T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAZILIM TANIMLI AĞLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI İLE ANOMALİ TESPİTİ VE SİBER SALDIRI SINIFLANDIRMASI

YÜKSEK LİSANS TEZİ

Hasan ZEYNİYEV

Yapay Zekâ Mühendisliği Anabilim Dalı

Yapay Zekâ Mühendisliği Tezli Yüksek Lisans Programı

EKİM 2024
İSTANBUL

T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAZILIM TANIMLI AĞLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI İLE ANOMALİ TESPİTİ VE SİBER SALDIRI SINIFLANDIRMASI

YÜKSEK LİSANS TEZİ

Hasan ZEYNİYEV
210039010
0000-0003-2399-4477

Yapay Zekâ Mühendisliği Anabilim Dalı

Yapay Zekâ Mühendisliği Tezli Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Fatih TARLAK

İstanbul 2024

T.C.

İSTANBUL GEDİK ÜNİVERSİTESİ

Lisansüstü Eğitim Enstitüsü Müdürlüğü

Jüri Tez Onay Formu

24.10.2024

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

Bu çalışma 24.10.2024 tarihinde aşağıdaki jüri tarafından Yapay Zeka Mühendisliği Anabilim Dalı, Yapay Zeka Mühendisliği (Tezli Yüksek Lisans) Programı Yüksek Lisans Tezi olarak kabul edilmiştir.

TEZ JÜRİSİ

Doç. Dr. Fatih TARLAK
Danışman
Gebze Teknik Üniversitesi

Dr. Öğr. Üyesi Tuğbay Burçin Gümüş
Üye (İmza)
İstanbul Gedik Üniversitesi

Prof. Dr. Bahaddin SINSOYSAL
Üye (İmza)
İstanbul Beykent Üniversitesi

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum “Yazılım Tanımlı Ağlarda Derin Öğrenme Yöntemi İle Anomali Tespiti” başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını, patent ve telif haklarını ihlal edici bir davranışımın olmadığını belirtir ve bunu onurumla doğrularım (24/10/2024)

Hasan ZEYNİYEV

ÖNSÖZ

Bu tezin hazırlanmasında emeği geçen ve beni her adımda destekleyen herkese teşekkür etmek istiyorum. Öncelikle, bana her zaman rehberlik eden ve değerli bilgiler sağlayan danışman hocam Doç. Dr. Fatih Tarlak'a sonsuz teşekkürlerimi sunarım.

Ayrıca, bu süreçte bana yardımcı olan ve sürekli destek veren arkadaşım Hüseyin Nasirov'a minnettarım. Onun yardımları ve moral desteği benim için çok değerliydi.

Son olarak, her zaman yanımda olan ve beni destekleyen aileme teşekkür ederim. Onların sevgisi, sabrı ve teşvikleri olmasaydı bu başarıyı elde edemezdim.

Ekim 2024 Hasan ZEYNİYEV

İÇİNDEKİLER

Sayfa No:

KISALTMALAR

YTA : Yazılım Tanımlı Ağ

ANN : Yapay Sinir Ağı

CNN : Evrişimli Sinir Ağı

SDN : Software-Defined Networks

RNN : Tekrarlayan Sinir Ağı

IDS : Intruision Detection System

IPS : Intrusion Prevention Systems

NLP : Doğal dil işleme

IoT : Nesnelerin İnterneti

DNN : Derin Sinir Ağı

BGP : Border Gateway Protocol

UNB : University of New Brunswick

ŞEKİL LİSTESİ

Sayfa No:

ÇİZELGE LİSTESİ

Sayfa No:

YAZILIM TANIMLI AĞLARDA MAKİNE ÖĞRENMESİ ALGORİTMALARI İLE ANOMALİ TESPİTİ VE SİBERSALDIRI SINIFLANDIRMASI

ÖZET

Yazılım Tanımlı Ağlar (YTA), ağ yönetimini merkezi, programlanabilir ve dinamik bir yapıya kavuşturarak ağ yönetiminde devrim yaratmaktadır. Ancak, YTA'ın açıklığı ve karmaşıklığı, ağları çeşitli güvenlik tehditlerine karşı savunmasız hale getirmektedir. Bu çalışmada, YTA ortamında anomali tespiti için derin öğrenme tekniklerinin uygulanabilirliği ve etkinliği incelenmiştir.

Bu tezde, anomali tespiti için üç farklı derin öğrenme modeli olan Yapay Sinir Ağı (ANN), Evrişimli Sinir Ağı (CNN) ve Autoencoder kullanılmıştır. Bu modellerin performansları karşılaştırılmış ve YTA güvenliğine olan etkileri değerlendirilmiştir. Veri seti olarak, Canadian Institute for Cybersecurity at the University of New Brunswick (UNB) tarafından sağlanan geniş bir ağ trafiği veri seti kullanılmıştır

Veri seti, ön işleme adımlarından geçirilmiş ve eğitim ile test setlerine ayrılmıştır. Modeller, belirli hiperparametrelerle eğitilmiş ve doğruluk, precision, recall ve F1-score gibi performans metrikleri kullanılarak değerlendirilmiştir. Elde edilen sonuçlar, ANN modelinin en yüksek doğruluk oranına sahip olduğunu, ancak nadir görülen saldırılar için yetersiz kaldığını göstermektedir. CNN modeli, uzaysal verileri işleme kapasitesi sayesinde belirli tehditlerde etkili olmuştur. Autoencoder modeli ise verilerin düşük boyutlu bir uzaya indirgenmesinde başarılı olmuştur, ancak nadir saldırılarda düşük performans göstermiştir.

Bu çalışma, YTA'ların güvenliğini artırmak için derin öğrenme tekniklerinin nasıl optimize edilebileceğine dair değerli bilgiler sunmaktadır. Gelecekteki çalışmalar, bu modellerin daha geniş ve dengeli veri setleri ile eğitilerek performanslarının artırılmasına yönelik olabilir. Ayrıca, gerçek zamanlı anomali tespiti için modellerin entegrasyonu ve performans değerlendirmesi, gelecekteki araştırma alanları arasında yer almaktadır. Bu tez, YTA'ların güvenliğini sağlamak için derin öğrenme tekniklerinin uygulanabilirliğini doğrulamakta ve gelecekteki araştırmalar için değerli bir temel sunmaktadır.

Anahtar kelimeler: *Yazılım Tanımlı Ağlar, anomali tespiti, derin öğrenme, ağ güvenliği.*

ANOMALY DETECTION AND CYBER ATTACK CLASSIFICATION WITH MACHINE LEARNING ALGORITHMS IN SOFTWARE DEFINED NETWORKS

ABSTRACT

Software-Defined Networks (SDN) revolutionize network management by providing a centralized, programmable, and dynamic infrastructure. However, the openness and complexity of SDNs make networks vulnerable to various security threats. This study investigates the applicability and effectiveness of deep learning techniques for anomaly detection in SDN environments.

In this thesis, three different deep learning models, namely Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Autoencoder, are used for anomaly detection. The performance of these models is compared, and their impacts on SDN security are evaluated. The dataset used is a comprehensive network traffic dataset provided by the Canadian Institute for Cybersecurity at the University of New Brunswick (UNB).

The dataset underwent preprocessing steps and was divided into training and testing sets. The models were trained with specific hyperparameters and evaluated using performance metrics such as accuracy, precision, recall, and F1-score. The results showed that the ANN model had the highest accuracy but was insufficient for detecting rare attacks. The CNN model was effective in specific threats due to its spatial data processing capability. The Autoencoder model was successful in reducing the dimensionality of the data but showed low performance in detecting rare attacks.

This study provides valuable insights into how deep learning techniques can be optimized to enhance the security of SDNs. Future studies may focus on improving the performance of these models by training them with more extensive and balanced datasets. Additionally, integrating and evaluating these models for real-time anomaly detection could be a potential area for future research. This thesis confirms the applicability of deep learning techniques for securing SDNs and lays a valuable foundation for future research.

Keywords: *Software-Defined Networks (SDN), anomaly detection, deep learning, network security.*

1. GİRİŞ

Yazılım Tanımlı Ağlar (YTA), ağ yönetimini merkezi, programlanabilir ve dinamik bir yapıya kavuşturarak ağ yönetiminde devrim yaratmaktadır. Kontrol düzlemi ile veri düzlemini ayıran YTA, ağ yöneticilerinin ağ davranışlarını yazılım uygulamaları aracılığıyla yönetmelerine olanak tanır. Bu mimari esneklik, ağların daha verimli, esnek ve özelleştirilebilir hale gelmesini sağlar. Ancak, YTA'ın bu avantajlarına rağmen, ağın açıklığı ve karmaşıklığı, onu çeşitli güvenlik tehditlerine karşı savunmasız hale getirmektedir. SDN ortamlarında karşılaşılan başlıca güvenlik tehditleri arasında Dağıtılmış Hizmet Reddi (DDoS) saldırıları, izinsiz girişler, kötü niyetli trafik desenleri ve diğer anomali türleri bulunmaktadır. Bu tehditler, ağın performansını, güvenliğini ve bütünlüğünü ciddi şekilde tehlikeye atabilir. Geleneksel güvenlik önlemleri, YTA'ın dinamik ve programlanabilir doğası gereği yetersiz kalabilmektedir. Bu nedenle, YTA'ların güvenliğini artırmak için yenilikçi ve etkili yöntemlerin araştırılması gerekmektedir. Son yıllarda, derin öğrenme teknikleri, ağ güvenliği ve anomali tespiti alanında önemli başarılar elde etmiştir. Derin sinir ağları (DNN), özellikle Evrişimli Sinir Ağları (CNN'ler) ve Tekrarlayan Sinir Ağları (RNN), karmaşık verilerdeki anomalileri tespit etme yetenekleriyle dikkat çekmektedir. Bu modeller, geniş veri kümelerinden öğrenerek, normal ve anormal davranışları ayırt edebilmekte ve bilinmeyen tehditlere karşı adaptasyon gösterebilmektedir. Ancak, YTA ortamındaki güvenlik tehditlerini tespit etmek için derin öğrenme modellerinin etkinliğini değerlendiren kapsamlı çalışmalar sınırlıdır. Bu çalışmada, YTA içinde anomali tespiti için üç farklı derin öğrenme modelinin (Yapay Sinir Ağı (ANN), Evrişimli Sinir Ağı (CNN) ve Autoencoder) performansı karşılaştırılacaktır. Çalışma, derin öğrenme modellerinin YTA ortamındaki anomali tespitinde sağladığı doğruluk oranlarını ve farklı saldırı türlerini tespit etme başarılarını analiz etmeyi amaçlamaktadır. Hangi derin öğrenme modeli, YTA ortamında anomali tespiti için en yüksek doğruluk oranına sahiptir? Modeller, farklı türlerdeki saldırıları ne derece başarılı bir şekilde tespit edebilmektedir? Derin öğrenme modellerinin YTA güvenliğine olan etkileri nelerdir ve bu modellerin performansı nasıl optimize edilebilir?

Bu çalışmanın amacı, YTA'ların güvenliğini artırmak için derin öğrenme tekniklerinin uygulanabilirliğini değerlendirmek ve bu yöntemlerin ağ güvenliği üzerindeki etkilerini incelemektir. Çalışma, YTA ortamlarında anomali tespiti için derin öğrenme modellerinin etkinliğini ortaya koyarak, ağ güvenliği alanında önemli bir katkı sağlayacaktır. Ayrıca, elde edilen bulgular, YTA'ların daha güvenli ve etkili bir şekilde yönetilmesine olanak tanıyacak yeni stratejilerin geliştirilmesine yardımcı olacaktır.

2. YAZILIM TANIMLI AĞLAR

Mevcut ağ gereksinimleri, güvenilirlik, esneklik ve güvenlik gibi kritik detaylarda artışla birlikte önem kazanmıştır. Bu gereksinimler, bulut bilişim, akıllı telefonlar ve internetin büyümesi gibi faktörlerin etkisiyle ortaya çıkmıştır. Ne yazık ki, mevcut geleneksel ağlar bu hızla artan talebi karşılamak için yeterli kapasiteye sahip değildir. Ancak, Yazılım Tanımlı Ağların (YTA) tanıtılmasıyla bu zorluklar hafifletilmiştir. YTA, ağ kontrol düzlemini temel veri düzleminden fiziksel olarak ayırır ve bu, geleneksel ağlardan farklı bir yaklaşımdır. Kontrol düzlemi programlanabilir ve mantıksal olarak merkezileştirilmiştir. Bu nedenle, YTA'daki düzlem ayrımı, topolojide herhangi bir değişiklik yapmadan ağ trafiğini yazılım programlama yoluyla kontrol etmeyi kolaylaştırır (Feamster, N., Rexford, J., & Zegura, E. 2014).

Yazılım tanımlı ağ (SDN), kuruluşların uygulamaları dağıtmasını ve esnek teslimatı mümkün kılmasını kolaylaştırmakta, ağ kaynaklarını uygulama ve veri ihtiyaçlarına göre ölçeklendirme olanağı sunmakta ve hem CapEX hem de OpEX'i azaltmaktadır. Bu ağ segmentasyonu, ağ esnekliği ve kontrol edilebilirliği açısından çok sayıda avantaj sunar. Bir yandan sistem sanallaştırma ve bulut bilişimin avantajlarını bir araya getirirken, diğer yandan da kolay ağ yönetimi ve bakımının yanı sıra gelişmiş ağ kontrolü ve tepkiselliği için ağ üzerinde net bir görünürlük sağlayan merkezi bir zeka'nın uygulanmasına olanak tanır. Geleneksel altyapıda, ağ uygulaması, yapılandırması ve sorun giderme, teknik açıdan üst düzey becerilere sahip ağ ve sistem mühendislerinin müdahalesini ve büyük çok satıcılı ağların sağlanması ve yönetilmesiyle ilgili operasyonel maliyetleri gerektirmektedir (You, Y., Gitman, I., & Ginsburg, B.2017).

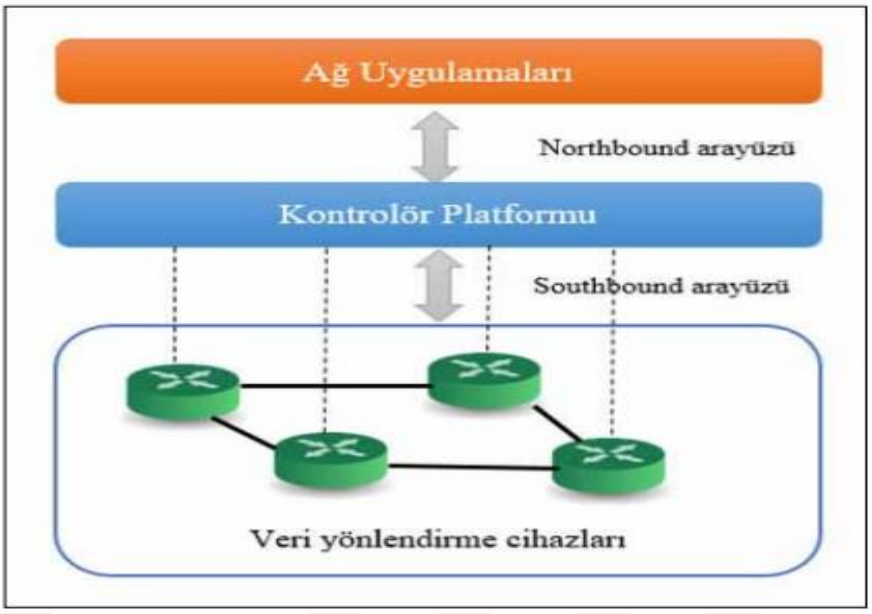
Yazılım Tanımlı Ağlar (YTA), veri düzlemi ve programlanabilir kontrol düzlemini birbirinden ayırarak ağ işlevlerinin merkezi bir denetleyici tarafından yönetilmesini sağlayan modern bir teknolojidir. Geleneksel ağ yapısında birleşik olan kontrol platformu ve yönlendirme işlevleri, YTA'da ayrı katmanlara ayrılmış ve kontrol platformu, sunucu üzerinde çalışan YTA denetleyicisi olarak bilinen bir

uygulamaya devredilmiştir. Bu merkezi denetleyici, ağın tüm bileşenlerini yönetebilmek için merkezi verilere sahip olup, veri düzlemindeki ağ cihazlarını kontrol edebilir (Cicioğlu, M., & Çalhan, A. 2017). Bu teknoloji, mevcut donanım altyapısını kullanarak sanal ağlar oluşturmayı ve düşük maliyetle esnek ağ yapıları tanımlamayı sağlar. YTA, programlanabilirlik ve esneklik sunarak geleneksel ağ yapılarına göre avantaj sağlar. YTA teknolojisi, ağ performansını artırmak ve daha dinamik, programlanabilir bir ağ yönetimi sağlamak amacıyla geliştirilmiştir (Yavuz, A., & Tuna, G. 2020).

SDN tasarımı, tüm kontrol işlevselliğini ağ genelinde eksiksiz bir görünüme sahip merkezi bir denetleyiciye itmeye olanak tanıdığından, bu özelliklerle kontrol uygulamaları geliştirmek ve politikaları uygulamak çok daha kolay hale gelir. Ağın boyutu büyüdükçe, denetleyiciye daha fazla olay ve istek gönderilir ve bir noktada, gelen tüm isteklerin denetleyici tarafından ele alınması ciddi bir zorluk haline gelir. Bu, her sistemde var olan yaygın bir sorundur ve yalnızca YTA tasarımı için geçerli değildir. Kontrol ve veri düzlemlerinin birbirinden ayrılması ve büyük bir ağ üzerindeki ağ trafiğinin kontrolünün denetleyiciye emanet edilmesiyle, ağ trafiğinin büyümesi, performans açısından denetleyicinin yetenekleriyle ölçeklenemeyebilir ve böylece ölçeklenebilirlik ve esneklik sorunları ortaya çıkabilir. Görevleri birden fazla kontrolör arasında dağıtarak ve paylaştırarak tek bir kontrolör üzerindeki iş yükünü hafifletmek için çok sayıda çalışma önerilmiştir (Balestriero, R., & LeCun, Y. 2023).

2.1 Yazılım Tanımlı Ağ Mimarisi

SDN'nin gelişimi, 1990'ların başında çeşitli destekleyici teknolojilerden ilham alınarak başlamıştır. Veri düzlemi ile kontrol düzleminin ayrılması fikri, telefon ağlarında uygulanan ve veri ile kontrolün birbirinden bağımsız olduğu ağ kontrol noktalarından doğmuştur. Bu yaklaşım, maliyet açısından etkin ve güvenli bir çözüm olarak kanıtlanmıştır. Aktif ağlar ağların programlanabilirliğini bir uygulama arayüzü ile sağlamıştır. Kapsül modeli, Tempest , Sanal Ağ Altyapısı ve programlanabilir yönlendirici anahtar modeli , farklı görevleri ve olayları kontrol etme esnekliği sunan aktif ağ modellerine örnektir. Bu modeller daha erken bir dönemde keşfedilmiş olsalar da, uygun altyapı ve donanım desteğinin eksikliği nedeniyle uygulanamamışlardır. YTA'a yönelik büyük bir ilerleme, 2008 yılında OpenFlow'un tanıtılmasıyla gerçekleşmiştir (Rawat, D. B., & Reddy, S. R. 2016).



Şekil 2.1: Yazılım Tanımlı Ağ Mimarisi

Kaynak: (Cicioğlu, M., & Çalhan, A. 2017).

SDN, ağ cihazlarının zekasını kaldırarak ağ işlevlerinin tamamını merkezi bir denetleyici ile yönetmeyi amaçlayan yenilikçi bir teknoloji olarak değerlendirilmektedir. YTA'ların temel yapısı üç ana katmandan oluşur: Uygulama katmanı, Kontrol katmanı ve Veri katmanı. Bu teknoloji, ağ cihazlarındaki zekayı ortadan kaldırarak tüm ağ işlevlerini merkezi bir denetleyiciyle yönetmeyi hedefler.

2.2 Kontrol Katmanı

Kontrol katmanı, YTA'ın tüm işlevlerini yöneten bir denetleyiciden oluşur. Bu katman, altyapı katmanı ile uygulama katmanı arasında aracı görevi görür. Denetleyici, trafik akışını yönetmekten sorumludur ve yönlendirme, akış iletimi ve paket düşürme gibi konularda programlama yoluyla kararlar alır. Dağıtık ortamda bulunan denetleyiciler, doğu ve batı yönlü arayüzler üzerinden birbirleriyle iletişim kurar. Kontrol katmanı ile altyapı katmanı arasındaki iletişim, OpenFlow, NetConf gibi güney yönlü API'ler aracılığıyla gerçekleşir (Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. 2014).

2.3 Uygulama Katmanı

YTA'ın en üst katmanını oluşturur. Bu katman, yazılım ile ilgili iş ve güvenlik uygulamalarını yönetmekle görevlidir. Ağ sanallaştırma, Saldırı Tespit

Sistemleri (IDS), Saldırı Önleme Sistemleri (IPS), güvenlik duvarı uygulamaları ve mobilite yönetimi gibi uygulamalar bu katman tarafından yönetilir. Uygulama Katmanı, kontrol katmanı ile Kuzey yönlü uygulama arayüzü olarak da bilinen Uygulama Kontrol Düzlemi Arayüzü (A-CPI) kullanarak iletişim kurar (Voellmy, A., Kim, H., & Feamster, N. 2012).

2.4 Veri/Altyapı Katmanı

YTA'da veri katmanı, yönlendirici, anahtar ve erişim noktası gibi ağ cihazlarından oluşur. Veri katmanının ana görevi, ağ üzerinden gelen ve giden veri paketlerini iletmektir. Bu, belirli kurallar ve politikalar doğrultusunda gerçekleştirilir.Bu katman, paketlerin hızlı bir şekilde işlenmesi ve yönlendirilmesi için optimize edilmiştir. Bu, düşük gecikme süreleri ve yüksek bant genişliği sağlar (Hu, F., Hao, Q., & Bao, K. 2014).

Kuzey ve Güney Arayüzleri

Kuzey arayüzü: Kuzey yönlü arayüz, yönetim düzlemi ile kontrol düzlemi arasında veri değişimini sağlayan, uygulama tabanlı bir ağ katmanıdır. Bu arayüz, yönetim düzlemindeki uygulamalar aracılığıyla kontrolcüye komut iletir ve genellikle yazılım tabanlı bir ekosistemdir. YTA mimarisinin iki temel soyutlama katmanı olan Kuzey yönlü ve Güney yönlü arayüzler, bu mimarinin temelini oluşturur. Güney yönlü arayüz için yaygın olarak kullanılan protokol OpenFlow'dur; ancak, Kuzey yönlü arayüz için henüz bir standart protokol yoktur ve bu konuda çalışmalar sürmektedir (Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., & Gayraud, T. 2014).

Kuzey yönlü arayüz, ağ politikalarının ve operasyonel görevlerin nasıl ifade edileceğini belirler ve bunları kontrolcünün anlayabileceği bir biçime dönüştürür. Bu arayüz, ağ politikalarının oluşturulduğu katman olup, farklı kontrol platformları arasında uygulamaların taşınabilirliği ve uyumluluğu açısından büyük önem taşır. Floodlight, Trema, NOX, Onix ve OpenDaylight gibi kontrolcüler, kendi Kuzey yönlü arayüzlerini sunar, ancak her biri kendine özgü tanımlamalara sahiptir (Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S.2014).

Güney yönlü arayüzü: Güney yönlü arayüz, SDN anahtarları ile kontrol katmanı bileşenleri arasında köprü işlevi gören ve iletişim protokolü olarak tanımlanan bir katmandır. Bu arayüz, kontrol ve veri katmanlarının tam olarak ayrılmasını sağlar. Güney arayüzü, veri düzlemi ile kontrol düzlemi arasında bir protokol sunar ve YTA'nın katmanlı mimarisinde ağ altyapısı ile ağ hypervisor arasında yer alır. OpenFlow ve Network Configuration Protocol (NetConf), YTA için en yaygın olarak kullanılan Güney arayüz standartlarıdır. OpenFlow, veri ve kontrol cihazları arasında bir iletişim kanalı sağlamakta ve yönlendirme cihazlarına ortak özellikler sunmaktadır. Ayrıca, OpenFlow ile birlikte ForCES, Open vSwitch Database (OVSDB), POF, OpFlex ve OpenState gibi uygulamalar da YTA için Güney arayüz çözümleri arasında bulunmaktadır. Bu protokoller ve uygulamalar, ağın kontrol ve veri katmanlarını etkin bir şekilde ayırarak ağ yönetimini ve esnekliğini artırmayı hedefler. (Cicioğlu, M., & Çalhan, A.2017).

Batı yönlü arayüzü: Batı Yönlü arayüzü, farklı ağların dağıtılmış YTA denetleyicileri arasında bilgi alışverişi sağlayan bir iletişim kanalı olarak çalışır. Bu API, ağ durumu hakkında bilgi değişimini mümkün kılar ve her bir denetleyicinin yönlendirme kararlarını etkileyerek, birden fazla ağ alanında birleşik ağ akışı operasyonlarını kolaylaştırır. Bilgi değişimi için BGP (Border Gateway Protocol) gibi bir yönlendirme protokolünün kullanılması uygun olabilir (Zinner, T., Jarschel, M., Hossfeld, T., Tran-Gia, P., & Kellerer, W.2013).

Doğu Yönlü Arayüzü: MPLS (Çok Protokollü Etiket Anahtarlama) gibi YTA dışı alanların kontrol düzlemleri ile iletişim kurmak için bir araç görevi görür. Eastbound API'nin uygulanması,YTA dışı alan teknolojisine dayanır. Geleneksel ağ ile YTA arasındaki iletişimi anlamak ve yorumlamak için bir modül gereklidir (Zinner, T., Jarschel, M., Hossfeld, T., Tran-Gia, P., & Kellerer, W.2013).

2.5 Openflow Protokolü

Openflow protokolü, veri düzlemindeki cihazların kontrol düzlemindeki bir kontrolör ile etkileşimde bulunmasını sağlayan bir yöntemdir. Stanford Üniversitesi'nin ağ araştırma merkezi tarafından geliştirilen ve uygulanan Openflow protokolü, başlangıçta kampüs ağı üzerinde deneyler ve araştırmalar yürütmek amacıyla deneysel bir protokol olarak tasarlanmıştır. Geleneksel ağlarda, ağ protokollerinin belirli bir göreve odaklanmış ve tek bir satıcıya ait cihazlarda izole

edilmiş olması en büyük sınırlamalardan biridir. Bu durum, ağ üzerinde topolojik değişiklikler yapmayı son derece zorlaştırır. Bu zorlukların üstesinden gelmek için OF protokolü oluşturulmuştur. Bu protokol, 2011 yılında ağ hizmet sağlayıcıları, hükümet ve akademik kurumlar tarafından kurulan ve OF'nin üretim ağlarında standartlaştırılmasını, ticarileştirilmesini ve yaygınlaştırılmasını amaçlayan kâr amacı gütmeyen bir dernek olan ONF tarafından desteklenmektedir (McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J.2008).

2.6 YTA'ın Avantajları ve Dezavantajları

Yazılım Tanımlı Ağlar aşağıdaki avantajları sağlayabilir:

Maliyet:Tüm ağ yönetimi merkezileştirildiği ve otomatikleştirildiği için YTA toplam maliyet tasarrufu sağlar. YTA ayrıca sunucu kullanımını ve sanallaştırmayı geliştirerek kullanıcıların maliyetleri düşürmesine olanak tanır. Bu faydaların yanı sıra YTA, çoklu görevlere olanak sağlayarak ağ operasyonlarını kolaylaştırır. Bu aynı zamanda pahalı donanım ihtiyacını da azaltır.

Güvenlik: YTA, tüm ağın güvenliğini sağlayan bir denetleyici içerir. Ayrıca bu denetleyici, ağın güvenlik düzenlemelerine ve bilgilerine uyulmasını garanti eder. YTA ayrıca merkezi bir yönetim sistemine sahiptir. Güvenlik ve özellikler tek bir organ tarafından yönetilecektir. Sadece tek bir merkezi nokta kullandığı için bu yöntem çok güvenlidir. Yönetici ayrıca güvenlik risklerini engelleyebilir ve bunların sisteme bulaşmasını önleyebilir.

Merkezileştirme: YTA merkezi ağ yönetimini mümkün kılar. Ayrıca, tüm ağ izleme ve yönetimi tek bir konumdan gerçekleştirilebilir. Geleneksel yöntemlerin altyapı yönetimine getirdiği engeli ortadan kaldırır. YTA ayrıca bireysel bazda sistem yönetimini de mümkün kılar.

Optimizasyon:

YTA kullanıldığında, donanım cihazlarını optimize etmek için yeni bir yöntem gerçekleşir. YTA denetleyicilerinin kullanımı ile mevcut ve yeni tüm donanımlara belirli bir rol atayabilir. Sonuç olarak, tek bir göreve adanmış donanım cihazlarının sınırlamasını ortadan kaldırır.

YTA ve OpenFlow, ağ elemanlarının prototiplenmesini, dağıtımını ve yönetimini kolaylaştırır; ancak bu teknolojilerin bazı önemli sorunları vardır:

Denetleyici Erişilebilirliği:

Denetleyici ve anahtarlar arasındaki güçlü bağımlılık, kuralların değiştirilmesi gerektiğinde problemlere yol açabilir. Ağda sadece bir merkezi denetleyici bulunması durumunda, bu bir "tek hata noktası" oluşturabilir. Dağıtık bir yapı benimseyerek erişilebilirlik artırılabilir ve istenmeyen kesintiler önlenebilir. Ayrıca, ağın dayanıklılığını artırmak için yedekleme çözümleri uygulanabilir.

Akış Tablolarının Tutarlılığı:

Birden fazla denetleyicinin aynı akış tablolarını yönetmesi durumunda tutarsızlıklar ortaya çıkabilir. Örneğin, bir üretim denetleyicisi ve deneysel bir denetleyici aynı akış tablolarını yönetiyorsa, daha düşük güvenlik kontrollerine sahip olan deneysel denetleyici "zincirin en zayıf halkası" olabilir. Bu da tutarsız akış grafiklerine yol açabilir. Potansiyel tehditleri önlemek için bir akış gözlemcisi kullanılabilir.

Ağın Ölçeklenebilirliği:

Çok sayıda paket denetleyiciye ulaştığında ağ performansı olumsuz etkilenebilir. Bu nedenle, kontrol düzleminin dağıtık olması önemlidir. Aksi takdirde, denetleyici bir "darboğaz" oluşturabilir.

Ağ Performansı:

Akış tablosu boyutları sınırlıdır ve çok sayıda akışı yönetmek zor olabilir. Proaktif bir yaklaşım benimsenerek performans sorunları azaltılabilir. Proaktif yaklaşım, denetleyici ile anahtarlar arasındaki mesaj alışverişini sınırladığı için reaktif moddan daha iyi performans sunar.

Yukarıda belirtilen konular, topluluk araştırmacıları tarafından ele alınmış ve SDN'nin gelecekte karşılaşabileceği zorlukları temsil etmektedir (Ongaro, F.2014).

2.7 YTA Güvenliği

Yazılım tanımlı ağlar esneklik programlanabilirlik ve basitlik, sayesinde ağ yöneticilerine önemli kolaylıklar sunmaktadır. Bu özellikler hem akademik hem de endüstriyel kuruluşların büyük ilgisini çekmiştir. Bununla birlikte, YTA'ların

güvenliğini ele alan çalışmalar yeterince kapsamlı değildir. YTA'lar geleneksel ağların doğasında bulunan bazı güvenlik sorunlarını ele alırken, kontrol ve veri düzlemlerinin ayrılması yeni güvenlik zorluklarını beraberinde getirmektedir. Bu zorluklar ağ trafiğinin ve verilerin gizliliğini, bütünlüğünü, gerçekliğini ve tutarlılığını tehlikeye atma potansiyeline sahiptir. YTA'larda gözlemlenen güvenlik tehditleri arasında yetkisiz erişim, veri sızıntısı, veri modifikasyonu, kötü amaçlı yazılım, hizmet reddi (DoS) saldırıları, yapılandırma hataları ve sistem düzeyinde güvenlik açıkları yer almaktadır. Bu tehditler uygulama katmanındaki kimlik doğrulama ve yetkilendirme sorunları, kontrol düzlemindeki uygulamalardan kaynaklanan tehditler, ölçeklenebilirlikle ilgili tehditler, dağıtık kontrol düzlemindeki DoS saldırıları ve zorluklar ve veri düzlemindeki zorluklar olarak kategorize edilmektedir (Yavuz, A., & Tuna, G.2020)

"Diego Kreutz ve meslektaşları (2015), YTA mimarisinde yedi ana tehdit unsuru tespit etmiştir. Bu unsurlardan dördü geleneksel ağlarda da bulunurken, üçü yalnızca YTA'lara özgüdür ve YTA'lar için en büyük riski temsil etmektedir. Bu tehdit unsurları, aşağıda detaylı olarak incelenmiştir.

Tehdit Unsuru 1

Ağın içinde sahte trafik oluşturarak yönlendiricilere ve anahtarlara saldıran bir yöntemdir. Bu saldırı, OpenFlow anahtarlarına ve denetleyicinin kaynaklarına DoS saldırıları başlatmak için kullanılır.

Tehdit Unsuru 2

Ağ anahtarlarındaki güvenlik açıklarını hedef alır. Ele geçirilen bir anahtar, ağ trafiğini kopyalama, yönlendirme, yavaşlatma, düşürme ve sahte trafik üreterek cihazları aşırı yükleme gibi sorunlara yol açabilir.

Tehdit Unsuru 3

Veri hırsızlığı veya DoS saldırıları gerçekleştirmek için kontrol düzlemine yapılan saldırıları ifade eder. Transport Layer Security (TLS) / Secure Socket Layer (SSL) bu iletişimi tamamen güvence altına alamaz. Bir saldırgan kontrol düzlemine eriştiğinde, veri kaçırma ve DoS saldırıları gerçekleştirmek için sanal kara delikler oluşturabilir.

Tehdit Unsuru 4

SDN'lere karşı en tehlikeli saldırı vektörüdür. Hatalı veya kötü niyetli bir denetleyici, tüm ağın güvenliğini riske atabilir. Denetleyici, davranışını değiştirebildiği için ağdaki kötü amaçlı yazılımlara karşı etkisiz kalabilir.

Tehdit Unsuru 5

Denetleyici ve uygulamalar arasında güvenilir bir ilişki kurulamaması nedeniyle yukarıda bahsettiğimiz 3. Tehdit Vektörü'ne benzer. Bu durum, denetleyicinin üzerinde kötü amaçlı yazılımların kullanılmasına yol açabilir.

Tehdit Unsuru 6

Yönetici istasyonlarındaki güvenlik açıklarını istismar eden bu saldırı geleneksel bilgisayar ağlarında da görünür. SDN'lerde bu saldırının tehdit seviyesi oldukça yüksektir.

Tehdit Unsuru 7

Sorunun nedenini anlamayı ve hızlıca çözmeyi engelleyen saldırılardır. Ağın güvenilirliğini tehlikeye atar.”

3. ANOMALİ TESPİTİ

Anomali tespiti, veri setinde normal davranıştan farklılık gösteren veri noktalarını veya desenleri tanımlama sürecidir. Bu farklılıklar genellikle ilgi çekici, nadir ya da potansiyel olarak zararlı olaylara işaret eder ve dolandırıcılık tespiti, ağ güvenliği, sağlık izleme ve üretim hatalarının belirlenmesi gibi birçok alanda önemli bir rol oynar.

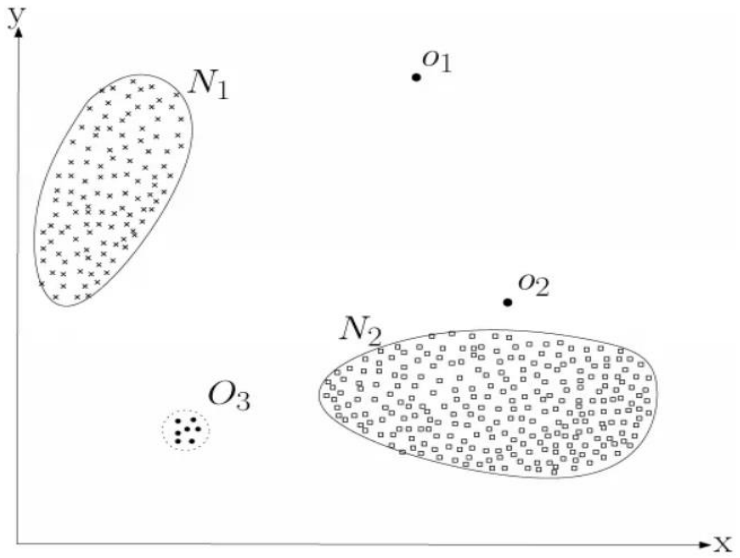
Chandola, Banerjee ve Kumar (2009), anormallikleri normal durumlarla olan ilişkileri ve bağlantıları temelinde üç kategoriye ayırmışlardır.

Nokta anomalileri: Bu anomali, veri kümesindeki diğer verilerle büyük ölçüde uyumsuz bir özelliğe sahip olan verilerdir ve bu tür anomaliler nokta anomalileri olarak bilinir. Bu en basit anomali türüdür. Örneğin, bir ağda normalde belirli bir port üzerinden çok az miktarda trafik geçerken, bir anda bu port üzerinden büyük miktarda veri transferi yapılması, ağın normal kullanım modelinden önemli ölçüde sapar ve Tip-1 anomali olarak değerlendirilir. Nokta anomaliler, büyük bir veri kümesinde istatistiksel gürültü olarak veya anomali tespit sistemleri için önemli kısa süreli olaylar olarak görülebilir (Cook, Mısırlı, & Fan, 2019).

Bağlamsal anomaliler: Bu anomaliler, bağlamsal anomaliler olarak adlandırılır ve izole olarak incelendiğinde anormal görünmeyen, ancak belirli bir bağlamda değerlendirildiğinde anomali olarak tanımlanan veri noktalarını kapsar. Örneğin, iş saatleri dışında bir sunucudan gelen yüksek miktarda veri trafiği, iş saatleri içinde aynı miktarda trafiğe kıyasla anormal olarak değerlendirilebilir. Bu tür anomaliler, belirli bir zaman diliminde veya bağlamda meydana gelen beklenmedik olaylar olarak kabul edilir.

Toplu anomaliler: Anomaliler ise toplu anomaliler olarak bilinir ve tek tek incelendiğinde anormal olmayan, ancak birlikte değerlendirildiğinde anormal bir desen oluşturan veri noktalarını içerir. Örneğin, bir kullanıcı aynı günde birçok farklı IP adresine küçük veri paketleri gönderiyorsa, bu durum tek başına anormal görünmeyebilir, ancak bu tür faaliyetler topluca analiz edildiğinde bir DDoS saldırısını gösterebilir.

Aşağıda iki boyutlu bir veri kümesi üzerinde anomali tespiti gösteren bir grafik sunulmaktadır. Grafikte, farklı veri kümeleri ve bu kümeler arasındaki anomaliler işaretlenmiştir.



Şekil 3.1: Anomali Tespiti

Kaynak: (Coşkun, M.2018)

Grafikte iki ana normal veri kümesi bulunmaktadır: N1 ve N2.

N1 kümesi, grafik üzerinde "x" işaretleri ile gösterilmektedir ve "N1" olarak etiketlenmiştir. N1 kümesi, yoğun bir veri noktaları topluluğu şeklinde belirli bir alanda toplanmıştır. Bu, N1 kümesindeki veri noktalarının birbirine oldukça yakın ve belirgin bir yoğunlukta olduğu anlamına gelir. Böylece, N1 kümesi, veri setinde normal davranışın bir örneği olarak kabul edilir.

N2 kümesi ise grafik üzerinde kare sembolleri ile gösterilen ve "N2" olarak adlandırılan veri grubudur. N2 kümesi de yoğun bir veri noktaları topluluğudur ve kendi belirli alanı içinde yer almaktadır. N2'deki veri noktaları da birbirine oldukça yakındır ve belirgin bir yoğunluk oluşturmaktadır. Bu, N2 kümesinin de veri setinde normal davranışı temsil ettiğini gösterir.

Her iki küme de (N1 ve N2), veri setinde normal davranış örneklerini temsil eder ve veri noktalarının yoğunlaştığı belirli alanlar olarak grafik üzerinde açıkça görülebil

Anomaliler (O1, O2 ve O3):

Grafik üzerinde "O1" ve "O2" olarak işaretlenen iki nokta, her iki normal veri grubundan (N1 ve N2) uzakta bulunmaktadır. Bu noktalar, herhangi bir gruba ait olmadıkları için anomali olarak değerlendirilir. Grafik üzerinde "O3" olarak belirtilen küçük bir kümecik, normal veri gruplarından (N1 ve N2) ayrı bir alanda bulunmaktadır. Bu kümecik, küçük bir veri noktaları topluluğu olup belirgin bir sınırla çevrelenmiştir. O3, grup anomalisine örnek olarak gösterilebilir.

Grafikte, normal veri kümeleri (N1 ve N2) ve bu kümelerden sapma gösteren anormal noktalar (o1 ve o2) ile anormal kümecik (O3) belirgin bir şekilde gösterilmiştir. Bu tür görseller, anomali tespitinde hangi veri noktalarının veya kümelerin normal veri desenlerinden sapma gösterdiğini görsel olarak anlamamıza yardımcı olur. Bu tür anomalilerin tespiti, özellikle ağ trafiği, siber güvenlik, finansal dolandırıcılık ve sağlık gibi alanlarda kritik öneme sahiptir. Derin öğrenme ve makine öğrenmesi algoritmaları, bu tür anomalileri tespit etmek için veri kümesindeki özellikleri öğrenir ve normdan sapma gösteren noktaları veya kümeleri işaretler. Böylece, olası sorunların erken tespiti ve müdahalesi mümkün olur.

Son zamanlarda Yazılım Tanımlı Ağ (SDN) kavramının ortaya çıkışı ve gelişimi, ağ mimarisini daha esnek hale getirerek ağ yönetimini kolaylaştırmıştır (Ongaro, F. 2014). SDN, ağ fonksiyonlarını kontrol mantığı ve yönlendirme mantığı olarak ikiye ayırır: anahtar, basit bir yönlendirme ve işleme cihazına dönüşürken, kontrol mantığı merkezi bir kontrol cihazı tarafından yürütülür. Ancak, başlangıç tasarımında güvenlik unsurlarına yeterince dikkat edilmemesi nedeniyle bazı potansiyel güvenlik tehditleri ortaya çıkmıştır. Örneğin, Shin ve Gu, Openflow protokolünün DDoS (Dağıtık Hizmet Engelleme) saldırısına karşı savunmasız olduğunu belirtmiştir (Shin, S., & Gu, G. 2013) . SDN'nin özel mantık yapısı gereği, kontrol cihazı ele geçirildiğinde, ağ cihazlarının çoğu veya hatta tüm ağ normal şekilde çalışamaz hale gelir. Kontrol cihazlarına yönelik üç ana saldırı türü mevcuttur: sahtecilik saldırısı (örneğin, DoS), yetkisiz erişim gibi izinsiz giriş saldırıları ve veri değiştirme saldırısı, örneğin, anahtarlardaki verilerin değiştirilmesi (Qin, Y., Wei, J., & Yang, W.2019).

Düzenli veya belirgin kalıplarla ilgilenen çoğu sorun ve görevlerin aksine, anomali tespiti azınlık, tahmin edilemez ve nadir olaylarla uğraşır, bu da hem

derinlemesine hem de yüzeysel tespit yöntemlerine özgü bazı benzersiz problem karmaşıklıklarıyla sonuçlanır:

- **Tahmin Edilemezlik**. Anomaliler, ani ve beklenmedik davranışlar, veri yapıları ve dağılımlar gibi birçok bilinmeyenle ilişkilidir. Yeni terör saldırıları, dolandırıcılık ve ağ saldırıları gibi, meydana gelene kadar bilinmezler.

- **Heterojen Anomali Sınıfları**. Anomaliler heterojendir, bu nedenle bir sınıf anomali, başka bir sınıf anomaliden tamamen farklı anormal özellikler gösterebilir. Örneğin, video gözetiminde, soygunlar, trafik kazaları ve hırsızlık gibi anormal olaylar görsel olarak oldukça farklıdır.

- **Nadirlik ve Sınıf İstikrarsızlığı**. Anomaliler genellikle nadir veri örnekleridir, bu da normal örneklerin çoğunlukta olduğu veri setleriyle tezat oluşturur. Bu nedenle, büyük miktarda etiketli anormal örnek toplamak zor veya hatta imkânsız olabilir. Bu, çoğu uygulamada büyük ölçekli etiketli verilerin eksikliğine yol açar. Sınıf dengesizliği ayrıca, anomali yanlış sınıflandırmalarının, normal örneklerin yanlış sınıflandırılmasından çok daha maliyetli olmasından kaynaklanır.

- **Çeşitli anomali türleri**. Üç farklı anomali türü araştırılmıştır. Nokta anomalileri, diğer bireysel örneklere kıyasla anormal olan tekil örneklerdir, örneğin bir hastanın anormal sağlık göstergeleri gibi. Koşullu anomaliler, bağlamsal anomaliler olarak da bilinir ve belirli bir bağlamda anormal olan bireysel anomali örnekleridir, yani veri örnekleri belirli bir bağlamda anormal olup, diğer durumlarda normaldir. Gerçek dünyada, bağlamlar oldukça farklı olabilir, örneğin belirli bir zaman diliminde ani sıcaklık düşüşü/artışı veya alışılmadık mekansal bağlamlarda hızlı kredi kartı işlemleri gibi. Grup anomalileri ise, toplu anomaliler olarak da bilinir ve diğer veri örneklerine göre bir bütün olarak anormal olan veri örneklerinin alt kümesidir; bu alt kümedeki bireysel üyeler anormal olmayabilir. Örneğin, sosyal ağlarda sahte hesaplar tarafından oluşturulan son derece yoğun alt grafikler bir bütün olarak anomali oluşturur, ancak bu alt grafiklerdeki bireysel düğümler, gerçek hesaplar kadar normal görünebilir (Chandola, V., Banerjee, A., & Kumar, V.2009).

3.1 Anomali Tespitinin Önemi

Anomali tespiti, günümüz veri analitiği ve ağ güvenliği dünyasında önemli bir yere sahiptir. Anomaliler, normal veri örüntülerinden farklılık gösteren nadir ve

beklenmedik olaylardır. Bu tür sapmalar, potansiyel olarak zararlı veya istenmeyen durumlara işaret edebilir ve bu nedenle erken tespit edilmesi büyük önem taşır.

Anomali tespitinin önemi şu başlıklar altında incelenebilir:

Güvenlik ve Koruma: Anomali tespiti, ağ güvenliği, siber güvenlik ve bilgi güvenliği alanlarında önemli bir savunma mekanizmasıdır. Saldırganlar, ağlarda olağandışı faaliyetlerle kendilerini belli ederler ve bu anomalilerin erken tespiti, veri ihlalleri, DDoS saldırıları ve diğer siber tehditlerin önlenmesine yardımcı olur. Örneğin, bir ağda aniden artan veri trafiği veya beklenmeyen veri erişim talepleri, bir siber saldırının habercisi olabilir ve zamanında müdahale ile büyük zararlardan kaçınılabilir.

Dolandırıcılık Tespiti: Finansal kurumlar, kredi kartı şirketleri ve e-ticaret platformları, dolandırıcılık tespitinde anomali tespit tekniklerinden yararlanır. Anormal harcama kalıpları veya alışılmadık işlem davranışları, dolandırıcılık faaliyetlerini işaret edebilir. Bu tür anomalilerin hızlı bir şekilde belirlenmesi hem kullanıcıların hem de şirketlerin finansal kayıplarını en aza indirir.

Sağlık ve Tıp: Anomali tespiti, tıbbi teşhislerde ve hasta izleme sistemlerinde hayati bir rol oynar. Hastaların tıbbi verilerindeki olağandışı değişiklikler, erken teşhis ve tedavi gerektiren sağlık sorunlarını gösterebilir. Örneğin, bir hastanın kalp ritmindeki ani değişiklikler, kalp krizi veya diğer ciddi sağlık sorunlarının habercisi olabilir ve hızlı müdahale hayat kurtarıcı olabilir.

Endüstriyel Uygulamalar: Üretim hatlarında ve endüstriyel süreçlerde anomali tespiti, makine arızalarının ve operasyonel sorunların önceden belirlenmesine olanak tanır. Bu, bakım maliyetlerini düşürür, duruş sürelerini azaltır ve genel operasyonel verimliliği artırır. Örneğin, bir üretim makinesinde normalden sapma gösteren titreşim veya sıcaklık verileri, olası bir arızanın işareti olabilir ve zamanında bakım yapılmasını sağlar.

Veri Kalitesi ve Temizliği: Anomali tespiti, büyük veri kümelerindeki hataları ve veri kalitesindeki sapmaları belirlemek için kullanılır. Verideki hatalar veya yanlışlıklar, analiz sonuçlarını ve karar verme süreçlerini olumsuz etkileyebilir. Bu nedenle, anomalilerin erken tespiti ve düzeltilmesi, verinin doğruluğunu ve güvenilirliğini artırır.

Sonuç olarak, anomali tespiti, çeşitli alanlarda kritik öneme sahiptir ve güvenlik, doğruluk ve verimlilik sağlamak için vazgeçilmez bir araçtır. Anomalilerin zamanında ve doğru bir şekilde tespit edilmesi, potansiyel sorunların erken müdahale ile çözülmesine olanak tanır ve bu da sistemlerin genel güvenliğini ve performansını artırır.

3.2 Makine Öğrenmesi Yöntemleri

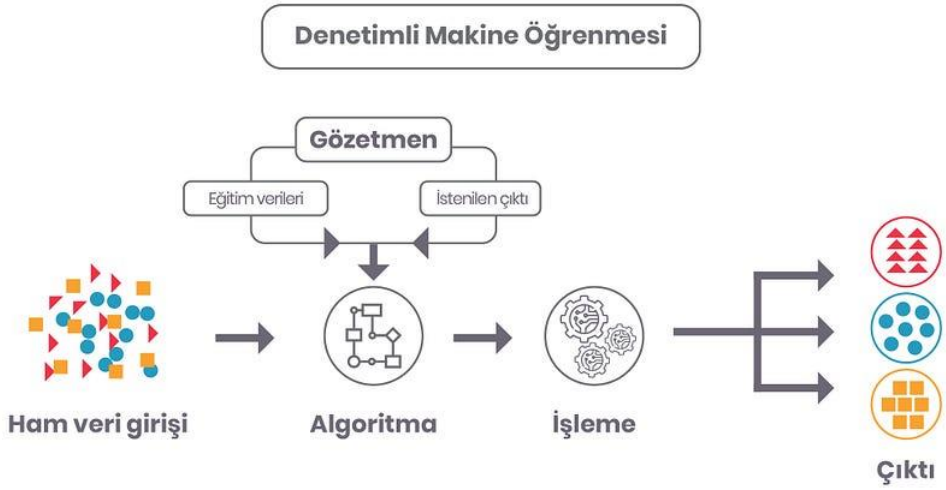
Günümüzde, veri hacmindeki hızlı artış ve hesaplama gücündeki gelişmeler sayesinde, makine öğrenimi ve derin öğrenme yöntemleri veri analitiği ve yapay zekâ alanlarında giderek daha önemli hale gelmektedir. Bu yöntemler, büyük veri setlerinden anlamlı desenler çıkarmak, tahminlerde bulunmak ve karmaşık sorunları çözmek için kullanılır. Makine öğrenimi, istatistiksel yöntemler ve algoritmalar kullanarak veriden öğrenme sürecini ifade ederken, derin öğrenme, çok katmanlı yapay sinir ağları kullanarak bu süreci daha ileri bir seviyeye taşır.

Makine öğrenimi, çeşitli veri türleri üzerinde çalışabilen geniş bir algoritma yelpazesi sunar. Denetimli, denetimsiz ve yarı denetimli öğrenme gibi farklı öğrenme paradigmaları, çeşitli veri yapıları ve problem türlerine uyarlanabilir. Diğer taraftan, derin öğrenme, özellikle büyük ve karmaşık veri setlerinde görüntü, ses ve doğal dil işleme gibi alanlarda olağanüstü başarılar elde etmiştir.

Bu giriş bölümünde, makine öğrenimi ve derin öğrenme yöntemlerinin genel prensiplerini, uygulama alanlarını ve aralarındaki farkları ele alacak ve neden bu teknolojilerin günümüzde mevcut olan en heyecan verici ve güçlü araçlar arasında yer aldığını inceleyeceğiz.

Goldstein ve Uchida (2016), kullanılan yöntemlere göre anomali tespitini üç alt kategoriye ayırmıştır:

Denetimli Anomali Tespiti: Bu yöntem, tamamen etiketlenmiş bir eğitim veri seti kullanır. Ancak, sınıf dengesizliği gibi sorunlar nedeniyle, destek vektör makineleri veya yapay sinir ağları gibi algoritmaların kullanılması daha uygun olabilir. Anomaliler önceden tanımlanamadığı ve eğitim seti doğru şekilde etiketlenemediği için bu yöntemi her durumda kullanmak mümkün olmayabilir.

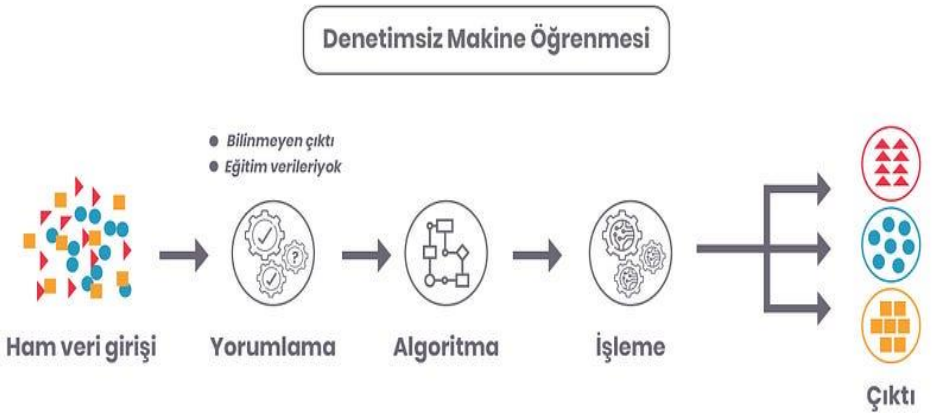


Şekil 3.2: Denetimli Öğrenme

Kaynak: (Erkan İ.2020)

Yarı-Denetimli Anomali Tespiti: Bu yöntemde, yalnızca normal veriler içeren bir eğitim seti kullanılır. Model, normal davranışları öğrenir ve bu davranışların dışındaki verileri anomali olarak tanımlar. Bu yöntem, "tek sınıf" sınıflandırıcılar olarak da bilinir.

Denetimsiz Anomali Tespiti: Bu yöntemde, etiketlenmiş veri kullanılmaz ve eğitim ile test verileri ayrılmaz. Bu yöntemde k-En Yakın Komşu (k-NN), yerel aykırı değer faktörü (LOF) ve etkili aykırı değer (INFLO) gibi algoritmalar kullanılabilir."



Şekil 3.3: Denetimsiz Öğrenme

Kaynak:(Erkan İ.2020)

4. DERİN ÖĞRENME

Derin öğrenme, makinelerin özellik çıkarma, tespit ve öğrenme işlemleri için ileri düzey bir makine öğrenmesi yöntemidir ve bu işlemleri ardışık çok sayıda katman kullanarak gerçekleştirir. Her katman, önceki katmanın çıktısını girdi olarak alır. Derin öğrenme, manuel olarak çıkarılan özellikler yerine verilerin temsillerinden öğrenmeye dayanır ve veriyi en iyi şekilde temsil eden hiyerarşik özellik çıkarımı için etkili algoritmalar kullanır (Karataş, G.2020).

Derin öğrenme ağında bulunan nöron, bir sinir ağı içindeki temel hesaplama birimidir. Nöron girdiyi alır, bu girdileri ağırlıklarla çarpar, bir sapma değeri ekler ve ardından bu toplamı bir aktivasyon fonksiyonundan geçirerek bir çıktı üretir. Nöronlar, veriyi işlemek ve tahminlerde bulunmak için katmanlar halinde düzenlenmiş ve birbirlerine bağlanmıştır. Yapay sinir ağlarındaki nöronlar, biyolojik sinir hücrelerini taklit eder ve ağırlıklar, öğrenme süreci boyunca en iyi hale getirilir.

İlk çok katmanlı öğrenme algoritması, Ivakhnenko ve Lapa tarafından 1966 yılında tanıtılmıştır. Bu algoritma, her katmanda verinin en iyi özelliklerini seçerek ve bir sonraki katmana ileterek uçtan uca eğitim sağlamak için geri bildirim mekanizmasını kullanır. İlk başarılı derin sinir ağı uygulaması, Yann LeCun ve ekibi tarafından posta kutusu metni üzerinde gerçekleştirilmiş, ancak uzun eğitim süresi nedeniyle pratik bulunmamıştır. LeCun, el yazısı rakamları sınıflandırmak için evrişimli ağları geri yayılım algoritması ile birleştirerek "LeNet" ağını oluşturmuştur.

1995 yılında Brendan Frey, Peter Dayan ve Geoffrey Hinton, yüzlerce tamamen bağlantılı gizli katmana sahip bir ağın "Wake-Sleep" algoritması kullanılarak eğitilebileceğini kanıtlamıştır. Ancak, yüksek hesaplama gücü gereksinimleri nedeniyle, 1990'larda yapay sinir ağları yerine destek vektör makineleri tercih edilmiştir. "Derin Öğrenme" terimi, 2000 yılında Igor Aizenberg ve ekibi tarafından ortaya atılmıştır. 2006'da Geoffrey Hinton, geri yayılım kullanarak çok katmanlı ileri beslemeli sinir ağlarının nasıl etkili bir şekilde eğitileceğini göstermiştir. GPU hızlarındaki artış sayesinde, ön eğitim gerektirmeden derin ağları eğitmek mümkün hale gelmiştir. 2012 yılında, Krizhevsky, Sutskever ve Hinton,

GPU destekli çalışmalarında "dropout" normalizasyon yöntemini kullanarak aşırı öğrenmeyi azaltmıştır (Karataş, G.2020).

4.1 Derin Öğrenme Çalışma Alanları

Derin öğrenme, çeşitli alanlarda karşılaşılan sorunlara yaygın bir şekilde uygulanmıştır. İnsan hareketi algılama, müşteri ilişkileri yönetimi otomasyonu, araçların otonom sistemleri, multimedya yönetimi ve iç mekân yönlendirme gibi alanlarda derin öğrenme algoritmaları kullanılmaktadır. DARPA, büyük belgeleri etkili bir şekilde aramak ve anormallikleri tespit etmek için Doğal Dil İşleme yeteneklerini kullanan Derin Arama ve Metin Filtreleme programını geliştirmiştir. NVIDIA tarafından üretilen derin öğrenme kartları, bilgisayarlarda derin öğrenme algoritmalarının daha verimli çalışmasını sağlar (Niu, X., Zhu, Y., Zhang, X.2014).

Derin öğrenme, görüntü işleme ve doğal dil işleme alanlarında geniş bir kullanım alanı bulmuştur. Görüntü işlemede, görüntü sınıflandırma ve nesne tanıma gibi konular ön plana çıkmaktadır. Derin öğrenme ayrıca otonom araç teknolojilerinde önemli bir rol oynamaktadır. Doğal dil işleme alanında, derin öğrenme algoritmaları metin özetleme, sınıflandırma ve soru-cevaplama gibi problemleri çözmede başarılı bir şekilde kullanılmaktadır (Karataş, G.2020).

4.2 Derin Öğrenme Modelleri

Derin öğrenme, makine öğreniminin bir alt kolu olarak kabul edilir ve insan beyninin işleyişinden ilham alarak geliştirilen çok katmanlı yapay sinir ağları (ANN'ler) kullanılarak gerçekleştirilir. Derin öğrenme, büyük ve karmaşık veri setlerinden anlamlı bilgiler çıkarma yeteneğiyle tanınır ve pek çok alanda köklü değişikliklere yol açmıştır.

Geleneksel makine öğrenimi algoritmalarıyla karşılaştırıldığında, derin öğrenme modelleri verideki özellikleri otomatik olarak öğrenme ve keşfetme kabiliyetine sahiptir. Bu özellik, özellikle büyük veri kümeleriyle çalışırken manuel özellik mühendisliği gereksinimini ortadan kaldırır. Verinin ham formatından başlayarak, derin öğrenme aşamalı olarak daha karmaşık ve soyut özellikler elde eder.

Önemli derin öğrenme modelleri arasında Yapay Sinir Ağları (ANN), Evrişimli Sinir Ağları (CNN), Tekrarlayan Sinir Ağları (RNN) ve Transformer modelleri bulunmaktadır. Bu modeller, görüntü tanıma, doğal dil işleme, ses tanıma, otonom sürüş, tıbbi görüntü analizi gibi çeşitli alanlarda başarılı bir şekilde uygulanmıştır.

4.2.1 Yapay sinir ağları (ANN)

Yapay sinir ağları (ANN'ler), biyolojik sinir sistemlerinin, özellikle de beyindeki sinir hücrelerinin (nöronların) işleyiş biçiminden esinlenerek oluşturulmuş bir tür yapay zekâ modelidir. Bu ağlar, öğrenme, yorumlama ve otonom karar verme yeteneklerini gerçekleştirmeyi amaçlar. Yapay sinir ağları, giriş katmanı (input layer), bir veya daha fazla gizli katman (hidden layer) ve çıkış katmanı (output layer) olmak üzere üç ana katmandan oluşur. Giriş katmanı gözlemleri alır ve aktarır, gizli katman bu değerleri işlemden geçirir ve çıkış katmanı sonuca ulaşır. Her katman, bir dizi nöron içerir ve bu nöronlar, aktivasyon fonksiyonları aracılığıyla girdileri işler. Nöronlar arasındaki bağlantılar, 'ağırlıklar' olarak adlandırılan parametrelerle belirlenir ve bu ağırlıklar, ağın öğrenme sürecinde ayarlanarak modelin performansını optimize eder. Bu şekilde, yapay sinir ağları karmaşık veri örüntülerini tanımlayabilir, sınıflandırabilir ve öğrenebilir.

Yapay sinir ağları, ilk olarak 1943 yılında Warren S. McCulloch ve Walter Pitts tarafından tanıtılmıştır. Ancak, o dönemde bilgisayar teknolojisinin yetersizliği nedeniyle bu modellerin potansiyeli anlaşılamamıştır. 2000'li yıllarda bilgisayar gücünün artmasıyla yapay sinir ağları daha başarılı ve popüler hale gelmiştir. İnsan beyninde yaklaşık 100 milyar nöron bulunmasına rağmen, mevcut teknolojilerle bu yapıyı tam anlamıyla modellemek mümkün değildir. Ancak, teknolojik ilerlemelerle birlikte yapay sinir ağları da sürekli olarak gelişmektedir.

Yapay Sinir Ağları (ANN), öğrenme sürecini gerçekleştirmek için birçok önceden belirlenmiş örneği kullanır ve daha önce karşılaşmadığı örnekler hakkında bilgi edinme yeteneğine sahiptir. Bilgiyi bir veritabanı veya programda değil, ağ üzerinde saklamak amacıyla Makine Öğrenimi yöntemlerini kullanırlar. Bilgiyi işlemeleri, klasik bilgisayar programlama yöntemlerinden tamamen farklıdır. Desen tanıma, sınıflandırma ve ilişkilendirme gibi görevleri yerine getirebilirler. Bilgi, tüm ağın bir olayı işlemek için birlikte çalışmasıyla ağ üzerinde dağıtılır. Eğitim

sürecinden sonra, gelen verilerde bazı eksiklikler olsa bile sonuç üretebilirler, bu da onları hata toleranslı hale getirir. Ancak, Yapay Sinir Ağları sadece sayısal verilerle çalışabilir (Patterson, 1998).

Yapay Sinir Ağlarının (ANN'ler) bazı dezavantajları bulunmaktadır: ANN'ler sadece sayısal verilerle çalıştığı için, sayısal olmayan veriler ağa sunulmadan önce ayrı bir işlemden geçirilmelidir. Eğitim süresini tahmin etmek zordur ve ANN'ler donanım ve kurallara bağlıdır. Her problem için özel özelliklerin incelenmesi gerekmektedir, bu da zaman alıcı bir süreçtir. Uygun yapıyı belirlemek için çeşitli yöntemlerle sürekli deneyler yapmak gerekir. Ayrıca, ağın nasıl davranacağını ve öğrenme mekanizmasını nasıl oluşturacağını önceden kestirmek güçtür (Patterson, 1998).

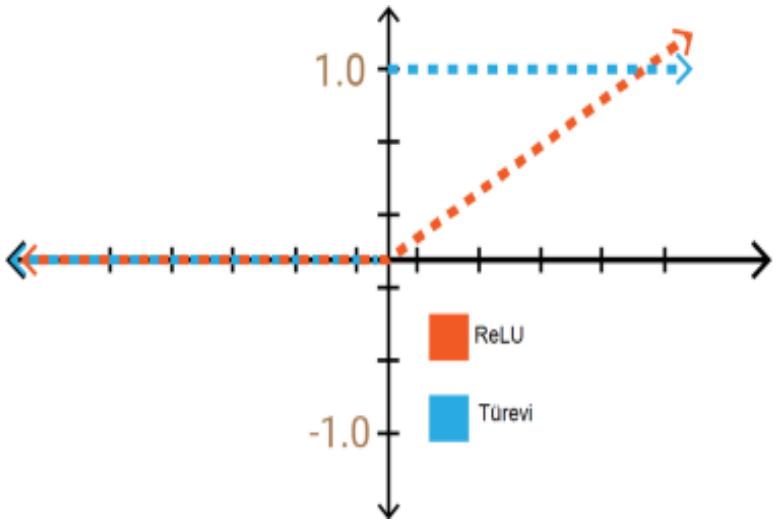
Yapay Sinir Ağları'nın (ANN) ilk hesaplamalı modeli, W.S. McCulloch ve W.A. Pitts tarafından 1943'te yayımlanan bir makalede sunulmuştur (McCulloch ve Pitts, 1943). Bu araştırmacılar, insan beyninin öğrenme kapasitesinden esinlenerek elektrik devreleri kullanmış ve yapay bir sinir hücresi yapısı geliştirmişlerdir. Bu yapı, insan benzeri mantıksal ifadelerin formüle edilmesini sağlamıştır. Sonrasında, birden fazla hücrenin birlikte çalışması ilkesine dayanan paralel yürütme tekniği ile öğrenme kuralları oluşturulmuştur.

1949 yılında Donald Hebb, "Organization of Behavior" adlı kitabında "Hebbian öğrenme" adlı temel öğrenme teorisini ele almıştır (Hebb, 1949). Hebb kuralı, sinir ağlarının bağlantı sayısı ile öğrenebilme ve uyum sağlayabilme yeteneği arasında bir ilişki olduğunu göstermiştir. Bu kuram, YSA'yı geliştirmeye yönelik çalışmalara ilham vermiştir (Karataş, G.2020).

Aktivasyon Fonksiyonları

Düzeltilmiş Doğrusal Birim Aktivasyon Fonksiyonu (ReLU):

Derin öğrenmede, özellikle görüntü işleme alanında yaygın olarak kullanılan ReLU fonksiyonu, negatif girdiler için 0, pozitif girdiler için ise girdi değerini alır. ReLU, diğer aktivasyon fonksiyonlarına kıyasla daha hızlı ve hesaplanması daha kolaydır. Konvolüsyonel sinir ağı (CNN) mimarilerinde, yüksek parametre ve nöron sayıları nedeniyle ReLU kullanımı, hesaplama maliyetini azaltmada önemli bir rol oynar.



Şekil 4.1: Relu Fonksiyonu

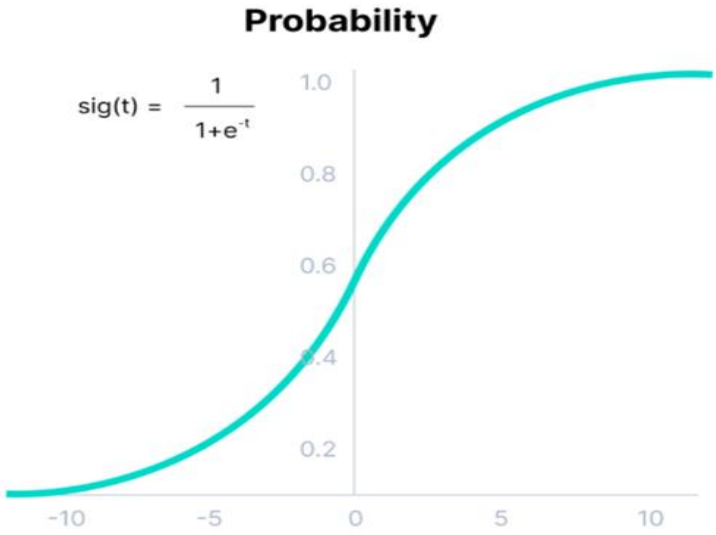
Kaynak:(Çolmabey, F.2021)

$$f(x_i)=\frac{e^{xi}}{\sum_{k=1}^{m} e^{x_k}} \tag{4.1}$$

Softmax Aktivasyon Fonksiyonu:

Softmax aktivasyon fonksiyonu, birden fazla sınıfın bulunduğu çoklu sınıflandırma problemlerinde yaygın olarak kullanılan bir fonksiyondur. Bu tür sınıflandırmalarda, sinir ağının çıktıları her sınıf için negatif veya pozitif değerler alabilir ve bu çıktılar toplamda her zaman 1'e eşit olmayabilir. Bu durumda, sınıflandırma sonuçlarını anlamak ve yorumlamak zorlaşır. İşte bu noktada Softmax fonksiyonu devreye girer ve her bir sınıf için olasılık değeri hesaplayarak bu sorunu çözer.

Softmax fonksiyonu, giriş olarak aldığı her sınıfın çıkış değerini eksponansiyel bir dönüşüme tabi tutar ve ardından bu değerleri toplam eksponansiyel değere bölerek normalize eder. Böylece, her sınıf için olasılık değeri hesaplanır ve bu olasılık değerleri 0 ile 1 arasında normalize edilir. Sonuç olarak, tüm sınıfların olasılık değerlerinin toplamı 1 olur. Bu, modelin her bir sınıf için ne kadar güvenle tahmin yaptığını anlamamıza olanak tanır.



Şekil 4.2: Softmax Aktivasyonu

Kaynak: (Çolmabey, F.2021)

4.2.2 Evrişimsel sinir ağları (CNN)

Konvolüsyonel Sinir Ağları (CNN'ler), derin öğrenme modelleri arasında en kritik türlerden biridir ve katmanlı yapısıyla dikkat çeker. CNN'ler, özellikle görüntü tanıma ve sınıflandırma gibi görevlerde olağanüstü performans göstermektedir. Bu ağlar, giriş katmanı, konvolüsyon katmanları, havuzlama (pooling) katmanları ve çıkış katmanı gibi birkaç temel bileşenden oluşur.

Giriş Katmanı: İlk aşamada, modelin işlemek üzere aldığı ham veriler (örneğin, görüntüler) bu katmanda depolanır ve sonraki katmanlara iletilir. Giriş katmanı, modelin başlangıç noktasıdır ve verinin sinir ağına ilk kez sunulduğu yerdir.

Konvolüsyon Katmanları: İkinci aşamada, konvolüsyon katmanları işleme girer. Bu katmanlar, belirli filtreler kullanarak giriş verisindeki özellikleri çıkarır. Filtreler, küçük nöron grupları olarak çalışır ve verinin her bölgesine uygulanarak özellik haritaları (feature maps) oluşturur. Bu işlem, verideki belirli desenleri veya öznitelikleri tanımlamaya yarar. Filtrelerin sayısı ve boyutu, modelin karmaşıklığına ve işlenecek verinin türüne göre ayarlanır.

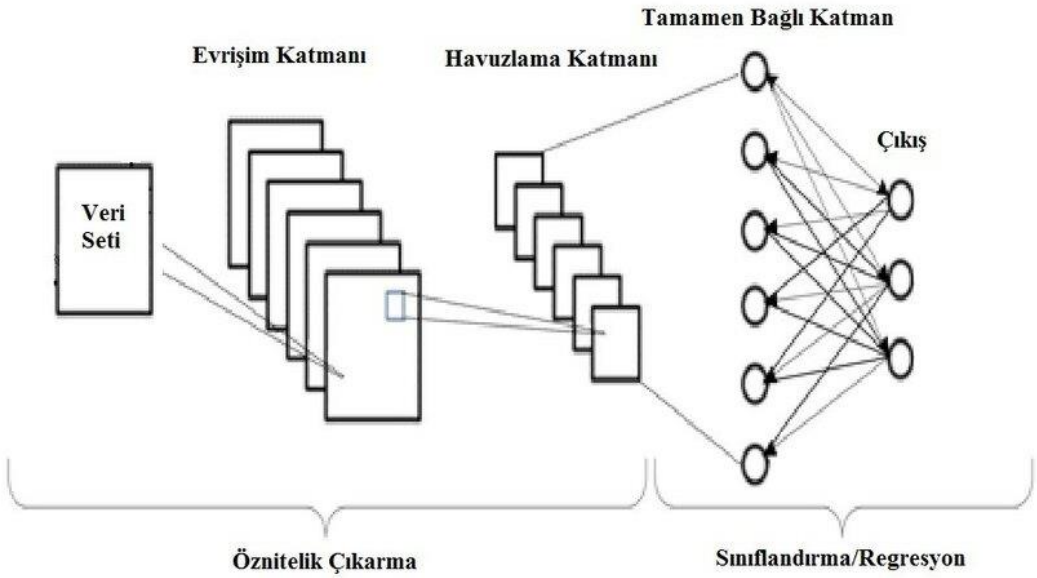
Havuzlama (Pooling) Katmanları: Konvolüsyon katmanlarının çıktısı, havuzlama katmanlarına iletilir ve burada veriler daha kompakt bir hale getirilir. En yaygın havuzlama türleri, maksimum havuzlama (max pooling) ve ortalama havuzlama (average pooling) olup, veri boyutunu küçültür ve hesaplama yükünü

azaltır. Havuzlama katmanları, modelin genel performansını artırırken, konvolüsyon katmanları tarafından öğrenilen önemli özelliklerin korunmasını sağlar.

Tam Bağlantılı (Fully Connected) Katmanlar: Havuzlama katmanlarından gelen veriler, tam bağlantılı katmanlara iletilir. Bu katmanlar, verinin sınıflandırılması veya regresyon analizi gibi son işlemleri gerçekleştiren klasik yapay sinir ağı katmanlarıdır. Bu katmanlarda, her nöron önceki katmandaki tüm nöronlarla bağlantılıdır.

Çıkış Katmanı: Son olarak, çıkış katmanı veriyi alır ve nihai sonucu üretir. Örneğin, bir görüntü tanıma modelinde çıkış katmanı, verinin hangi sınıfa ait olduğunu belirler. Bu sınıflandırma işlemi, genellikle softmax gibi aktivasyon fonksiyonları kullanılarak yapılır.

Filtreler ve Algoritmalar: CNN'lerin temel bileşenlerinden biri, sistem tasarımında kullanılan filtrelerdir. Bu filtreler, modelin verideki belirli özellikleri tanımasını ve öğrenmesini sağlar. Her filtre, verinin belirli bir bölgesinde kaydırılarak (stride) uygulanır ve bu işlem tüm veri üzerinde tekrar edilerek özellik haritaları oluşturulur.

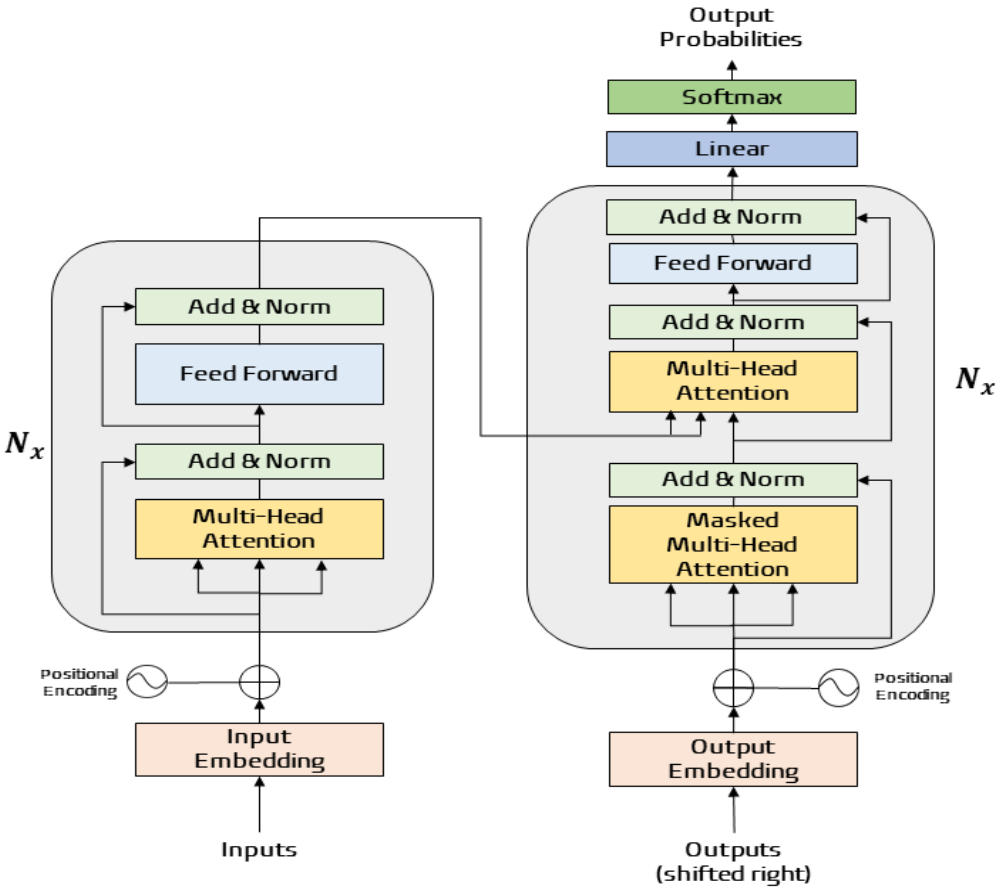


Şekil 4.3: Evrişimli Sinir Ağı Mimarisi

Kaynak: (Hamit, E.2023)

4.2.3 Transformer

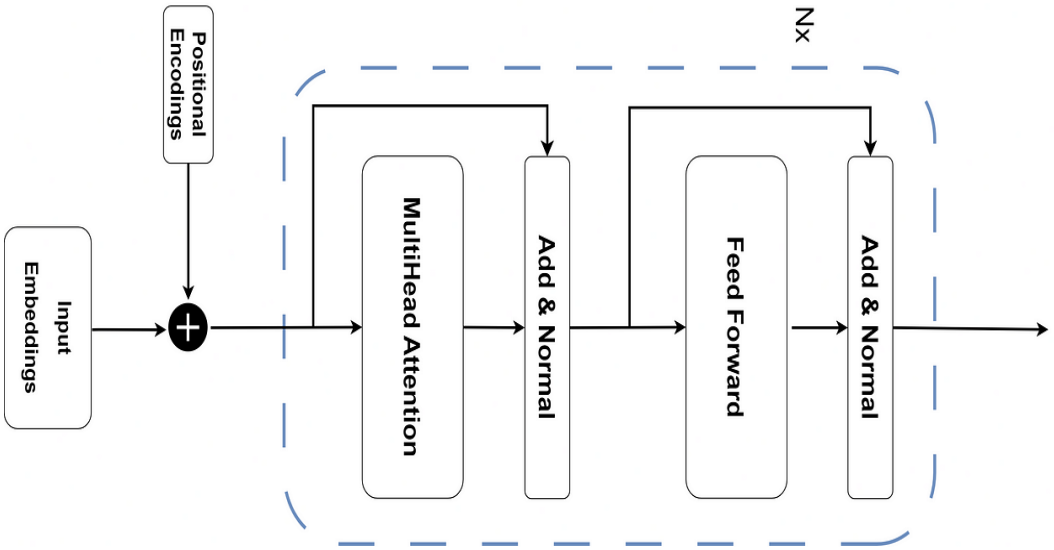
Transformer modeli, Vaswani ve arkadaşları tarafından 2017 yılında "Attention is All You Need" başlıklı makalede tanıtılmıştır. Bu model, sıralı verilerle (örneğin metin) çalışırken daha önce yaygın olarak kullanılan RNN (Recurrent Neural Network) ve LSTM (Long Short-Term Memory) gibi modellere kıyasla birçok avantaj sunar.



Şekil 4.4: Transformer Mimarisi

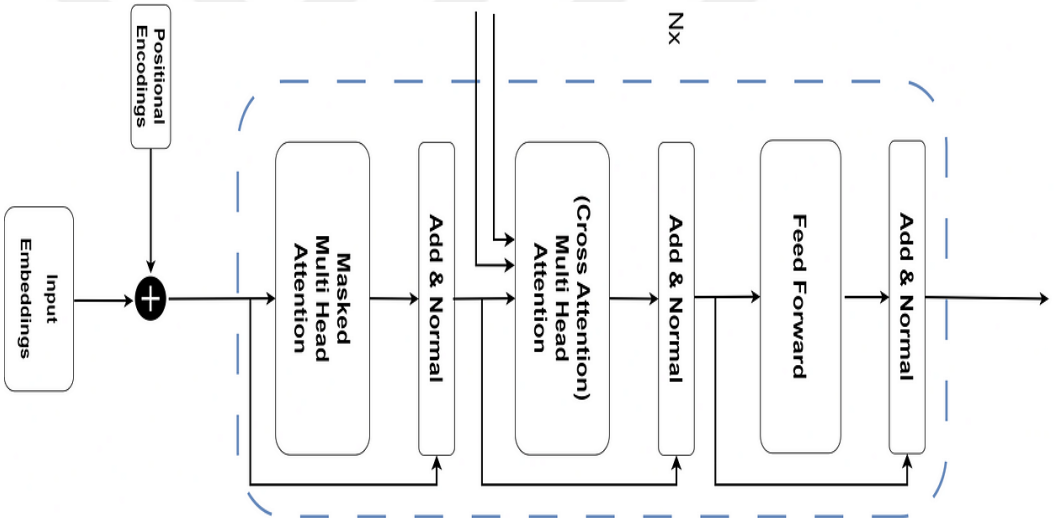
Kaynak: (Codegenius, 2021)

Transformer, encoder-decoder mimarisi üzerine kuruludur. Encoder kısmı, girdi verisini kodlar ve decoder kısmı bu kodlamayı çözerek hedef çıktıyı üretir. Encoder ve decoder her biri çok katmanlı yapılardır ve her katman, multi-head self-attention mekanizması ve konumlandırma (positional) eklenmiş feed-forward sinir ağı bileşenlerinden oluşur.



Şekil 4.5: Encoder Diyagramı

Kaynak: (Konak, İ.2023)



Şekil 4.6: Decoder Diyagramı

Kaynak: (Konak, İ.2023)

Attention mekanizması, modelin belirli bir anda girdinin hangi kısmına odaklanması gerektiğini belirler. Self-attention, aynı dizinin farklı pozisyonları arasındaki ilişkileri yakalarken, multi-head attention, farklı dikkat mekanizmalarının çıktılarının birleştirilmesiyle daha zengin temsil sağlar. Transformer modeli, girdilerin konum bilgisini tanıtmak için positional encoding adı verilen bir teknik kullanır. Bu teknik, her pozisyon için sinüs ve kosinüs fonksiyonları ile elde edilen değerleri girdi vektörlerine ekler.

Transformer modelleri, paralel işlemeyi destekledikleri için eğitimi hızlandırır ve uzun bağlamları etkili bir şekilde yakalayabilirler. Ayrıca, yalnızca

metin verileriyle sınırlı kalmayıp, görüntü işleme, ses tanıma ve daha birçok alanda kullanılabilirler. Doğal dil işleme (NLP) görevlerinde makine çevirisi, metin özetleme, duygu analizi gibi birçok alanda başarıyla uygulanmıştır. Vision Transformer (ViT) gibi modellerle görüntü sınıflandırma ve segmentasyon yapılabilir. Ayrıca, ses verilerini işleyerek konuşma tanıma ve sentezlemede de kullanılabilirler.

Transformer modelleri, YTA'da anomali tespiti için de oldukça uygundur. SDN'den gelen ağ trafiği verileri, uygun şekilde ön işlenir ve modele uygun hale getirilir. Eğitim sırasında model, normal ve anormal trafik örneklerini öğrenir. Eğitilmiş model, test verisi üzerinde değerlendirilir ve anomali tespit oranı, doğruluk, hatırlama (recall) gibi metriklerle performansı ölçülür. Gerçek zamanlı verileri analiz ederek anomali tespiti yapabilen Transformer modeli, YTA'ın güvenliğini artırır ve potansiyel saldırıları önceden tespit edebilir.

Transformer modeli, sunduğu esneklik ve güçlü dikkat mekanizmaları sayesinde, YTA gibi karmaşık sistemlerde anomali tespiti için ideal bir çözümdür. Bu modelin etkin kullanımı, ağ güvenliğini önemli ölçüde artırabilir.

4.3 Literatür Taraması

Hilal ve arkadaşları (2023), akıllı şebekelerin yönetim ve güvenlik alanındaki eksiklikleri gidermek amacıyla bir araştırma gerçekleştirmiştir. Bu çalışmanın temel amacı, Yazılım Tanımlı Ağ (SDN) ve Nesnelerin İnterneti (IoT) teknolojilerini birleştirerek akıllı şebeke altyapısının güvenliğini artırmaktır. SDN, merkezi kontrol sağlayarak şebeke performansını artırırken, IoT, veri toplama ve analiz etme yetenekleriyle şebekenin etkin yönetimine katkıda bulunur. Bu iki teknolojinin entegrasyonu ile önerilen akıllı şebeke mimarisi, Mininet simülasyon ortamında başarıyla modellenmiştir. Araştırma, akıllı şebekelerin güvenliğinde kullanılabilecek makine öğrenme algoritmalarını özel bir veri seti üzerinde değerlendirerek güvenlik açısından olumlu sonuçlar elde etmiştir.

Rounak Gupta ve ekibinin (2022) yaptığı bu çalışma, Yazılım Tanımlı Ağlar (SDN) için anormal ağ sızma tespit sistemleri geliştirmeye odaklanmaktadır. Araştırmada, derin sinir ağı (DNN) yöntemi kullanılmış ve dropout tekniği sayesinde rastgele alt kümelerle daha sağlam özellikler öğrenme olanağı sağlanmıştır. NSL-

KDD veri seti kullanılarak gerçekleştirilen bu çalışmada, DNN modeli için k-katlamalı çapraz doğrulama ve holdout gibi tahmin yöntemleri uygulanmıştır. Deneysel sonuçlar, önerilen modelin gerçek zamanlı uygulamalarda daha hesaplama açısından verimli olduğunu ve toplam tespit oranının %92.65'e kadar artırılabileceğini göstermektedir.

Mesut Toğaçar'ın (2022) yaptığı çalışmada, yazılım tanımlı ağ verilerindeki saldırıları tespit etmeye yönelik yapay zeka tabanlı hibrit bir yaklaşım incelenmiştir. Araştırmada kullanılan veri seti, normal trafik yanı sıra kaba kuvvet saldırıları, hizmet engelleme (DoS), cross-site scripting, SQL enjeksiyonu ve scripting enjeksiyonu gibi çeşitli ağ saldırılarını içermektedir. Etkili özellikleri seçme sürecinde başarı elde etmek amacıyla Archimedes optimizasyon algoritması kullanılmıştır. Bu özellikler seçildikten sonra, veri seti transformer tabanlı konvolüsyonel sinir ağı (CNN) modeli ile eğitilmiştir ve normal veya saldırı trafiğini tespit etmek için softmax yöntemi uygulanmıştır. Deneysel analizler sonucunda elde edilen genel doğruluk oranı %98,94 olarak kaydedilmiştir.

W. Krzemień ve ekibi (2021) tarafından gerçekleştirilen araştırma, programlanabilir SDN bilgisayar ağlarında saldırı tespitine odaklanmış ve bu amaçla altı farklı veri seti üzerinde RandomForest, XGBoost algoritmaları ve sinir ağları kullanılmıştır. Bu çalışmada çapraz doğrulama yöntemi uygulanmış ve test verilerinde saldırı tespit doğruluğunun %88.49 ile %99.28 arasında değiştiği bulunmuştur. Çalışmanın dikkat çeken bir yeniliği, algoritmanın giriş setinden bağımsız olarak etkili bir şekilde çalışabilmesidir. Programlanabilir SDN ağlarında anormallik tespiti üzerine yapılan başka bir çalışmada ise XGBoost sınıflandırıcı algoritması kullanılmış ve yüksek doğruluk seviyeleri elde edilmiştir. Her iki çalışma da, programlanabilir SDN ağlarında saldırı tespiti için algoritmaların ve parametrelerin dikkatli bir şekilde seçilmesinin ve uygulanmasının önemine vurgu yapmaktadır.

Jun Bai ve ekibi (2020) tarafından gerçekleştirilen araştırma, OpenFlow tabanlı Yazılım Tanımlı Ağlar (SDN) ortamlarında derin öğrenme tekniklerinin uygulanarak anormallik tespiti yapılmasını incelemektedir. Bu çalışmanın ana hedefi, YTA altyapısında OpenFlow protokolü ile elde edilen çeşitli metrikler kullanılarak derin öğrenme yöntemleriyle ağlardaki anormallikleri belirlemektir. Araştırma kapsamında, dört farklı derin öğrenme modeli kullanılarak çok değişkenli zaman

serileri sınıflandırılmış ve bu modellerin performansları titizlikle değerlendirilmiştir. Sonuçlar, derin öğrenme tekniklerinin anormallik tespitinde ortalama %83.8 doğruluk oranına ulaştığını göstermektedir. Bu bulgular, OpenFlow tabanlı YTA ortamlarında derin öğrenme yöntemlerinin etkinliğine dair önemli kanıtlar sunmaktadır.

Xinshuo Bai ve ekibinin 2020 yılında gerçekleştirdiği bu araştırma, OpenFlow tabanlı bir YTA ortamında çeşitli güvenlik saldırılarından kaynaklanan anormallikleri tespit etmek için yenilikçi bir yöntem sunmaktadır. Bu yaklaşım, YTA mimarisinde değişiklik yapmadan, mevcut yöntemlere göre anormallik tespitine yeni bir bakış açısı getirmektedir. Araştırma, OpenFlow anahtar verilerinden elde edilen çoklu metrikleri kullanarak YTA'larda anormallik tespiti yapmak için bir derin öğrenme yöntemi önermektedir. NSL_KDD veri setinin kullanıldığı bu çalışmada, dört farklı derin öğrenme modeli ile yapılan değerlendirmelerde ortalama doğruluk oranı %83,8 olarak bulunmuştur.

Yang Qin ve ekibinin (2019) gerçekleştirdiği bu araştırma, Yazılım Tanımlı Ağlar (SDN) üzerindeki güvenlik sorunlarını ele almayı ve bu sorunlara çözümler geliştirmeyi amaçlamaktadır. Çalışmanın ana hedefi, insan müdahalesini en aza indirerek veri paketlerinin özelliklerini otomatik olarak çıkaran bir model geliştirmektir. Bu model, derin öğrenme tekniklerinden CNN (Evrişimli Sinir Ağı) ve RNN (Tekrarlayan Sinir Ağı) kullanılarak oluşturulmuştur. Araştırmada, CTU-13, CSIC ve araştırmacılar tarafından özel olarak oluşturulan sim_data olmak üzere üç farklı veri seti kullanılmıştır. CTU-13 veri setinde %99.86, sim_data veri setinde ise %99.84 gibi yüksek doğruluk oranları elde edilmiştir. Modelin uygulanmasında TensorFlow Kütüphanesi kullanılmıştır.

Abdullah ve diğer araştırmacılar, 2019 yılında Yazılım Tanımlı Ağ tabanlı bir Sızma Tespit Sistemi (IDS) üzerine bir çalışma gerçekleştirmiştir. YTA, kontrol düzlemi ile yönlendirme düzlemini fiziksel olarak ayırarak, ağ cihazları üzerinde merkezi kontrol imkanı sunar. Bu çalışmanın temel amacı, bu teknolojiyi kullanarak hafif bir akış tabanlı IDS'in oluşturulabilirliğini araştırmaktır. Araştırma sonuçları, önerilen IDS'in makine öğrenimi ve derin öğrenme algoritmaları ile başarıyla değerlendirildiğini ortaya koymaktadır. Belirli algoritmalarla elde edilen sonuçlar, tespit oranının %90'ın üzerinde olduğunu göstermektedir.

Soumaine Bouba Mahamat'ın (2019) gerçekleştirdiği bu çalışma, Yazılım Tanımlı Ağlar (SDN) üzerinde makine öğrenme algoritmaları kullanarak anomali tespiti yapmayı amaçlamıştır. Araştırmada, akışları analiz ederek anormal durumları belirlemek için karar ağacı makine öğrenme algoritması uygulanmıştır. Araştırmacılar, mevcut sistemlerdeki eksiklikleri dikkate alarak kendi algoritmalarını geliştirmiş ve özgün veri setleri kullanmışlardır. Bu yöntem, YTA ortamında ping seli saldırılarını başarılı bir şekilde tespit etmiştir. Sonuçlar, sistemin etkili çalıştığını göstermiştir. Ancak, POX'un yalnızca Python 2.7'yi desteklemesi ve OpenFlow'un daha yeni sürümlerini kapsamaması gibi bazı sınırlamalar belirtilmiştir.

Ahmed Dawoud ve ekibi (2018) tarafından yapılan bu çalışma, özellikle derin öğrenmedeki en son yeniliklere odaklanarak ağ anormalliklerinin tespitini incelemektedir. Araştırmada, gözetimsiz derin öğrenme algoritmaları kullanılarak bir sızma tespiti çerçevesi önerilmektedir. Derin öğrenme algoritmalarının, özellikle de derin otomatik kodlayıcıların, ağ trafiği kayıtlarının yeniden yapılandırma hatalarını hesaplamak için nasıl kullanılabileceği ayrıntılı bir şekilde açıklanmıştır. Bu hataların K-means kümeleme algoritması ile analiz edilmesi sonucu elde edilen bulgular, makul miktarda eğitim verisiyle sağlam bir tahmin performansı sergilemiştir. Araştırma, bilgisayar ağlarındaki güvenlik zorluklarına derin öğrenme tabanlı çözümler sunmanın potansiyelini vurgulamakta ve bu konuda önemli bir katkı sağlamaktadır. Ayrıca, gözetimsiz derin öğrenme yaklaşımının YTA güvenliğini artırmada etkili olabileceği gösterilmiştir.

Goyal, Abhilash ve ekibinin (2018) araştırması, Yazılım Tanımlı Ağlar (SDN) üzerindeki Sızma Tespiti ve Önleme konusunu ele almaktadır. Çalışmanın odak noktası, açık kaynaklı Opendaylight modülü geliştirmek üzerinedir. Araştırma, eski nesil ağlarda sızma tespiti ve önleme için çeşitli çözümler bulunmasına rağmen, YTA bağlamında bu tür çözümlerin sınırlı olduğunu belirtmektedir. Bu makalenin amacı, YTA alanındaki sızma tespiti ve önleme sorunlarına yönelik bir çözüm sunmaktır. Geliştirilen çözüm, herhangi bir eğitim veri setiyle çalışabilen genel bir sınıflandırma sistemi sunar. Denetleyici, eğitim ve test süreçleri tamamlandığında, paket davranışlarını tahmin etmeye hazır hale gelir. Proje başlangıçta sadece Opendaylight YTA denetleyicisi için tasarlanmıştır ve geliştirilen OSGI paketleri bu denetleyiciyle uyumlu şekilde çalışmaktadır. Gelecekte, birden fazla denetleyici için kullanılabilecek bağımsız bir denetleyici çözümü oluşturmayı hedeflemektedirler.

Ayrıca, tahminlerin doğruluğunu ve verimliliğini artırmak amacıyla gelecekteki akıllı öğrenme algoritmalarının entegrasyonunu planlamaktadırlar.

Quamar Niyaz ve ekibi (2016) tarafından gerçekleştirilen başka bir araştırma, YTA ortamında çoklu vektörlü DDoS saldırılarını tespit etmek amacıyla Sparse Autoencoder (SAE) adı verilen bir derin öğrenme modelini kullanmıştır. Bu model, farklı koşullarda toplanan trafik veri kümelerine uygulanmış ve elde edilen sonuçlara göre, önerilen model bireysel DDoS saldırılarını %95,65 doğruluk oranıyla tespit etmiş ve normal ile saldırgan trafiği %99,82 doğruluk oranıyla sınıflandırmayı başarmıştır. Diğer çalışmalara kıyasla, bu sistemin yanlış pozitif oranı oldukça düşüktür. Araştırmacılar, gelecekteki çalışmalarında, denetleyicinin performansını artırmayı ve çeşitli türlerdeki ağ saldırılarını tespit edebilen bir Ağ İçi Tehdit Algılama Sistemi (NIDS) geliştirmeyi planlamaktadır.

Kaur ve ekibi (2015), Yazılım Tanımlı Ağ (YTA) mimarisi için OpenFlow tabanlı, programlanabilir bir güvenlik duvarı uygulaması önermişlerdir. Bu çalışmada, YTA tabanlı ağların güvenliğini sağlamak amacıyla güçlü ve esnek güvenlik duvarı çözümleri oluşturmayı hedeflemişlerdir. Güvenlik duvarı fonksiyonlarının birçoğunun özel donanım gerekmeden, yazılım aracılığıyla gerçekleştirilebileceğini vurgulamışlardır. Araştırmalarında, açık kaynaklı Python ile yazılmış POX denetleyiciyi kullanmışlar; sanallaştırma için VMPlayer'ı ve ağ topolojilerini oluşturmak için Mininet emülatörünü tercih etmişlerdir.

Ni Gao ve ekibinin (2014) gerçekleştirdiği bu çalışma, derin inanç ağları (DBN'ler) kullanarak oluşturulan bir sızma tespit sınıflandırıcısına odaklanmaktadır. Bu derin model, geriye yayılım ağı ile birlikte kullanılan, çok katmanlı denetimsiz öğrenme ağı olan RBM'nin bir kombinasyonunu içermektedir. Deneyler, KDD CUP 1999 veri kümesi üzerinde yürütülmüştür. Modelin tespit oranı %91,7 olarak tespit edilmiştir. Bu oran, ağın belirli bir tehdidi ne kadar doğru tanıdığını gösterir ve sızma tespiti konusundaki etkinliğini değerlendirmek açısından önemlidir.

5. VERİ SETİ VE MODEL EĞİTİMİ

Bu çalışmada kullanılan CSE-CIC-IDS2018 isimli veri seti (Datasets | Research | Canadian Institute for Cybersecurity | UNB), New Brunswick Üniversitesi'ndeki Kanada Siber Güvenlik Enstitüsü'nün web sitesinden alınmıştır. Veri seti, ağ trafiği davranışını yansıtan 1.188.333 satır ve 78 özellik sütunundan oluşmaktadır. Bu özellikler, ağ akışlarının çeşitli özelliklerini ayrıntılı olarak tanımlar. Ek olarak, veri seti, ağ trafiğini beş farklı sınıfa ayıran bir sınıf etiketi sütunu içermektedir. Bu sınıflar, normal trafik ve çeşitli anomali sınıflarını kapsamaktadır. Sınıf etiketi sütunu, ağ trafiğinin türünü tanımlar ve normal ile anormal trafik türlerini kapsar. Anormal trafik türleri arasında Dağıtılmış Hizmet Reddi (DDoS) saldırıları, izinsiz girişler ve kötü niyetli trafik gibi çeşitli saldırı türleri bulunur. Bu sınıflandırma, derin öğrenme modellerinin eğitim ve test aşamaları için büyük önem taşır. Veri setinin geniş kapsamı ve çeşitliliği, YTA'larda anomali tespiti amacıyla derin öğrenme modellerinin eğitimi ve değerlendirilmesi için oldukça elverişlidir. Büyük ve çeşitli bir veri seti, modellerin farklı anomali türlerini daha etkili bir şekilde tespit edebilmesi ve genelleme yeteneklerinin artırılması açısından önemlidir. Bu nedenle, veri seti üzerinde yapılan ön işleme ve analiz çalışmaları, modellerin performansını doğrudan etkiler.

Veri seti üzerinde gerçekleştirilen ilk analizler, veri setinin yapısını ve içeriğini anlamak amacıyla yapılmıştır. Bu analizler, veri setinin boyutlarının gözden geçirilmesi, her sütunun veri türlerinin ve eksik değerlerin kontrol edilmesi gibi adımları kapsamaktadır. Bu ön işleme adımları, veri setinin ileri düzey analizler ve modelleme için uygun hale getirilmesini sağlar.

Sonuç olarak, UNB'den temin edilen bu kapsamlı veri seti, YTA'larda anomali tespiti için derin öğrenme modellerinin geliştirilmesi ve değerlendirilmesi için mükemmel bir kaynak sunar. Verilerin çeşitliliği ve ayrıntılı yapısı, modellerin eğitim sürecinde daha iyi performans göstermesine ve gerçek dünya uygulamalarında daha güvenilir sonuçlar üretmesine olanak tanır.

5.1 Veri Ön İşleme

Veri ön işleme, derin öğrenme modellerini eğitmek için veri setini hazırlamada hayati bir adımdır. Bu adım, modellerin performansını doğrudan etkileyebilir ve modelin başarısını en üst düzeye çıkarmak için dikkatlice gerçekleştirilmelidir. Veri ön işleme sürecindeki adımlar aşağıdaki gibidir

Veri Setinin Yüklenmesi: Veri seti, analiz ve manipülasyonu kolaylaştırmak amacıyla pandas DataFrame'e yüklendi. Bu adım, verilerin yapılandırılmış bir formatta olmasını sağlayarak inceleme ve işlemeyi daha kolay hale getirdi. Pandas DataFrame, verilerin satır ve sütunlar halinde düzenlenmesine imkan tanır, böylece veriler kolayca işlenebilir.

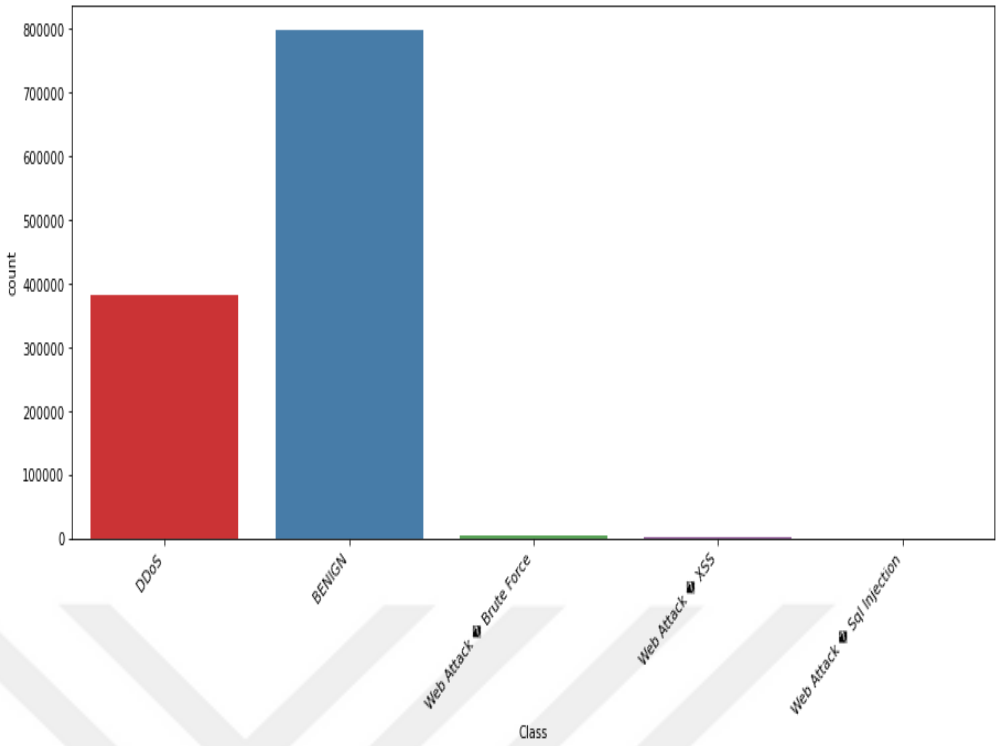
Veri setinin yapısını ve içeriğini anlamak için aşağıdaki analizler gerçekleştirildi:

Veri Setinin Boyutu: Veri setinin boyutları incelendi. Bu, veri setinde kaç örnek (satır) ve kaç özellik (sütun) bulunduğunu anlamamıza yardımcı oldu. Bu bilgi, veri setinin kapsamını ve büyüklüğünü anlamak açısından önemlidir.

Veri Seti Bilgisi: Her sütunun veri tipi ve eksik değerlerin varlığı kontrol edildi. Pandas'ın info() fonksiyonu kullanılarak her sütunun veri tipi (örneğin, tam sayı, kayan nokta, nesne) ve eksik değerlerin sayısı belirlendi. Bu adım, veri temizleme sürecinin gerekliliğini ve hangi tür veri dönüşümlerinin yapılması gerektiğini belirlemek için kritiktir.

Sınıf Dağılımı: Veri setindeki benzersiz sınıflar ve bunların sayıları belirlenerek etiketlerin dağılımı analiz edildi. Bu bilgi, sınıflar arasındaki dengesizliği tespit etmek ve gerekirse veri dengeleme stratejilerini (örneğin, yeniden örnekleme) uygulamak için kullanıldı.

Veri setindeki sınıfların dağılımını anlamak, anomali tespiti gibi görevler için oldukça önemlidir. Bu yüzden, sınıf dağılımını görselleştirmek amacıyla bir sayım grafiği oluşturuldu. Bu grafik, veri setindeki her sınıfın frekansını görsel olarak gösterir ve sınıflar arasındaki dengesizliği hızlı bir şekilde belirlememize olanak tanır.



Şekil 5.1: Çalışmada Kullanılan Çeşitli Veri Türlerinin Dağılımı

Kaynak: (Python, 2024).

5.1.1 Veri temizleme

Veri temizleme, veri setinin kalitesini artırmak ve analiz sürecinde oluşabilecek hataları en aza indirmek için hayati bir adımdır. Bu süreç, veri manipülasyonu ve modelleme aşamalarında tutarlılığı sağlamak için gerçekleştirilir. Veri temizleme sürecinde atılan önemli adımlar şunlardır:

Sütun Adlarının Düzenlenmesi: Sütun adlarının düzenlenmesi, veri işleme sırasında tutarlılığı sağlamak ve hatalardan kaçınmak için kritik bir adımdır. Bu aşamada, sütun adlarındaki boşluk karakterleri kaldırılarak, adların daha okunabilir ve tutarlı hale getirilmesi sağlanır. Böylece, sütun adlarının veri işleme sırasında doğru şekilde kullanılması ve olası yazım hatalarının önlenmesi mümkün olur.

Etiket Temizliği: Veri setindeki sınıf etiketlerinin temizlenmesi de önemli bir adımdır. Bu aşamada, sınıf etiketlerindeki özel karakterler çıkarılarak ve boşluklar alt çizgilerle değiştirilerek etiketler daha okunabilir ve tutarlı hale getirilir. Bu işlem, etiketlerin modelleme sürecinde doğru bir şekilde işlenmesine yardımcı olur ve veri analizinin doğruluğunu artırır.

5.2 Model Eğitimi

Veri seti, derin öğrenme modellerinin eğitiminde ve performanslarının ölçülmesinde önemli bir rol oynar. Bu süreçteki ilk adım, veri setinin eğitim ve test setlerine ayrılmasıdır. Bu bölümde, veri setinin nasıl hazırlandığını ve bu hazırlık adımlarının her birini detaylı olarak açıklayacağız.

Veri Setinin Bölünmesi

Veri seti, modelin daha önce görmediği veriler üzerindeki performansını değerlendirmek amacıyla eğitim ve test setlerine ayrılmıştır. Bu bölünme, veri setinin %80'inin eğitim seti ve %20'sinin test seti olacak şekilde gerçekleştirilmiştir. Eğitim seti, modelin öğrenme sürecinde kullanılırken, test seti modelin yeni veriler üzerindeki performansını değerlendirmek için kullanılır. Bu bölme işlemi, train_test_split fonksiyonu kullanılarak yapılmıştır.

5.2.1 ANN Modeli

Çizelge 5.1: Yapay Sinir Ağı Katmanları ve Parametreleri

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 78)	0
dense (Dense)	(None, 57)	4503
dropout (Dropout)	(None, 57)	0
dense_1 (Dense)	(None, 5)	290

Total params: 4,793
Trainable params: 4,793
Non-trainable params: 0

- `*tf.keras.layers.Flatten(input_shape=(78,))*`

Flatten katmanı, giriş verilerini iki boyutlu bir diziden (78,) tek boyutlu bir diziye dönüştürür. Yani, bu katman matris biçimindeki veriyi tek bir vektör haline getirir. Örneğin, (78,) boyutundaki bir matris 78 elemanlı bir vektöre dönüştürülür.

Amacı: Sinir ağları, özellikle yoğun katmanlar gibi belirli katmanlar, giriş verilerinin belirli bir formatta olmasını gerektirir. Yoğun katmanlar, her giriş elemanı

için ağırlıkları hesapladığından, düzleştirilmiş vektörlerle çalışmayı gerektirir. Bu gereksinimi karşılamak için Flatten katmanı, giriş verilerini düzleştirerek veri formatını yoğun katmanlar için uygun hale getirir. Bu sayede, veriler sonraki katmanlarda işlenebilir hale gelir.

- *`tf.keras.layers.Dense(57, activation='relu')`*

Bu katman, sinir ağlarının tam bağlantılı (yoğun) bir katmanıdır ve içinde 57 nöron barındırır. Her bir nöron, önceki katmandaki tüm nöronlarla tam bir bağlantıya sahiptir. Bu, her nöronun önceki katmandan gelen tüm girdileri alarak, kendi ağırlık ve önyargı değerleriyle işlemden geçirdiği anlamına gelir. Bu katman, girdileri daha yüksek boyutlu bir uzaya dönüştürerek verinin daha karmaşık özelliklerini keşfedebilir ve öğrenebilir.

Aktivasyon Fonksiyonu: Bu katmanda aktivasyon fonksiyonu olarak ReLU (Düzeltilmiş Doğrusal Birim) kullanılır. ReLU, modele doğrusal olmayan özellikler katarak daha karmaşık desenlerin öğrenilmesini mümkün kılar. Bu fonksiyon, negatif değerli girdiler için sıfır, pozitif değerli girdiler için ise girdinin kendisini çıktıya verir. Bu özellik, ReLU'yu hesaplama açısından son derece verimli hale getirir ve derin öğrenme ağlarında sıkça tercih edilmesini sağlar.

- *`tf.keras.layers.Dropout(0.2)`*

Bu katman, eğitim süresi boyunca her güncelleme aşamasında giriş birimlerinin %20'sini rastgele olarak sıfırlar. Bu işlem, eğitim sırasında belirli bir kısmı nöronların devre dışı bırakılması anlamına gelir. Yani, her eğitim döngüsünde, katmandaki bazı nöronlar geçici olarak kapatılır, bu nedenle bu nöronların çıktı değerleri sıfır olur ve ağın bu nöronlardan gelen bilgilere bağımlılığı azalır.

Dropout, sinir ağlarında aşırı öğrenmeyi (overfitting) önlemek için kullanılan bir düzenleme (regularization) tekniğidir. Aşırı öğrenme, modelin eğitim verisine çok fazla uyum sağlaması ve bu yüzden yeni, daha önce görmediği verilerle karşılaştığında düşük performans sergilemesi durumudur. Dropout, ağın belirli nöronlarını rastgele kapatarak modelin belirli ağırlıklara aşırı bağımlı hale gelmesini engeller. Bu teknik, modelin farklı nöronların işlevlerini öğrenmesini sağlar ve böylece modelin genelleme yeteneği artar. Sonuç olarak, model, eğitim verilerinde bulunmayan yeni verilerle karşılaştığında daha iyi performans gösterir.

❖ *`tf.keras.layers.Dense(5, activation='softmax')`*

Bu katman, 5 nörondan oluşan bir tam bağlı (yoğun) katmandır. Bu katman, softmax aktivasyon fonksiyonunu kullanır. Softmax fonksiyonu, her nöronun çıktısını 0 ile 1 arasında bir olasılık değeri olarak dönüştürür ve toplamlarının 1 olmasını sağlar. Bu, her bir sınıf için tahmin edilen olasılıkların toplamının 1 olmasını garanti eder ve modelin çıktısını bir olasılık dağılımı olarak yorumlamamıza olanak tanır. Softmax, çoklu sınıflandırma problemlerinde yaygın olarak kullanılır, çünkü her sınıfın olasılığını hesaplayarak modelin hangi sınıfa ait olduğunu belirlememizi sağlar.

Softmax aktivasyon fonksiyonu, modelin çıktısını çoklu sınıflandırma problemlerine uygun hale getirir. Her nöron belirli bir sınıfı temsil eder ve bu katmandaki her nöronun çıktısı, girdinin o sınıfa ait olma olasılığını gösterir. Bu şekilde, bir model girdisinin hangi sınıfa ait olduğunu belirleyebilir ve çıktıları olasılık olarak yorumlayabiliriz.

Fonksiyon: Adam (Adaptive Moment Estimation) algoritması, her parametre için öğrenme oranını adaptif olarak ayarlayarak eğitim sürecini iyileştirir. AdaGrad ve RMSProp algoritmalarının avantajlarını birleştirir. AdaGrad, sık güncellenen parametreler için öğrenme oranını düşürürken, RMSProp, nadir güncellenen parametreler için öğrenme oranını yüksek tutar. Adam, bu iki yöntemi birleştirerek, her parametre için farklı öğrenme oranları hesaplar ve bu oranları momentum adı verilen bir yöntemle günceller. Momentum, gradyanların hareketli ortalamalarını kullanarak daha hızlı ve kararlı bir öğrenme süreci sağlar.

Adam, Öğrenme oranını dinamik olarak ayarlayarak, modelin eğitim sürecinde daha hızlı ve daha stabil bir şekilde yakınsamasını sağlar. Bu, hem zaman kazandırır hem de modelin performansını artırır.

Kayıp Fonksiyonu: *sparse_categorical_crossentropy*

Sparse categorical crossentropy, özellikle çok sınıflı sınıflandırma problemleri için kullanılan bir kayıp fonksiyonudur. Bu kayıp fonksiyonu, etiketlerin tam sayı (integer) olarak verildiği durumlarda kullanılır. Her bir örneğin doğru sınıfının tam sayı olarak belirtildiği veri setlerinde, bu fonksiyon doğru sınıfın olasılığını maksimize etmeyi amaçlar. Sparse categorical crossentropy, modelin

tahmin ettiği olasılık dağılımı ile gerçek dağılım (one-hot kodlanmış vektör) arasındaki farkı hesaplar.

Bu kayıp fonksiyonu, modelin tahmin performansını ölçmede kullanılır. Modelin çıktısı, her sınıf için 0 ile 1 arasında bir olasılık değeri üretir. Bu fonksiyon, tahmin edilen olasılık dağılımını gerçek etiketlerle karşılaştırarak, modelin ne kadar iyi performans gösterdiğini değerlendirir. Modelin eğitimi sırasında, bu kayıp fonksiyonunun minimize edilmesi hedeflenir. Bu, modelin sınıfları doğru bir şekilde tahmin etme olasılığını artırır ve modelin genel performansını iyileştirir.

Parametreler:

- Flatten Katmanı: Flatten katmanı, çok boyutlu bir girdi verisini tek boyutlu bir diziye dönüştürür. Bu katman, özellikle konvolüsyonel sinir ağlarından (CNN) sonra kullanılmak üzere tasarlanmıştır. Çok boyutlu veriyi düzleştirerek, tamamen bağlı (dense) katmanlara uygun hale getirir.

Parametreler: Bu katmanda eğitilecek herhangi bir parametre bulunmamaktadır. Sadece veriyi yeniden şekillendirir.

- Birinci Yoğun (Dense) Katman: Birinci yoğun katman, 57 nörona sahiptir. Bu katman, her bir girdi özelliği için ağırlıklar ve bias terimleri hesaplar. Girdi özellikleri ve nöronlar arasındaki bağlantılar tam bağlantılıdır, yani her nöron, giriş katmanındaki her bir özelliğe bağlıdır.

Parametreler:

- Ağırlıklar: 78 giriş özelliği * 57 nöron = 4,446 ağırlık parametresi.
- Biaslar: Her bir nöron için bir bias terimi olduğundan 57 bias parametresi.
- Toplam: 4,446 + 57 = 4,503 parametre.

Hesaplama: Bu katman, her bir girdiyi ağırlıklarla çarpar ve bias ekleyerek çıktıyı hesaplar. Aktivasyon fonksiyonu genellikle bu çıktıyı işleyerek nöronun aktivasyonunu belirler.

Bir nöronun çıktısını hesaplamak için kullanılan formül:

$$z = \sum_{i=1}^{n} (x_i \cdot w_i) + b \tag{5.1}$$

❖ Dropout Katmanı: Dropout katmanı, eğitim sırasında nöronların belirli bir kısmını rastgele devre dışı bırakır. Bu işlem, aşırı öğrenmeyi (overfitting) önlemeye yardımcı olur ve modelin genelleme kabiliyetini artırır.

Parametreler: Dropout katmanında eğitilecek herhangi bir parametre bulunmamaktadır. Bu katman sadece eğitim sürecinde nöronları rastgele devre dışı bırakma oranını belirler.

Eğitim Süreci: Dropout oranı genellikle 0.2 ila 0.5 arasında seçilir. Örneğin, dropout oranı 0.5 ise, her eğitim adımında nöronların %50'si rastgele devre dışı bırakılır.

❖ Çıktı Yoğun (Dense) Katmanı: Çıktı yoğun katmanı, modelin son katmanıdır ve tahmin edilen çıktıları üretir. Bu katmanda 5 nöron bulunmaktadır, her biri modelin tahmin ettiği sınıflardan birini temsil eder. Bu katman genellikle softmax aktivasyon fonksiyonu kullanarak çıktıları olasılık dağılımı olarak verir.

Parametreler:

➢ Ağırlıklar: 57 özellik (bir önceki katmandan gelen) * 5 nöron = 285 ağırlık parametresi.

➢ Biaslar: Her bir nöron için bir bias terimi olduğundan 5 bias parametresi.

➢ Toplam: 285 + 5 = 290 parametre.

Hesaplama: Bu katman, bir önceki katmandan gelen özellikleri ağırlıklarla çarpar, bias ekler ve softmax aktivasyon fonksiyonu kullanarak her bir sınıf için olasılık değerleri hesaplar.

ANN Model özeti

● Giriş Verileri: Model, 78 boyutlu giriş verilerini kabul eder. Bu, modelin her bir örnek için 78 farklı özelliği veya değişkeni analiz ettiği anlamına gelir. Bu giriş verileri, modelin öğrenme sürecinde kullanacağı temel bilgilerdir.

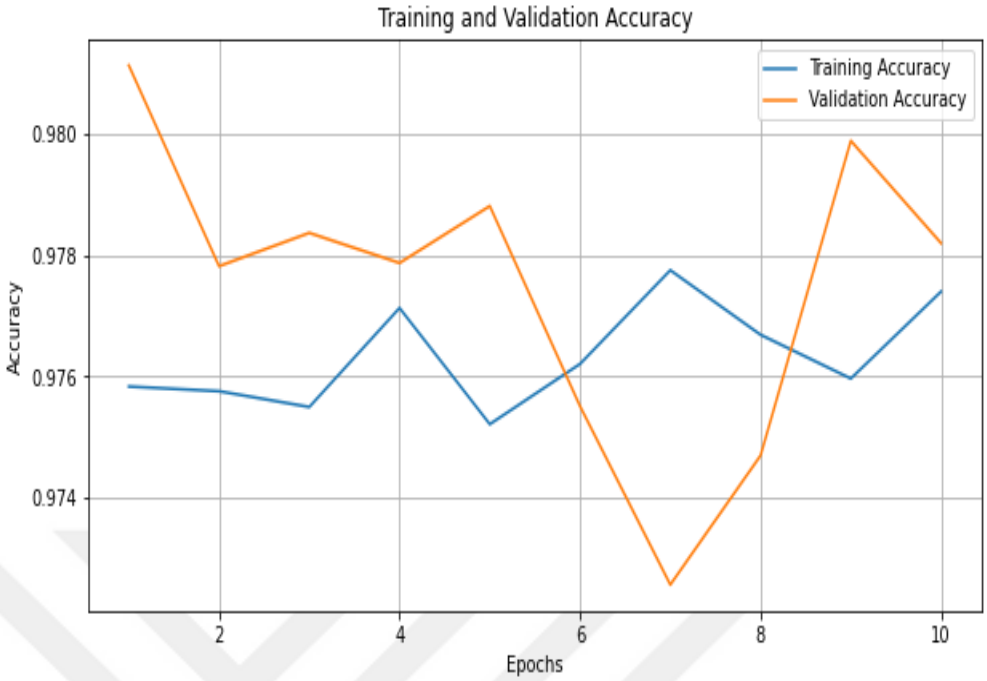
● Gizli Katman: Modelin bir tane gizli katmanı vardır ve bu katman 57 nörondan oluşur. Gizli katman, giriş verilerini daha soyut ve yüksek düzeyde temsil edecek şekilde öğrenir. Her bir nöron, önceki katmandaki tüm nöronlarla tam bağlantılıdır ve bu bağlantılar sayesinde verinin daha karmaşık özellikleri çıkarılır. Gizli katman, modelin veriyi anlaması ve sınıflandırması için gerekli olan temel desenleri öğrenir.

● Dropout Katmanı: Modeli düzenlemek ve aşırı öğrenmeyi (overfitting) önlemek için Dropout tekniği kullanılır. Dropout, eğitim süresi boyunca her güncelleme aşamasında giriş birimlerinin belirli bir yüzdesini (bu modelde %20) rastgele olarak sıfırlar. Bu, modelin belirli ağırlıklara aşırı bağımlı olmasını engeller ve daha genelleyici bir yapıya sahip olmasını sağlar. Dropout, modelin yeni ve görülmemiş veriler üzerinde daha iyi performans göstermesine yardımcı olur.

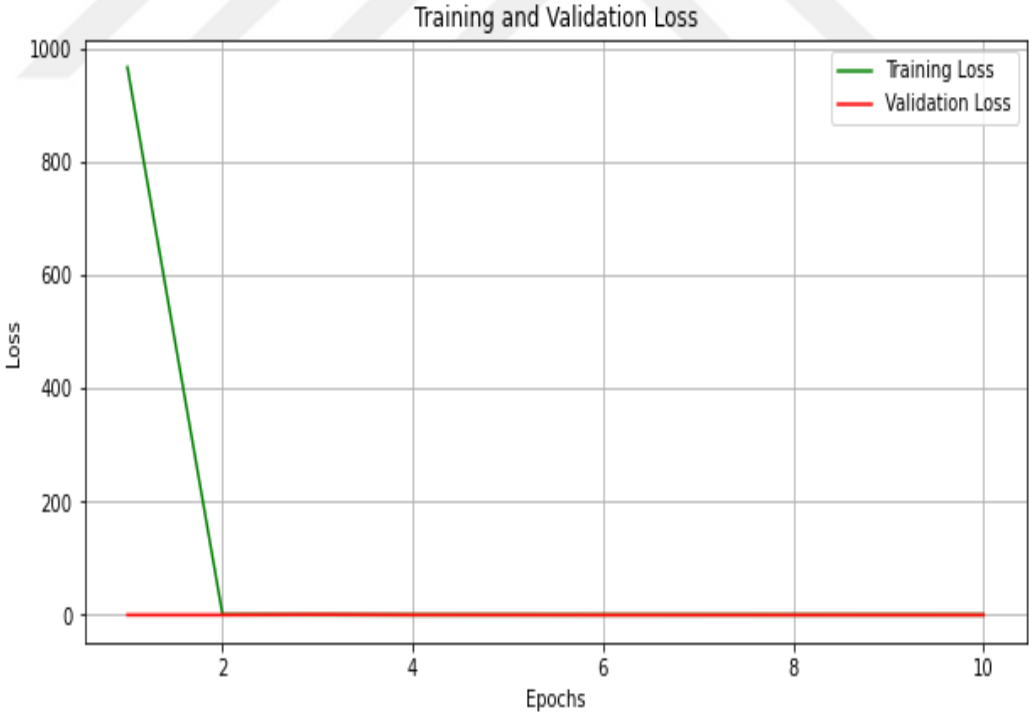
● Çıkış Katmanı: Son katman, softmax aktivasyon fonksiyonu ile donatılmış 5 nörona sahiptir. Softmax aktivasyon fonksiyonu, her bir sınıf için olasılık değerleri hesaplar ve toplamı 1 olan bir olasılık dağılımı oluşturur. Bu, modelin 5 farklı sınıf arasında bir tahmin yapmasını sağlar. Her bir nöron, belirli bir sınıfa karşılık gelir ve modelin tahmin ettiği sınıfın olasılığını temsil eder.

● Model Eğitimi ve Değerlendirmesi: Model, Adam optimizasyon algoritması kullanılarak eğitilir. Adam, adaptif öğrenme hızı sağlayan ve genellikle daha hızlı ve etkili öğrenme sağlayan bir optimizasyon algoritmasıdır. Modelin performansı, doğruluk metriği kullanılarak değerlendirilir. Doğruluk, modelin doğru sınıflandırma yaptığı örneklerin toplam örneklere oranını ifade eder ve modelin ne kadar iyi performans gösterdiğini ölçmek için kullanılır.

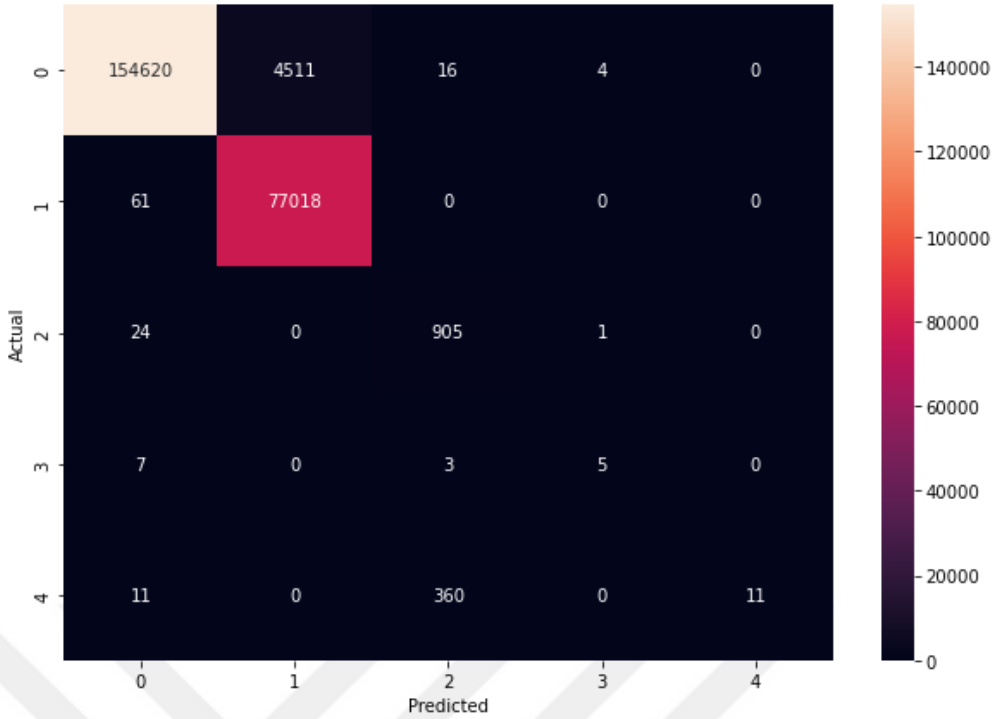
Model Sonuçları



Şekil 5.2: ANN Model Eğitim ve Doğrulama Grafiği



Şekil 5.3: ANN Model Eğitim ve Doğrulama Kaybı



Şekil 5.4: ANN Karışıklık Matrisi

Çizelge 5.2: ANN Sınıflandırma Raporu

	precision	recall	f1-score	support
BENIGN	1.00	0.97	0.99	159151
DDoS	0.94	1.00	0.97	77079
Web_Attack_Brute_Force	0.70	0.97	0.82	930
Web_Attack_Sql_Injection	0.50	0.33	0.40	15
Web_Attack_XSS	1.00	0.03	0.06	382
accuracy			0.98	237557
macro avg	0.83	0.66	0.65	237557
weighted avg	0.98	0.98	0.98	237557

5.2.2 CNN Modeli

Çizelge 5.3: Evrişimli Sinir Ağı Katmanları ve Parametreleri

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 78, 1)	0
conv1d (Conv1D)	(None, 76, 32)	128
conv1d_1 (Conv1D)	(None, 74, 64)	6208
max_pooling1d (MaxPooling1D)	(None, 37, 64)	0
dropout (Dropout)	(None, 37, 64)	0

Reshape katmanı:

- ` *model2.add(Reshape((78, 1), input_shape=(78,))* `

Reshape katmanı, modeldeki giriş verilerinin formatını değiştirmek için kullanılır. Bu örnekte, giriş verisi, (78,) şeklindeki bir tek boyutlu diziden (78, 1) şeklindeki bir iki boyutlu diziye dönüştürülür.

Mevcut Giriş Şekli (input_shape): (78,) - Bu ifade, giriş verisinin 78 elemanlı bir tek boyutlu dizi olduğunu gösterir.

Yeni Şekil (hedef şekil): (78, 1) - Giriş verisi, her biri 1 eleman içeren 78 alt diziye sahip iki boyutlu bir diziye dönüştürülür.

Reshape katmanı, özellikle Conv1D (1 boyutlu evrişimli) katmanlar gibi belirli tipteki katmanların beklediği giriş veri formatını sağlamak için kullanılır. Conv1D katmanları, giriş verisinin çok boyutlu bir yapıda olmasını gerektirir. Bu nedenle, modelin bu katmanları doğru bir şekilde işleyebilmesi için veri yeniden şekillendirme işlemi gereklidir.

İlk Evrişim Katmanı:

- *model2.add (Conv1D(32, kernel_size=3, activation='relu'))*

Bu katman, 32 filtre kullanarak giriş verilerinde 1D evrişim işlemi yapar. Her bir filtre, 3 giriş veri noktası üzerinde işlem yapar.

Aktivasyon: ReLU (Doğrusal Doğrultmalı Birim) aktivasyon fonksiyonu, modelin daha karmaşık ilişkileri öğrenmesine yardımcı olacak şekilde doğrusallığı olmayanlık ekler.

Amaç: Bu katman, giriş verilerindeki uzamsal hiyerarşileri öğrenir ve önemli özellikleri çıkarır. Evrişim işlemleri, verideki önemli desenleri ve yapıları tanımak için kullanılır.

İkinci Evrişim Katmanı:

- *model2.add (Conv1D(64, kernel_size=3, activation='relu'))*

Bu katman, ek olarak 64 filtre kullanarak 1D evrişim işlemi yapar. Her bir filtre, 3 giriş veri noktası üzerinde işlem yapar.

ReLU aktivasyon fonksiyonu, modelin daha karmaşık ilişkileri öğrenmesine yardımcı olacak şekilde doğrusallığı olmayanlık ekler.

Amaç: Bu ikinci evrişim katmanı, önceki katman tarafından öğrenilen uzamsal özellikleri daha derinleştirir ve geliştirir. Daha fazla filtre kullanarak, verideki daha ince ve karmaşık özellikleri tanımaya yardımcı olur.

Maximum Havuzlama Katmanı

- *model2.add (MaxPooling1D(pool_size=2))*

Maximum havuzlama katmanı, giriş özellik haritalarını belirli bir boyuta (bu örnekte 2) kadar küçülterek her havuzlama penceresi içindeki en yüksek değerleri seçer. Maximum havuzlama, özellik haritasının boyutunu azaltarak her bir havuzlama penceresindeki en büyük değeri seçer.

Amaç: Maximum havuzlama katmanı, modelin hesaplama zorluğunu azaltır ve özellik haritasının boyutunu küçültürken aşırı öğrenmeyi kontrol altına alır. Ayrıca, modelin daha belirgin ve önemli özellikleri öğrenmesine katkıda bulunur.

- *model2.add (Dropout(0.25))*

Dropout katmanı, eğitim sırasında giriş birimlerinin belirli bir yüzdesini rastgele olarak devre dışı bırakır. Bu örnekte, giriş birimlerinin %25'i rastgele olarak devre dışı bırakılmaktadır. Bu işlem, modelin belirli nöronlara fazla bağımlı olmasını engeller ve genel performansını artırır.

Amaç: Dropout katmanı, modelin aşırı öğrenme (overfitting) yapmasını önler. Aşırı öğrenme, modelin eğitim verisine çok iyi uyum sağlaması ancak test verisinde düşük performans göstermesi durumudur. Dropout, modelin daha genel özellikler öğrenmesine yardımcı olarak genelleme yeteneğini artırır.

- *`model2.add (Flatten())*

Düzleştirme katmanı, evrişim katmanlarından gelen çok boyutlu veriyi (örneğin (yükseklik, genişlik, filtre sayısı)) tek boyutlu bir vektöre dönüştürür. Bu süreç, evrişim katmanlarının çıktılarını yoğun katmanlar gibi tam bağlantılı katmanlar için uygun bir formata getirir.

Amaç: Evrişim katmanlarının çıktıları genellikle çok boyutludur ve tam bağlantılı (yoğun) katmanlar gibi doğrudan bağlantılı katmanlar tarafından işlenemez. Düzleştirme katmanı, bu çok boyutlu yapıyı tek boyutlu bir vektöre dönüştürerek verinin yoğun katmanlara aktarılmasını sağlar.

- *model2.add (Dense (128, activation='relu'))*

Yoğun Katman tamamen bağlı (fully connected) bir katmandır ve 128 nöron içerir. Bu katman, önceki konvolüsyon ve düzleştirme katmanlarından gelen özellikleri alır ve girdi verilerinin karmaşık temsillerini öğrenir. Her nöron, önceki katmanın çıktıları üzerinde ağırlıklı işlemler yaparak yeni özellikler oluşturur.

ReLU (Doğrusal Doğrultulmuş Birim) aktivasyon fonksiyonu, negatif değerler için çıktıyı sıfır yapar ve pozitif değerler için aynı değeri döndürerek modelin öğrenme sürecini hızlandırır.

Amaç: Bu katman, modelin veri kümesindeki derin ilişkileri ve özellikleri tanımasını sağlar, böylece daha karmaşık desenleri tespit edip öğrenebilir.

- *`model2.add (Dropout(0.5))*

Dropout katmanı eğitim sırasında giriş birimlerinin %50'sini rastgele devre dışı bırakarak ağın genel yapısını düzenler.

Amaç: Dropout katmanı, aşırı öğrenmeyi (overfitting) önlemek için kullanılır. Eğitim sırasında birimleri belirli bir yüzdeyle rastgele devre dışı bırakmak, modelin farklı özellikleri daha dengeli bir şekilde öğrenmesine yardımcı olur ve genelleme yeteneğini artırır.

❖ *`model2.add (Dense(5, activation='softmax'))`*

5 nöron içeren tamamen bağlı (fully connected) bir çıkış katmanıdır. Bu katman, modelin nihai tahminlerini üretir ve genellikle sınıflandırma problemlerinde kullanılır. Her nöron, giriş özelliklerini kullanarak belirli bir sınıf için olasılık değerleri üretir. Softmax aktivasyon fonksiyonu, çıktı değerlerini sınıflar arasında bir olasılık dağılımına dönüştürür. Bu şekilde model, her sınıf için olasılık değerleri üreterek en olası sınıfı belirler.

Amaç: Softmax aktivasyonu sayesinde, çıkış katmanı modelin 5 farklı sınıf arasında doğru bir şekilde sınıflandırma yapmasını sağlar ve sonuçları bir olasılık dağılımı olarak sunar.

Optimizasyon: Adam

Adam (Adaptive Moment Estimation), öğrenme oranını uyarlamalı olarak ayarlayan bir optimizasyon algoritmasıdır. Bu algoritma, her parametre için öğrenme oranını ayrı ayrı ayarlayarak eğitim sürecini hızlandırır ve daha verimli hale getirir. Adam, momentum ve RMSProp gibi iki önemli optimizasyon tekniğini birleştirir. Bu algoritma, her bir ağırlık parametresi için hareketli ortalamaları ve ikinci moment tahminlerini kullanarak öğrenme oranını dinamik olarak ayarlar.

Adam optimizasyon algoritması, kayıp fonksiyonunu minimize etmek için model parametrelerini verimli bir şekilde günceller. Özellikle büyük veri setlerinde ve yüksek boyutlu modellerde etkili sonuçlar sağlar.

Parametreler:

Modelimizdeki farklı katmanların parametrelerini ve işlevlerini daha iyi anlamak için, her bir katmanın parametre hesaplamalarını ve işlevlerini detaylı olarak ele alacağız.

Conv1D Katmanları

Conv1D (1 Boyutlu Konvolüsyon) katmanları, zaman serisi verileri veya bir boyutlu veri dizileri üzerinde konvolüsyon işlemleri gerçekleştirir. Bu katmanların parametreleri, filtre sayısı, çekirdek boyutu ve giriş boyutlarına göre hesaplanır.

Filtre Sayısı: Her bir konvolüsyon katmanı belirli sayıda filtre kullanır. Bu filtreler, giriş verilerindeki belirli özellikleri çıkarır ve her biri için farklı bir ağırlık matrisi vardır.

Çekirdek Boyutu (Kernel Size): Çekirdek boyutu, her bir filtrenin giriş verisi üzerinde kaç öğe boyunca kaydırılacağını belirler.

Giriş Boyutları: Giriş verisinin boyutları, katmana giren verinin şekli ile ilgilidir.

Örneğin, ilk Conv1D katmanının parametrelerini hesaplayalım:

Filtre Sayısı: 32

Çekirdek Boyutu: 3

Giriş Kanalları: 1 (tek boyutlu veri)

Parametreler şu şekilde hesaplanır:

Parametre sayısı=Filtre sayısı × (Çekirdek boyutu×Giriş kanalları+1)

32 × (3×1+1) =128

Bu hesaplamada, her bir filtre için 3 ağırlık parametresi ve 1 bias parametresi bulunur.

MaxPooling1D Katmanı:

MaxPooling1D katmanı, özellik haritasının boyutunu azaltarak hesaplama maliyetini düşürür ve modelin daha karmaşık desenleri öğrenmesini kolaylaştırır. Havuzlama katmanları, giriş özellik haritalarını belirli bir boyutta (havuzlama penceresi) havuzlayarak her havuzlama penceresi içindeki maksimum değeri seçer. Bu işlem, giriş verisinin boyutunu küçültür ve önemli özelliklerin korunmasını sağlar.

Amaç: Boyut azaltma, modelin hesaplama karmaşıklığını düşürür ve aşırı öğrenmeyi (overfitting) kontrol altında tutar. Aynı zamanda, modelin daha belirgin özellikleri tanımasına yardımcı olur.

Yoğun Katmanlar (Dense Layers)

Yoğun katmanlar, her nöronun bir önceki katmandaki tüm nöronlarla tam bağlantılı olduğu katmanlardır. Bu katmanlar, modelin öğrenme kapasitesini artırır ve verilerdeki karmaşık ilişkileri öğrenmesini sağlar.

Giriş Özellikleri: Giriş verisinin boyutları.

Nöron Sayısı: Katmandaki nöron sayısı.

Bias Terimleri: Her nöron için bir bias terimi bulunur.

Örneğin, ilk yoğun katmanın parametrelerini hesaplayalım:

Giriş Özellikleri: 2368

Nöron Sayısı: 128

Parametreler şu şekilde hesaplanır:

Parametre sayısı = Giriş Özellikleri × Nöron sayısı + Bias Terimleri

2368×128+128=303.232

Bu hesaplamada, her bir nöron için 2368 ağırlık parametresi ve bir bias parametresi bulunur.

Model Özeti:

Modelimizdeki her katmanın parametreleri, belirli özelliklere ve giriş boyutlarına dayanarak hesaplanır. Conv1D katmanları, filtre sayısı ve çekirdek boyutuna göre; MaxPooling1D katmanı, boyut azaltma amacıyla ve Yoğun Katmanlar, giriş özellikleri ve nöron sayısına göre parametre hesaplamalarını içerir. Bu hesaplamalar, modelin öğrenme sürecini ve performansını doğrudan etkiler.

1. Giriş Verilerinin Alınması ve Yeniden Şekillendirilmesi

Model, 78 boyutlu giriş verilerini alır. Bu veriler, her bir veri noktasının 78 özellikten oluştuğu anlamına gelir. Bu giriş verileri, `Reshape` katmanı kullanılarak Conv1D katmanları için uygun formata getirilir. Örneğin, giriş verisi (78,) şeklinden (78, 1) şekline dönüştürülür. Bu, Conv1D katmanlarının veriyi işlemesi için gereklidir.

2. Konvolüsyonel Katmanların Öğrenmesi

Konvolüsyonel Katmanlar (Conv1D): Modeldeki Conv1D katmanları, verilerden mekansal özellikler öğrenir. Bu katmanlar, belirli bir çekirdek boyutu ve filtre sayısı kullanarak giriş verisindeki önemli özellikleri çıkarır.

İlk Conv1D Katmanı: 32 filtre ve çekirdek boyutu 3 olan bir Conv1D katmanı eklenir. Bu katman, giriş verisindeki temel özellikleri öğrenir.

İkinci Conv1D Katmanı: 64 filtre ve çekirdek boyutu 3 olan başka bir Conv1D katmanı eklenir. Bu katman, önceki katmanın öğrendiği özellikleri daha da rafine eder.

3. Maksimum Havuzlama ile Boyut Azaltma

MaxPooling1D Katmanı: Maksimum havuzlama katmanı, özellik haritalarının boyutlarını azaltarak modelin hesaplama maliyetini düşürür ve önemli özelliklerin korunmasını sağlar. Örneğin, `MaxPooling1D(pool_size=2) ` kullanılarak, her iki değerden biri seçilir ve özellik haritası yarıya indirilir.

4. Dropout Katmanları ile Aşırı Öğrenmenin Önlenmesi

Dropout katmanları, eğitim sırasında giriş birimlerinin belirli bir yüzdesini rastgele devre dışı bırakarak modelin aşırı öğrenmesini önler. Bu işlem, modelin genelleme yeteneğini artırır ve overfitting riskini azaltır. Örneğin, `Dropout(0.5)` kullanılarak giriş birimlerinin %50'si rastgele devre dışı bırakılır.

5. Düzleştirme Katmanı ile Verilerin Hazırlanması

Konvolüsyonel ve havuzlama katmanlarından çıkan çok boyutlu veriler, `Flatten` katmanı ile tek boyutlu bir vektöre dönüştürülür. Bu, verilerin yoğun katmanlar (Dense Layers) tarafından işlenebilmesi için gereklidir.

6. Yoğun Katmanlar ile Sınıflandırma

Yoğun katmanlar, verilerdeki karmaşık ilişkileri öğrenir ve sınıflandırma işlemini gerçekleştirir.

İlk Yoğun Katman: 128 nörona sahip bir yoğun katman eklenir ve ReLU aktivasyon fonksiyonu kullanılarak doğrusal olmayan özellikler öğrenilir.

Dropout Katmanı: İlk yoğun katmandan sonra, aşırı öğrenmeyi önlemek için bir Dropout katmanı eklenir (`Dropout(0.5)`).

Çıkış Katmanı: Son olarak, 5 nörona sahip bir çıkış katmanı eklenir ve Softmax aktivasyon fonksiyonu kullanılarak 5 sınıf üzerinde olasılık dağılımı üretilir. Bu, modelin her sınıf için bir olasılık değeri üretmesini sağlar.

7. Modelin Derlenmesi ve Değerlendirilmesi

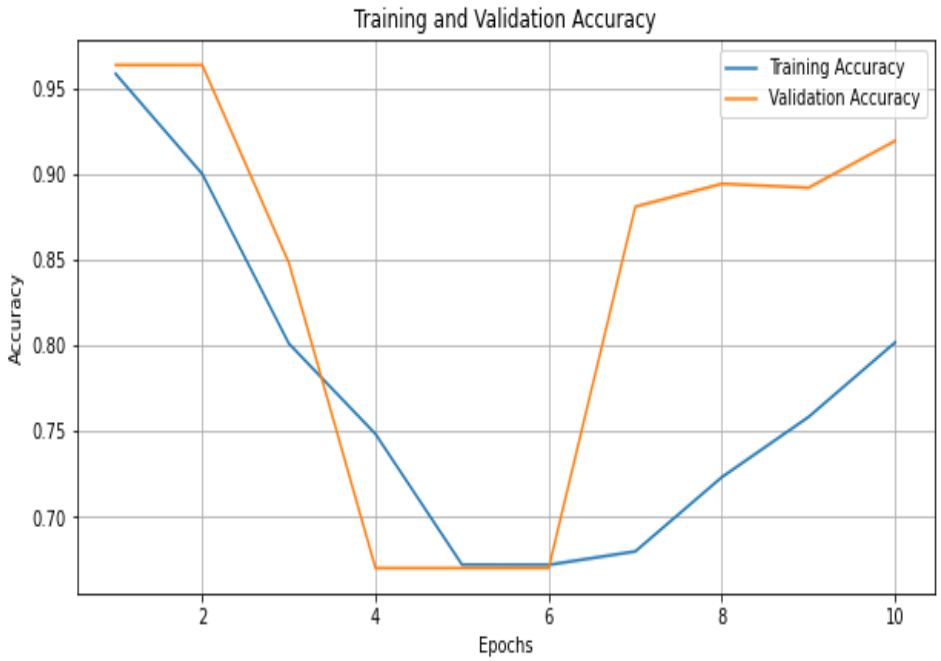
Optimizasyon: Model, Adam optimizasyon algoritması ile derlenir. Adam, öğrenme oranını uyarlamalı olarak ayarlayarak eğitim sürecini hızlandırır ve verimli hale getirir.

Kayıp Fonksiyonu: `sparse_categorical_crossentropy` kayıp fonksiyonu kullanılır. Bu fonksiyon, çok sınıflı sınıflandırma problemlerinde kullanılan bir kayıp fonksiyonudur ve tahmin edilen olasılık dağılımı ile gerçek etiketler arasındaki farklılığı ölçer.

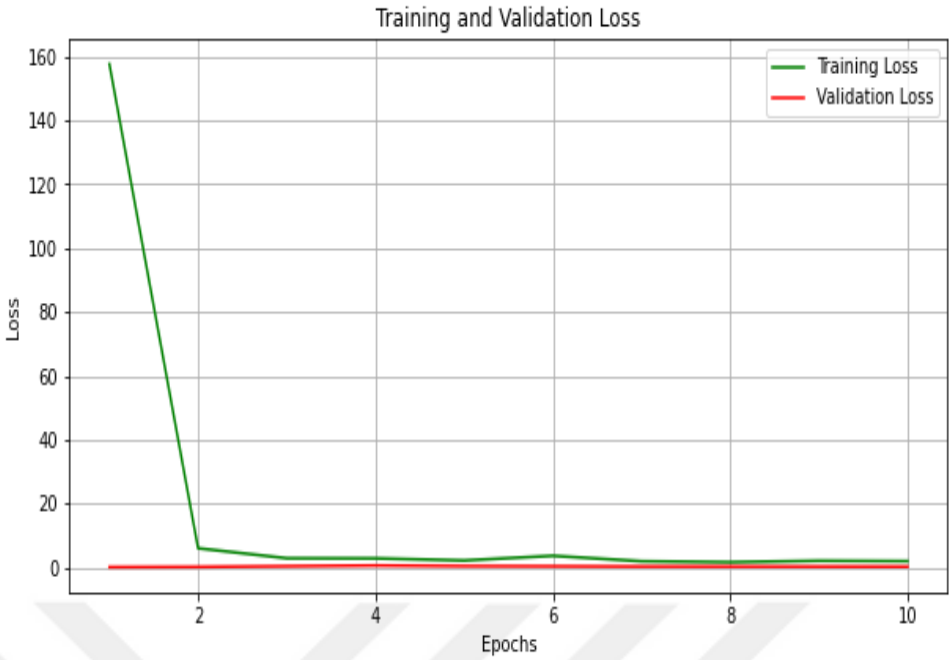
Değerlendirme Metriği: Model, doğruluk (accuracy) metriği kullanılarak değerlendirilir. Bu metrik, modelin tahminlerinin etiketlerle ne sıklıkta eşleştiğini ölçer ve modelin performansını basit ve anlaşılır bir şekilde gösterir.

Özet: Bu model, 78 boyutlu giriş verilerini işleyerek mekansal özellikler öğrenir, özellik haritalarının boyutlarını azaltır, aşırı öğrenmeyi önler, verileri yoğun katmanlar için hazırlar ve nihayetinde veriyi 5 sınıfa ayırır. Model, Adam optimizasyon algoritması ile derlenir ve doğruluk metriği ile değerlendirilir. Bu yapı, modelin verimli ve etkili bir şekilde öğrenmesini sağlar ve yüksek doğruluk oranları elde etmeye yardımcı olur.

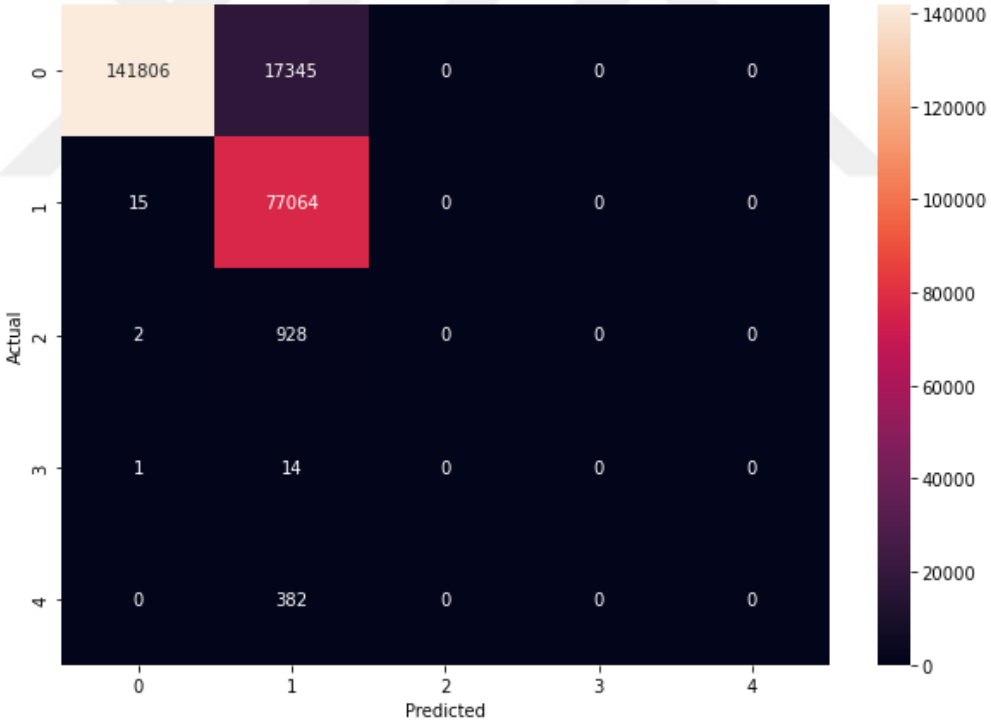
Model Sonuçları



Şekil 5.5: CNN Model Eğitim ve Doğrulama Grafiği



Şekil 5.6: CNN Model Eğitim ve Doğrulama Kaybı



Şekil 5.7: CNN Karışıklık Martisi

Çizelge 5.4: CNN Sınıflandırma Raporu

	precision	recall	f1-score	support
BENIGN	1.00	0.89	0.94	159151
DDoS	0.80	1.00	0.89	77079
Web_Attack_Brute_Force	0.00	0.00	0.00	930
Web_Attack_Sql_Injection	0.00	0.00	0.00	15
Web_Attack_XSS	0.00	0.00	0.00	382
accuracy			0.92	237557
macro avg	0.36	0.38	0.37	237557
weighted avg	0.93	0.92	0.92	237557

5.2.3 Transformer modeli

Çizelge 5.5: Transformer Katmanları ve Parametreleri

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 78)]	0
dense (Dense)	(None, 64)	5056
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 78)	5070

Total params: 10,126
Trainable params: 10,126
Non-trainable params: 0

Parametreler:

Yoğun Katman (64 nöron): Bu katman, modelde 64 nöron bulunan tamamen bağlı bir yapıdır. Her bir nöron, önceki katmandan gelen tüm girişlerle bağlantılıdır, yani her bir nöron 78 girişten veri alır. Buna ek olarak, her nöronun bir bias (sapma) terimi de bulunur. Toplam parametre sayısı şu şekilde hesaplanır:

(78 giriş + 1 bias) × 64 nöron = 5056 parametre.

Dropout Katmanı: Bu katman, modelin aşırı uyum (overfitting) yapmasını engellemeye yardımcı olur. Eğitim sürecinde belirli nöronları rastgele devre dışı bırakarak çalışır. Dropout katmanının kendine ait eğitilecek herhangi bir parametresi yoktur, yalnızca eğitim sürecini etkiler.

Yoğun Katman (78 nöron): Bu katman, modelde 78 nöron bulunan başka bir tamamen bağlı yapıdır. Her bir nöron, önceki katmandan gelen 64 girişi alır ve ayrıca bir bias terimine sahiptir. Toplam parametre sayısı şu şekilde hesaplanır:

(64 giriş + 1 bias) × 78 nöron = 5070 parametre.

Model Özeti:

Girdi Şekli: Modelin aldığı girdi, 78 farklı özelliği barındıran bir vektördür. Bu özellikler, modelin öğrenmesi gereken tüm bilgileri temsil eder ve her bir veri noktasının anlamlandırılması için kullanılır.

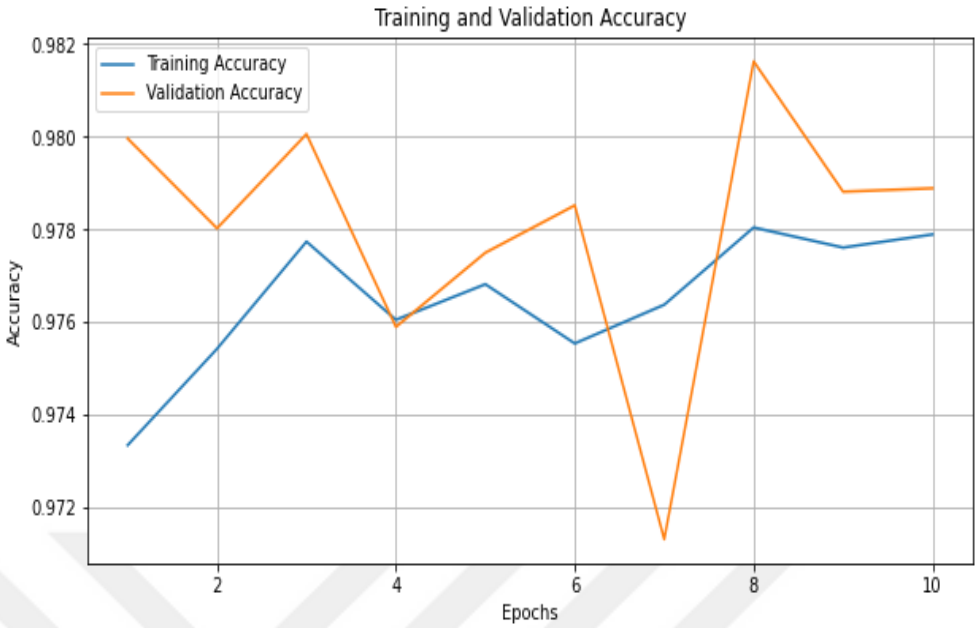
Kodlayıcı (Encoder): Kodlayıcı bölümü, girdiyi daha az boyutlu bir temsile dönüştürmek için tasarlanmıştır. Bu işlem sırasında, 78 özellikten oluşan giriş vektörü, 64 özellik içeren daha kompakt bir temsile sıkıştırılır. Aşırı öğrenmeyi (overfitting) önlemek için bu süreçte dropout tekniği uygulanır. Dropout, eğitim sırasında belirli nöronları rastgele devre dışı bırakarak modelin genel performansını artırmayı amaçlar.

Kod Çözücü (Decoder): Kodlayıcı tarafından oluşturulan düşük boyutlu temsili, orijinal 78 boyutlu girdiye geri döndüren bileşendir. Kod çözücü, modelin orijinal veriyi yeniden üretmesini sağlar ve böylece modelin öğrendiği temsili doğru bir şekilde geri çevirmesi beklenir.

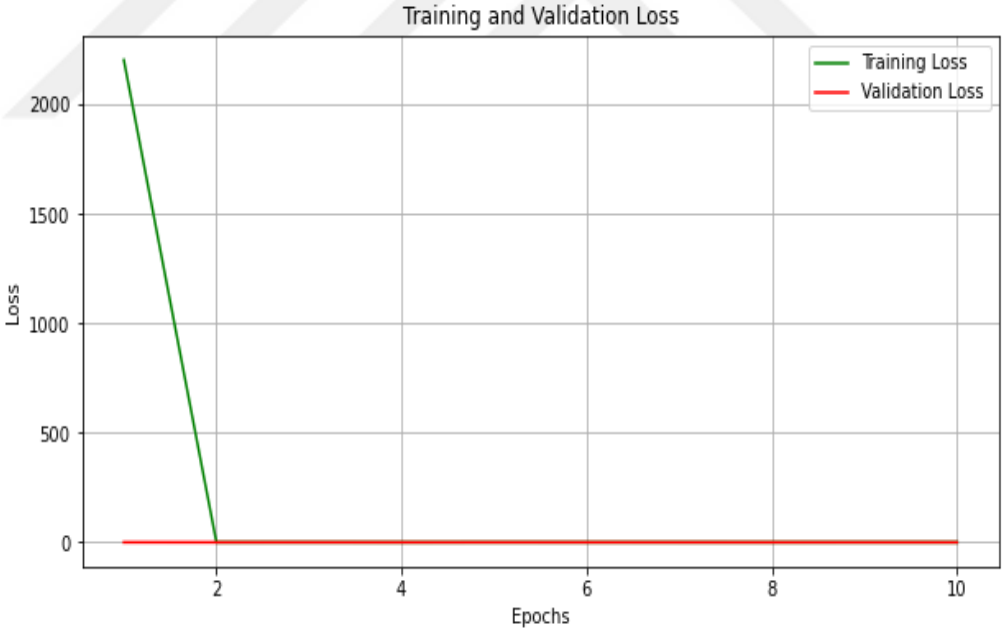
Kayıp Fonksiyonu: Modelin performansını değerlendirmek ve optimize etmek için kullanılan kayıp fonksiyonu, yeniden yapılandırma hatasını ölçer. Bu modelde, binary cross-entropy kayıp fonksiyonu kullanılır. Binary cross-entropy, orijinal ve yeniden oluşturulan veriler arasındaki farkı hesaplayarak modelin ne kadar iyi çalıştığını gösterir.

Çıktı: Modelin çıktısı, orijinal girdi verisinin yeniden yapılandırılmış halidir. Kod çözücü tarafından üretilen bu veri, modelin girdi verisini ne kadar doğru bir şekilde yeniden oluşturabildiğini gösterir.

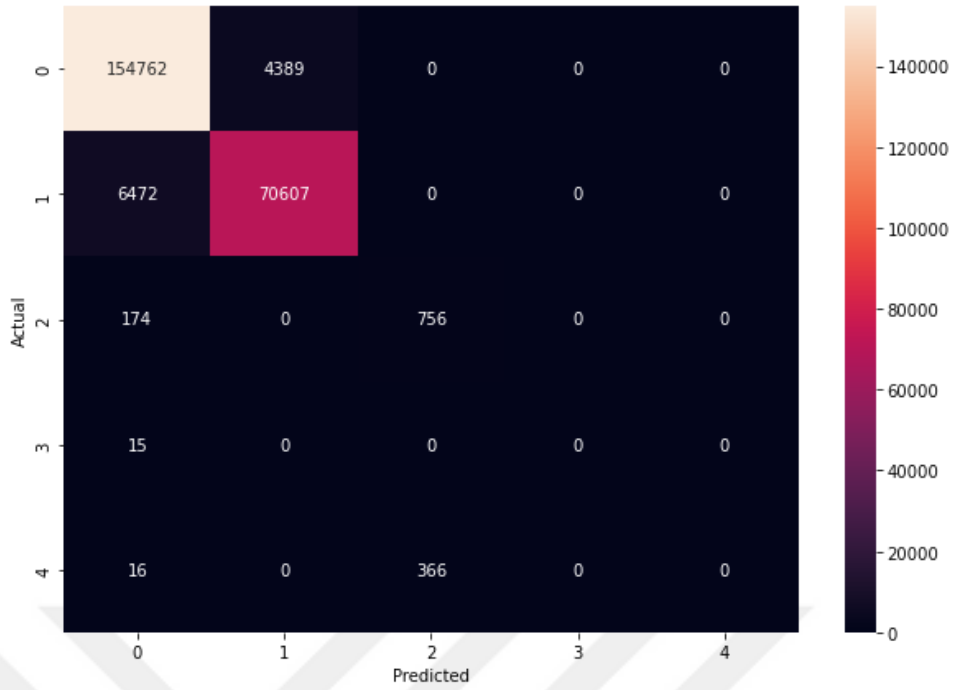
Model Sonuçları



Şekil 5.8: Transformer Model Eğitim ve Doğrulama Grafiği



Şekil 5.9: Transformer Model Eğitim ve Doğrulama Kaybı



Şekil 5.10: Transformer Karışıklık Matrisi

Çizelge 5.6: Transformer Sınıflandırma Raporu

	precision	recall	f1-score	support
BENIGN	0.96	0.97	0.97	159151
DDoS	0.94	0.92	0.93	77079
Web_Attack_Brute_Force	0.67	0.81	0.74	930
Web_Attack_Sql_Injection	0.00	0.00	0.00	15
Web_Attack_XSS	0.00	0.00	0.00	382
accuracy			0.95	237557
macro avg	0.51	0.54	0.53	237557
weighted avg	0.95	0.95	0.95	237557

6. TARTIŞMA VE SONUÇ

Bu tezde, Yazılım Tanımlı Ağlar (YTA) için anomali tespiti alanında derin öğrenme tekniklerinin etkinliğini incelemek amacıyla üç farklı derin öğrenme modeli kullanılmıştır: Yapay Sinir Ağı (ANN), Evrişimli Sinir Ağı (CNN) ve Transformer tabanlı bir model. Bu bölümde, her bir modelin performansı tartışılacak, elde edilen sonuçlar değerlendirilecek ve bu sonuçların YTA güvenliği üzerindeki potansiyel etkileri analiz edilecektir.

Model Performansları:

Yapay Sinir Ağı (ANN):

ANN modeli, anomali tespitinde yüksek bir doğruluk oranı elde etmiştir. Benign ve DDoS saldırılarını tespit etme konusunda başarılı sonuçlar verirken, özellikle daha az görülen Web Saldırılarını (Brute Force, SQL Injection, XSS) tespit etme konusunda zorluklar yaşamıştır. Bu durum, modelin yaygın saldırıları öğrenirken nadir saldırıları yeterince temsil edememesi ile açıklanabilir. Bununla birlikte, ANN modelinin genel doğruluğu, SDN ortamlarındaki temel anomali tespit görevleri için yeterli olduğunu göstermektedir.

Evrişimli Sinir Ağı (CNN):

CNN modeli, veri kümesinin uzaysal özelliklerini öğrenme yeteneği sayesinde belirli saldırı türlerinde (örneğin DDoS) başarılı olmuştur. Ancak, genel performansı ANN modeline göre daha düşük kalmıştır. Özellikle Web Saldırıları için düşük F1-skorları, CNN'in bu tür anomali desenlerini yeterince öğrenemediğini göstermektedir. CNN'in düşük performansı, SDN ağlarında daha karmaşık anomali tespit görevleri için yetersiz kalabileceğini düşündürmektedir.

Transformer Modeli:

Transformer tabanlı model hem doğruluk hem de F1-skorları açısından en dengeli performansı göstermiştir. Model, tüm anomali türlerinde tutarlı bir şekilde yüksek performans sergileyerek, YTA ortamlarında çeşitli saldırıları tespit edebilme

potansiyelini göstermiştir. Transformer modellerinin sekans veri analizi konusundaki üstün yetenekleri, YTA trafik verilerini işlerken önemli avantajlar sağlamıştır.

Sonuçların Değerlendirilmesi.

Bu çalışmanın bulguları, derin öğrenme modellerinin YTA ortamlarında anomali tespiti için etkili araçlar olduğunu ortaya koymaktadır. Her modelin güçlü ve zayıf yönleri, belirli kullanım senaryoları ve ihtiyaçlar doğrultusunda değerlendirilmelidir:

1. ANN modeli: Temel anomali tespiti görevlerinde başarılıdır ve yaygın saldırıları tespit etmek için yeterli olabilir. Ancak, daha az görülen saldırıları tespit etmek için ek veri işleme veya model geliştirme gerektirebilir.

2. CNN modeli: Uzaysal veri analizinde güçlü olmasına rağmen, YTA ortamlarında çeşitlilik gösteren anomali türlerini tespit etmekte zorlanabilir. Bu modeller, belirli saldırı türlerine odaklanmak için optimize edilebilir.

3. Transformer Modeli: Geniş bir yelpazedeki anomali türlerini tespit etme konusunda üstün performans göstermiştir. YTA gibi dinamik ve karmaşık ağ ortamlarında güvenliği artırmak için ideal bir seçim olabilir.

YTA Güvenliği Üzerindeki Potansiyel Etkiler:

Derin öğrenme modellerinin YTA güvenliği üzerindeki potansiyel etkileri çeşitli açılardan değerlendirilebilir. Gelişmiş Tespit Kabiliyeti Derin öğrenme modelleri, özellikle nadir ve karmaşık saldırıları tespit edebilme yetenekleri ile SDN güvenliğini önemli ölçüde artırabilir.

Adaptif Öğrenme: Bu modellerin adaptif öğrenme yetenekleri, dinamik YTA ortamlarına hızlı bir şekilde uyum sağlamalarını ve yeni tehditleri tespit etmelerini sağlar.

Otomasyon ve Ölçeklenebilirlik: Derin öğrenme tabanlı çözümler, otomatik ve ölçeklenebilir güvenlik önlemleri sunarak, manuel müdahale gereksinimini azaltır ve geniş ağ yapılarında etkin güvenlik sağlar.

Bu tezde gerçekleştirilen çalışmalar, derin öğrenme modellerinin YTA ortamlarında anomali tespiti için güçlü araçlar olduğunu göstermektedir. ANN, CNN ve Transformer modelleri arasında yapılan karşılaştırma, her bir modelin belirli

avantajlarını ve sınırlamalarını ortaya koymuştur. Özellikle Transformer modellerinin genel performansı, YTA güvenliği için umut vaat etmektedir.

Gelecekteki çalışmalar, bu modellerin performansını daha da artırmak ve çeşitli saldırı türlerini daha etkin bir şekilde tespit etmek amacıyla model mimarilerini ve eğitim süreçlerini optimize etmeye odaklanabilir. Ayrıca, gerçek zamanlı anomali tespiti ve müdahale sistemlerinin geliştirilmesi, YTA ortamlarında daha güvenli ve dayanıklı ağlar oluşturulmasına katkıda bulunacaktır.

Bu sonuçlar, YTA teknolojisinin yaygınlaşması ile birlikte, ağ güvenliğinde derin öğrenme yaklaşımlarının önemini vurgulamakta ve bu alandaki gelecekteki araştırmalara ışık tutmaktadır.

7. ÖNERİLER

Bu çalışmada, yazılım tanımlı ağlarda derin öğrenme yöntemleri kullanılarak anomali tespiti gerçekleştirilmiştir. Ancak, daha ileri çalışmalar için bazı öneriler sunulabilir:

1. Model Geliştirme ve Optimizasyon: Çalışmada kullanılan derin öğrenme modelinin doğruluğunu ve etkinliğini artırmak için hiperparametre optimizasyonu yapılabilir. Ek olarak, farklı derin öğrenme mimarileri (örneğin, Transformers veya GNN'ler) incelenerek daha verimli sonuçlar elde edilebilir.

2. Veri Setinin Zenginleştirilmesi: Mevcut çalışmada kullanılan veri seti genişletilerek ve farklı veri kaynaklarından yeni anomali tipleri eklenerek modelin genelleme kapasitesi artırılabilir. Özellikle gerçek dünya ağ verileri kullanılarak modelin gerçek koşullarda daha güvenilir sonuçlar verebilmesi sağlanabilir.

3. Gerçek Zamanlı Anomali Tespiti: Çalışmada geliştirilen modelin gerçek zamanlı olarak kullanılabilirliği üzerine çalışmalar yapılabilir. Yazılım tanımlı ağlarda hızlı yanıt süreleri gerektiren durumlar için modelin performansını optimize etmek önemli bir araştırma alanıdır.

4. Siber Saldırı Türlerine Göre Özelleştirme: Model, çeşitli siber saldırı türlerine göre özelleştirilerek her bir saldırı türüne yönelik daha spesifik tespit yöntemleri geliştirilebilir. Bu sayede, farklı saldırı türlerine karşı daha yüksek hassasiyetle çalışan modeller elde edilebilir.

5. Yeniden Eğitim ve Adaptif Sistemler: Yazılım tanımlı ağlar hızla değişen bir yapıya sahip olduğundan, modelin belirli aralıklarla yeniden eğitilmesi ve yeni saldırı türlerine karşı adapte olabilmesi için adaptif bir sistem tasarımı yapılabilir. Böylece model, yeni anomali türlerini hızlıca öğrenebilir ve tespit edebilir.

6. Güvenlik Önlemleriyle Entegrasyon: Modelin çıktılarını analiz eden ve otomatik olarak güvenlik önlemlerini devreye sokan bir güvenlik yönetim sistemi ile entegrasyon sağlanabilir. Bu tür bir yaklaşım, anomali tespitini daha etkin hale getirecek ve müdahale süresini azaltacaktır.

Bu öneriler doğrultusunda, çalışmanın daha geniş kapsamlı hale getirilmesi ve yazılım tanımlı ağlarda derin öğrenme yöntemleri ile daha etkin bir anomali tespit sistemi geliştirilmesi mümkündür

KAYNAKÇA

Abhilash, G., & Divyansh, G. (2018, December). Intrusion detection and prevention in software defined networking. In 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) (pp. 1-4). IEEE.

Al-Sultani, Z. N. M. A. (2012). An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System (Doctoral dissertation, Middle East University).

Balestriero, R., & LeCun, Y. (2023, June). POLICE: Provably optimal linear constraint enforcement for deep neural networks. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE.

Bai, X., Bai, J., Yang, X., Liu, H., Wang, B., & Liu, Y. (2020, November). A deep learning approach to detect anomaly in software-defined network. In 2020 12th International Conference on Advanced Infocomm Technology (ICAIT) (pp. 100-106). IEEE.

Coşkun, M. M. (2018, September 16). Anomaly detection (anomali tespiti) nedir. Medium. https://medium.com/yaz%C4%B1l%C4%B1m-bilimi/anomaly-detection-anomali-tespiti-nedir-989a956df7a7.

Codegenius (2021, May 16).Transformer Neural Networks – 1 – Genel Mimari ve Giriş. https://www.codegeni.us/transformer-neural-networks/#google_vignette

Cicioğlu, M., & Çalhan, A. (2017). Yazılım tanımlı ağlar–YTA. Karaelmas Fen ve Mühendislik Dergisi, 7(2), 684-695.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 1-58.

Çekmez, U., Erdem, Z., Yavuz, A. G., Sahingoz, O. K., & Buldu, A. (2018, May). Network anomaly detection with deep learning. In 2018 26th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

Dawoud, A., Shahristani, S., & Raun, C. (2018, May). Yazılım tanımlı ağ güvenliğini geliştirmeye yönelik derin bir öğrenme çerçevesi. In 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 709-714). IEEE.

Feamster, N., Rexford, J., & Zegura, E. (2014). The road to SDN: An intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, 44(2), 87-98.

Gao, N., Gao, L., Gao, Q., & Wang, H. (2014, November). An intrusion detection model based on deep belief networks. In 2014 Second International Conference on Advanced Cloud and Big Data (pp. 247-252). IEEE.

Gupta, R., Kumar, Z. C., & Patil, V. N. (2022, April). Design of deep neural network based anomaly detection system. In 2022 IEEE 7th International Conference for Convergence in Technology (I2CT) (pp. 1-5). IEEE.

Erdal, H. (March 2023). Dalgacık Dönüşümü ve Evrişimsel Sinir Ağları Kullanılarak Günlük Borsa Endeksinin Derin Öğrenmeye Dayalı Tahmini. ResearchGate. https://www.researchgate.net/figure/Oernek-bir-evrisimsel-sinir-agi-mimarisi_fig2_369588622

Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., & Gayraud, T. (2014). Software-defined networking: Challenges and research opportunities for future internet. Computer Networks, 75, 453-471.

Hu, F., Hao, Q., & Bao, K. (2014). A survey on software-defined network and OpenFlow: From concept to implementation. IEEE Communications Surveys & Tutorials, 16(4), 2181-2206.

Karataş, G. (2020). Derin öğrenme tabanlı saldırı tespit sistemi.

Konak, İ.(2023, Aug 5). Yapay Zekanın Dönüştürücüsü: Transformer Mimarisi-Türkçe Anlatım — Part 3: Transformer'a adım adım. https://medium.com/@i_konak/transformer-mimarisi-t%C3%BCrk%C3%A7e-anlat%C4%B1m-part-3-transformera-ad%C4%B1m-ad%C4%B1m-6b19f5cde5ae

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.

Krzemień, W., Jędrasiak, K., Nawrat, A., & Daniec, K. (2021, December). Anomaly detection in software-defined networks using cross-validation. In 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1-7). IEEE.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69-74.

Mahamat, S. B., & Çeken, C. (2019). Anomaly detection in software-defined networking using machine learning. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 7(1), 748-756.

Niyaz, Q., Sun, W., & Javaid, A. Y. (2016). A deep learning based DDoS detection system in software-defined networking (SDN). arXiv preprint arXiv:1611.07400.

Ongaro, F. (2014). Enhancing quality of service in software-defined networks. Diss. Alma Mater Studiorum-University Of Bologna.

Pimentel, T., Monteiro, M., Veloso, A., & Ziviani, N. (2020, July). Deep active learning for anomaly detection. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

Qin, Y., Wei, J., & Yang, W. (2019, September). Deep learning based anomaly detection scheme in software-defined networking. In 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS) (pp. 1-4). IEEE.

Rawat, D. B., & Reddy, S. R. (2016). Software defined networking architecture, security and energy efficiency: A survey. IEEE Communications Surveys & Tutorials, 19(1), 325-346.

Scott-Hayward, S., Natarajan, S., & Sezer, S. (2015). A survey of security in software-defined networks. IEEE Communications Surveys & Tutorials, 18(1), 623-654.

Shin, S., & Gu, G. (2013, August). Attacking software-defined networks: A first feasibility study. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 165-166).

Voellmy, A., Kim, H., & Feamster, N. (2012, August). Procera: A language for high-level reactive network control. In Proceedings of the first workshop on Hot topics in software defined networks (pp. 43-48).

Yavuz, A., & Tuna, G. (2020). Yazılım tanımlı ağların ve geleneksel bilgisayar ağlarının güvenlik karşılaştırması (Master's thesis, Trakya Üniversitesi Fen Bilimleri Enstitüsü).

Zhang, G., Qiu, X., & Gao, Y. (2019, June). Software defined security architecture with deep learning-based network anomaly detection module. In 2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN) (pp. 784-788). IEEE.

Zinner, T., Jarschel, M., Hossfeld, T., Tran-Gia, P., & Kellerer, W. (2013). A compass through SDN networks.

ÖZGEÇMİŞ

EĞİTİM

- **Lisans:** Azerbaycan Teknik Üniversitesi - Otomosyon ve Kontrol Mühendisliği (2017-2021)
- **Yüksek Lisans:** T.C İstanbul Gedik Üniversitesi – Yapay Zeka Mühendisliği (2021-2024)