94.04 | **99.81** |
| F1-Puanı | 92.93 | 97.96 | 94.65 | 98.94 | 96.16 | **99.82** |
| FAR | 5.26 | 0.51 | 3.47 | 0.39 | 2.06 | **0.23** |
| Spesifiklik | 94.74 | 99.49 | 96.53 | 99.61 | 97.94 | **99.77** |
| FNR | 9.75 | 3.62 | 7.79 | 1.82 | 5.96 | **0.19** |
| Negatif kesinlik | 88.03 | 95.41 | 90.36 | 97.65 | 92.56 | **99.74** |
| Hata oranı | 7.82 | 2.28 | 5.93 | 1.2 | 4.28 | **0.21** |
| BDR | 95.77 | 99.6 | 97.23 | 99.7 | 98.37 | **99.83** |
| BTNR | 88.03 | 95.41 | 90.36 | 97.65 | 92.56 | **99.74** |
| g_mean1 | 92.47 | 97.92 | 94.34 | 98.89 | 95.97 | **99.79** |
| g_mean2 | 92.97 | 97.97 | 94.69 | 98.94 | 96.18 | **99.82** |
| MCC | 84.39 | 95.43 | 88.16 | 97.57 | 91.45 | **99.57** |
| Eğitim süresi (saniye) | 7938 | **740** | 8241 | 936 | 9059 | 1552 |
| Test süresi (saniye) | 1421 | **132** | 1475 | 168 | 1621 | 278 |

Beklediğimiz gibi, eğitim öncesi aşamalı derin öğrenme modelleri, tüm metrikler ve tüm veri setleri için ön eğitim olmadan aynı modellerden daha iyi performans gösterdi. Bu, derin öğrenme modellerinin eğitim öncesi aşamasında çift PSO tabanlı algoritmanın kullanılmasının etkisiyle mümkün olmuştur. Bölüm 4.3'te açıklandığı gibi, çift PSO tabanlı algoritma, modelin yapısını basitleştiren optimum özellik alt kümesi ile azaltılmış veri setini oluşturur. Buna ek olarak, çift PSO tabanlı algoritma, verilen veri setindeki doğruluğu en üst düzeye çıkaran modelin optimal hiperparametrelerini seçmeye çalışır. Denklem 4.9'a göre, doğruluk değeri arttıkça, paydaki sabit olduğu için, paydaki TP ve TN toplamı önemli ölçüde artacaktır. Bu nedenle, geri çağırma gibi sınıflandırma

metriklerinin değerini kesinlikle artırmanın yanı sıra FAR gibi yanlış sınıflandırma metriklerinin değerini de düşürür. Bulgulara göre, ön eğitim almış olan derin öğrenme modelleri, geri çağırma metriklerini (4%-6%) arttırdı ve FAR metriklerini (1%-5%) azalttı, ön eğitim almadan aynı modellerin karşılık gelen değerlerinden aynı veri setinde.

Çizelge 4.7. CICIDS2017 deneyinin ikili sınıflandırma sonuçları

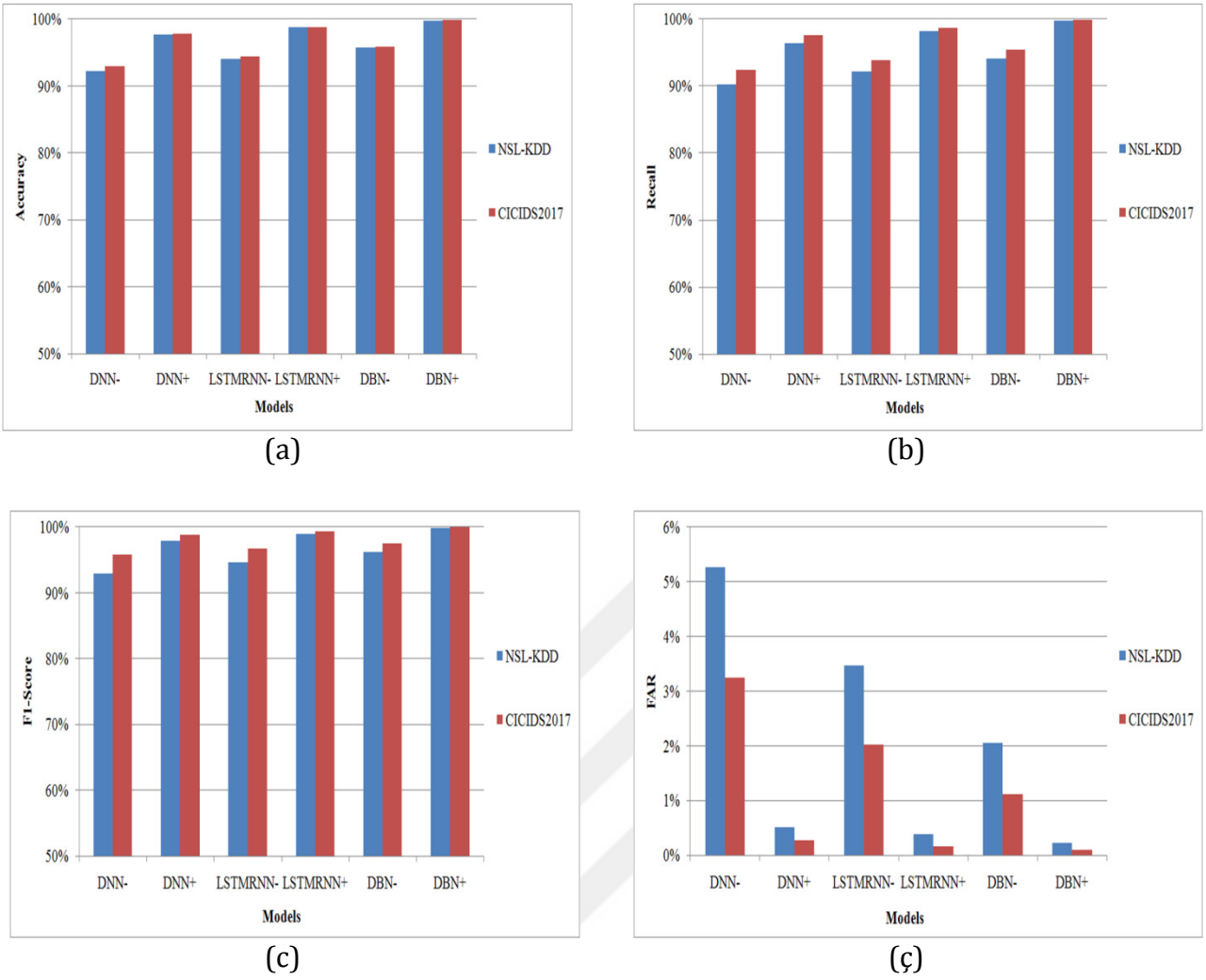
| **Metrik (%)** | **DNN** | | **LSTM-RNN** | | **DBN** | |
|---|---|---|---|---|---|---|
| | **-** | **+** | **-** | **+** | **-** | **+** |
| Doğruluk | 92.92 | 97.85 | 94.41 | 98.83 | 95.82 | **99.91** |
| Kesinlik | 99.5 | 99.96 | 99.69 | 99.98 | 99.83 | **99.99** |
| Geri çağırma | 92.38 | 97.58 | 93.9 | 98.68 | 95.38 | **99.92** |
| F1-Puanı | 95.81 | 98.76 | 96.71 | 99.33 | 97.56 | **99.95** |
| FAR | 3.24 | 0.28 | 2.02 | 0.16 | 1.11 | **0.1** |
| Spesifiklik | 96.76 | 99.72 | 97.98 | 99.84 | 98.89 | **99.9** |
| FNR | 7.62 | 2.42 | 6.1 | 1.32 | 4.62 | **0.08** |
| Negatif kesinlik | 64.45 | 85.5 | 69.65 | 91.55 | 75.37 | **99.41** |
| Hata oranı | 7.08 | 2.15 | 5.59 | 1.17 | 4.18 | **0.09** |
| BDR | 99.5 | 99.96 | 99.69 | 99.98 | 99.83 | **99.99** |
| BTNR | 64.45 | 85.5 | 69.65 | 91.55 | 75.37 | **99.41** |
| g_mean1 | 94.54 | 98.64 | 95.92 | 99.26 | 97.12 | **99.91** |
| g_mean2 | 95.87 | 98.76 | 96.75 | 99.33 | 97.58 | **99.95** |
| MCC | 75.5 | 91.19 | 79.82 | 94.96 | 84.2 | **99.61** |
| Eğitim süresi (saniye) | 9310 | **1274** | 10043 | 1492 | 10688 | 1617 |
| Test süresi (saniye) | 4665 | **637** | 5029 | 846 | 5439 | 915 |

Bir bakışta, LSTM-RNN modelleri, tüm veri setleri ile ilgili tüm değerlendirme ölçütleri açısından DNN modellerinden daha üstün performans gösterdi. Bunun arkasındaki gerçek, geleneksel yapay nöronları tüm gizli katmanlarda kullanmak yerine, LSTM-RNN modelinin LSTM bellek hücrelerini kullanmasıyla sonuçlanmaktadır. Ayrıca, LSTM-RNN modelleri, aynı gizli katmandaki işleme elemanları arasında, gizli-gizli bağlantılara sahiptir. Böylece, DNN modellerinde bulunmayan LSTM-RNN modellerinin bu yerleşik özellikleri, sınıflandırıcının

önceki durumları ezberlemesine, aralarındaki bağımlılıkları keşfetmesine ve son olarak çıktıyı tahmin etmek için mevcut girdilerle birlikte kullanmasına olanak tanır. Bu arada, LSTM-RNN ve DNN modellerinin tüm veri setlerindeki performansı arasındaki fark (1%-2%) arasında olan geri çağırma metriği için önemlidir, ancak (0.1%-0.2%) arasında olan FAR metriği için çok küçüktür tüm vakalarda.

Her ne kadar DNN modelleri en iyi eğitim ve test sürelerini atsalar da, deneysel sonuçlar DBN modellerinin tüm veri setlerindeki diğer derin öğrenme modellerine göre üstünlüğünü göstermiştir. Bu, DBN'nin derin yapısının doğası gereği, bir dizi RBM'nin ardından tam olarak bağlanmış bir BP katmanı içermektedir. RBM'ler, gizli ve görünür katmanlarını kullanarak girdilerden faydalı özellikleri öğrenmeye çalışırlar. Öğrenme sürecinde, RBM'ler denetimsiz bir şekilde önceden eğitilmiştir. Daha sonra, çıkarılan kısaltılmış özellikler, geri yayılımın denetimli bir şekilde kullanılmasıyla tüm ağ ile ince ayarlanmış bir BP katmanına beslenir. Bu nedenle, bu işlevsellik DBN'yi temeldeki görevi derinden anlayabilen güçlü bir sınıflandırıcı yapar. DBN modelleri, tüm veri setlerinde geri çağırma için (99.81%-99.92%) ve FAR için (0.1%-0.23%) arasında kalanları hatırlayan mükemmel sonuçlar elde etti. Buna rağmen, DNN ve LSTM-RNN modelleri, kullanılan veri setlerinde ağ saldırı tespitinde çok iyi performans gösterdi.

Kısalık ve alan sınırlaması açısından, Şekil 4.10'da görsel olarak sunulacak sadece standart sınıflandırma ölçütlerini seçtik. Şekil 4.10 (a), 4.10 (b), 4.10 (c) ve 4.10 (ç), her veri setindeki tüm modellerin Doğruluk, Geri çağırma, F1-Puanı ve FAR yüzdelerini göstermektedir. Oysa, *M-* notasyonu, tam özellikli veri setine uygulanan eğitim öncesi aşaması olmayan derin bir öğrenme modelini *M* belirtirken, *M+* önerilen çift PSO tabanlı algoritmayı kullanarak ön eğitim ile derin bir öğrenme modelini *M*'yi belirtir. Şekil 4.10, her veri setindeki tüm modellerin performansının görsel olarak karşılaştırılmasını sağlayabilir.




Şekil 4.10. Her veri seti için modeller arasında standart metrik karşılaştırma a) Doğruluk, b) Geri çağırma, c) F1-Puanı, ç) Yanlış Alarm Oranı

#### 4.6.1.2. Çok sınıflı sınıflandırma sonuçları

Çizelge 4.8 ve 4.9, sırasıyla NSL-KDD deneyi ve CICIDS2017 deneyi için değerlendirme ölçütlerinin değerlerini göstermektedir. Eğitim süresi ve Test süresi saniye cinsinden hariç tüm değerler yüzde olarak sunulur. Çizelge 4.8 ve 4.9'daki eksi ve artı işaretleri, Çizelge 4.6 ve 4.7'deki ile aynı anlama sahiptir.

Benzer şekilde, NSL-KDD veri seti dengesiz bir yapıya sahiptir. Dengesiz veri seti genellikle tüm sınıfların eşit şekilde temsil edilmediği sınıflandırma görevleriyle ilgili bir sorunu ifade eder. NSL-KDD veri seti durumunda, test seti dağılımı: Normal, DoS, Probe, R2L ve U2R sınıfları için sırasıyla 43%, 33%,

10.7%, 13% ve 0,3%'tür. Bu nedenle, öğrenme modelleri küçük azınlıklardan ziyade örneklerin büyük çoğunluğuyla sınıflara yönelmektedir. Bu, NSL-KDD veri setinin, çok sınıflı sınıflandırma yerine Normal (43%) ve Saldırı (57%) sınıfları arasında küçük bir fark olduğu ikili sınıflandırma için daha uygun olduğuna yol açar. Öte yandan, çoğu durumda sonuçları zenginleştiren dengeli bir CICIDS2017 veri seti kullandık.

Çizelge 4.8. NSL-KDD deneyinin çok sınıflı sınıflandırma sonuçları

| **Metrik (%)** | **DNN** | | **LSTM-RNN** | | **DBN** | |
|---|---|---|---|---|---|---|
| | **-** | **+** | **-** | **+** | **-** | **+** |
| Genel doğruluk | 73.35 | 90.63 | 74.68 | 93.6 | 86.53 | **96.91** |
| Ortalama doğruluk | 89.34 | 96.25 | 89.87 | 97.44 | 94.61 | **98.77** |
| Genel hata oranı | 25.09 | 7.84 | 23.09 | 5.49 | 12.03 | **2.45** |
| Ortalama hata oranı | 10.66 | 3.75 | 10.13 | 2.56 | 5.39 | **1.23** |
| $Precision_M$ | 83.37 | 93.86 | 83.3 | 95.85 | 91.39 | **98.1** |
| $Recall_M$ | 47.61 | 80.61 | 50.4 | 86.19 | 69.82 | **92.29** |
| $F1\_Score_M$ | 60.61 | 86.73 | 62.81 | 90.76 | 79.16 | **95.11** |
| $Precision_\mu$ | 73.35 | 90.63 | 74.68 | 93.6 | 86.53 | **96.91** |
| $Recall_\mu$ | 73.35 | 90.63 | 74.68 | 93.6 | 86.53 | **96.91** |
| $F1\_Score_\mu$ | 73.35 | 90.63 | 74.68 | 93.6 | 86.53 | **96.91** |
| MR | 43.53 | 14.81 | 42.37 | 10.38 | 21.85 | **5.07** |
| WR | 9.62 | 4.83 | 9.47 | 2.68 | 5.27 | **1.53** |
| FPC | 84.59 | 98.57 | 85.69 | 99.36 | 96.81 | **99.85** |
| Eğitim süresi (saniye) | 9452 | **881** | 9813 | 1115 | 10787 | 1848 |
| Test süresi (saniye) | 1692 | **157** | 1756 | 200 | 1930 | 331 |

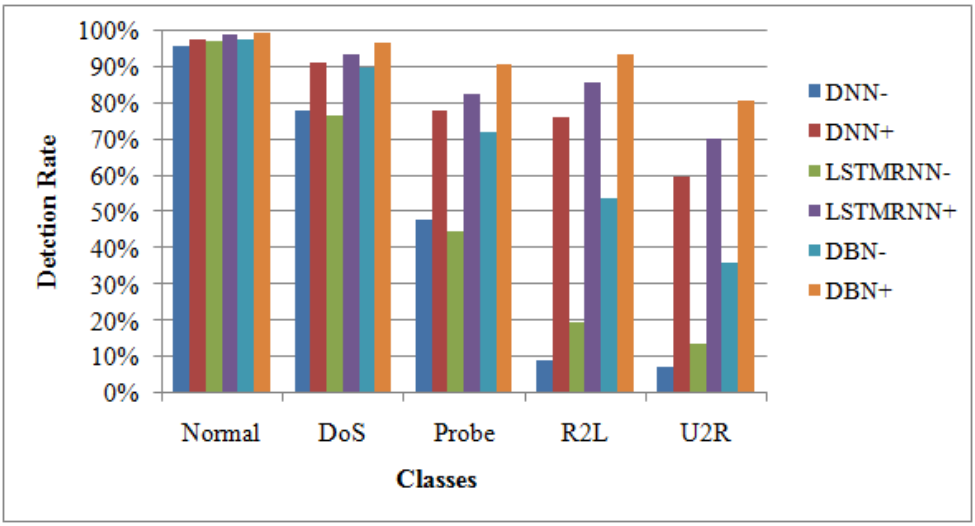
Nitekim doğruluk değeri sadece veri setindeki temel sınıf dağılımını yansıtmaktadır. Bu koşul, doğruluk değerinin modelin tam performansını yansıtmadığı doğruluk paradoksu olarak da bilinir. Genel olarak, mikro ortalamalı ölçümlerin dengesiz veri setleriyle kullanılması daha çok tercih edilir, yani tüm sınıfları sık sınıfa yönlendirmek istediğimizde. Buna karşılık, makro ortalamalı metrikler dengeli veri setleri için yararlıdır ve her bir sınıfa eşit ağırlık vererek tüm sınıflara aynı vurguyu koymak istiyoruz. Ancak, eğitim ve

test süreleri değerleri ikili sınıflandırmada karşılık gelen değerlerden daha yüksektir. Bunun nedeni, çok sınıflı bir sınıflandırmadaki modellerin mimarisi, ikili bir sınıflandırmadakilerden daha karmaşıktır.

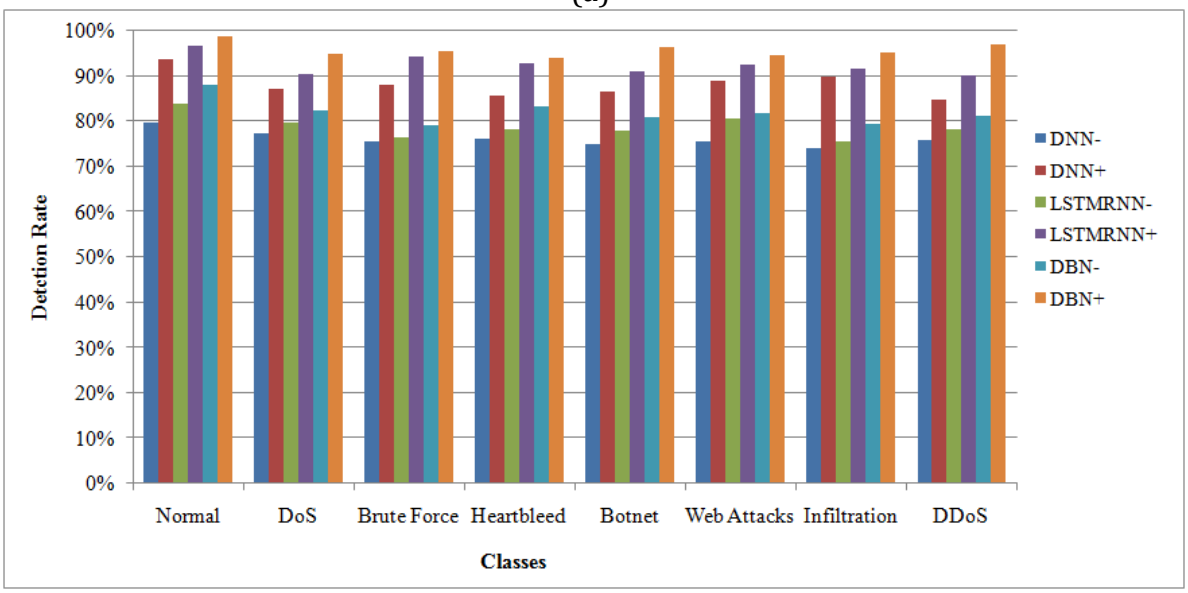
Çizelge 4.9. CICIDS2017 deneyinin çok sınıflı sınıflandırma sonuçları

| **Metrik (%)** | **DNN** | | **LSTM-RNN** | | **DBN** | |
|---|---|---|---|---|---|---|
| | **-** | **+** | **-** | **+** | **-** | **+** |
| Genel doğruluk | 76.19 | 88.04 | 78.81 | 92.41 | 82 | **95.81** |
| Ortalama doğruluk | 94.05 | 97.01 | 94.7 | 98.1 | 95.5 | **98.95** |
| Genel hata oranı | 6.29 | 2.34 | 5.25 | 1.23 | 4.03 | **0.58** |
| Ortalama hata oranı | 5.95 | 2.99 | 5.3 | 1.9 | 4.5 | **1.05** |
| $Precision_M$ | 76.35 | 88.08 | 78.92 | 92.44 | 82.08 | **95.82** |
| $Recall_M$ | 76.19 | 88.04 | 78.81 | 92.41 | 82 | **95.81** |
| $F1\_Score_M$ | 76.27 | 88.06 | 78.87 | 92.43 | 82.04 | **95.81** |
| $Precision_\mu$ | 76.19 | 88.04 | 78.81 | 92.41 | 82 | **95.81** |
| $Recall_\mu$ | 76.19 | 88.04 | 78.81 | 92.41 | 82 | **95.81** |
| $F1\_Score_\mu$ | 76.19 | 88.04 | 78.81 | 92.41 | 82 | **95.81** |
| MR | 24.33 | 12.76 | 21.9 | 8.21 | 18.89 | **4.63** |
| WR | 23.24 | 12.02 | 20.81 | 7.77 | 17.88 | **4.3** |
| FPC | 92.94 | 98.26 | 94.42 | 99.31 | 95.96 | **99.79** |
| Eğitim süresi (saniye) | 13950 | **1911** | 150645 | 2238 | 16032 | 2426 |
| Test süresi (saniye) | 6998 | **956** | 7544 | 1269 | 8156 | 1373 |

Açıkçası, bulgularımız ön eğitim aşamalı modellerin tüm ölçümlerde ve tüm veri setlerinde ön eğitimsiz modellerden daha iyi performans gösterdiğini doğruladı. Şekil 4.11 (a) ve 4.11 (b), NSL-KDD ve CICIDS2017 veri setlerinde her bir sınıfın tespit oranını göstermektedir. Şekil 4.11'e göre, önerdiğimiz yaklaşım, modellerimizin sadece bizim modelimizi kullanmadan modellerin performansını arttırmakla kalmayıp, aynı zamanda her bir sınıfın tespit oranını da önemli ölçüde arttırdı.




(a)




(b)

Şekil 4.11. Veri seti her sınıf için bir model tespit oranı a) NSL-KDD, b) CICIDS2017

#### 4.6.1.3. Önceki çalışmalara karşılaştırılması

Çizelge 4.6 ve 4.7'deki sonuçlarımızın, derin öğrenme modellerimizin özellikle DNN'nin Çizelge 2.2'deki tüm listelenen modellerden daha iyi performans gösterdiğini kolayca fark edebiliriz. Çalışmalara rağmen Tang vd. (2016) ve Ahmad (2015), önerdiğimiz çift PSO tabanlı algoritmamızdan daha küçük özellik alt kümeleri oluşturmuştur, ancak çalışmamızda seçilen özellikler, modellerin hem ikili hem de çok sınıflı sınıflandırmalardaki genel performansını iyileştirmiştir.

### 4.6.2. Friedman testi analizi

Friedman testi birkaç tekrarlı tedavi arasındaki farklılıkları keşfetmek için iyi bilinen bir parametrik olmayan istatistiksel testtir (Daniel, 1990). Parametrik olmayan, testin verilerinizin belirli bir dağıtımdan geldiğini varsaymadığı anlamına gelir. Bu çalışmada, her biri veri setlerinden biri için iki ilgili tedavimiz ($k$ = 2) ve her tedavide altı deneğin ($R$ = 6) olduğu gibi, her deneğin modellerden biriyle ilişkili olduğu görülmektedir. Friedman testi için boş hipotez, tüm tedavilerin aynı etkilere sahip olduğu, yani tedaviler arasında hiçbir fark olmadığıdır. Matematiksel olarak, eğer sadece *FS FC*'den büyükse ve *P-değeri* seçilen anlamlılık seviyesinden ($\alpha$) düşükse boş hipotezi reddedebiliriz. Çizelge 4.10, *TP*, *TN*, *FP* ve *FN* ölçümleri için Friedman testinin sonuçlarını göstermektedir. Tüm testlerde anlamlılık seviyesini 0,05'e eşit seçtik çünkü oldukça yaygın.

Çizelge 4.10. Her bir sonuç için Friedman test sonuçları ($\alpha$= 0.05)

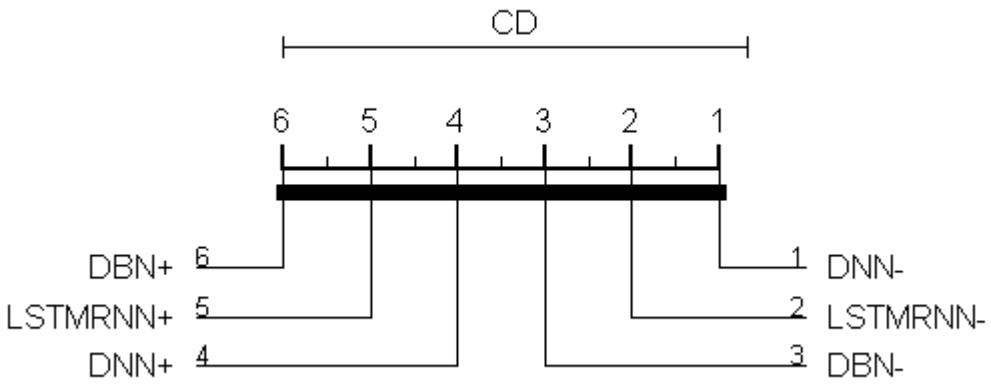
| Ölçümler | FS | FC | P-değeri |
|---|---|---|---|
| TP | 12 | 7 | 0.01431 |
| TN | 12 | 7 | 0.01431 |
| FP | 9.33 | 7 | 0.0094 |
| FN | 12 | 7 | 0.01431 |

Buna göre, Friedman testinin boş hipotezini reddettik, çünkü her durumda (*FS*> *FC*) ve (*P-değeri*<$\alpha$). Bu nedenle, her bir ölçüm için modellerin puanlarının birbirinden önemli ölçüde farklı olduğu sonucuna varılabilir. Friedman testinin sonuçlarını görsel olarak yorumlayabilmek için, Kritik Fark Diyagramını da çizdik (Demsar, 2006). Şekil 4.12, kullanılmış modellerin tüm veri setlerinde Kritik Fark Diyagramını göstermektedir. Son olarak, rakamın üzerinde çubuk olarak gösterilen *CD* değerini 5.3319'a eşittir.

### 4.6.3. Sıralama yöntemler analizi

Sıralama yöntemleri genellikle farklı makine öğrenme algoritmaları arasındaki dengeyi göstermek ve en uygun sınıflandırıcıyı seçmek için performanslarını

değerlendirmek için kullanılır. Sıralama yöntemlerinin ana avantajları, sınıflandırıcının tam çalışma aralığında görselleştirme veya özetleme performansı sağlamak ve verilerin eğriliği ile ilgili bilgileri vurgulamaktır (He ve Ma, 2013). Bu çalışmada, Alıcı Çalışma Karakteristik (ROC) eğrileri ve Kesinlik-Geri çağırma (PR) eğrileri olmak üzere iki popüler sıralama yöntemi seçtik. Aralarındaki en büyük fark, PR eğrileri olup, dengesiz veri setleriyle çalışırken modelin performansıyla ilgili daha bilgilendirici bir resim verir (Davis ve Goadrich, 2006). Başka bir deyişle, PR eğrileri sınıf dengesizliği problemi için daha iyidir. Aksi takdirde, ROC eğrileri dengeli veri setleri için daha iyidir.

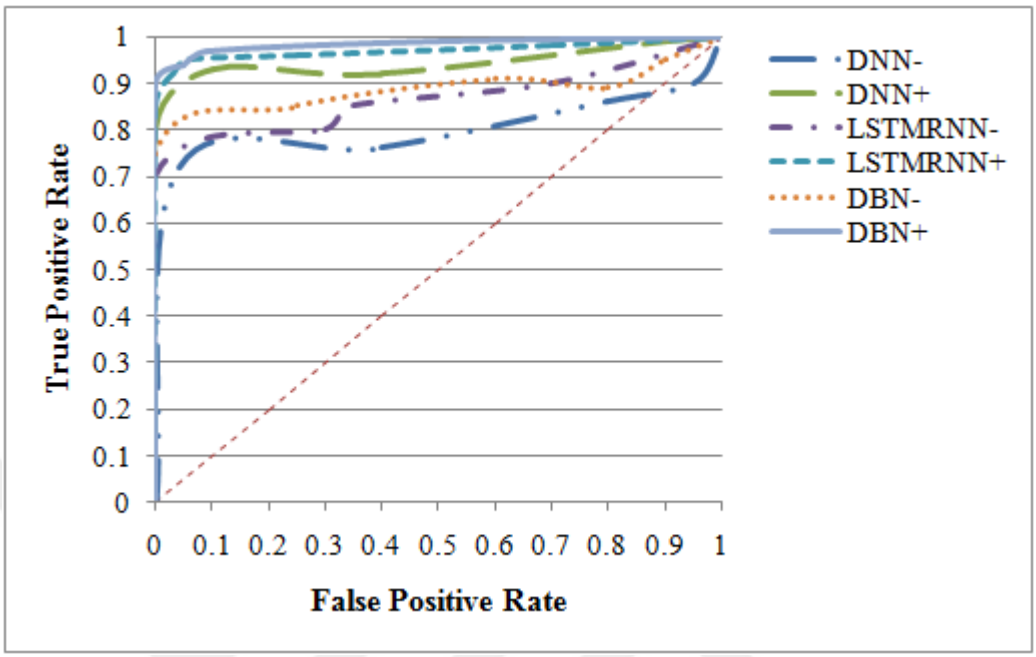



Şekil 4.12. Tüm veri setleri üzerinde kullanılan modellerin Kritik Fark Diyagramı

Başlangıçta, bir ROC eğrisi, bir sınıflandırıcının yanlış pozitif oranının (veya FAR) bir fonksiyonu olarak gerçek pozitif oranın (veya Geri Çağırma) bir grafiğidir (Fawcett, 2006). ROC'nin çapraz çizgisinin, performansın 50%'sinin elde edildiğini gösteren bir referans çizgisi olduğu kabul edilir. ROC'nin sol üst köşesi, 100% ile en iyi performansı ifade eder. Şekil 4.13, CICIDS2017 veri setinde test edilen modellerin ROC eğrilerini göstermektedir.

ROC eğrisi Altındaki Alan (AUC-ROC veya basitçe AUROC), çeşitli ROC eğrileri arasında nicel olarak karşılaştırma yapmak için yaygın olarak kullanılan bir ölçüdür (Bradley, 1997). AUROC değeri, karşılık gelen ROC eğrisini 0 ile 1 arasında tek bir değerde özetler. Optimal sınıflandırıcı, AUROC değerine 1 sahip

olacaktır. Çizelge 4.11, Şekil 4.13'te çizilen ROC eğrilerinin AUROC değerlerini göstermektedir.




Şekil 4.13. CICIDS2017 veri seti üzerinde modellerin ROC eğrileri

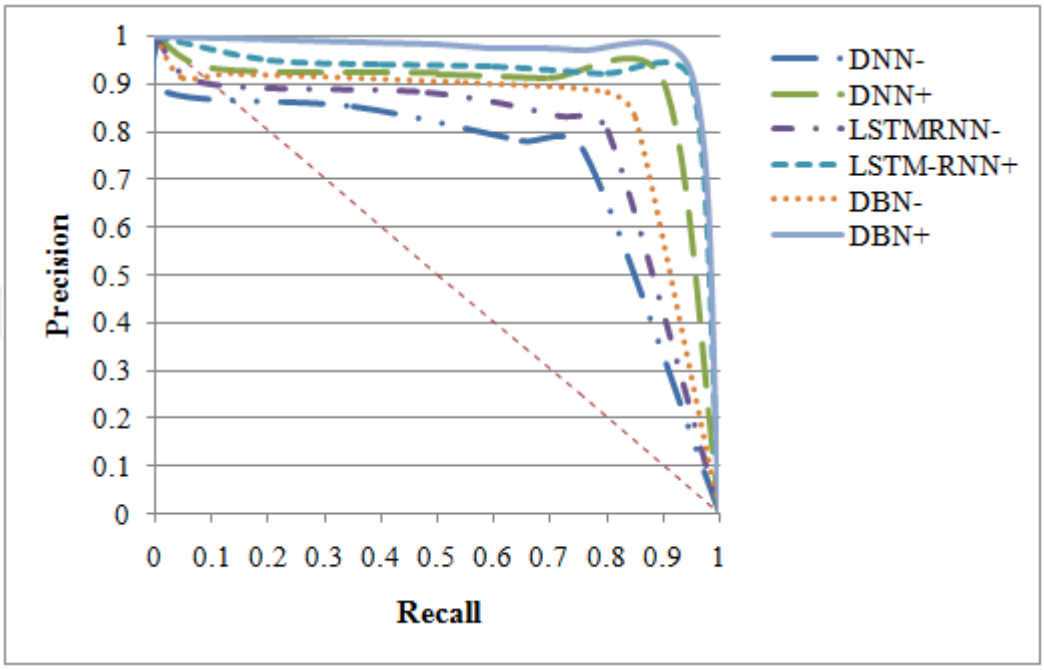
Çizelge 4.11. ROC eğrilerinin AUROC değerleri

| Model | AUROC |
|---|---|
| DNN- | 0.9457 |
| DNN+ | 0.9865 |
| LSTMRNN- | 0.9594 |
| LSTMRNN+ | 0.9926 |
| DBN- | 0.9713 |
| DBN+ | 0.9991 |

Öte yandan, bir PR eğrisi, X ekseni üzerinde Geri çağırma karşı Y eksenindeki Kesinlik bir grafiğidir. İdeal model sağ üst köşede bir PR eğrisi verir; bu, modelin yalnızca yanlış pozitif ve yanlış negatif olmayan gerçek pozitifleri elde ettiği anlamına gelir (Davis ve Goadrich, 2006). Şekil 4.14, test edilen modellerin PR eğrilerini NSL-KDD veri setinde göstermektedir.

Son olarak, PR eğrisi Altındaki Alan (AUC-PR veya sadece AUPR), bir modelin PR eğrisinin altındaki alandır (Landgrebe vd. 2006). Temel olarak, AUPR değeri [0,1] aralığındadır ve mükemmel sınıflandırıcı AUPR değerine 1 eşittir. Çizelge

4.12, Şekil 4.14'te çizilen PR eğrilerinin AUPR değerlerini göstermektedir. ROC ve PR eğrileri, sıralı modellerin: DBN, LSTM-RNN ve DNN'nin tüm veri setleri üzerinde ağ saldırı tespitinde etkin performansa sahip olduğunu göstermektedir. Ancak, kullanılan tüm modeller hala oldukça iyi bir uyum sergiliyor.




Şekil 4.14. NSL-KDD veri seti üzerinde modellerin PR eğrileri

Çizelge 4.12. PR eğrilerinin AUPR değerleri

| Model | AUPR |
|---|---|
| DNN- | 0.9301 |
| DNN+ | 0.9799 |
| LSTMRNN- | 0.9472 |
| LSTMRNN+ | 0.9894 |
| DBN- | 0.962 |
| DBN+ | 0.9982 |

# 5. BULUT SALDIRI TESPİTİ

Bu bölümde, bulut saldırı tespiti için entegre bir IDS ve üçüncü taraf bir servis oluşturmak için önceki bölümlerdeki bulgularımızı bir araya getirdik. Yaklaşan alt bölümler, bazı materyalleri ve yöntemleri açıklar ve ardından önerilen IDS ve bulut hizmetini sunar.

## 5.1. Giriş

Son yıllarda, bulut bilişim, Bilgi Teknolojisi (BT) dünyasında en yeni ortaya çıkan internet tabanlı teknoloji olarak kabul edilir. Google'ın 2006'da bulut bilişim kavramını ortaya çıkarmasıyla başladı. Ardından BT endüstrisinde yer alan her bireyin ve kuruluşun tercih ettiği tercih haline gelene kadar hızla büyüyor. Bulut bilişimin yaygınlığının arkasındaki sebep, bilişim dünyasında yeni bilişim ve iletişim paradigmalarıyla yapılan devrimdir. Örneğin, kullanıcılardan endişe duymadan yönetim çabasıyla hızlı bir şekilde sağlanabilen ve yayınlanabilen dinamik olarak ölçeklenebilir ve sanallaştırılmış kaynaklar sunar. Ayrıca, depolama, bilgi, platformlar, uygulamalar ve sunucular gibi paylaşılan bu kaynaklar, bulut kullanıcılarına talep üzerine servis olarak sunulur. Bu nedenle, kullanıcılar yalnızca kullandıkları kaynaklar için ödeme yapar. Buna göre, bu kullanım başına ödeme şekli pek çok işletme kuruluşunun geleneksel BT için karşılayamayacakları sermaye harcamalarından kaçınmasını mümkün kılmaktadır (Mehmood vd., 2013).

Temel olarak bulut bilişim, sistem katmanı, platform katmanı ve uygulama katmanı olmak üzere üç soyut katmandan oluşur. Sanal Makineler (VM) ve işletim sistemiyle ilgili ilk iki katman, sonuncusu bulut tabanlı web tabanlı uygulamalar gibi sağlanan uygulamaları içerir. Ayrıca, hizmetler kullanıcılara üç farklı modelde sunulmaktadır: Hizmet Olarak Altyapı (IaaS), Hizmet Olarak Platform (PaaS) ve Hizmet Olarak Yazılım (SaaS). IaaS, yöneticileri, VM'leri ve ağın tam kontrolü ve bakımı için hedefler. Bu arada, PaaS modeli, geliştiricilerin kullanıcı tarafından oluşturulan uygulamaları bulutta dağıtmalarını hedeflemektedir, SaaS, kullanıcıların sağlayıcıların uygulamalarını

yürütmelerini sağlar. Bulutun mimarisiyle ilgili olarak, iki sıralı bileşenden oluşur. ilk kullanıcı tarafından görülen kısım olan ön uç olarak bilinir; kullanıcının ağı veya bilgisayarı ve buluta bir kullanıcı arayüzü aracılığıyla erişmek için kullanılan uygulama. İkinci bileşen, aslında çeşitli sunuculardan oluşan bulutun kendisi olan arka uçtur (Modi vd., 2013).

Bulut bilişimin yararları ve popülerliğinin yanı sıra, başarısında engel teşkil eden ciddi engellerden muzdariptir. Bunlardan biri, çoğu kurum tarafından bulutun benimsenmesi için büyük bir engel olarak kabul edilen bulut güvenliğidir. Bu, bulut ortamının doğasının açık ve tamamen dağılmış olması nedeniyle güvenlik tehditlerine ve güvenlik açıklarına daha yatkın hale gelir. Bu nedenle, davetsiz misafirleri buluta veya bulut içindeki cihazlara karşı potansiyel saldırılar başlatmaya teşvik eder. Bulut güvenliğinin bir diğer beklentisi ve şüphesi, kullanıcıların yalnızca Bulut Hizmet Sağlayıcısı (CSP) tarafından yapılan verilerin saklanması ve sürdürülmesi sorumluluğu olmadan buluttaki uzak sunuculardaki verilerine erişmeleridir. Veri ve uygulama üzerindeki kontrolün sonlandırılması, veri bütünlüğü, gizlilik ve erişilebilirliğin kritik endişelerini ortaya koymaktadır (Dhage vd., 2011).

Bulut güvenliği aktif bir araştırma alanıdır ve birçok çözüm önerilmekte ve çoğu geliştirilmektedir. Buna rağmen, geleneksel bulut mimarisi bir bulut girme riski için kapı açar. İzinsiz Saldırı, bir bilgisayarın, ağın veya bulutun Gizlilik, Bütünlük ve Kullanılabilirliğini (CIA) tehlikeye atma girişimi olarak tanımlanır. Bu, bir veya daha fazla izinsiz giriş saldırı gerçekleştirerek gerçekleştirilebilir. Bulut bilişim, örneğin DDoS, sel, liman taraması vb. gibi birçok saldırı gerçekleştirilebilir. Bu tür saldırıları önlemek için iyi bilinen mekanizmalardan biri bulut tabanlı bir IDS kullanmaktır (Kadam, 2011).

Aslında, bulut tabanlı IDS, türüne ve kullandıkları algılama yöntemine bağlı olarak değişir. Üç ünlü bulut tabanlı IDS türü vardır: Ana Bilgisayar tabanlı IDS (HIDS), Ağ tabanlı IDS (NIDS) ve Dağıtılmış IDS (DIDS). HIDS, müdahaleci olayları tespit etmek için toplanan verileri izlemek ve analiz etmek için belirli bir ana bilgisayar üzerinde çalışır. NIDS, ağdaki tüm trafiği izleyerek ve

yakalayarak kilit ağ noktalarındaki izinsiz giriş saldırıları tespit eder. DIDS, hem HIDS hem de NIDS kullanır. Tespit yöntemleri ile ilgili olarak, literatürde, imza bazlı tespit, anomali bazlı tespit ve hibrit tespit olmak üzere üç iyi bilinen teknik vardır. İmza tabanlı algılama veya yanlış kullanım tespiti, toplanan verileri, bilinen saldırıların veya önceden tanımlanmış bir kurallar kümesinin bir veri tabanıyla eşleştirerek izinsiz giriş saldırıları tanımlar. Önceki tekniğin ana dezavantajı, yalnızca bilinen saldırıları tespit edebilmesidir. Öte yandan, anomali temelli tespit, toplanan verileri yerleşik bir temele göre karşılaştırarak izinsiz giriş saldırıları tespit eder. Mevcut etkinlik normal durumdan saparsa, olası kötü amaçlı saldırı alarmını verir. Anomaliye dayalı tespitin bilinen veya bilinmeyen atakları tespit edebilmesinin temel avantajı. Son olarak, imza tabanlı ve anomaliyi bir arada bir araya getiren karma algılama. Hibrit tespit kullanan IDS'nin diğer tekniklerden daha iyi sonuçlar elde ettiği bildirilmiştir (Patel vd. 2013).

Literatürde, bulut bilişime izinsiz girişi saldırı tespit etmek için birçok IDS önerilmiştir. Buna rağmen, mevcut bulut tabanlı IDS'lerin çoğu, bulut ağında yüksek hızda erişime uyum sağlama yeteneği, yeni ve dağıtılmış saldırıları tespit etme güvenlik açığı, kullanıcıların kontrolünün eksikliği gibi bazı sınırlamalara sahiptir (Gul ve Hussain, 2011). Bu nedenle, tüm bu ortaya çıkan sorunları dikkate alarak güvenilir bir bulut tabanlı IDS geliştirmeye ihtiyaç vardır.

### 5.2. Özellik Çıkarma

Makine öğrenmesi temelli güvenlik yaklaşımları, meşru bir savunucunun yanlış sınıflandırılmasından veya eğitim veri setini kirleterek tespit edilmekten kaçınmak isteyen saldırganların neden olabileceği zehirli veri setlerine karşı savunmasızdır. Laboratuar ortamı ile gerçek dünya arasında da belirgin boşluklar var. Bulut tabanlı herhangi bir IDS'nin bir parçası olarak bir denetim sistemine ihtiyaç duyulmasını gerektiren eski gerçeğe önem vermek. Tespit yönteminin türünden bağımsız olarak, denetim sistemi bulutun tüm altyapısından geçen veri akışını yakalar. Daha sonra, toplanan akış, özellik çıkarımı olarak adlandırılan önemli bir işlem gerçekleştirilerek analiz

edilmelidir. Daha sonra, bu çıkarılan özellikler eğitim veya bina tahmin modelleri için kullanışlıdır.

Temel olarak, "özellik" terimi kayıttaki bilgilerin belirli bir yönünü ifade eder. Bu nedenle, özellik çıkarımı, her bir paket için ayrı ayrı trafik akışının bir açıklayıcı istatistik setinin çıkarıldığı bir süreçtir. Bundan sonra, bu çıkarılan özellikler bir arada, bu kaydın her alanının çıkarılmış özelliklerden birini temsil ettiği yerde kaydedilir. Bu şekilde devam ederek, eğitim veya test amacıyla farklı veri paketlerinin bir dizi kaydı elde edilir ve bir veri tabanında veya veri deposunda saklanır. Açıkçası, toplanan ham verilerden hangi özelliklerin çıkarılacağının belirlenmesi büyük bir ikilemdir. Çünkü yalnızca kaynak veri paketlerinin aktivitesini başarılı bir şekilde karakterize edebilecek özellikleri çıkarmak çok önemlidir. Bu, paketin kötü niyetli bir faaliyet olup olmadığını belirlemeye yönelik tahmin modellerine yardımcı olacak ve ardından hangi trafiğin geçebileceğine ve hangi trafiğin engelleneceğine buna göre bir karar alınacaktır. Bu alt bölümde, önerilen IDS'nin denetim sistemi tarafından çıkarılacak bir dizi özellik belirtiyoruz. Bu özellikler iyi bilinmektedir ve literatürdeki önceki çalışmaların bazılarında bildirilmiştir.

Lashkari vd. (2017), Kanada Siber Güvenlik Enstitüsü web sitesinde kamuya açık olan CICFlowMeter olarak bilinen yeni bir yazılım sundu (CICFlowMeter, 2017). CICFlowMeter kullanarak, Tor akışlarını oluşturdular ve Tor trafiğini tanımlamak ve karakterize etmek için zamana bağlı bir dizi özellik önerdiler. Buna ek olarak, zamanla ilgili özellikleri kullanarak sadece Tor trafiğini bir dereceye kadar tanımlayabildiklerini ve niteleyebildiklerini kanıtladılar.

Bundan sonra, Sharafaldin vd. (2018) önceki çalışmayı genişletmiş ve CICIDS2017 (Kanada Siber Güvenlik Enstitüsü - IDS 2017, 2017) adlı yeni ve güvenilir bir IDS veri seti yayınlamıştır. Alt bölüm 4.2.2'de belirtildiği gibi, CICIDS2017 tamamen veri setinde etiketlenmiştir ve CICFlowMeter kullanarak *pcap* ham verilerinden elde edilen 80 ağ trafiği özelliğine sahiptir. Ayrıca, çeşitli uygulamalara ve ağ protokollerine dayanan hem iyi hem de müdahaleci akışlar için özellikler çıkarılır ve hesaplanır. Çok çeşitli ortak saldırı senaryolarını ele

almak için Brute Force, Heartbleed, Botnet, DoS, DDoS, Web ve Infiltration olmak üzere yedi ana saldırı ailesine veya kategorisine ayrılabilen 20 farklı saldırı tipi uyguladılar. Dahası, yedi saldırı kategorisinin tümü için ayarlanmış en iyi algılama özellikleri olabilecek en iyi kısa bir özellik alt kümesini seçmek için RandomForestRegressor kullanarak 80 çıkarılan özelliği test ettiler. Dolayısıyla, 23 özellikten oluşan bir alt kümenin tüm saldırı tipleri için en uygun özellik olduğunu bildirmişlerdir (Sharafaldin vd., 2018). Yermi uç optimal özelliklerin listesi ve bunlara karşılık gelen saldırı kategorileri ve protokolleri Çizelge 5.1'in ilk satırında gösterilmiştir. İleri terimi, kaynaktan hedefe giden trafik akışı anlamına gelir. Oysa, Geriye dönük terim ters yönde, yani hedeften kaynağa olan trafik akışını belirtir. Son olarak, Akış terimi, her iki yönde de iki yönlü bir akış anlamına gelir.

Benzer şekilde, Coburg İzinsiz giriş saldırı Tespiti Veri Seti (CIDDS), anomalliğe dayalı IDS'nin değerlendirilmesi için etiketli bir akış bazlı veri setidir ve ayrıca İnternet üzerinden kamuya açıktır (CIDDS-Coburg Intrusion Detection Data Sets, 2017). Gerçekten de, CIDDS'nin CIDDS-001 (Ring vd., 2017a) ve CIDDS-002 (Ring vd., 2017b) olmak üzere iki sürümü vardır. CIDDS-001'in toplam 92 saldırı ile çeşitli saldırı türlerine sahip olduğu arasındaki temel fark, bu arada CIDDS-002, yalnızca toplam 43 port tarama saldırısı olan veri kümesi bağlantı noktası tarama saldırısıdır. Ayrıca, tek yönlü *NetFlow* denetim verilerini topladılar ve 10 farklı özelliği ve uygun etiket için ek bir özellik elde etmek için bunları analiz ettiler. CIDDS-001'deki 92 saldırı aynı zamanda DoS, Brute Force, Port Scan ve Ping Scan gibi dört ana saldırı kategorisinde gruplandırılmıştır. Bu nedenle, kalan özelliğin çoğaltıldığı Çizelge 5.1'deki ikinci sıraya dokuz CIDDS özelliği dahil ettik.

Dahası, NSL-KDD iyi bilinen bir IDS veri setidir ve izinsiz giriş saldır tespit alanında yaygın olarak kullanılmaktadır (Tavallaee vd., 2009; NSL-KDD Dataset, 2009). Alt bölüm 4.2.1'de belirtildiği gibi, NSL-KDD, KDD CUP 99 veri setinin tüm sınırlamalarını çözen geliştirilmiş bir versiyondur (Stolfo vd., 2000; UCI Machine Learning Repository: KDD CUP 1999 Data Set, 1999). *Tcpdump* raw verilerini yakaladılar ve 41 farklı özellik çıkarmak için işlediler.

Çizelge 5.1. Alınan özelliklerin listesi ve karşılık gelen saldırı aileleri ve protokolleri

| Özellikleri | Saldırı kategorileri | Protokoller |
|---|---|---|
| Backward Packet Length Minimum<br>Subflow Forward Bytes<br>Total Length Forward Packets<br>Forward Packet Length Mean<br>Backward Packet Length Standard deviation<br>Flow Inter Arrival Time Minimum<br>Forward Inter Arrival Time Minimum<br>Flow Inter Arrival Time Mean<br>Flow Duration<br>Flow Inter Arrival Time Standard deviation<br>Active Minimum<br>Active Mean<br>Backward Inter Arrival Time Mean<br>Initial Window Forward Bytes<br>Acknowledge Flag Count<br>Forward Push Flags<br>Synchronization Flag Count<br>Forward Packets/second<br>Initial Window Backward Bytes<br>Backward Packets/second<br>Push Flag Count<br>Average Packet Size<br>Forward Inter Arrival Time Mean | Brute Force<br>Heartbleed<br>Botnet<br>DoS<br>DDoS<br>Web Attack<br>Infiltration | HTTP<br>HTTPS<br>FTP<br>SSH<br>Email protocols |
| Source IP Address<br>Source Port<br>Destination IP Address<br>Destination Port<br>Transport Protocol<br>Start Time Flow First Seen<br>Number of Submitted Bytes<br>Number of Transmitted Packets<br>TCP Flags | DoS<br>Brute Force<br>Port Scan<br>Ping Scan | TCP<br>UDP<br>ICMP |
| Network Service<br>Number of Received Bytes<br>Source Connections Count<br>Synchronization Flag Error Rate<br>Different Services Rate<br>Destination Connections Count<br>Reject Flag Error Rate<br>Rate of connection to different destinations | DoS<br>Probe<br>R2L<br>U2R | SMTP<br>HTTP<br>FTP<br>Telnet<br>ICMP<br>SNMP |

Buna ek olarak, NSL-KDD: DoS, Probe, R2L ve U2R olmak üzere dört ana saldırı kategorisinde birleştirilen 38 tip saldırıya sahiptir. Her saldırı kategorisine

uygun özellik alt kümelerini seçmek için NSL-KDD'ye bir özellik seçim süreci gerçekleştirilmiş birçok çalışma vardır (Natesan ve Balasubramanie, 2012; Wahba vd., 2015; Naidoo vd., 2015; Aminanto ve Kim, 2016; Ambusaidi vd., 2016). Bu özellik alt kümelerini çoğaltılmış öznitelikleri çıkardıktan sonra Çizelge 5.1'in üçüncü satırında tamamladık.

Bu çalışmada, yukarıda tartışılan önceki çalışmalarda kullanılan prosedürlerin aynısını kullanarak, denetim sistemimizdeki trafik akışlarından hesaplanıp çıkartılmak üzere Çizelge 5.1'deki 40 özelliğin tümünü seçtik. Bu özellik kümesinin, kullanıcıların yaygın saldırı ailelerini kapsayacak şekilde farklı etkinliklerini temsil edebildiğine inanıyoruz. Son olarak, Gharib vd. (2016), IDS veri setleri için geçerli bir veri kümesinin özelliklerini yansıtan bir değerlendirme çerçevesi önermiştir ve Komple Ağ Konfigürasyonu, Komple Trafik, Etiketli Veri Kümesi, Komple Etkileşim, Komple Yakalama, Mevcut Protokoller, Saldırı Çeşitliliği, Anonimlik, Heterojenlik, Özellik Seti ve Meta Veriler gibi on bir özellikten oluşur. Seçilen özelliklerin yukarıda belirtilen değerlendirme çerçevesine tamamen uygun olduğu gösterilmiştir (Garib vd., 2016).

### 5.3. Topluluk Sistemi

Topluluk sistemi veya topluluk öğrenme olarak da bilinen, gelişmiş bir bileşik model oluşturmak amacıyla çeşitli sınıflandırıcıların kombinasyonunun kullanıldığı bir tekniktir (Dasarathy ve Sheela, 1979). Bir topluluk sisteminin yapısı ya farklı türdeki sınıflandırıcılara (heterojen topluluk) ya da aynı türdeki (homojen topluluk) sınıflandırmaya dayalı olabilir (Folino ve Sabatino, 2016). Bu tür bir topluluk sistemini kullanmanın asıl yararı, doğruluğu ve sağlamlığı, daha iyi genelleme ve daha fazla veri işleme yeteneğini arttırmaktır. Topluluk sisteminin arkasındaki ana fikir, tüm birleşik sınıflandırıcıları eğitim setinde aynı anda eğitmektir. Daha sonra, görünmeyen her veriyi bir dizi öğrenilen sınıflandırıcıya karşı, her sınıflandırıcının kararını vereceği şekilde test etmek. Daha sonra, nihai kararı almak için kararları bir araya getirilir. Literatürde, tüm kararları bir araya getirmek için iki yöntem vardır, yani Torbalama metodu ve

Artırma metodu. Torbalama metodu, tüm sınıflandırıcıları bir sınıflandırıcı topluluğu üzerindeki tahminin ortalamasını alarak birleştirir. Oysa Artırma yöntemi, bir sınıflandırıcı koleksiyonu ile oy ağırlıklı olarak gerçekleşti.

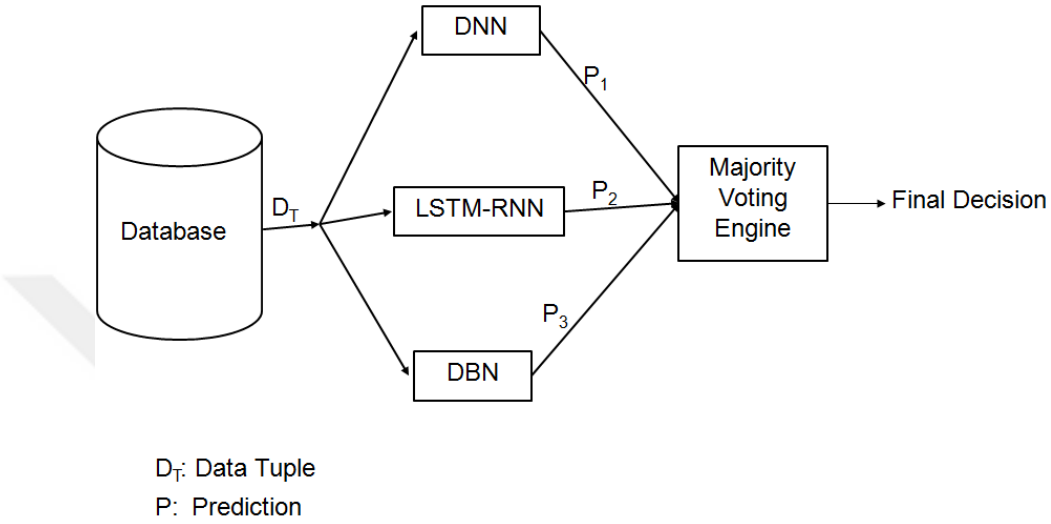
Son zamanlarda, izinsiz giriş saldırı tespit alanındaki topluluk sisteminden yararlanmak, Ji ve Ma (1997), Dzeroski ve Zenko (2002), Chebrolu vd. (2005), Borji (2007), Ludwig (2017) ve Marir vd. (2018). Önceki çalışmaların deneysel sonuçları, topluluk temelli yaklaşımın, tek sınıflandırıcı ile karşılaştırıldığında daha güvenilir sonuçlara ulaştığını göstermiştir.

Bu çalışmada, bulut tabanlı IDS'imizdeki izinsiz giriş saldırıları tespit etmek için Torbalama topluluk sistemini kullanmaya karar verdik. Önerilen Torbalama topluluk sistemi, Şekil 5.1'de gösterildiği gibi iki bitişik parçadan oluşur. İlk bölüm, DNN, LSTM-RNN ve DBN olmak üzere üç derin öğrenme modelinden oluşur. Diğer derin öğrenme modellerinden ziyade bu modellerin seçilmesinin nedenleri şunlardır: i) 3. ve 4. bölümdeki bulgularımız hem maskeli saldırı tespiti hem de ağ saldırı tespitinde etkinliklerini doğruladı; ii) birçok inceleme makalesi izinsiz giriş saldırı tespit probleminin çözülmesindeki başarısına dikkat çekmiştir (Aminanto ve Kim, 2016; Vani, 2017); iii) izinsiz giriş saldırı tespiti gibi statik sınıflandırma görevlerinde ortaktırlar. Öte yandan, ikinci kısım, tek tek modellerin tüm oylarını çoğunluğa (en fazla oylama) bağlı olarak nihai karara bağlamak için kullanılan bir oy çoğunluğudur. Çizelge 5.2, çoğunluk oylama motorunun doğruluk tablosunu sunmaktadır. Kesin olarak, son kararı alırken çoğunluk güvence altına alınmıştır, çünkü uç modelimiz var. Son olarak, böyle bir Torbalama topluluk sistemi kurmanın, diğer derin öğrenme modelleri tarafından tek bir derin öğrenme modelinin hatalarını telafi etmenin faydalı olduğuna inanıyoruz.

### 5.4. Önerilen Bulut Tabanlı IDS

Bulut bilişimin açık ve dağıtılmış doğası ve hizmet odaklı paradigması, bulut altyapısını çeşitli potansiyel saldırılara karşı çok savunmasız hale getiriyor. Geleneksel IDS'ler bu tür çevre ve tehditler için yetersizdir. Bu nedenle, tasarımı

sırasında ortaya çıkan tüm sorunları ve zorlukları göz önünde bulunduran entegre bir bulut tabanlı IDS geliştirmeye gerçek bir ihtiyaç vardır. Entegre bir IDS, tek bir platform üzerinden iletişim kurmak için ünlü çözümler içermesi anlamına gelir. Bu bölümde, tümleşik bulut tabanlı IDS'imizin tasarımını öneriyoruz.

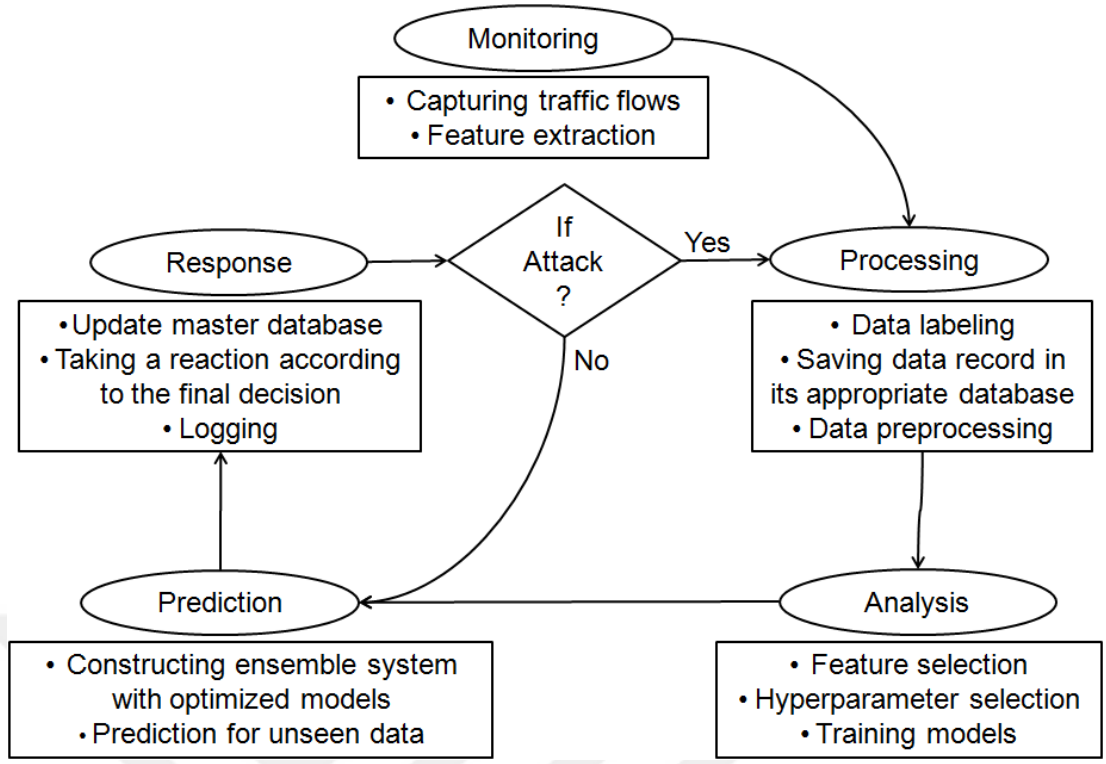



Şekil 5.1. Önerilen torbalama topluluk sisteminin temel yapısı

Çizelge 5.2. Çoğunluk oy motorunun doğruluk tablosu

| Tahminler | | | Son karar |
|---|---|---|---|
| **DNN** | **LSTM-RNN** | **DBN** | |
| Normal | Normal | Normal | Normal |
| Normal | Normal | Saldırı | Normal |
| Normal | Saldırı | Normal | Normal |
| Normal | Saldırı | Saldırı | Saldırı |
| Saldırı | Normal | Normal | Normal |
| Saldırı | Normal | Saldırı | Saldırı |
| Saldırı | Saldırı | Normal | Saldırı |
| Saldırı | Saldırı | Saldırı | Saldırı |

Şekil 5.2, önerilen bulut tabanlı IDS'nin akış şemasını göstermektedir. İzleme, işleme, analiz, tahmin ve yanıt modülleri olmak üzere beş ana modülden oluşur. Her bir modülün detayları gelecek alt bölümlerde ele alınmaktadır.




Şekil 5.2. Önerilen bulut tabanlı IDS'nin akış şeması

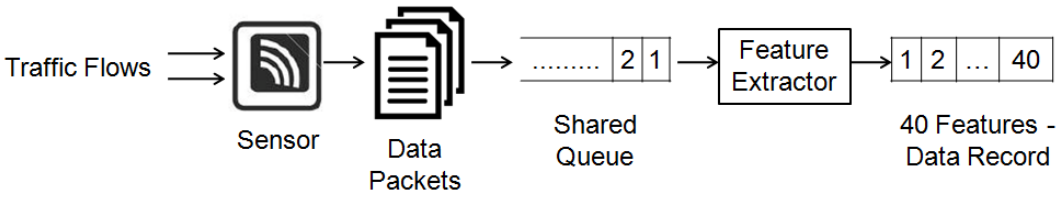
### 5.4.1. İzleme modülü

Öncelikle, bulut kullanıcıları bulut ağ üzerinden CSP'nin sitesindeki uzak sunuculardaki verilerine ve uygulamalarına erişirler. Bu nedenle, kullanıcıların eylemleri ve istekleri bir denetim sistemi aracılığıyla izlenmeli ve kaydedilmelidir. İzleme modülü, önerilen bulut tabanlı IDS'nin denetim sistemi olarak kabul edilir. Trafik akışlarını yakalamak ve özellik çıkarımı olmak üzere iki ayrı işlevi yerine getirir.

İzleme modülünün ilk işlevi, bulut ağı içinde dolaşan tüm dahili ve harici veri paketlerini yakalamaktır. Bu görevi kolaylaştırmak için izleme modülü ağ trafiğini hassaslaştırmak için sensörler kullanır. Ayrıca, izleme modülü, TCP, UDP, ICMP, IP, HTTP, SMTP, vb. gibi farklı uygulama, taşıma ve ağ protokollerinin veri paketlerini yakalama esnekliğine sahiptir. Toplanan trafik akışları derhal paylaşılan sıraya kaydedilir. Paylaşılan kuyruğu kullanmanın amacı, paket yakalama ve özellik çıkarımı arasında bir ara istasyon olmak, yani toplanan veri paketlerini, özellik çıkarıcı bunları sırayla işleyene kadar

depolamaktır. Paylaşılan kuyruğu, toplanan veri paketlerini taşmadan depolamak için depolama kapasitesi açısından yeterli alana sahiptir.

Daha sonra, özellik çıkarma işlemi bir özellik çıkarıcı aracılığıyla gerçekleştirilir. Özellik çıkarıcımız, paylaşılan kuyruktaki her veri paketini, alt bölüm 5.2'de belirtildiği gibi veri paketinden 40 özellik çıkaracak şekilde işler. Daha sonra, veri paketinin çıkarılan tüm özelliklerini bir veri kaydına veya dizgisine yerleştirir ve sonraki modüle, yani işleme modülüne gönderir. Bu veri kayıtları, sahnenin özü ve eğitim ve öngörü için ihtiyaç duyduğumuz hammaddedir. Şekil 5.3, izleme modülünün işlemini göstermektedir. İzleme modülünün çalışmasının kalıcı, sürekli ve diğer modüllerden bağımsız olduğunu söylemek önemlidir.




Şekil 5.3. İzleme modülünün süreci

### 5.4.2. İşleme modülü

İşleme modülü, öngörü işleminden önce gerekli tüm görevleri yerine getirir. Öncelikle, özellik çıkarıcısından veri kayıtları alır ve bunları uygun sınıf etiketine ("Normal" veya "Saldırı") atamaya çalışır. Bu, denetlenen veri kaydını bilinen saldırı modelleriyle eşleştirmek için önceden tanımlanmış bir dizi kural (imza) kullanan imza bazlı bir algılama prosedürü kullanılarak gerçekleştirilebilir. Bir eşleşme varsa, veri kaydı saldırı olarak etiketlenir. Aksi takdirde, normal olarak etiketlenir. Bu kurallar bulut servis sağlayıcısı tarafından tanımlanır ve belirlenir. Bundan sonra, işlem modülü etiketli veri kaydını "ana veri tabanı" olarak adlandırdığımız bir veri tabanında saklar.

Önceki süreç sadece sistemin ilk çalışmasında gerçekleşecek. Başka bir deyişle, ilk çalıştırmada, işlem modülü sıfırdan ana veri tabanını yaratacak, veri kayıtlarını alacak, yukarıda belirtilen yöntemi kullanarak veri kayıtlarını

etiketleyecek ve etiketli veri kayıtlarını sırayla ana veri tabanına kaydedecektir. Ana veri tabanı çeşitlilik ve miktar bakımından yeterince zenginleştiğinde, işlem modülü önceki işlemi durduracak ve "görünmeyen veri tabanı" olarak adlandırdığımız sıfırdan başka bir veri tabanı oluşturacak. Ardından, gelen tüm yeni veri kayıtları, etiketlenmeden görünmeyen veri tabanına kaydedilecektir. Buna göre, etiketli veri kayıtlarını içeren ana veri tabanı, bulut tabanlı IDS'nin eğitim seti olarak kullanılacaktır. Öte yandan, etiketlenmemiş veri kayıtlarını içeren görünmeyen veri tabanı test seti olacaktır.

İşleme modülünün en son işlevi veri ön işlemedir. İşlem modülü ana veri tabanına doldurma işlemi bittiğinde gerçekleşir. Veri ön işleme, alt bölüm 4.4.1.1'de açıkladığımız prosedür kullanılarak ana veri tabanında ele alınacaktır.

### 5.4.3. Analiz modülü

Analiz modülü, derin öğrenme modellerinin eğitim öncesi ve eğitim aşamalarını gerçekleştirir. Eğitim öncesi aşamada, analiz modülü, hem özellik hem de hiperparametre seçimi için alt bölüm 4.3.3'te önerdiğimiz çift PSO tabanlı algoritmayı yürütür. İlk olarak, analiz modülü, veri ön işleme işleminden sonra ana veri tabanının bir kopyasını alır. Daha sonra, optimum özellik alt kümesini bulmak için ana veri tabanında çift PSO tabanlı algoritmanın üst seviyesini çalıştırır. Ondan sonra, sadece optimal özellikleri kullanarak ana veri tabanını azaltır. Ardından, istenen derin öğrenme modelinin optimal hiperparametre vektörünü elde etmek için azaltılmış ana veri tabanını kullanarak çift PSO tabanlı algoritmanın alt seviyesini uygular. Üç derin öğrenme modelimiz olduğundan, ikinci adım DNN, LSTM-RNN ve DBN modelleri için sırasıyla üç kez tekrarlanacaktır. Daha sonra, eğitim aşaması, azaltılmış ana veri tabanında ayrı ayrı her öğrenme modeli için gerçekleştirilir.

### 5.4.4. Tahmin modülü

Tahmin modülü, iki sıralı işlevi gerçekleştiren bulut tabanlı IDS'nin özüdür. Bunlardan ilki, analiz modülünden elde edilen öğrenilmiş DNN, LSTM-RNN ve

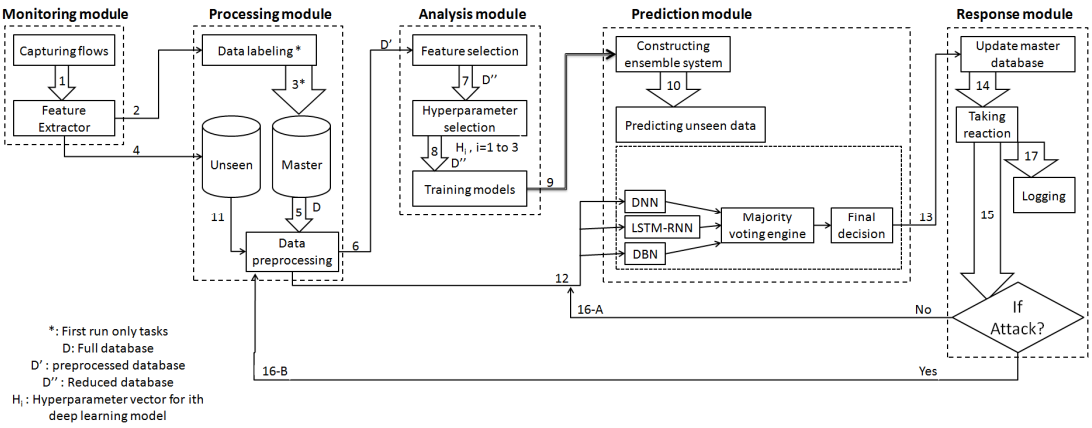
DBN modellerini kullanarak Bölüm 5.3'te önerdiğimiz Torbalama Topluluk sisteminin inşasıdır. Ardından, işleme modülü görünmeyen veri tabanının bir kopyasını alır ve üzerinde de ön işleme yapan bir veri gerçekleştirir. Tahmin modülünün ikinci işlevi, önceden işlenmemiş görünmeyen veri tabanından bir veri kaydını sırayla almaktır. Bundan sonra, toplanan veri kayıtları Torbalama Topluluk sistemi kullanılarak test edilir ve Torbalama Topluluk sisteminin çoğunluk oylama motorundan alınan son karar yanıt modülüne iletilir.

### 5.4.5. Yanıt modülü

Yanıt modülü, üç ana işlevi yerine getirerek teklif edilen bulut tabanlı IDS'nin tamamının tamamını kontrol etmekten sorumludur. İlk olarak, test edilen veri kaydı ile birlikte son kararı alır ve ardından test edilen veri kaydını son kararla etiketler. Bundan sonra, yeni etiketli veri kaydını kaydederek ana veri tabanını günceller. İkinci işlev, normal mı yoksa saldırı mı olduğuna karar vermesi üzerine tepki almaktır. Normal durumda, yanıt modülü, tahmin modülüne, etiketini tahmin etmek üzere görünmeyen veri tabanındaki bir sonraki veri kaydına devam etmesi için bir komut gönderir.

Öte yandan, alınan nihai karar saldırı ise, yanıt modülü, bulut ağının potansiyel bir kötü niyetli faaliyet altında olduğu konusunda bir alarm verir. Daha sonra, yanıt modülü saldırganın oturumunu sonlandırır ve aynı zamanda bu kötü niyetli faaliyete ait tüm trafik akışlarını engeller. Buna ek olarak, yanıt modülü işlem modülünü güncellenmiş ana veri tabanının yeni bir kopyasında ön işleme işlemi yapmak için zorlar ve önceden işlenmiş ana veri tabanını analiz modülüne gönderir. Ardından, analiz modülü yeniden optimize edilmiş yeni derin öğrenme modelleri üretmek için tekrar çalışır ve bunları Torbalama Topluluk sistemini tekrar oluşturan ve görülmeyen veri tabanından yeni bir veri kaydı tahmin eden tahmin modülüne gönderir.

Yanıt modülünün son işlevi, daha sonraki işlemler için tüm olayları özel bir günlük dosyasına kaydetmektir. Şekil 5.4, önerilen bulut tabanlı IDS çerçevesini göstermektedir.




Şekil 5.4. Önerilen bulut tabanlı IDS'in çerçevesi

## 5.5. Tartışma

Bu bölümde, önerilen bulut tabanlı IDS'nin avantajlarını ve IDS'imizin, Bölüm 2.3'te tartıştığımız mevcut bulut tabanlı IDS'lerin sınırlamalarını nasıl çözdüğünü sunuyoruz.

### 5.5.1. Hibrit IDS

Geleneksel IDS'lerin bulut ortamı için uygun olmadığı, çünkü imza bazlı tespit IDS'lerinin yeni veya bilinmeyen saldırıları tespit edemediği, bu arada, anomali bazlı tespit IDS'lerinin hesaplama açısından pahalı olduğu ve çok miktarda saf normal verilere ihtiyaç duyduğu bildirildi (Modi vd., 2013). Bölüm 5.4'te belirtildiği gibi, önerilen bulut tabanlı IDS iki algılama tekniğini bir araya getirir, işlem modülünde bilgiye dayalı algılama (önceki saldırıların bilinen izleri) gerçekleştirdiğinden ve tahmin modülünde davranışa dayalı (son kullanıcı eylemlerinin olağan davranışla karşılaştırılması) gerçekleştirir. Sonuç olarak, önerilen bulut tabanlı IDS, bilinen tüm saldırı imzalarını ve yeni tehditlerin bilgisini kapsayacaktır. Ayrıca, bir hibrit algılama IDS'sinin kullanılması doğruluğu artıracak ve yanlış alarm oranı önemli ölçüde azaltacaktır. (Gül ve Hüseyin, 2011).

### 5.5.2. Dinamik IDS

Önerilen bulut tabanlı IDS, üç farklı perspektifle dinamiktir. Birincisi, performans açısından geleneksel makine öğrenimi yöntemlerine göre üstünlüğü olan iyi bilinen üç derin öğrenme modelini birleştiren izinsiz giriş saldırı tespiti için bir Torbalama Topluluk sistemi kullanır. İkincisi, derin öğrenme modellerinin mimarisini dinamik olarak değiştiren hem özellik hem de hiperparametre seçimi için çift PSO tabanlı algoritmayı kullanır. Bu, analiz modülünün çift PSO tabanlı algoritmayı her çalıştırdığında, ana veri tabanından yeni bir optimum özellik alt kümesi oluşturması, ardından azaltılmış ana veritabanı olanların doğruluğunu maksimize eden her derin öğrenme modeli için optimal hiperparametreleri çıkarmasıyla açıklanabilir.

Üçüncü bakış açısı, yanıt modülünün ana veri tabanını periyodik olarak yeni öngörülen normal ve saldırı veri kayıtları ile güncellemesidir. Buna göre, bu ana veri tabanının gerçek dünya trafiğinin temsili bir veri kümesi olmasını sağlar. Ayrıca, bir saldırı durumunda, yanıt modülü analiz modülünü, görünmeyen bir veri kaydı için yeni bir tahmin oluştururken yeni özellik ve hiperparametre seçimi için çift PSO tabanlı algoritmayı tekrar yürütmek için zorlar.

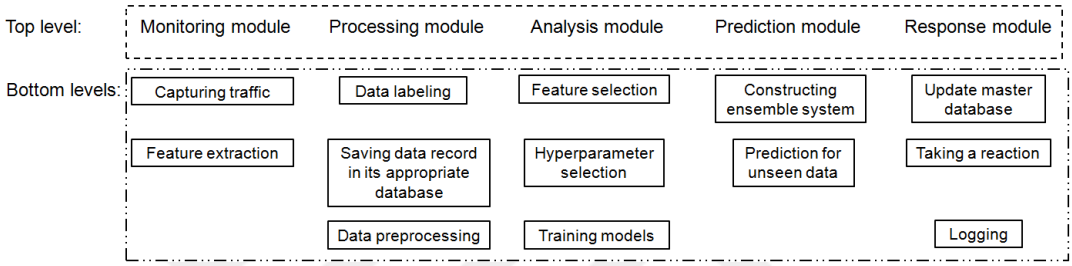
### 5.5.3. Çok okuyuculu IDS

Aynı derecede önemli olan, bulut bilişimin yüksek bir ağ erişim hızı nedeniyle büyük miktarda veri üretmesidir. Bu nedenle, bulut tabanlı IDS'ler yanlış pozitiflere ve gürültü verilerine karşı sağlam olmalıdır. Genel olarak, geleneksel IDS'ler aşağıdaki gibi birçok nedenden dolayı bulut benzeri ortamlar için yeterli değildir: bu IDS'ler tek dişlidir ve bulut bilgi işlem bu kadar büyük bir veri akışını gerçekleştirememeye mahkum olan ve zengin veri kümesi nedeniyle büyük bir ağ trafiğine sahiptir akış (Gül ve Hüseyin, 2011).

Bu endişenin üstesinden gelmek için, bulut tabanlı IDS'imizi çok iş parçacıklı bir IDS olacak şekilde tasarladık, yani her modül aynı sunucudaki bağımsız bir iş parçacığı içinde ayrı ayrı çalışır. Bu, çok iş parçacıklı bulut tabanlı IDS'nin büyük

miktarda veri işleyebilmesini, paket kaybını azaltmasını ve VM barındıran IDS'nin aşırı yüklenmesini önlemesini sağlar. Sonuç olarak, önerilen bulut tabanlı IDS, bulut altyapısı üzerindeki performansı artıracak.

Daha fazla gelişme uğruna, bulut tabanlı IDS tasarımımızdaki çoklu kullanım seviyelerine de ekledik. Üst seviye tüm ana modülleri içerirken, alt seviye her modülün tüm fonksiyonlarını içerir. Şekil 5.5, önerilen çok iş parçacıklı bulut tabanlı IDS'nin mekanizmasını göstermektedir. Son olarak, böyle bir Veli-Çocuklar hiyerarşisinin hızlı bir işlem sağlayacağına inanıyoruz.



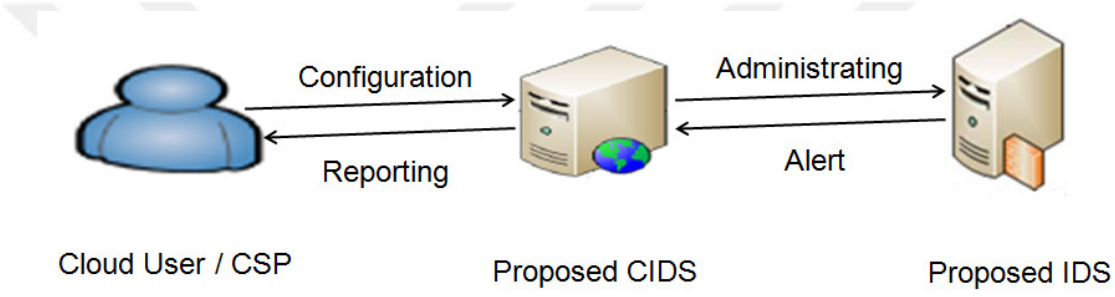

Şekil 5.5. Önerilen bulut tabanlı IDS'de çok okuyuculu mekanizması

## 5.6. Önerilen CIDS

Geleneksel IDS'lerin bir başka eksikliği, kullanıcının kontrolünün olmamasıdır, çünkü genellikle geleneksel IDS'ler bulut servis sağlayıcısı tarafından izlenir ve yönetilir. Bu nedenle, bir davetsiz misafir kullanıcının verilerine girip onu çalmayı veya ona zarar vermeyi başarırsa, kullanıcıya doğrudan bildirimde bulunulmaz. Bunun yerine IDS, CSP'ye izinsiz giriş saldırıyle ilgili tüm bilgileri gönderir ve bulut kullanıcısı ona güvenmek zorunda kalır. İmajı ve şöhreti uğruna, CSP bulut kullanıcısını kayıp hakkında bilgilendirmeyebilir ve bilgileri gizleyebilir. Böyle bir senaryoda, tarafsız bir bulut hizmeti, bulut kullanıcıları için yeterli uyarı ve izleme sağlayabilir.

Önceki duruma önem vererek, teklif edilen bulut tabanlı IDS'yi izleyebilecek ve hem bulut kullanıcıları hem de CSP için uyarı raporları sağlayabilecek bir üçüncü taraf bulut hizmeti öneriyoruz. Aslında, yönetim ve raporlama olmak üzere iki ana işlevi olan web tabanlı bir bulut hizmetidir. İdare işlevi, önerilen

bulut hizmetinin, önerilen bulut tabanlı Kimlikleri belirli ayrıcalıklara göre yapılandırmalarına izin vermek için hem kullanıcı hem de CSP ile ilgilenebileceği anlamına gelir. Öte yandan, bir saldırganın kullanıcının verileri veya uygulamaları üzerinde kötü amaçlı bir etkinlik yapmayı başarması durumunda, önerilen bulut hizmeti, günlük dosyasında en son bilgileri içeren yanıt modülünden bir uyarı alır. Bu nedenle, önerilen bulut hizmeti ayrıntılı bir rapor hazırlar ve doğrudan bulut kullanıcısına ve CSP'ye de gönderir. Şekil 5.6, üçüncü taraf bulut hizmetinin ana işlevlerini göstermektedir. Üçüncü taraf CIDS'i kullanarak, bilgilerin şeffaflığı ve verilerin güvenliğinin sağlanabileceğine inanıyoruz.

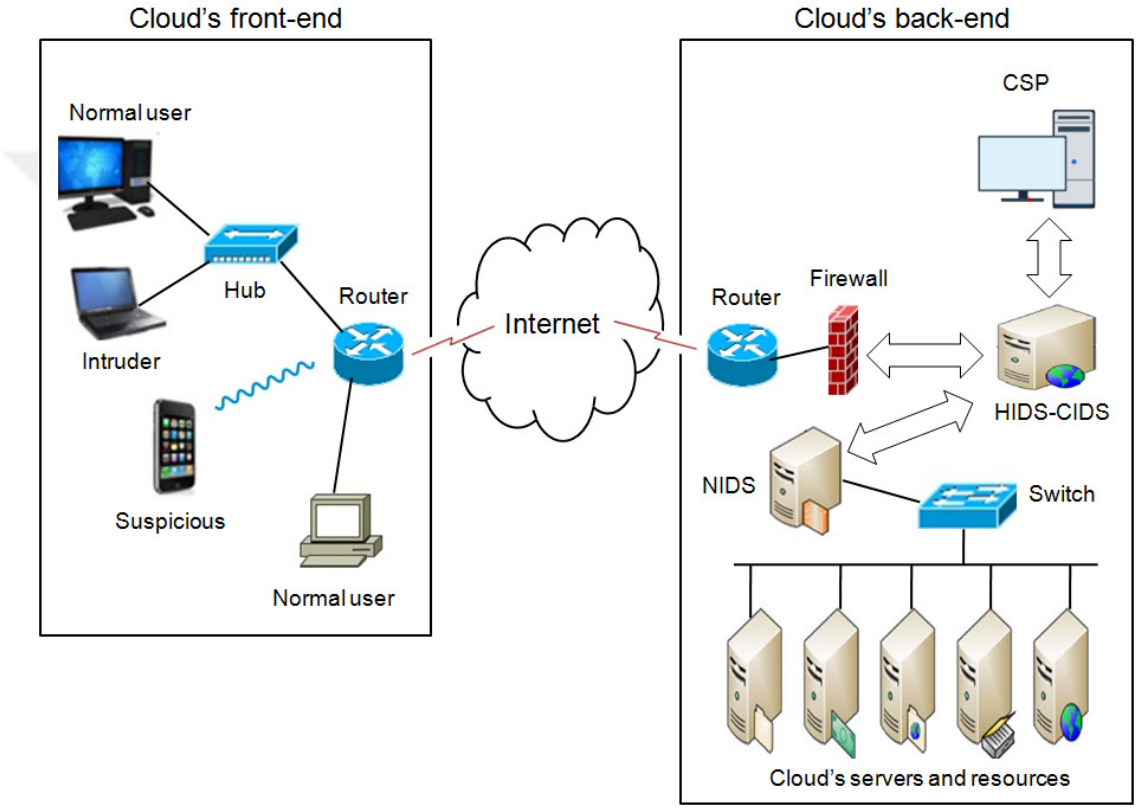



Şekil 5.6. Üçüncü taraf bulut hizmetinin ana işlevleri

Bulut altyapısı içindeki IDS türü ve konumu, bulut tabanlı bir IDS tasarlarken ortaya çıkan çok kritik konulardır. Temel olarak, bulut bilgi işlem, birçok bulut kullanıcısının bulut veri merkezindeki bir fiziksel makine "hiper yönetici sunucusu" üzerinde barındırıldığı kaynakların "sanallaştırılması" kavramını kullanır. IDS'yi hipervizörde HIDS olarak kullanmak, CSP'nin bu hipervizördeki VM'leri izlemesine izin verecektir. Ancak muazzam miktarda trafik ve veri akışı olduğunda, IDS'yi barındıran VM'yi düşüren ve aşırı yükleyen veri paketleri olacaktır. Ayrıca, rahatsız edici bir saldırgan ana makineyi tehlikeye sokarsa, bu ana makinede barındırılan HIDS nötrleştirilir. Bu nedenle, IDS'yi NIDS olarak dağıtmak bulut ortamlarında en iyi seçenek olacaktır (Gul ve Hussain, 2011).

Buna göre, önerilen bulut tabanlı IDS bir NIDS olacak şekilde tasarladık ve bunu dahili ve harici izinsiz giriş saldırı tespiti için bulut altyapısının arka tarafına yerleştirdik. Daha spesifik olarak, önerilen bulut tabanlı IDS, bulut

sunucularının bulunduğu yerdeki anahtar gibi anahtar ağ boğulma noktası üzerindeki VM sunucularının dışına yerleştirilecektir. Sistemin bu küresel görüşü, önerilen bulut tabanlı IDS'nin ağ trafiğini verimli bir şekilde izlemesini sağlayacaktır. Buna ek olarak, önerilen bulut hizmetini, bulut altyapısının en üstüne (doğrudan bulutun sonundan sonra) yönlendirici veya ağ geçidi gibi ağ noktalarının şişe boynuna yerleştirdik. Şekil 5.7, önerilen bulut tabanlı IDS ve üçüncü taraf bulut hizmetinin mimarisini göstermektedir.




Şekil 5.7. Önerilen bulut tabanlı IDS ve üçüncü taraf bulut hizmetinin mimarisi

Şekil 5.7'den görülebileceği gibi, önerilen bulut hizmetini bulutun üstünde HIDS olarak yerleştirmek ve önerilen bulut tabanlı IDS'yi bulutun arka ucundaki merkezi noktada NIDS olarak yerleştirmek, tasarımımızı şu şekilde yapıyor, bulutun tüm bölgelerine dağılmış bir yaklaşım.

# 6. SONUÇ VE ÖNERİLER

Bu bölümde, gelecekteki çalışmalar için önerilen bazı öneriler sunmanın yanı sıra işimizden sonuçlar çıkardık.

## 6.1. Sonuç

Maskeli saldırı tespiti, bilgisayar güvenliği alanındaki en önemli konulardan biridir. Çeşitli araştırma çalışmaları bile, on yıldan uzun bir süre boyunca maskeli saldırı algılamaya odaklanmıştır, ancak bu alanda derin öğrenme modellerini kullanan derin bir çalışmanın varlığı nadirendir. Bölüm 3'te, DNN, LSTM-RNN ve CNN modellerini kullanarak maskeli saldırıların tespiti için kapsamlı bir deneysel çalışma sunduk. Literatürde en çok kullanılan üç UNIX komut satırı veri setini kullandık. Buna ek olarak, bu veri setlerinden altı farklı veri yapılandırması yaptık. Bu veri konfigürasyonları üzerindeki maskeli saldırı algılama, iki yaklaşım kullanılarak gerçekleştirilir: birincisi statik ve ikincisi dinamiktir. Bu arada statik yaklaşım, statik sayısal özelliklere sahip veri konfigürasyonlarında uygulanan DNN ve LSTM-RNN modelleri kullanılarak gerçekleştirilirken, dinamik yaklaşım kullanıcının komut metni dosyalarından dinamik olarak kullanıcının özelliklerini alan CNN modeli kullanılarak gerçekleştirilir.

Hiperparametre seçimi problemini çözmek ve aynı zamanda yüksek performans elde etmek için, DNN hiperparametrelerini optimize etmek için PSO tabanlı bir algoritma da önerdik. Önerilen PSO tabanlı algoritma, doğruluğu en üst düzeye çıkarmaya çalışır ve hem DNN hem de LSTM-RNN modellerinin deneylerinde kullanılır. Ayrıca, kullanılmış modellerin performansını değerlendirmek için on iki tanınmış değerlendirme ölçütünü ve istatistiksel testleri kullandık ve performans sonuçlarını ve ROC eğrileri analizini kullanarak deney sonuçlarını analiz ettik. Elde ettiğimiz sonuçlar, kullanılan modellerin, kullanılan veri setleriyle ilgili maskeli saldırı algılamada başarı sağladığını ve tüm değerlendirme ölçütleri bakımından tüm geleneksel makine öğrenme yöntemlerinin performansını geride bıraktığını göstermektedir. Ayrıca, CNN

modeli tüm veri yapılandırmalarında hem DNN hem de LSTM-RNN modellerinden daha üstündür; bu, dinamik maskeli saldırı algılamasının statik olandan daha iyi olduğu anlamına gelir. Bununla birlikte, sonuç analizleri tüm kullanılmış modellerin maskeli saldırı tespitindaki etkinliğini, Doğruluk ve Vuruş oranı ile 1% ila 10% oranında artırdıkları birlikte FAR yüzdelerini 1% ila 10% oranında azalttıkları şekilde kanıtlamıştır. Son olarak, sonuçlara göre, derin öğrenme modellerinin siber güvenlik alanında kullanılabilecek çok umut verici araçlar gibi göründüğünü söyleyebiliriz.

Dörtüncü bölümde, üç derin öğrenme modeli kullanarak ağ saldırı tespiti üzerine kapsamlı bir deneysel çalışma sunduk: DNN, LSTM-RNN ve DBN. Bu modeller çift PSO tabanlı bir algoritmadan yararlanılarak ön eğitilmiştir. Önceki algoritma iki seviyeden oluşur: verilen veri seti için en uygun özellik alt kümesinin seçildiği üst seviye, daha sonra alt seviyede, indirgenmiş veri setindeki doğruluğu en üst düzeye çıkaran modelin optimize edilmiş hiperparametreleri vektörü otomatik olarak belirlenir. Buna ek olarak, deneylerimizde NSL-KDD ve CICIDS2017 olarak iki ortak IDS veri seti kullanılmıştır. Birincisi, bir referans noktası olarak kabul edilir ve literatürde geniş çapta kullanılır, ikincisi ise en güncel güvenilir NIDS veri setidir.

Ayrıca, deneylerimizi iki farklı perspektifte, ikili ve çok sınıflı sınıflandırmalarda yaptık. Modellerin performansı hakkında derin bir fikir vermek için, deney sonuçlarını performans analizi, Friedman istatistik testi ve iki sıralama yöntemi kullanarak analiz ettik. Sonuçlarımız, yaklaşımımızın eğitim öncesi aşamada olmayan modellerin performansına kıyasla ağ saldırı tespitinde bir başarı sağladığını gösteriyor. Dahası, derin öğrenme yaklaşımımız, önceki çalışmalarda derin öğrenme modellerinin performansını daha iyi bir performansa soktu ve aynı zamanda çoğu durumda yanlış alarm oranını 1% ila 5% arasında düşürmenin yanı sıra doğruluk ve algılama oranını 4% ila 6% arasında artırma yeteneğini kanıtladı.

Beşinci Bölümde, bulut bilişimde izinsiz giriş saldırı tespiti alanında ortaya çıkan sorunları ve zorlukları araştırdık. Geleneksel ve varolan IDS'lerin çoğunun

bulut benzeri ortamlar için yeterince yetersiz olduğunu gördük. Bu nedenle, literatürdeki mevcut IDS'lerin tüm sınırlamalarını çözmek için etkili ve verimli bir derin öğrenme yaklaşımı tasarladık. Yaklaşımımız bulut tabanlı IDS ve bulut hizmeti olmak üzere iki bileşenden oluşur.

Önerilen bulut tabanlı IDS, izleme, işleme, analiz, tahmin ve yanıt olmak üzere beş modülden oluşan entegre bir NIDS'tir. Bu modüllerin her biri belirli görevleri ve işlevleri yerine getirir. Dahası, güvenilir veri tabanları oluşturmak için çevrimiçi olarak çıkarılacak 40 özellik belirledik. Ayrıca, bir dizi DNN, LSTM-RNN ve DBN modellerinden oluşan bir çoğunluk oylama motorundan oluşan bir torbalama topluluğu sistemi önerdik. Torbalama topluluk sistemi kullanmanın amacı, izinsiz giriş saldırı tespitinin doğruluğunu arttırmaktır. Ayrıca, derin öğrenme modelleri, topluluk sisteminin kurulmasından önce çift PSO tabanlı algoritmanın kullanılmasıyla optimize edilmiştir. Son modül olan yanıt modülü olayları kaydeder ve topluluk sisteminin nihai kararına bağlı olarak bir tepki alır.

Buna ek olarak, hibrit algılama, dinamik yapı ve çok okuyuculu gibi bulut tabanlı IDS'imizin avantajlarını, büyük trafik ve veri akışını ebeveyn-çocuk tarzında ele almak için ele aldık. Bilgi ve veri güvenliğinin şeffaflığını sağlamak adına, önerilen bulut tabanlı IDS'yi izlemekten ve yönetmekten sorumlu üçüncü taraf bir bulut hizmeti de önerdik. Yanıt modülünden uyarılar alır ve hem bulut kullanıcısı hem de bulut servis sağlayıcısı için raporlar gönderir. Son olarak, teklif edilen bulut tabanlı IDS'yi bulutun arka ucunun ortasına yerleştirdik ve aynı zamanda bulutun üstündeki önerilen bulut hizmetini yerleştirdik. Bu dağıtılmış yaklaşım, verimli bir yaklaşım olan eşzamanlı veri analizi işlemlerini gerçekleştirme yeteneğine sahiptir.

### 6.2. Öneriler

Gelecekteki bir çalışma olarak iki öneride bulunmak istiyoruz. İlki, önerilen bulut tabanlı IDS ve bulut hizmetimizi gerçek dünyadaki bir bulut ortamında

test etmediğimizdir. Bu nedenle, önceki dağıtılmış yaklaşımı dağıtmaya ve gerçek dünya bulut saldırı tespitinde etkinliğini ölçmeye ihtiyaç var.

İkinci endişe, önerilen bulut tabanlı IDS'de performans açısından çok okuyuculu kullanmanın etkisidir. Alternatif olarak, önerilen bulut tabanlı IDS'yi uygulamak için bulut programlama tarafından sağlanan büyük kaynakları, çok okuyuculu yerine özellikle Hadoop'a paralel programlama teknikleri kullanarak kullanabiliriz. Bu tür tekniklerin kullanılması, izinsiz giriş saldırı tespit etmenin etkinliğinde paralel programlama kullanmanın yanı sıra hesaplama maliyeti açısından performansın etkisinde derinlemesine çalışmayı gerektirir.

# KAYNAKLAR


Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., vd., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint , s. arXiv:1603.04467.

Abdalla, O. A., Elfaki, A. O., Almurtadha, Y. M., 2014. Optimizing the multilayer feed-forward artificial neural networks architecture and training parameters using genetic algorithm. International Journal of Computer Applications , 96 (10).

Aggarwal, C. C., Zhai, C., 2012. A survey of text classification algorithms. Mining text data (s. 163-222). Boston: Springer.

Ahmad, I., 2015. Feature Selection Using Particle Swarm Optimization in Intrusion Detection. International Journal of Distributed Sensor Networks , 11 (10), s. Article ID 806954.

Albelwi, S., Mahmood, A., 2017. A Framework for Designing the Architectures of Deep Convolutional Neural Networks. Entropy , 19 (6), s. 242.

Alguliev, R., Abdullaeva, F., 2014. Illegal Access Detection in the Cloud Computing Environment. Journal of Information Security , 5 (2), s. 65.

Alsafi, H. M., Abduallah, W. M., Pathan, A. K., 2012. IDPS: An Integrated Intrusion Handling Model for Cloud. arXiv preprint , s. arXiv:1203.3323.

Ambusaidi, M. A., He, X., Nanda, P., Tan, Z., 2016. Building an intrusion detection system using a filter-based feature selection algorithm. IEEE transactions on computers , 65 (10), s. 2986-2998.

Aminanto, E., Kim, K., 2016. Deep learning in intrusion detection system: An overview. 2016 International Research Conference on Engineering and Technology (2016 IRCET). Higher Education Forum.

Aminanto, M. E., Kim, K., 2016. Deep learning-based feature selection for intrusion detection system in transport layer.

Axelsson, S., 1999. The base-rate fallacy and its implications for the difficulty of intrusion detection. Proceedings of the 6th ACM Conference on Computer and Communications Security (s. 1-7). ACM.

Azevedo, G., Cavalcanti, G., Filho, E., 2007. An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting. 2007 IEEE Congress on Evolutionary Computation (s. 3577-3584). IEEE.

Bakshi, A., Dujodwala, Y. B., 2010. Securing cloud from ddos attacks using intrusion detection system in virtual machine. 2010 Second International

Conference on Communication Software and Networks (s. 260-264). IEEE.

Belharbi, S., H´erault, R., Chatelain, C., Adam, S., 2016. Deep multi-task learning with evolving weights. European Symposium on Artificial Neural Networks (ESANN), 2016.

Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks , 5 (2), s. 157-166.

Bergstra, J. S., Bengio, Y., 2012. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research , 13 (Feb.), s. 281–305.

Bergstra, J. S., Bardenet, R., Bengio, Y., K´egl, B., 2011. Algorithms for Hyper-Parameter optimization. Advances in neural information processing systems, (s. 2546–2554).

Bertacchini, M., Fierens, P., 2008. A survey on masquerader detection approaches. Proceedings of V Congreso Iberoamericano de Seguridad Inform´atica, Universidad de la Rep´ublica de Uruguay.

Borji, A., 2007. Combining heterogeneous classifiers for network intrusion detection. Annual Asian Computing Science Conference (s. 254-260). Springer.

Bosin, A., Dessi, N., Pes, B., 2008. A service based approach to a new generation of intrusion detection systems. 2008 Sixth European Conference on Web Services (s. 215-224). IEEE.

Boughorbel, S., Jarray, F., El-Anbari, M., 2017. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. PloS one , 12 (6), s. e0177678.

Bradley, A. P., 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition , 30 (7), s. 1145-1159.

Canadian Institute for Cybersecurity - IDS 2017, 2017. Erişim Tarihi: August 2018. http://www.unb.ca/cic/datasets/ids-2017.html.

Cervante, L., Xue, B., Zhang, M., Shang, L., 2012. Binary Particle Swarm Optimisation for Feature Selection: A Filter Based Approach. 2012 IEEE Congress on Evolutionary Computation (s. 1-8). IEEE.

Chae, H., Jo, B., Choi, S., Park, T., 2013. Feature Selection for Intrusion Detection using NSL-KDD. Recent advances in computer science, (s. 184-187).

Chakraborty, B., 2008. Feature subset selection by particle swarm optimization with fuzzy fitness function. 2008 3rd international conference on intelligent system and knowledge engineering. 1, s. 1038-1042. IEEE.

Chebrolu, S., Abraham, A., Thomas, J. P., 2005. Feature deduction and ensemble design of intrusion detection systems. Computers security , 24 (4), s. 295-307.

Chen, L., Aritsugi, M., 2006. An svm-based masquerade detection method with online update using co-occurrence matrix. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (s. 37-53). Berlin: Springer.

Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., vd., 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. arXiv preprint , s. arXiv:1406.1078.

Chollet, F., 2015. Keras. Erişim Tarihi: March 2018. https://github.com/fchollet/keras.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint , s. arXiv:1412.3555.

CICFlowMeter, 2017. Erişim Tarihi: May 2019. https://www.unb.ca/cic/research/applications.html#CICFlowMeter.

CIDDS-Coburg Intrusion Detection Data Sets, 2017. Erişim Tarihi: March 2018. https://www.hs-coburg.de/index.php?id=927.

Clerc, M., Kennedy, J., 2002. The particle swarm: Explosion, stability and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation , 6 (1), s. 58-73.

Cortes, C., Mohri, M., 2004. AUC optimization vs. error rate minimization. Advances in neural information processing systems , s. 313-320.

CUDA- Compute Unified Device Architecture, 2018. Erişim Tarihi: March 2019. https://developer.nvidia.com/about-cuda.

cuDNN- The NVIDIA CUDA Deep Neural Network library, 2018. Erişim Tarihi: March 2019. https://developer.nvidia.com/cudnn.

Daniel, W. W., 1990. Friedman two-way analysis of variance by ranks. Applied nonparametric statistics , s. 262-274.

Dasarathy, B. V., Sheela, B. V., 1979. A composite classifier system design: Concepts and methodology. Proceedings of the IEEE , 67 (5), s. 708--713.

Dash, M., Liu, H. (1997). Feature selection for classification. Intelligent data analysis , 1 (1-4), s. 131-156.

Dastjerdi, A. V., Bakar, K. A., Tabatabaei, S. G., 2009. Distributed intrusion detection in clouds using mobile agents. 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences (s. 175-180). IEEE.

David, O. E., Greental, I., 2014. Genetic algorithms for evolving deep neural networks. Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation (s. 1451–1452). ACM.

Davis, J., Goadrich, M., 2006. The Relationship Between Precision-Recall and ROC Curves. In Proceedings of the 23rd international conference on Machine learning (s. 233-240). ACM.

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research , 7, s. 1-30.

Deng, L., 2014. A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing. 3. Cambridge University Press.

Dhage, S. N., Meshram, B., Rawat, R., Padawe, S., Paingaokar, M., Misra, A. , 2011. Intrusion detection system in cloud computing environment. Proceedings of the International Conference Workshop on Emerging Trends in Technology (s. 235-239). ACM.

Dong, B., Wang, X., 2016. Comparison Deep Learning Method to Traditional Methods Using for Network Intrusion Detection. In Proceedings of 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN) (s. 581-585). IEEE.

Du, X., Cai, Y., Wang, S., Zhang, L., 2016. Overview of deep learning. 31st Youth Academic Annual Conference of Chinese Association of Automation (s. 159-164). Wuhan: IEEE.

Dzeroski, S., Zenko, B., 2002. Is combining classifiers better than selecting the best one? ICML , 2002, s. 123e30.

Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. Proceedings of the 6th international symposium on Micro Machine and Human Science (s. 39-43). IEEE.

Eid, H. F., Salama, M. A., Hassanien, A. E., 2013. A Feature Selection Approach for Network Intrusion Classification: The Bi-Layer Behavioral-Based. International Journal of Computer Vision and Image Processing (IJCVIP) , 3 (4), s. 51-59.

Elhamahmy, M. E., Elmahdy, H. N., Saroit, I. A., 2010. A New Approach for Evaluating Intrusion Detection System. CiiT International Journal of Artificial Intelligent Systems and Machine Learning , 2 (11).

Elmasry, W., Akbulut, A., Zaim, A. H., 2018. Deep Learning Approaches for Predictive Masquerade Detection. Security and Communication Networks , 2018, s. Article ID 9327215, 24 pages.

Erbacher, R. F., Prakash, S., Claar, C. L., Couraud, J., 2007. Intrusion Detection: Detecting Masquerade Attacks Using UNIX Command Lines. Proceedings of the 6th Annual Security Conference. Las Vegas.

Escalante, H. J., Montes, M., Sucar, L. E., 2009. Particle Swarm Model Selection. Journal of Machine Learning Research , 10 (Feb.), s. 405-440.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern recognition letters , 27 (8), s. 861-874.

Folino, G., Sabatino, P., 2016. Ensemble based collaborative and distributed intrusion detection systems: A survey. Journal of Network and Computer Applications , 66, s. 1-16.

Gebski, M., Wong, R. K., 2005. Intrusion detection via analysis and modelling of user commands. International Conference on Data Warehousing and Knowledge Discovery (s. 388-397). Berlin: Springer.

Gharib, A., Sharafaldin, I., Lashkari, A. H., Ghorbani, A. A., 2016. An evaluation framework for intrusion detection dataset. 2016 International Conference on Information Science and Security (ICISS) (s. 1-6). IEEE.

Greenberg, S., 1988. Using Unix: Collected traces of 168 users. Technical Report, University of Calgary, Department of Computer Science, Calgary.

Gul, I., Hussain, M., 2011. Distributed cloud intrusion detection model. International Journal of Advanced Science and Technology , 34 (38), s. 135.

He, H., Ma, Y., 2013. Imbalanced Learning: Foundations, Algorithms, and Applications. John Wiley Sons.

Hinton, G. E., 2002. Training products of experts by minimizing contrastive divergence. Neural Computation , 14 (8), s. 1771-1800.

Hinton, G. E., Osindero, S., Teh, Y. W., 2006. A fast learning algorithm for deep belief nets. Neural Computation , 18 (7), s. 1527-1554.

Hinton, G., Deng, L., Yu, D., Dhal, G. E., Mohamed, A., Jaitly, N., vd,. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared

views of four research groups. IEEE Signal processing magazine , 29 (6), s. 82-97.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation , 9 (8), s. 1735-1780.

Hossin, M., Sulaiman, M., 2015. A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS. International Journal of Data Mining Knowledge Management Process (IJDKP) , 5 (2), s. 1-11.

Huang, C., Dun, J., 2008. A distributed PSO-SVM hybrid system with feature selection and parameter optimization. Applied soft computing , 8 (4), s. 1381-1391.

Huang, L., 2010. A study on masquerade detection.

Ji, C., Ma, S., 1997. Combinations of weak classifiers. Advances in Neural Information Processing Systems, (s. 494-500).

Johnson, R., Zhang, T., 2014. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. Association for Computational Linguistics (ACL).

Kadam, Y., 2011. Security Issues in Cloud Computing A Transparent View. International Journal of Computer Science Emerging Technology , 2 (5), s. 316-322.

Ke, L., Feng, Z., Xu, Z., Shang, K., Wang, Y., 2010. A multi objective ACO algorithm for rough feature selection. 2010 Second Pacific-Asia Conference on Circuits, Communications and System. 1, s. 207-210. IEEE.

Kennedy, J., Eberhart, R., 1997. A discrete binary version of the particle swarm optimization. Computational cybernatics and simulation , 5 (1), s. 4104-4108.

Kennedy, J., Mendes, R., 2002. Population structure and particle swarm performance. Proceedings of the 2002 Congress on Evolutionary Computation. 2, s. 1671-1676. IEEE.

Kennedy, J., Spears, W., 1998. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. IEEE World Congress on Computational Intelligence,1998 IEEE International Conference on Evolutionary Computation Proceedings (s. 78-83). IEEE.

Keras, 2018. Erişim Tarihi: March 2019. https://keras.io.

Kholidy, H. A., Baiardi, F., 2012. CIDS: A framework for intrusion detection in cloud systems. 2012 Ninth International Conference on Information Technology-New Generations (s. 379-385). IEEE.

Kholidy, H. A., Erradi, A., Abdelwahed, S., 2014. Attack prediction models for cloud intrusion detection systems. 2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation (s. 270-275). IEEE.

Kholidy, H. A., Erradi, A., Abdelwahed, S., Baiardi, F., 2013. A hierarchical, autonomous, and forecasting cloud IDS. 2013 5th International Conference on Modelling, Identification and Control (ICMIC) (s. 213-220). IEEE.

Kim, H. S., Cha, S. D., 2005. Empirical evaluation of svm-based masquerade detection using unix commands. Computers and Security , 24 (2), s. 160-168.

Kim, J., Kim, J., Thu, H. L., Kim, H., 2016. Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. 2016 International Conference on Platform Technology and Service (PlatCon) (s. 1-5). IEEE.

Kim, Y., 2014. Convolutional neural networks for sentence classification. arXiv preprint , s. arXiv:1408.5882.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., Barnes, L. E., 2017. Hdltex: Hierarchical deep learning for text classification. arXiv preprint , s. arXiv:1709.08267.

Kubat, M., Matwin, S., 1997. Addressing the curse of imbalanced training sets: one-sided selection. Proceedings of the 14th International Conference on Machine Learning (ICML), 97, s. 179-186. Nashville.

Lai, S., Xu, L., Liu, K., Zhao, J., 2015. Recurrent Convolutional Neural Networks for Text Classification. AAAI , 333, s. 2267-2273.

Landgrebe, T. C., Paclik, P., Duin, R. P., 2006. Precision-recall operating characteristic (P-ROC) curves in imprecise environments. In Proceedings of the 18th International Conference on Pattern Recognition. 4, s. 123-127. IEEE.

Lane, T., Brodley, C. E., 1997. An application of machine learning to anomaly detection. Proceedings of the 20th National Information Systems Security Conference, 377, s. 366-380. Baltimore.

Langley, P., 1994. Selection of relevant features in machine learning. Proceedings of the AAAI Fall symposium, 1-5.

Lashkari, A. H., Draper-Gil, G., Mamun, M. S., Ghorbani, A. A., 2017. Characterization of Tor Traffic using Time based Features. In

Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), (s. 253-262).

Laureano, M., Maziero, C., Jamhour, E., 2007. Protecting host-based intrusion detectors through virtual machines. Computer Networks , 51 (5), 1275-1283.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE , 86 (11), 2278-2324.

Lee, J., Park, M., Eom, J., Chung, T., 2011. Multi-level intrusion detection system and log management in cloud computing. 13th International Conference on Advanced Communication Technology (ICACT2011) (s. 552-555). IEEE.

Li, Z., Li, Z., Liu, B., 2006. Masquerade detection system based on correlation eigen matrix and support vector machine. 2006 International Conference on Computational Intelligence and Security. 1, 625-628. IEEE.

Lin, S., Ying, K., Chen, S., Lee, Z., 2008. Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert systems with applications , 35 (4), s. 1817-1824.

Liu, L., Luo, J., Deng, X., Li, S., 2015. FPGA-based acceleration of deep neural networks using high level method. 2015 10th International Conference on Parallel, Grid, Cloud and Internet Computing (3PGCIC) (s. 824-827). IEEE.

Liu, P., Qiu, X., Huang, X., 2016. Recurrent Neurl Network for Text Classification with Multi-Task Learning. arXiv preprint , s. arXiv:1605.05101.

Lo, C., Huang, C., Ku, J., 2010. A cooperative intrusion detection system framework for cloud computing networks. 2010 39th International Conference on Parallel Processing Workshops (s. 280-284). IEEE.

Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., Pastor, J. R., 2017. Particle swarm optimization for hyper-parameter selection in deep neural networks. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017) (s. 481-488). New York: ACM.

Lorenzo, P. R., Nalepa, J., Ramos, L. S., Pastor, J. R., 2017. Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2017) (s. 1864-1871). New York: ACM.

Ludwig, S. A., 2017. Intrusion detection of multiple attack classes using a deep neural net ensemble. 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (s. 1-7). IEEE.

Mahmood, T., Afzal, U., 2013. Security Analytics: Big Data Analytics for Cybersecurity: A Review of Trends, Techniques and Tools. 2013 2nd National Conference on Information Assurance (NCIA) (s. 129-134). IEEE.

Marir, N., Wang, H., Feng, G., Li, B., Jia, M., 2018. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. IEEE Access , 6, 59657-59671.

Martin, A., Fuentes-Hurtado, F., Naranjo, V., Camacho, D., 2017. Evolving deep neural networks architectures for Android malware classification. 2017 IEEE Congress on Evolutionary Computation (CEC) (s. 1659-1666). IEEE.

Matthews, B. W., 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure , 405 (2), 442-451.

Maxion, R. A., 2003. Masquerade Detection Using Enriched Command Lines. 2003 International Conference on Dependable Systems and Networks (DSN-03) (s. 514). IEEE.

Maxion, R. A., Townsend, T. N., 2002. Masquerade detection using truncated command lines. Proceedings of international conference on dependable systems and networks (DSN-02) (s. 219-228). IEEE.

Mazzariello, C., Bifulco, R., Canonico, R., 2010. Integrating a network ids into an open source cloud computing environment. 2010 Sixth International Conference on Information Assurance and Security (s. 265-270). IEEE.

McHugh, J., 2000. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Transactions on Information and System Security (TISSEC) , 3 (4), 262-294.

Mehmood, Y., Shibli, M. A., Habiba, U., Masood, R., 2013. Intrusion detection system in cloud computing: Challenges and opportunities. 2013 2nd National Conference on Information Assurance (NCIA) (s. 59-66). IEEE.

MIT Lincoln Laboratory, 1998. DARPA Intrusion Detection Evaluation Data Set. Erişim Tarihi: August 2018. https://www.ll.mit.edu/rd/datasets/1999-darpa-intrusion-detection-evaluation-data-set.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M., 2013. A survey of intrusion detection techniques in cloud. Journal of network and computer applications , 36 (1), 42-57.

Mohemmed, A., Zhang, M., Johnston, M., 2009. Particle Swarm Optimization based Adaboost for face detection. 2009 IEEE Congress on Evolutionary Computation (s. 2494-2501). IEEE.

Naidoo, T., McDonald, A., Tapamo, J. R., 2015. Feature Selection for Anomaly–Based Network Intrusion Detection Using Cluster Validity Indices. http://www. satnac. org. za.

Nalepa, J., Lorenzo, P. R., 2017. Convergence Analysis of PSO for Hyper-Parameter Selection in Deep Neural Networks. International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (s. 284-295). Cham: Springer.

Natesan, P., Balasubramanie, P., 2012. Multi stage filter using enhanced adaboost for network intrusion detection. International Journal of Network Security & Its Applications , 4 (3), 121.

Neshatian, K., Zhang, M., Andreae, P., 2012. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. IEEE Transactions on Evolutionary Computation , 16 (5), 645-661.

Nkiama, H., Syed, S. Z., Saidu, M., 2016. A Subset Feature limination Mechanism for Intrusion Detection System. International Journal of Advanced Computer Science and Applications , 7 (4), 148-157.

NSL-KDD Dataset, 2009. Erişim Tarihi: August 2018. https://github.com/defcom17/NSL_KDD.

NumPy, 2018. Erişim Tarihi: March 2019. http://www.numpy.org.

Okamoto, T., Watanabe, T., Ishida, Y., 2003. Towards an immunity-based system for detecting masqueraders. International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (s. 488-495). Berlin: Springer.

Oliveira, L., Sabourin, R., Bortolozzi, F., Suen, C., 2002. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. International Conference on Pattern Recognition, 16, 568-571.

Patel, A., Qassim, Q., Shukor, Z., Nogueira, J., Junior, J., Wills, C., vd., 2011. Autonomic agent-based self-managed intrusion detection and prevention system. Proceedings of the South African Information Security Multi-Conference (SAISMC 2010), 223-234.

Patel, A., Taghavi, M., Bakhtiyari, K., Junior, J. C., 2013. An intrusion detection and prevention system in cloud computing: A systematic review. Journal of network and computer applications , 36 (1), 25-41.

Pervez, M. S., Farid, D. M., 2014. Feature Selection and Intrusion classification in NSL-KDD Cup 99 Dataset Employing SVMs. The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014) (s. 1-6). IEEE.

Prusa, J. D., Khoshgoftaar, T. M., 2017. Improving deep neural network design with new text data representations. Journal of Big Data , 4 (1), 7.

Purohit, A., Chaudhari, N., Tiwari, A., 2010. Construction of classifier with feature selection based on genetic programming. IEEE Congress on Evolutionary Computation (s. 1-5). IEEE.

Python. 2018. Erişim Tarihi: March 2019. https://www.python.org.

Ragsdale, D. J., Carver, C., Humphries, J. W., Pooch, U. W., 2000. Adaptation techniques for intrusion detection and intrusion response systems. Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'. 4, s. 2344-2349. IEEE.

REDDY, K. V., PUSHPALATHA, N., 2014. CONDITIONAL NAIVE-BAYES TO DETECT MASQUERADES. International Journal of Computer Science and Engineering (IJCSE) , 3 (3), 13-22.

Ring, M., Wunderlich, S., Grudl, D., Landes, D., Hotho, A., 2017. Creation of Flow-Based Data Sets for Intrusion Detection. Journal of Information Warfare , 16 (4), 41-54.

Ring, M., Wunderlich, S., Grudl, D., Landes, D., Hotho, A., 2017. Flow-based benchmark data sets for intrusion detection. In Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS), 361-369.

Roschke, S., Cheng, F., Meinel, C., 2009. Intrusion detection in the cloud. 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (s. 729-734). IEEE.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1986. Learning representations by back-propagating errors. nature , 323 (6088), 533-536.

Salakhutdinov, R., Hinton, G. E., 2009. Deep Boltzmann Machines. In proceedings of the 12th International Conference on Artificial Intelligence and Statistics, 448-455.

Salama, M. A., Eid, H. F., Ramadan, R. A., Darwish, A., Hassanien, A. E., 2011. Hybrid intelligent intrusion detection scheme. Soft computing in industrial applications (s. 293-303). Springer.

Sasan, H. P., Sharma, M., 2016. INTRUSION DETECTION USING FEATURE SELECTION AND MACHINE LEARNING ALGORITHM WITH MISUSE

DETECTION. International Journal of Computer Science Information Technology (IJCSIT) , 8 (1), 17-25.

Schonlau, M., DuMouchel, W., Ju, W. H., Karr, A. F., Theus, M., Verdi, Y., 2001. Computer intrusion: Detecting masquerades. Statistical Science , 16 (1), 58-74.

Shaikh, R., Sasikumar, M., 2012. Trust framework for calculating security strength of a cloud service. 2012 International Conference on Communication, Information & Computing Technology (ICCICT) (s. 1-6). IEEE.

Sharafaldin, I., Gharib, A., Lashkari, A. H., Ghorbani, A. A., 2017. Towards a reliable intrusion detection benchmark dataset. Software Networking , 1 (1), 177-200.

Sharafaldin, I., Lashkari, A. H., Ghorbani, A. A., 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. 4th International Conference on Information Systems Security and Privacy (ICISSP), (s. 108-116). Purtogal.

Shelke, M. P., Sontakke, M. S., Gawande, D. A., 2012. Intrusion Detection System for Cloud Computing. International Journal of Scientific & Technology Research , 1 (4), 67-71.

Shi, Y., Eberhart, R. C., 1999. Emprirical study of particle swarm optimization. Proceedings of the 1999 congress on Evolutionary Computation. 3, s. 1945-1950. IEEE.

Shi, Y., Eberhart, R. C., 1998. Parameter selection in particle swarm optimization. International conference on evolutionary programming (s. 591-600). Berlin: Springer.

Sivagaminathan, R., Ramakrishnan, S., 2007. A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert systems with applications , 33 (1), 49-60.

Snoek, J., Larochelle, H., Adams, R. B., 2012. Practical Bayesian Optimization of Machine Learning Algorithms. Advances in neural information processing systems, 2951-2959.

Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Information Processing & Management , 45 (4), 427-437.

Spafford, E. H., Zamboni, D., 2000. Intrusion detection using autonomous agents. Computer networks , 34 (4), 547-570.

Stolfo, S. J., Fan, W., Lee, W., Prodromidies, A., Chan, P. K., 2000. Cost-based modeling for fraud and intrusion detection: Results from the JAM project. NEW YORK: COLUMBIA UNIV. DEPT. OF COMPUTER SCIENCE.

Subbotin, S., Oleynik, A., 2008. The multi objective evolutionary feature selection. 2008 International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), 115-116. IEEE.

Swets, J. A., 1988. Measuring the accuracy of diagnostic systems. Science, American Association for the Advancement of Science , 240 (4857), 1285-1293.

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A., Ghogho, M., 2016. Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (s. 258-263). IEEE.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A., 2009. A Detailed Analysis of the KDD CUP 99 Data Set. 2009. CISDA 2009. IEEE Symposium on Computational Intelligence for Security and Defense Applications (s. 1-6). IEEE.

TensorFlow, 2018. Erişim Tarihi: March 2019. https://www.tensorflow.org.

Tirumala, S. S., Ali, S., Ramesh, C. P., 2016. Evolving deep neural networks: A new prospect. 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD) (s. 69–74). IEEE.

Tran, B., Xue, B., Zhang, M., 2014. Overview of Particle Swarm Optimisation for Feature Selection in Classification. Asia-Pacific conference on simulated evolution and learning (s. 605-617). Springer.

UCI Machine Learning Repository: KDD CUP 1999 Data Set, 1999. Erişim Tarihi: Mart 2018. https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data.

Vani, R., 2017. Towards Efficient Intrusion Detection using Deep Learning Techniques: A Review. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE) , 6 (10).

Vieira, K., Schulter, A., Westphall, C., Westphall, C., 2009. Intrusion detection for grid and cloud computing. It Professional , 12 (4), 38-43.

Wahba, Y., ElSalamouny, E., ElTaweel, G., 2015. Improving the Performance of Multi-class Intrusion Detection Systems using Feature Reduction. arXiv preprint , s. arXiv:1507.06692.

Wang, K., Stolfo, S. J., 2003. One-class training for masquerade detection. Workshop on Data Mining for Computer Security, (s. 10-19). Melbourne.

Wang, Y., Wen, J., Wang, X., Tao, B., Zhou, W., 2019. A Cloud Service Trust Evaluation Model Based on Combining Weights and Gray Correlation Analysis. Security and Communication Networks.

Waqas, K., Baig, R., Ali, S., 2009. Feature subset selection using multi-objective genetic algorithms. 2009 IEEE 13th International Multitopic Conference (s. 1-6). IEEE.

Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics bulletin , 1 (6), 80-83.

Yang, C., Chuang, L., Li, J., Yang, C., 2008. Chaotic maps in binary particle swarm optimization for feature selection. 2008 IEEE Conference on Soft Computing in Industrial Applications (s. 107-112). IEEE.

Yang, M., Zhang, H., Cai, H. J., 2007. Masquerade detection using string kernels. 2007. International Wireless Communications, Networking and Mobile Computing (s. 3681-3684). IEEE.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E., 2016. Hierarchical attention networks for document classification. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1480-1489.

Ye, F., 2017. Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data. PloS one , 12 (12), e0188746.

Yee, C. G., Shin, W. H., Rao, G., 2007. An adaptive intrusion detection and prevention (ID/IP) framework for web services. 2007 International Conference on Convergence Information Technology (ICCIT 2007) (s. 528-534). IEEE.

Yin, C., Zhu, Y., Fei, J., He, X., 2017. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access , 5, 21954-21961.

Yung, K. H., 2003. Using feedback to improve masquerade detection. International Conference on Applied Cryptography and Network Security (s. 48-62). Berlin: Springer.

Yung, K. H., 2004. Using self-consistent naive-bayes to detect masquerades. Pacific-Asia Conference on Knowledge Discovery and Data Mining (s. 329-340). Berlin: Springer.

Zarrabi, A., Zarrabi, A., 2012. Internet intrusion detection system service in a cloud. International Journal of Computer Science Issues (IJCSI) , 9 (5), 308.

Zeng, Z., Gao, J., 2009. Improving SVM classification with imbalance data set. International Conference on Neural Information Processing (s. 389-398). Springer.

Zhang, X., LeCun, Y., 2015. Text Understanding from scratch. arXiv preprint , s. arXiv:1502.01710.

Zhang, X., Zhao, J., LeCun, Y., 2015. Character-level Convolutional Networks for Text Classification. Advances in neural information processing systems , 649-657.

Zhang, Y., Wallace, B., 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint , s. arXiv:1510.03820.

Zhou, Z., Liu, X., Li, P., Shang, L., 2014. Feature Selection Method with Proportionate Fitness Based Binary Particle Swarm Optimization. Asia-Pacific Conference on Simulated Evolution and Learning, Springer, 582-592.

# ÖZGEÇMİŞ

Adı Soyadı : Wisam Elmasry

Doğum Yeri ve Yılı : LİBYA, 05/12/1981

Medeni Hali : Bekar

Yabancı Dili : Arapça, İngilizce

E-posta : wisam.elmasry@istanbulticaret.edu.tr

**Eğitim Durumu**

Lise : Khalid Alhasan Lisesi, 1999

Lisans : The İslamic University of Gaza, Mühendisliği Fakültesi, Bilgisayar Mühendisliği Bölümü

Yüksek Lisans : The İslamic University of Gaza, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı

**Mesleki Deneyim**

| | |
|---|---|
| College of Science and Technology,<br>IT Department | 2011-2013 |
| Alaqsa University,<br>IT Department | 2012-2013 |
| Alazhar University,<br>IT Department | 2012-2013 |
| Palestine College of Nursing,<br>IT Department | 2010-2011 |
| General Security Training Directorate,<br>IT Department | 2005-2010 |

**Yayınları**

Elmasry W, Akbulut A, Zaim AH. Empirical study on multiclass classification-based network intrusion detection. *Computational Intelligence*. 2019;1–36. https://doi.org/10.1111/coin.12220.

Elmasry W., Akbulut A., and Zaim A. H., "Deep Learning Approaches for Predictive Masquerade Detection," Security and Communication Networks, Hindawi, Vol. 2018, Article ID 9327215, 24 pages, 2018. https://doi.org/10.1155/2018/9327215.

Kasapbasi M. C., and Elmasry W., "New LSB-based colour image steganography method to enhance the efficiency in payload capacity, security and integrity check," Sadhana, Springer, Vol. 43, pp. 1-14, 2018.

Elmasry W., "Securing WBAODV Routing Protocol in MANETs: Towards Efficient and Secure Routing Protocol", LAMBERT Academic Publishing (LAP), Germany, ISBN:978-3-659-26879-3.

Sahmoud S., Elmasry W., and Shadi Abudalfa, "Enhancement the security of AES against modern attacks by using Variable key block cipher", International Arab Journal of e-Technology (IAJeT) , Jordan.