
GAZİ ÜNİVERSİTESİ
Gazi
1926

YAZILIM TANIMLI AĞLARDA TRAFİK MÜHENDİSLİĞİ İÇİN DERİN ÖĞRENME TABANLI YAKLAŞIM GELİŞTİRME

Sudad ABDULRAZZAQ

YÜKSEK LİSANS TEZİ
BİLGİSAYAR BİLİMLERİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

MART 2021

Sudad ABDULRAZZAQ tarafından hazırlanan "YAZILIM TANIMLI AĞLARDA TRAFİK MÜHENDİSLİĞİ İÇİN DERİN ÖĞRENME TABANLI YAKLAŞIM GELİŞTİRME" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Bilimleri Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Dr. Öğr. Üyesi Mehmet DEMİRCİ

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ...……

Başkan: Prof. Dr. Hasan Şakir BİLGE

Elektrik - Elektronik Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

……..

Üye: Doç. Dr. Ahmet Burak CAN

Bilgisayar Mühendisliği Ana Bilim Dalı, Hacettepe Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

……..

Tez Savunma Tarihi: 09/03/2021

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

…….…….

Prof. Dr. Aslıhan TÜFEKCİ

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Sudad ABDULRAZZAQ

09/03/2021

YAZILIM TANIMLI AĞLARDA TRAFİK MÜHENDİSLİĞİ İÇİN DERİN ÖĞRENME TABANLI YAKLAŞIM GELİŞTİRME

(Yüksek Lisans Tezi)

Sudad ABDULRAZZAQ

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Mart 2021

ÖZET

Trafik mühendisliği, özellikle büyük miktarda veri taşıyan günümüzün büyük ağlarında ağ yönetimi için çok önemlidir. Trafik mühendisliği, kaynakların akıllıca tahsisi yoluyla ağın verimliliğini ve güvenilirliğini artırmayı amaçlamaktadır. Bu tez çalışmasında, çeşitli uygulamalar arasında bant genişliği tahsisini iyileştirmek için yazılım tanımlı ağlarda derin öğrenme tabanlı bir trafik mühendisliği yaklaşımı önerilmiştir. Önerilen yaklaşım ile trafik akışlarının sınıflandırılması için derin öğrenme tabanlı bir model geliştirilmek ve trafik sınıflarına göre ağdaki trafik akışlarının hizmet kalitesi dikkate alınarak yönlendirilmesi için bir trafik mühendisliği modülünü YTA kontrolcü üzerinde gerçekleştirilmek amaçlanmıştır. Derin sinir ağı (DNN) ve tek boyutlu evrişimli sinir ağı (1-B CNN) modellerine dayalı trafik sınıflandırması uygulanmıştır. Çeşitli uygulamalardan gelen akışları tanımlayarak ve tanımlanan akışı her kuyruğun farklı bir önceliğe sahip olduğu birden çok kuyruğa dağıtarak hizmet kalitesini (QoS) iyileştirmek hedeflenmiştir. Ayrıca, ağ bant genişliğini ve gelen trafiğin hacmini yönetmek için trafik şekillendirme uygulanmaktadır. Ağın performansını artırmak ve trafik sıkışıklığını önlemek için, genel yük dengelemesini gerçekleştirmek üzere bağlantı noktası kapasitesini dikkate alan bir teknik kullanılmıştır. Sentetik azınlık aşırı örnekleme tekniği (SMOTE) adı verilen bir yüksek hızda örnekleme tekniği uygulanarak dengesiz veri kümesi sorununa çözüm sağlanmıştır. DNN ve 1-B CNN'nin performansı, KNN, SVM, DT ve RF gibi bazı makine öğrenimi modelleriyle karşılaştırılmış ve değerlendirilmiştir. Sonuçlar, 1-B CNN ve DNN'nin 5 s ve 10 s sürelerde yakalanan trafikte yüksek doğruluk elde edebildiğini, KNN ve RF'nin ise 15 s, 30 s, 60 s ve 120 s sürelerde yakalanan trafikte yüksek doğruluk elde edebildiğini gösterilmiştir. CNN ve DNN, 15 s, 30 s, 60 s ve 120 s sürelerde DT ve SVM'den daha iyi doğruluk elde etmeyi başarmıştır. Hizmet kalitesi odaklı trafik mühendisliği sistemini değerlendirmek için farklı senaryolar uygulanmıştır. Sonuçlar, tanımlanan akışlara trafik şekillendirme uygulanmasının, gelen akışların hızını ve hacmini kontrol ederek ağın performansını ve bant genişliği kullanılabilirliğini artırdığını gösterilmiştir.

Bilim Kodu	:	92407
Anahtar Kelimeler	:	Derin öğrenme, hizmet kalitesi, yazılım tanımlı ağlar, trafik sınıflandırması, trafik mühendisliği, trafik şekillendirme
Sayfa Adedi	:	85
Danışman	:	Dr. Öğr. Üyesi Mehmet DEMİRCİ

DEVELOPING A DEEP LEARNING BASED APPROACH FOR TRAFFIC ENGINEERING IN SOFTWARE DEFINED NETWORKS

(M. Sc. Thesis)

Sudad ABDULRAZZAQ

GAZI UNIVERSITY

INSTITUTE OF INFORMATICS

March 2021

ABSTRACT

Traffic engineering is essential for network management, particularly in today's large networks carrying massive amounts of data. Traffic engineering aims to increase the network's efficiency and reliability through intelligent allocation of resources. In this thesis, we propose a deep learning-based traffic engineering system in software-defined networks (SDN) to improve bandwidth allocation among various applications. The proposed system implements traffic classification based on deep neural network (DNN) and one dimensional convolution neural network (1-D CNN) models, and implements a traffic engineering module on the SDN controller to direct traffic flows in the network according to traffic classes by considering the Quality of Service. The system aims to improve the Quality of Service (QoS) by identifying flows from various applications and distributing the identified flow to multiple queues where each queue has a different priority. Next, it applies traffic shaping in order to manage network bandwidth and the volume of incoming traffic. To increase the network's performance and avoid traffic congestion, we implement a technique that considers the port capacity to accomplish general load balancing. We solved the issue of imbalanced dataset by implementing an oversampling technique called Synthetic Minority Over-Sampling Technique (SMOTE). The performance of DNN and 1-D CNN have been compared and evaluated with some of machine learning models, such as KNN, SVM, DT, and RF. The results showed 1-D CNN and DNN are able to achieve high accuracy of traffic captured at 5s and 10s timeout, while KNN and RF are able to achieve high accuracy of traffic captured at 15s, 30s, 60s and 120s timeout, while CNN and DNN were able to achieve better accuracy than DT and SVM at 15 s, 30 s, 60 s and 120 s timeout. Different scenarios have been used to evaluate the QoS-based traffic engineering system. The results showed that applying traffic shaping to identified flows increases network performance by regulating of bandwidth usage helps in guaranteeing resources for each type of application and bandwidth availability by controlling the rate and volume of incoming flows.

Science Code	:	92407
Key Words	:	Deep learning, quality of service, software defined networks, traffic classification, traffic engineering, traffic shaping
Page Number	:	85
Supervisor	:	Assist. Prof. Dr. Mehmet DEMİRCİ

TEŞEKKÜR

Bugün nerede olduğum konusunda bana güç, sabır ve ilham veren, uzmanlık döngümü tamamlamamda bana destek olan aileme sonsuz teşekkürlerimi sunarım. Tez danışmanım Dr. Öğr. Üyesi Mehmet DEMİRCİ'ye güven, sabır, motivasyon ve muazzam bilgi birikiminden dolayı en içten teşekkürlerimi sunarım. Kendisi, bu tezin konusunun belirlenmesinden yazım ve teslim sürecine kadar her aşamada yol gösterici önerileriyle bana çok yardımcı olmuştur. Ayrıca, Gazi Üniversitesi Bilişim Enstitüsü'nde Bilgisayar Bilimleri yüksek lisans programında ders aldığım tüm hocalarıma ve Bilişim Enstitüsü Müdürü Prof. Dr. Aslıhan TÜFEKCİ'ye teşekkür ederim. Türk dili bilgisine sahip olmamda emeği geçen tüm Gazi TÖMER öğretmenlerime de ayrıca derin şükranlarımı sunarım. Yüksek lisans sürecinde kaybettiğim değerli babam ve anneannem'e Tanrıdan rahmet diliyorum. Son olarak, Türkiye Bursları Programına bana Yüksek Lisans eğitim fırsatını sağladıkları ve sınırsız destekleri için çok müteşekkirim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
ANN	Yapay Sinir Ağı (Artificial Neural Network)
API	Uygulama Programlama Arayüzü (Application Programming Interface)
CNN	Evrişimli Sinirsel Ağ (Convolutional Neural Network)
DBN	Derin İnanç Ağları (Deep Belief Networks)
DL	Derin Öğrenme (Deep Learning)
DPI	Derin Paket Muayenesi (Deep Packet Inspection)
DT	Karar Ağacı (Decision Tree)
FTP	Dosya Aktarma Protokolü (File Transfer Protocol)
FTPS	Dosya Aktarım Protokolü Güvenli (File Transfer Protocol Secure)
GAN	Üretken Çekişmeli Ağlar (Generative Adversarial Networks)
GPU	Grafik İşleme Birimi (Graphics Processing Unit)
HTB	Hiyerarşik Jetonu Kovası (Hierarchical Token Bucket)
ICMP	İnternet Denetim Mesaj Protokolü (Internet Control Message Protocol)
ICT	Bilgi ve İletişim Teknolojileri (Information and Communication Technology)
IMAP	İnternet Mesaj Erişim Protokolü (Internet Message Access Protocol)
IP	İnternet Protokolü (Internet Protocol)
KNN	K-En Yakın Komşu (K-Nearest Neighbor)
LSTM	Uzun Kısa Süreli Bellek (Long Short Term Memory)
ML	Makine Öğrenimi (Machine Learning)
MLP	Çok Katmanlı Algılayıcı (Multilayer perceptron)

OSI	Açık Sistemler Arabağlaşımı (Open Systems Interconnection)
OVSDB	Open vSwitch Veritabanı Yönetimi Protokolü (Open vSwitch Database Management Protocol)
P2P	Uçtan Uca (Peer to Peer)
PCA	Temel Bileşen Analizi (Principal Component Analysis)
POP3S	Postane Protokolü 3 Güvenli (Post Office Protocol 3 Secure)
QoE	Deneyim Kalitesi (Quality Of Experience)
QoS	Hizmet Kalitesi (Quality Of Service)
RBM	Kısıtlı Boltzmann Makinesi (Restricted Boltzmann Machine)
ReLU	Doğrultulmuş Doğrusal Birimler (Rectified Linear Units)
RF	Rastgele Orman (Random Forest)
RNN	Devirli Sinirsel Ağ (Recurrent Neural Network)
SDN	Yazılım Tanımlı Ağlar (Software Defined-Networking)
SFTP	Güvenli Dosya Aktarım Protokolü (Secure File Transfer Protocol)
SMOTE	Sentetik Azınlık Aşırı Örnekleme Tekniği (Synthetic Minority Over-sampling Technique)
SMTPS	Basit Posta Aktarım Protokolü Güvenli (Simple Mail Transfer Protocol Secure)
SNMP	Basit Ağ Yönetimi Protokolü (Simple Network Management Protocol)
SSH	Güvenli Kabuk (Secure Shell)
STP	Kapsayan Ağaç Protokolü (Spanning Tree Protocol)
SVM	Destek Vektör Makinesi (Support Vector Machines)
TCP	İletim Kontrol Protokolü (Transmission Control Protocol)
TCP/IP	Aktarma Kontrol Protokolü/İnternet Protokolü (Transmission Control Protocol/Internet Protocol)

ToS	Servis Tipi (Type of Service)
UDP	Kullanıcı Veri Paket Protokolü (User Datagram Protocol)
VLAN	Sanal Yerel Alan Ağı (Virtual Local Area Network)
VOIP	İnternet Protokolü Üzerinden Ses (Voice Over Internet Protocol)
YTA	Yazılım Tanımlı Ağlar

1. GİRİŞ

Konunun tanımı

Son yıllarda kurumsal ağlarda işlenerek müşteriler, çalışanlar ve iş ortaklarının kullanımına sunulan veri miktarında büyük bir artış yaşanmaktadır. Genel olarak, kurumsal ortamlar ağ akışlarına öncelik veren, ağ akışlarını karakterize eden ve bilinmeyen gelen trafiği tanımlayan kritik uygulamalar içermektedir. Bu nedenle, kaynak kullanımı ve ağ trafiğinin planlanması ağ yönetiminin önemli bir uygulaması haline gelmektedir (Ng, B., 2015).

Trafik mühendisliği (TM), ağ yönetimi için gereklidir ve kaynakların akıllıca tahsisi yoluyla ağın verimliliğini ve güvenilirliğini artırmayı amaçlamaktadır. Trafik tahmini, güvenlik izleme, trafik tanımlamalı trafik mühendisliği ve tıkanıklık tahmini gibi konularda makine öğrenmesi ve derin öğrenme yöntemlerinin uygulanması giderek yaygınlaşmaktadır.

Yazılım Tanımlı Ağlar (YTA - software-defined networks, SDN), kontrol ve veri iletim işlemlerini birbirinden ayırarak ağ yönetimini basitleştirmeyi, kaynak kullanımını iyileştirmeyi, yeniliği teşvik etmeyi ve ağların performansını optimize etmeyi amaçlayan bir teknolojidir. Yazılım Tanımlı Ağlar, esnek mimarisi ve ağ yönetimini kolaylaştırması sayesinde TM uygulamalarının hayata geçirilmesini kolaylaştırmaktadır.

Araştırmanın amacı

Bu tez çalışmasında, çeşitli uygulamalar arasında ağ kaynaklarının tahsisini iyileştirmek için bu teknolojileri birleştiren bir yaklaşım tasarlamak, geliştirmek ve değerlendirmek amaçlanmıştır. Bu yaklaşımda, çeşitli uygulamalardan gelen akışlar tanımlanarak hizmet kalitesini (quality of service, QoS) iyileştirmek için derin sinir ağı ve 1-B evrişimli sinir ağı modellerine dayalı trafik sınıflandırması yapılmaktadır. Ayrıca, bu yolla sınıflandırılmış akışlara, geliştirilen YTA kontrolcü modülüyle trafik şekillendirme uygulanmaktadır.

Araştırmanın önemi ve katkıları

Bu tez kapsamında,

- Trafik akışlarının zamana dayalı özelliklerini kullanan derin öğrenme modelleri uygulanarak trafiğin içeriğine bağımlı olmayan bir sınıflandırma yöntemi geliştirilmiştir.
- Sentetik Azınlık Aşırı Örnekleme Tekniği (Synthetic Minority Over-sampling Technique, SMOTE) kullanılarak dengesiz veri kümesi sorununun etkisi azaltılmıştır.
- Ağ kaynaklarının akıllı yönetimini sağlamak için derin öğrenme modellerini ve YTA'yı birleştiren bir yaklaşım önerilmiştir.
- Ağ cihazlarında her kuyruğun farklı bir önceliğe sahip olduğu birden fazla kuyruk kullanarak akışların hizmet kalitesinde iyileştirme yapılmıştır. Ayrıca, ağ bant genişliğini ve trafik hacmini yönetmek için trafik şekillendirme uygulanmıştır.
- Trafik sıkışıklığını azaltmak için trafik yükünü farklı fiziksel bağlantılara dağıtan bir yük dengeleme tekniği uygulanmıştır.

Sınırlılıklar

Tez çalışmasının kapsamını sınırlayan bazı hususular aşağıda sıralanmıştır:

- Derin öğrenme tabanlı modellerin eğitimi ve testi için literatürdeki açık bir veri kümesi kullanılmıştır. Bu veri kümesinin boyutunun küçük olması, model performansına kısıtlayıcı bir etki yapmıştır.
- Gerçekleştirim ve deneyler, zaman ve kolaylık açısından sanal ortamda yapılmıştır.
- Deneyler için kullanılan topolojilerın büyüklüğü ve trafik hacmi, donanım kaynakları ve kısıtları dikkate alınarak belirlenmiştir.

Bu tez çalışması şu şekilde düzenlenmiştir: 2. Bölümde yazılım tanımlı ağlar, trafik mühendisliği, hizmet kalitesi, makine öğrenimi ve derin öğrenme gibi çalışmadaki ana konularda genel bilgi verilmiştir. Ayrıca bu bölümde, yazılım tanımlı ağların mimarisi ve geleneksel ağlara göre avantajları, QoS için kullanılan araçlar ve mekanizmalar, bazı makine öğrenimi ve derin öğrenme algoritmaları ve derin öğrenme ile makine öğrenimi arasındaki

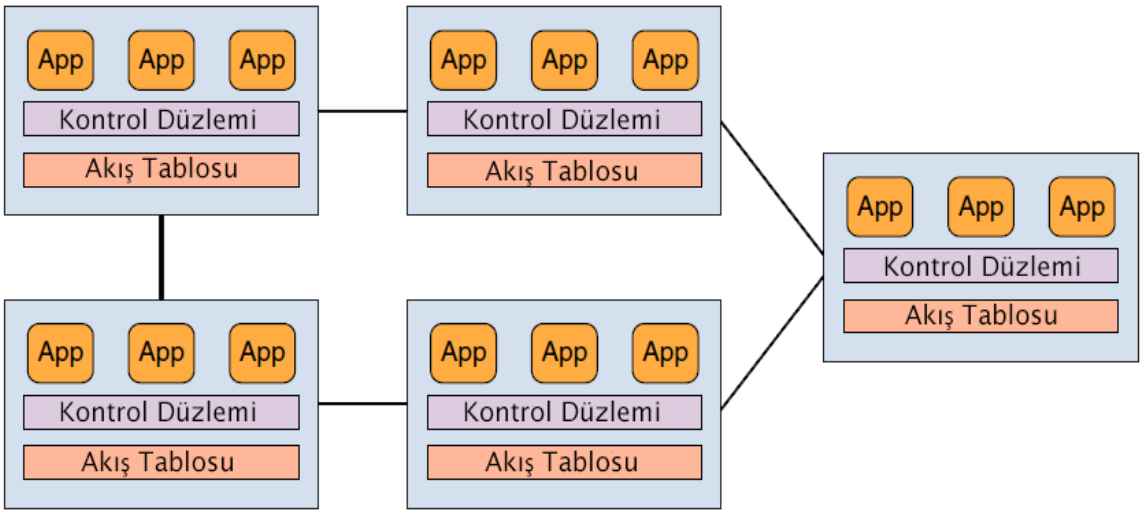
farklar incelenmiştir. 3. Bölümde yazılım tanımlı ağlarda hizmet kalitesi, makine öğrenmesine dayalı ve derin öğrenme tabanlı trafik sınıflandırması ile ilgili literatürde yapılan çalışmalar incelenmiştir. 4. Bölümde tez kapsamında önerilen yaklaşımın ana bileşenleri (derin öğrenme tabanlı sınıflandırıcı ve hizmet kalitesi odaklı trafik mühendisliği modülü) anlatılmıştır. 5. Bölümde, geliştirmede ve değerlendirmede kullanılan araçlar ve yöntemler hakkında bilgi verilmiş, derin öğrenme modellerinin ve trafik mühendisliği modülünün deneysel sonuçları sunulmuştur. Sonuç ve Öneriler bölümünde ise tez kapsamında yapılan çalışmalar özetlenerek tartışılmakta ve gelecekte yapılabilecek çalışmalara ilişkin fikir verilmiştir.

2. YAZILIM TANIMLI AĞLAR, İNTERNET TRAFİK MÜHENDİSLİĞİ VE DERİN ÖĞRENME

2.1. Yazılım Tanımlı Ağlar

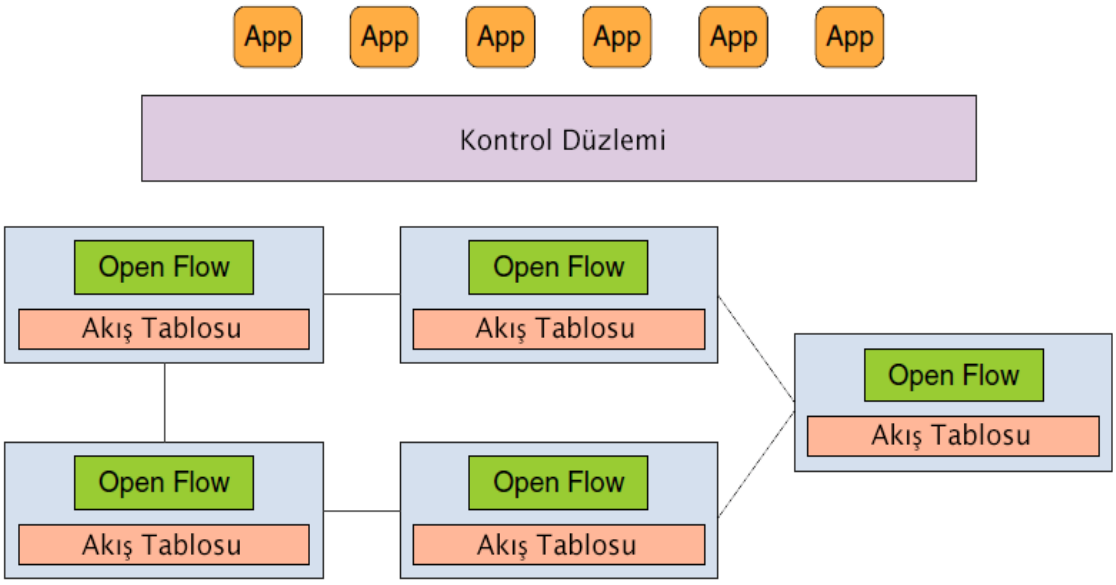
Yazılım Tanımlı Ağlar (YTA), veri düzlemi ile kontrol düzleminin ayrılmasına izin veren bir ağ mimarisidir ve veri düzlemindeki yönlendirme cihazları kontrol düzlemi tarafından yönetilmektedir. YTA, çeşitli uygulamaların veri düzleminde ağ donanımını kontrol etmesini ve yönetmesini sağlayan iyi tanımlanmış bir uygulama programlama arayüzü (Application Programming Interface, API) sağlamaktadır. Ayrıca, YTA mimarileri ortak ağ hizmetleri uygulamasına izin veren geniş yelpazede API'larınendüstriyel ihtiyaçları karşılamasını desteklemektedir. Örneğin; yönlendirme, çok yöne yayın, güvenlik, erişim denetimi, bant genişliği yönetimi, trafik mühendisliği, hizmet kalitesi, işlemci ve depolama optimizasyonu, enerji kullanımı ve her türlü politika yönetimi ve özel olarak tasarlanmış ortak ağ hizmetleri vardır. YTA, ağ zekası mantıksal olarak merkezileştirilmesine olanak tanır ve bu durum da ağın küresel bir görünümünü sağlar ve korur. Bu nedenle ağ, uygulamalara ve politika motorlarına tek bir büyük anahtar gibi görünmektedir. Başka bir deyişle, YTA ağ zekasını ağ donanımından etkili bir şekilde çıkarmış ve onu merkezi bir kontrolcünün içine yerleştirmiştir (Doherty, J., 2016).

Şekil 2.1'de geleneksel bir ağ için basit bir ağ örneği gösterilmiştir. Bu örnekte her bir cihaz bir kontrol düzlemi ve bir veri düzlemi görevi görür, her cihazın kendi uygulamaları vardır ve her birinin ayrı ayrı yapılandırılması gerekmektedir.



Şekil 2.1. Geleneksel ağ mimarisi

Şekil 2.2'de, zeka ve uygulamaların cihazlardan kaldırıldığı ve tüm cihazlar için merkezi bir kontrolcünün kontrol düzlemine dönüştüğü bir YTA ağını gösterilmektedir. Bütün ağ için trafik akışlarını programlamak amacıyla bir kontrolcü kullanılır. Böylece, trafik akışları bir akış tablosunu yöneten ve onu her cihaza dağıtan kontrolcünün kontrolü altındadır.



Şekil 2.2. Yazılım tanımlı ağlar

2.1.1. Yazılım tanımlı ağın avantajları

YTA'nin ana ilkesi kontrol ve veri düzlemlerini ayırma fikridir. Ağın mantığını ve ağ trafiğini kontrol eden katmanların ayrılması dikey entegrasyonu artırır. Kontrol ve veri düzlemlerinin bu ayrımı, ağ cihazlarını basit bir yönlendirme cihazı hâline getirir. Ağ yönetimi, ağ politikalarını uygulayan ve ağın yeniden yapılandırılmasını ve geliştirilmesini basitleştiren kontrolcüye bırakılmıştır.

Kontrol düzlemini ağ cihazlarından ayırıp harici bir parça hâline getirmenin bazı avantajları vardır. Bunlar şu şekilde özetlenebilir (Goransson, P., Black, C., and Culver, T., 2016):

- Kontrol düzlemi tarafından sağlanan soyutlama sayesinde ağ uygulamalarının programlanması daha kolay hâle gelmektedir.
- Kontrol düzlemi yazılım modülleri yeniden kullanılarak aynı ağ bilgisine sahip uygulamalar daha tutarlı ve daha etkili politika kararları verilmesi için temin edilmektedir.
- Bu kontrolcü uygulamaları, yönlendirme aygıtlarını ağın herhangi bir yerinden yeniden yapılandırabilmektedir. Böylece, ağa eklenecek yeni bir fonksiyonun konumu için kusursuz bir strateji tasarlamaya gerek kalmaz.
- Farklı türdeki uygulamaların entegrasyonu daha kolay hâle gelmektedir. Örneğin; yük dengeleme ve yönlendirme birleştirilebilir ve yük dengeleme kuralları yönlendirme politikalarına göre önceliklendirilmektedir.

2.1.2. Yazılım tanımlı ağ mimarisi

Yazılım tanımlı ağların temel konsepti düzlem ayrımıdır. Bu düzlemler, yazılım ve donanımın birleşmesi aracılığıyla oluşturulur. YTA mimarisi şunlardan oluşmaktadır:

i. Yönlendirme/Veri düzlemi
ii. Kontrol düzlemi
iii. Uygulama/Yönetim düzlemi

Veri düzlemi

Veri düzlemi (altyapı katmanı), kontrol düzlemi tarafından güzergâhlar belirlendiği anda verilerin aktarılmasıyla sorumludur. Veri düzlemindeki cihazlar kendi kendilerine herhangi bir veri oluşturmaz veya almaz. Ayrıca, kontrolcüle iletişim haricinde uç noktalar arasında iletişim için kullanılan protokolü bilmezler. Sadece yönlendirme cihazları görevi görürlmektedir. Veri düzlemindeki tüm ağ cihazları (fiziksel ve sanal cihazlar), kontrol düzlemi ile iletişim kurmak için bir güney yönlü arayüz API'yı desteklemelidir. Güney yönlü arayüzü API çoğunlukla OpenFlow'dur. Veri düzleminde 2 tip ağ cihazı vardır. Bunlar Open vSwitch, Pantou ve Indigo gibi Yazılım Tabanlı Anahtarlar ve Donanım Tabanlı Anahtarlardır (Huang, D., Chowdhary, A., and Pisharody, S., 2018).

Kontrol düzlemi

Kontrol düzlemi, trafik sinyallerinin taşınmasından ve yönlendirmeleri kurmak, iki cihaz arasındaki paketleri nakletmek, hizmet kalitesi, erişim kontrolü vb. görevlerden sorumludur. Kontrol düzlemi fonksiyonları ayrıca sistem yönetimi ve konfigürasyonu içermektedir. Ağ cihazları ve veri düzlemi için yönlendirme tablolarını kuran kontrol düzlemi, kontrol düzlemi tarafından yapılan yönlendirme tablolarını kullanarak yönlendirme kurallarını uygulmaktadır. Kontrol düzlemi ayrıca bir sunucu üzerindeki yazılımda da çalıştırılabilir. Böylece, ağ kaynaklarına ve yönetimine aktif erişim izni verir. Ayrıca, temel yönlendirme cihazlarını kontrol etmek için API'ları kullanan bir ya da daha fazla kontrolcüden oluşur. Kontrolcüler ayrıca yönlendirme kurallarını sanal anahtarlara iter ve ortamı gözler. Bu sayede kontrolcüler gerçek zamanlı trafik yönetimi ile entegre olmuş kararları iletme yeteneğine sahip olur. Kontrol düzlemi, üç iletişim arayüzü (güney, kuzey ve doğu/batı yönlü arayüzleri) aracılığıyla diğer düzlemlerle iletişim kurmaktadır (Huang, D., Chowdhary, A., and Pisharody, S., 2018) :

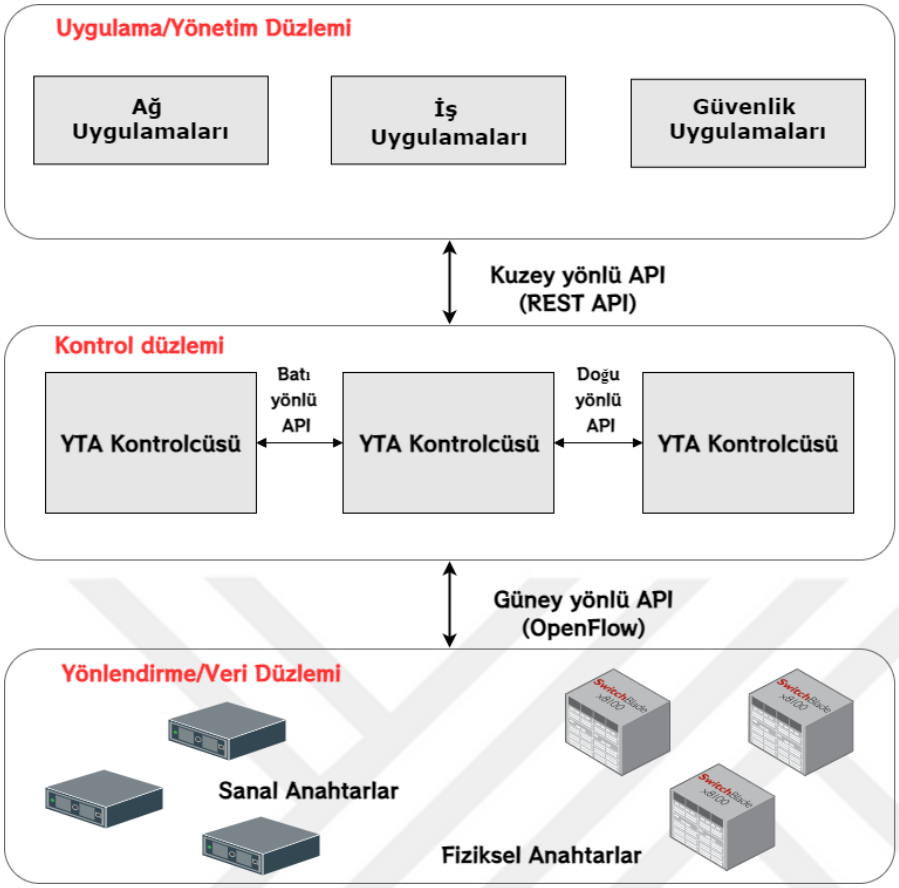
- Güney yönlü arayüzü (Southbound Interface): kontrolcünün yönlendirme cihazları ile iletişim kurmasına, etkileşimde olmasına ve onları yönetmesine olanak tanır. OpenFlow güney yönlü arayüzünün en yaygın uygulamalarından biridir.
- Kuzey yönlü arayüzü (Northbound Interface): uygulama katmanındaki uygulamaların kontrolcüleri programlamasına olanak tanır. Ayrıca, arayüz ağ cihazlarının içine doğru bir API olarak düşünülebilir.

- Doğu/Batı yönlü arayüzleri (Eastbound/Westbound Interface): kontrolcü grupları arasında iletişim kurmak amacıyla tasarlanmıştır.

Yönetim düzlemi

Yönetim düzlemi, ağ yönetiminden sorumludur ve politikaları bir ağda uygulmaktadır. Genellikle, yönetim düzlemi, kontrol düzleminin bir parçası olarak düşünülmektedir. Burada ağ yöneticileri, yönetim düzlemi ile iletişim kurmak için bir çeşit ağ yönetim sistemi kullanmaktadır. Bu düzlemde Basit Ağ Yönetim Protokolü (Simple Network Management Protocol, SNMP) cihazların işlemlerini ve performanslarını gözlemek için kullanılmaktadır. Ayrıca, iletişim uygulamaları, güvenlik uygulamaları, ağ hizmetleri ve yardımcı programları gibi diğer birçok ağ uygulamalarını da içermektedir (Mir, N. F., 2015).

Şekil 2.3'de Yazılım Tanımlı Ağların mimarisi göstermektedir. Veri düzlemi fiziksel anahtarlardan ve sanal anahtarlardan oluşur. Her iki durumda da anahtarlar paketlerin iletilmesinden sorumludur. Her anahtar, tek tip ve YTA kontrolcülerine açık olan bir paket yönlendirme modeli veya soyutlaması uygulamalıdır. Kontrol düzlemi, uygulama katmanı hizmet isteklerini veri düzlemi anahtarlarına yönelik belirli komutlar ve yönergelerle eşler ve uygulamalara veri düzlemi topolojisi ve etkinliği hakkında bilgi sağlamaktadır. Uygulama düzlemi, ağ kaynaklarını ve davranışını tanımlayan, izleyen ve kontrol eden uygulamalar ve hizmetler içermektedir.



Şekil 2.3. Yazılım tanımlı ağların mimarisi (Stallings, W., 2015)

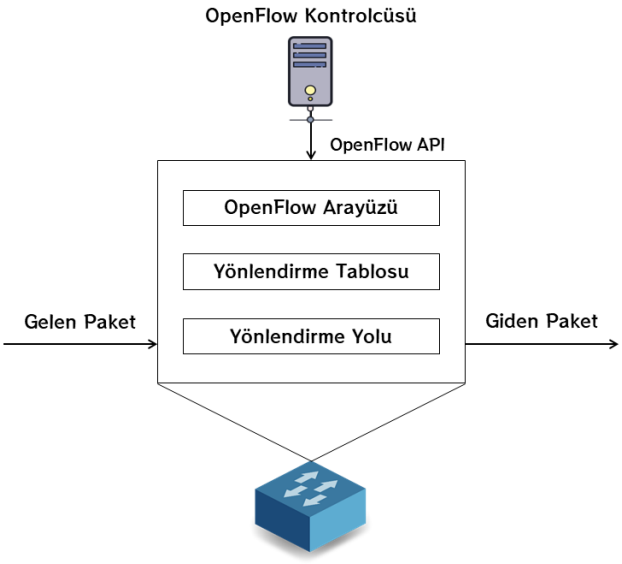
2.1.3. OpenFlow

OpenFlow, 2008 yılında Stanford Üniversitesinde ileri sürüldü ve bir araştırma projesi olarak başlanmıştır. İlk protokol tanımlaması 2009'da yayınlan başlanmıştır. Daha sonra, 2011 yılında Open Networking Foundation tarafından standartlaştırılmıştır. OpenFlow, aslında OpenFlow destekli cihazlar ile YTA mimarisindeki kontrolcü arasındaki iletişimi düzenlemek için geliştirilmiştir. Veri düzlemi ile kontrol düzlemi arasında tanımlanır ve temel olarak veri düzlemindeki cihazlar ile kontrol düzlemi arasındaki iletişimden sorumludur. Davranışı, cihazların farklı durumlarda nasıl tepki vermesi gerektiğini ve kontrolcüden gelen komutların nasıl yanıtlanacağını tanımlar. OpenFlow API'nın varlığı veri düzlemindeki yönlendirme cihazlarına doğrudan erişime izin vermek ve yönlendirme kararları vermek için gereklidir. Başka bir deyişle, satıcıların cihazlarının kaynak kodlarını gözler önüne sermelerine gerek kalmadan ağ cihazlarını kontrol eder. OpenFlow'un geniş ölçekteki en başarılı dağıtımlarından birisi 2013'teki Google B4'tür. Bu projede Google dünyanın her yerindeki veri merkezlerini birbirine bağlamak için Geniş Alan Ağında OpenFlow uygular (McKeown, 2008).

İki tip OpenFlow uyumlu anahtar vardır: yalnız OpenFlow anahtarlar ve OpenFlow işlemlerine ve geleneksel ağ işlemlerine destek sağlayan hibrit anahtarlardır. Genellikle, OpenFlow üç ana bileşenden oluşmaktadır: OpenFlow uyumlu anahtar, OpenFlow kanalı ve OpenFlow kontrolcüsüdür (Li, W., Meng, W., and Kwok, L. F., 2016).

- OpenFlow Uyumlu Anahtar: Bu cihazlar, OpenFlow protokolü kullanılarak güvenli bir kanal vasıtasıyla kontrolcü tarafından kontrol edilir ve yönetilir. Bu anahtarlar, paket aramasını ve trafik yönlendirmesini gerçekleştiren bir veya daha fazla akış tablosuna sahiptir. Akış tabloları, her girişin başlık alanları, sayaçları ve eylemleri varken girişlerin listesinden oluşur. Başlık alanları, paketleri eşleştirir ve kaynak ve hedef IP, ToS, VLAN ID, bağlantı noktası adresleri vb. gibi paket başlıkları hakkında veri içermektedir. Sayaçlar, bayt sayıları, paket sayıları vb. gibi paketler hakkındaki istatistiksel bilgileri muhafaza etmektdir. Son olarak, eylemler bir akıştaki paketlerin nasıl işleneceği ve eşleştirileceği hakkında bilgi içermektedir.

Şekil 2.4'de OpenFlow uyumlu anahtarlarda OpenFlow API göstermektedir.



Şekil 2.4. OpenFlow uyumlu anahtarlarda OpenFlow API

- OpenFlow kanal: OpenFlow tabanlı anahtarları kontrolcüye bağlayan bir arayüzdür. Kontrolcü, bu arayüzü kullanarak anahtarları yönetir ve yapılandırır. Ayrıca, olayları anahtarlardan alır ve paketleri anahtarların dışına yönlendirmektedir. Genellikle, bu

arayüz üzerinden gönderilen üç tip mesaj vardır. Bunlar kontrolcüden anahtara mesajlar, asenkron mesajlar ve simetrik mesajlardır. Kontrolcüden anahtara: bu tip mesajlar, anahtarın durumlarını doğrudan yönetmek ve incelemek için kullanılır. Asenkron mesajlar, kontrolcüsü ve anahtarın değişiklikleri ve olayları ile güncelleştirmektedir. Simetrik mesajlar, kontrolcü veya anahtar tarafından başlatılır ve başkalarından izin alınmadan gönderilmektedir.

- OpenFlow Kontrolcü: bir yazılım modülüdür. Ağın beyni olarak hizmet eden YTA ağlarının temel taşıdır. Tüm ağın bir görünümünü sağlar ve nakletme, yönlendirme, yönünü değiştirme, yük dengeleme vb. gibi politika kararlarını uygulanır; tüm yönlendirme cihazlarını kontrol eder ve uygulamalar için kuzey yönlü arayüzü API sağlanmaktadır. Ayrıca, güvenli kanal üzerinden anahtarlara akış girişleri yükleyerek ağdaki her akışı oluşturmaktan sorumludur. OpenFlow anahtarı, kontrolcülerden birinin yanıt vermeyi başaramaması durumunda ağ güvenilirliğini artırabilen kontrolcüden daha fazlasıyla iletişim kurabilir. Bir kontrolcüden daha fazlasına bağlı olan anahtarlar tüm kontrolcülere aynı anda bağlanmalıdır. Böylece, ilgili mesajlar uyumlu anahtara gönderilmektedir.

Çizelge 2.1'da en yaygın kullanılan kontrolcülerden bazıları gösterilmektedir.

Çizelge 2.1. Kontrolcü türleri

Kontrolcü	Dil	Açık Kaynak	Performans
NOX	C++	Evet	Yüksek
POX	Python	Evet	Düşük
Ryu	Python	Evet	Düşük
Floodlight	Java	Evet	Yüksek
Open Daylight	Java	Evet	Yüksek
Beacon	Java	Evet	Yüksek

Kontrolcünün modları

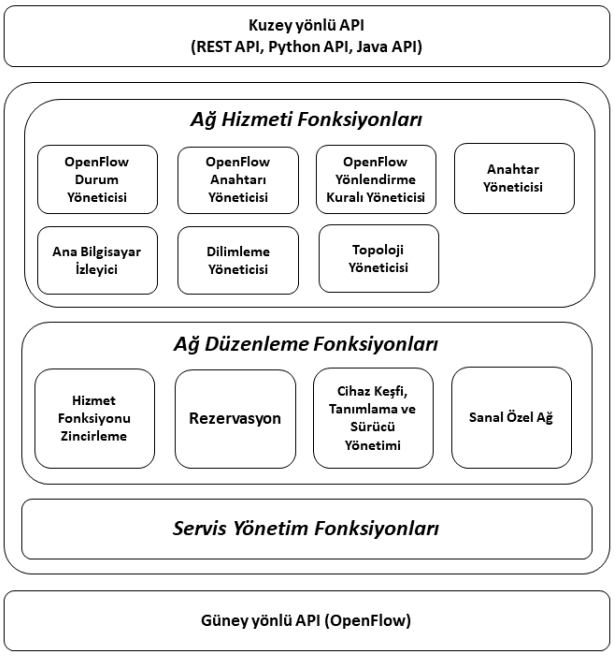
Kontrolcünün üç modu vardır: reaktif mod, proaktif mod ve hibrit mod (Karakus, M., and Durresi, A., 2017).

- Reaktif mod: Ağa yeni paket geldiğinde anahtar onu akış tablolarında arar. Bir eşleşme bulunmadıysa anahtar yeni gelen paketi işlemenin bir yolunu isteyerek onu kontrolcüye yönlendirir. Ardından paket ağ kurallarına göre kontrolcü tarafından işlenecektir. Bundan sonra bir akış girişi oluşturur ve bunu anahtara kurulmak üzere gönderilmektedir. Böylece, kurulu akış girişiyle eşleşen gelecek paketler uyumlu eşleştirme kuralına göre işlenecektir.

- Proaktif mod: bu modda kontrolcü herhangi bir işleme akış kuralı ile meşgul değildir çünkü ağa yeni gelen paketlerden dolayı anahtar daha önce kontrolcü tarafından kurulan akış girişlerine göre bu paketi nasıl işleyeceğini bilir.

- Hibrit mod: burada kontrolcü hem reaktif hem de proaktif modlar üzerinde çalışır. Ağ yöneticilerinin proaktif olarak ağ cihazlarına belirli akış girişlerini yüklemesi mümkündür. Kontrolcü reaktif olarak akış girişlerini değiştirebilir ya da gelen trafiğe göre yenisini ekleyebilir.

Şekil 2.5'de bir OpenFlow kontrolcüsü, hem kuzey hem de güney yönlü API olmak üzere kontrolcünün temel işlevselliğini sağlayan modülleri ve kontrolcüyi kullanabilecek birkaç örnek uygulamayı gösterilmektedir.



Şekil 2.5. OpenFlow kontrolcüsün modülleri (Goransson, P., Black, C., and Culver, T., 2016)

2.2. İnternet Trafik Mühendisliği

"İnternet trafik mühendisliği, internet ağ mühendisliğinin operasyonel IP ağlarının performans değerlendirmesi ve performans optimizasyonu meselesiyle ilgilenen yönü olarak tanımlanmaktadır. Trafik mühendisliği internet trafiğinin ölçümü, karakterizasyonu, modellenmesi ve kontrolüne teknoloji ve bilimsel ilkelerin uygulanmasını kapsamaktadır." (Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X., 2002)

Trafik mühendisliği tekniklerinden bazıları (Mendiola, A., Astorga, J., Jacob, E., and Higuero, M., 2016):

- Tıkanıklığın Azaltılma: Tıkanıklık, gecikmeyi, titremeyi ve paket kaybını etkilediği için en temel problemlerden birisi olarak görülmüştür. Bu yüzden, ağlardaki en önemli performans optimizasyon hedeflerinden birisidir. Tıkanıklık minimizasyonu farklı metotlar kullanılarak yapılabilir:

  - Ağ kaynaklarını birden çok trafik akışı ile paylaşma.
  - Altyapı üzerinden akışı tekrardan dağıtarak ağ kaynaklarını yeniden tahsis etme.
  - Sıkışık kaynaklara erişimi reddetme.

- Uçtan Uca Gecikme Minimizasyonu: Hizmet Kalitesini ve Deneyim Kalitesini etkiler. Uçtan Uca gecikmenin minimizasyonu gerçek zamanlı iletişim için çok büyük önem arz eder. Akış başına esasında ya da ağda iletilen tüm paketlerin Uçtan Uca gecikmesini dikkate alan genel bir hedef olarak uygulanabilir. Uçtan Uca gecikmeyi minimize etmedeki en yaygın tekniklerden birisi Uçtan Uca gecikmenin yol seçimi için bir sınır olarak kullanıldığı Kısıltanmış En Kısa Yol'dur (CSPF).

- Paket Kaybı Minimizasyonu: Akış başına ya da ağ çapında değerlendirebilir. Paket kaybı ayrıca ağdaki arızaların bir sonucu olabilir. Bu performans hedefi genellikle ağın arıza durumunda kullanılacak gereksiz kaynaklar vasıtasıyla esnekliği artırması için gereğinden fazla sağlama alınmasıdır.

- Enerji Tüketimi Minimizasyonu: ICT 'nin çevresel etkisini azaltmayı hedefler. Yeşil bilişimde yaygın olarak kullanılır. Bu performans hedefi tipik olarak ya ağ operasyon oranını sunulan iş yüküne uyarlayarak ya da aktif kaynakların miktarını azaltarak optimize edilir.

- Deneyim Kalitesi Minimizasyonu: bir ICT hizmeti veya ürünü kullanım performansını QoS gibi nesnel teknik parametreleri ve öznel psikolojik parametreleri dikkate alarak ölçen bir parametredir. QoE maksimizasyonu ağ performansının optimizasyonunu gerektirir. QoE'nin maksimize edilmesi her zaman ağ veriminin maksimizasyonunu göstermez ve QoE kriterleri ve ağ ile ilgili performans parametreleri arasında bir bağlantının tanımlanması gerekir.

- Kaynak Kullanım Optimizasyonu: Hesaplama, arabellek alanı ve bant genişliği, tıkanıklığı ve diğer parametreleri etkileyebileceklerinden dolayı etkili bir şekilde kullanılması gereken kaynaklardır. Ayrıca, kaynakların iyi bir şekilde kullanılması ağ operatörlerinin daha fazla sayıda hizmet talebini maliyetlerini artırmadan desteklemelerine yardımcı olur. Bu yüzden, ağ kaynaklarının iyi kullanımı ağ operatörlerinin daha yüksek miktarda trafik tahsis etmesine olanak sağlar.

2.3. IPv4 Paket Yapısı

IP protokolü, İnternet'in omurga protokolüdür ve TCP / IP protokollerinin ana protokolüdür. Bu protokol, OSI modelinin ağ katmanında bulunur. TCP, UDP ya da ICMP kullanılarak yapılacak veri iletiminde IP protokolü ile gerçek veri iletimi yapılır. Diğer protokollerde olduğu gibi her IP paketine (taşınan veriler dışında) farklı ayrıntılara sahip kontrol bilgileri eklenmiştir. Bu bilgiler paketin nasıl yorumlanacağına dair bilgilerden oluşmaktadır.

Şekil 2.6'de bir IPv4 paket başlık alanlarını göstermektedir.

Şekil 2.6. IPv4 paket başlığı

IPv4 paketinin başlığı hakkında ayrıntılar aşağıda listelenmektedir:

- Versiyon: Kullanılan internet protokolünün versiyon sayısı.
- IHL veya İnternet Başlığı Uzunluğu: Tüm IP başlığının boyutudur.
- Servis Tipi : 6 bit DS alanı ve 2 bit ECN alanı olmak üzere toplam 8 bitten oluşur.
- Toplam Uzunluk: IP paketinin toplam boyutu.
- Kimlik: Eğer IP paketi iletim sırasında parçalara ayrılırsa parçanın hangi IP paketine ait olduğunu belirtmek için bu alana her parça için aynı kimlik sayısı eklenir.
- Etiketler: Eğer IP paketi ağ kaynakları tarafından işlenmek için fazla büyükse bu bayraklara bakılarak IP paketinin bölünüp bölünemeyeceğine karar verilir.

- Parça Ofset: Parçalanmış IP paketindeki mevcut parçanın tam konumunu gösterir.
- Yaşam Süresi: Ağdaki döngüleri önlemek için her pakete bir TTL değeri eklenir ve bu şekilde ağa gönderilir. Bu değer, paketin kaç yönlendiriciyi geçebileceğini gösterir. Bu değer her yönlendirici için bir azaltılır ve bu değer sıfır olduğunda paket atılır.
- Protokol: Paketin hangi protokole (TCP, UDP, ICMP) ait olduğunu gösterir.
- Üstbilgi Sağlama Toplamı: Paketin herhangi bir hata olmadan alındığını kontrol etmek için kullanılır.
- Kaynak Adresi: Paketin göndericisinin 32 bitlik adresidir.
- Hedef Adresi: Paketin gönderildiği hedefin 32 bitlik adresidir.
- Seçenekler: Opsiyonel bir alandır ve IHL alanı 5'ten büyük olduğunda kullanılmaktadır. Bu alan, opsiyonel durumlarda güvenlik, rota kaydı ve zaman damgası vb.

8-bitlik ToS alanı hizmet kalitesi için kullanılır ancak genellikle servis sağlayıcıları tarafından göz ardı edilmektedir. Bu çalışma kapsamında, 8-bitlik ToS alanı IP paketlerini etiketlemek için kullanılmıştır..

2.4. Hizmet Kalitesi

Hizmet Kalitesi (Quality of Service, QoS) ağ altyapısı teknolojisidir. Farklı trafik akışlarına farklı öncelik seviyeleri tahsis etmek için bir takım araçlara güvenir ve yüksek kaliteli performans sağlamak için daha yüksek öncelikli trafik akışına özel kullanım sağlamak üzerine çalışır. Herhangi bir ağdaki herhangi bir yönetim politikasının temel bir öğesidir. Ağ yöneticilerinin ağ kaynaklarını etkin bir şekilde kontrol etmesine ve yönetmesine olanak sağlar ve son kullanıcı deneyimini maksimuma çıkarmak üzerine çalışır (Szigeti, T., Hattingh, C., Barton, R., and Briley Jr, K., 2013).

QoS araçları ve mekanizmalarına örnekler:

- Trafik sınıflandırma ve işaretleme herhangi bir başarılı QoS uygulamasının iki önemli fonksiyonu olarak görülür.
- Trafik kontrolü ve şekillendirmesi belirli bir uygulama, kullanıcı veya sınıf tarafından tüketilen bant genişliği miktarını sınırlayan araçlardır.

- Tıkanıklık yönetimi veya zamanlama, ağ trafiği mevcut kaynakları aştığında kullanılan araçlardır. Bu yüzden, ağ trafiği kaynakların kullanılabilirliğini beklemek için sıraya alınır.

Hizmet Kalitesinin üç uygulama modeli vardır (Karakus, M., and Durresi, A., 2017):

- Tümleşik Hizmetler (Integrated services, IntServ): bu model, gerçek zamanlı uygulamaların bir bant genişliği rezervasyonu ve VOIP gibi özel QoS işlemi yapması gerektiğinde uygulanmaktadır.
- Farklılaştırılmış Hizmetler (Differentiated services, DiffServ): bu model, ağ akışlarının farklı sınıfları özel QoS işlemi gerektirdiğinde uygulanmaktadır.
- En iyi çaba: Bu model, trafik akışları herhangi bir özel işlem gerektirdiğinde uygulanmaktadır.

2.4.1. Trafik sınıflandırma

Trafik sınıflandırma, ağ trafiğini özelliklerine ya da davranışlarına göre önceden belirlenmiş sınıf grubuna atama sorunlarıyla ilgilenen bir tekniktir. Trafik sınıflandırma, ağ yönetimi için kritik önem taşır ve QoS, akış planlama, ağ güvenliği, kaynak sağlama, performans gözlemi vb. gibi farklı ağ oluşturma alanlarında yaygın olarak kullanılır. Trafik sınıflandırmasında yaygın olarak kullanılan üç çözüm mevcuttur. Bunlar son kullanıcı uygulaması, trafik sınıflar ve uygulama protokolleridir. Ağ trafiği sınıflandırıldıktan sonra kurumsal ağ gereksinimlerine dayalı olarak QoS araçlarının herhangi bir kombinasyonu uygulanabilir (Robertazzi, T. G., and Shi, L., 2020).

En yaygın olarak kullanılan üç tip trafik sınıflandırma tekniği vardır:

- Port tabanlı
- Yük tabanlı
- İstatistik tabanlı

<u>Port tabanlı teknik</u>

En basit tekniklerden biri olarak görülmektedir. Çünkü uygulamaların port sayılarına haritasının yapılmasına bağlıdır. Ancak birçok uygulama gitgide artarak rastgele ve dinamik

port sayıları kullandığından bu teknik kusurlu hâle gelmiştir (Yan, J., and Yuan, J., 2018).

Yük tabanlı teknik veya derin paket inceleme

Bu teknik, yük tabanlı tekniğin (Deep packet inspection, DPI) getirdiği problemlerin üstesinden gelmek için sunulmuştur. Paket yüklerini denetlemeye ve onu bilinen protokoller imzasıyla karşılaştırmaya dayalı olarak çalışır. Bu teknik sınıflandırmayı doğru bir şekilde yerine getirebilir ancak bazı dezavantajları vardır. Şifrelenmiş trafik ile uğraşmak zor olabilir; yüksek hesaplama ve manuel imza koruması gerektirir (Yan, J., and Yuan, J., 2018).

İstatistik tabanlı teknik

Bu teknik, zaman serilerine ya da trafik akışlarının istatistiklerine dayanır. Şifrelenmiş ve şifrelenmemiş trafiğin üstesinden gelebilecek kabiliyettedir. Bu yaklaşım genellikle makine öğrenimi ve derin öğrenme algoritmalarını kullanır. Makine öğrenimi algoritmalarının performansı büyük ölçüde tasarlanmış özelliklere bağlıdır ve bu da genelleştirilebilirliklerini sınırlar. Ancak derin öğrenme, özellikleri eğitim vasıtasıyla otomatik olarak seçtiği için makine öğrenimi tarafından getirilen bu sınırlamaların üstesinden gelinmesine yardımcı oldu. Derin öğrenme algoritmaları, trafik sınıflandırması için onları çok istenen kılan çok karmaşık kalıpları da öğrenebilir (Rezaei, S., and Liu, X., 2019).

2.4.2. Trafik şekillendirme

Trafik şekillendirme, tıkanıklığı önlemek ve paket gecikmesini kontrol etmek için gelen akışların hızını ve yoğunluğunu kontrol ederek ağ kullanılabilir bant genişliğini yönetmek amacıyla kullanılan bir tekniktir. Trafik şekillendirme, ağın performansını optimize etmek, gecikmeyi düzeltmek ve bant genişliğinin kullanılabilirliğini artırmak üzerinde çalışmaktadır. Ayrıca, paket kaybına neden olan beklenmedik artan bant genişliği kullanımının önüne geçer (Stallings, W., 2015).

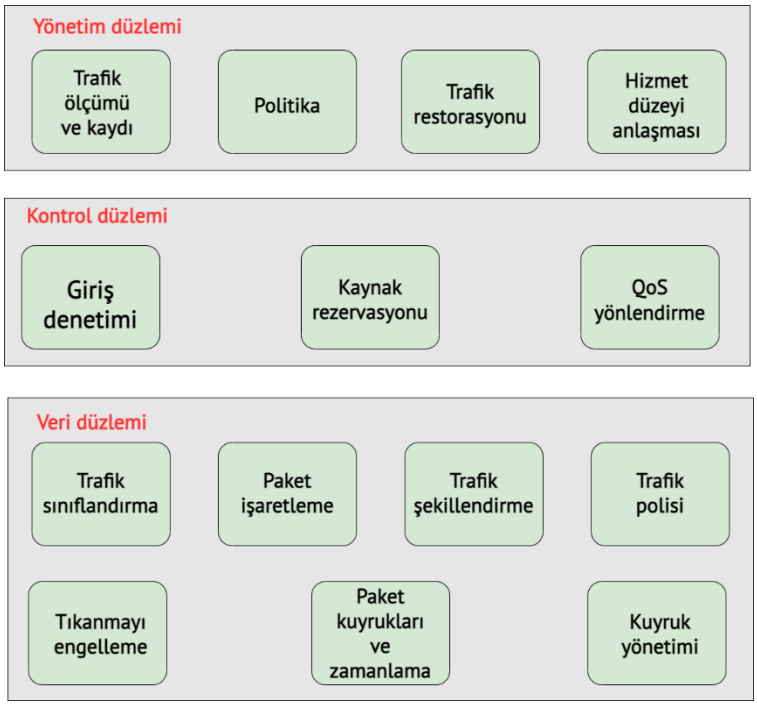
2.4.3. Paket kuyrukları ve zamanlama

Kuyruğa girme mekanizması iletilmek için sıralarını beklerken paketleri tutan kuyrukları yönetmek için kullanılan bir QoS araç setidir. Bir ağ cihazı bir paket aldığında bir

yönlendirme kararı verir ve ardından paketi dışarı giden arayüze gönderir. Ancak dışarı giden arayüz meşgul olduğunda dışarı giden iletiyi dışarı giden arayüzün hazır olmasını bekleyen kuyruklarda tutar. Kuyruğa girme sistemi, arayüz hazır olduğunda bir sonraki hangi paketin iletileceğine karar vermek için bir yönlendirme kararı verme üzerine çalışan bir programlama sistemine sahip olmayı gerektirir. Zamanlayıcı ayrıca bir kuyruğa öncelik verme kavramı olan kuyruk önceliklendirmesini yerine getirebilir. Bu yüzden, paketlerin yüksek önceliğe sahip olması onları kuyrukta öne koyar. Trafik şekillendirmeli kuyruk mekanizmasının Hizmet Kalitesi üzerinde etkisi vardır.

Zamanlama mekanizması o paketle ilgili QoS sağlamak için kuyruklardaki paketleri yönetme üzerine çalışır. Zamanlama mekanizması, yönlendirme kararları vermek için hangi sınıf akışının belirli bir gruba atanması gerektiğini belirlemek için bir haritalama tablosu kullanır. Her grup akışın uyumlu kuyruklarda nasıl işlenmesi gerektiğini belirlemektedir. Zamanlama özellikle gerçek zamanlı paketler ve gerçek zamanlı olmayan paketler bir arada gruplandığında birçok paket çeşidinin üstesinden gelinmek için gerekli görülür. Farklı çeşitlerdeki paketlerin eşit şekilde işlenmesini garantiye alır ve QoS sağlar (Stallings, W., 2015).

Şekil 2.7'de yazılım tanımlı ağlarda Hizmet Kalitesi desteğinin mimari çerçevesi gösterilmektedir.



Şekil 2.7. Yazılım tanımlı ağlarda hizmet kalitesi çerçevesi (Stallings, W., 2015)

2.5. Yük Dengeleme

Yük dengeleme, yükü aynı türden birçok kaynak arasında dağıtmak için kullanılan bir tekniktir. Doğru bir yük dengeleme tepki süresini minimuma indirme, ölçeklendirilebilirliği maksimuma çıkarma, kaynak tüketimini minimuma indirme, verimi maksimuma çıkarma, herhangi tek bir kaynağın aşırı yüklenmesinden kaçınılma, trafiği optimize etme vb. durumlara yardımcı olur. Yük dengelemenin uygulanması yazılım ya da fiziksel bir ekipman olarak olabilir ve statik, dinamik veya her ikisinin bir kombinasyonu gibi farklı metotlara sahiptir. İki tip yük dengeleme vardır. Bunlar IP ağında yük dengeleme ve YTA ağında yük dengelemedir (Neghabi, 2018).

2.6. Makine Öğrenimi

Makine öğrenimi bir bilgisayar bilimi alanı ve çözümleri otomatikleştirme yoluyla karmaşık problemleri çözmek için algoritmaları ve teknikleri çalışan bir yapay zekâ alt alanıdır. Makine öğrenimi büyük veri kümelerindeki kalıpların ve yapıların anlaşılmasını sağlama arayışındadır ve sonuçları veya davranışları tahmin etmek ya da önceden hesaplamak için var olan veri kümelerinden modeller oluşturmak için kullanılır (Rebala, G., 2019).
Birkaç tip makine öğrenimi mevcuttur:

- Gözetimli öğrenme: etiketli verilerden öğrenerek bir çıktıyı tahmin eder.
- Gözetimsiz öğrenme: verideki yapıları ve kalıpları bulmayı dener. Diğer bir deyişle veri noktalarını çeşitli türlerdeki kümeler hâlinde gruplara ayırmaya çalışır.
- Yarı gözetimli öğrenme: hem gözetimli hem de gözetimsiz öğrenimi birleştirir.
- Pekiştirmeli öğrenme: Kararlarına dayalı olarak geri bildirim alarak öğrenir ve hatalar açıkça düzeltilmez.

Makine öğrenimi sorunlarına örnekler:

- Sınıflandırma: Bir şeyi bir ya da daha fazla sınıfa ayırma vasıtasıyla problem çözmeye odaklanır. Örneğin; bir e-postanın spam olup olmadığını bulmak gibi.
- Kümeleme: Büyük bir veri noktaları setini kümelere ayırarak sorunu çözmeye odaklanır ve bu kümenin veri noktaları bazı ortak özellikler taşır.

- Tahmin: Tarihi verilere dayalı bir değeri tahmin ederek ya da önceden hesaplayarak problem çözmeye odaklanır.

Tez kapsamında kullanılan algoritmalar:

- K-en yakın komşu (K- Nearest Neighbor, KNN) algoritması, denetimli öğrenme yöntemlerinden biridir ve tüm makine öğrenme algoritmalarının en basitlerinden biri olarak kabul edilir. KNN, eğitim setini ezberleyerek çalışır ve eğitim setindeki en yakın komşularının etiketlerine göre yeni örneklerin etiketini tahmin etmeye çalışır. Numunelerin sınıflandırılması, etiketlenmemiş numune ile komşuları arasındaki mesafeyi belirlemek için uygulanır. Mesafeyi ölçmek için şehir bloğu, Öklid, Chebyshev ve Manhattan gibi mesafeler kullanılmaktadır.

- Destek vektör makineleri (Support Vector Machines, SVM) algoritması da denetimli öğrenme yöntemlerinden biridir, yaygın olarak nesneleri sınıflandırmak ve örüntü tanıma için kullanılır. SVM, giriş vektörünü yüksek boyutlu bir özellik kapsamına eşleyerek çalışır ve eşleme, farklı çekirdek işlevleri uygulanarak yapılır. Doğrusal, polinom ve radyal tabanlı işlev gibi çekirdek işlevi seçimi, SVM'de sınıflandırma sonucunun doğruluğunu etkileyebilecek önemli bir adım işi olarak kabul edilir. SVM'nin amacı, farklı sınıflar arasındaki marjı en üst düzeye çıkarmak için özellik kapsamında ayırıcı bir hiper düzlem bulmaktadır (Shalev-Shwartz, S., and Ben-David, S. , 2014).

- Karar ağacı (Decision Tree, DT) algoritması, bir öğrenme ağacına göre sınıflandırma yapmak için kullanılan denetimli bir öğrenme tekniğidir. DT, dahili karar düğümlerinden ve terminal yapraklarından oluşur. Ağaçtaki her düğüm bir özniteliği, dallar, sınıflandırmalara yol açan özniteliklerin bağlaçlarını belirtir ve her yaprak bir sınıf etiketidir. Etiketlenmemiş örnekler, karar ağacı düğümleriyle özellik değerine göre sınıflandırılır. DT'nin uygulanması kolaydır ve yüksek doğruluk elde etme yeteneğine sahiptir. En yaygın karar ağacı algoritmaları ID3, C4.5 ve CART'tır. Aralarındaki temel fark, bölme kriterleridir.

- Rastgele orman (Random Forest, RF) algoritması, sınıflandırma ve regresyon problemleri için kullanılan denetimli bir öğrenme tekniğidir. RF birçok karar ağacından oluşur. RF, aşırı uyumu en aza indirmek ve sınıflandırma doğruluğunu iyileştirmek için her bir karar ağacını oluşturmak için özellik kapsamının bir alt kümesini rastgele seçer. RF, örnekleri her ağaca koyarak çalışır, her ağaç ağacın oyu olan bir sonuç verir, daha sonra örnekler, en çok oyu alan sınıflara göre sınıflandırılmaktadır (Xie, J. et al., 2018).

2.7. Derin Öğrenme

Derin öğrenme yapay zekânın bir alt alanı ve makine öğreniminin belirli bir alt alanıdır. Makinelerin incelenmesi ve geliştirilmesine adanmıştır ve bilgisayarlı görü, doğal dil işleme ve otomatik konuşma metinleştirici gibi çeşitli alanlarda gerçek dünya görevlerini çözmek için kullanılmaktadır. Derin öğrenme, sinir ağı modelleri oluşturmaya odaklanır ve bu modeller veriden kalıpları belirleyip ayıklayarak veri güdümlü kararlar verebilir. Derin öğrenmenin karmaşık veri için ve büyük veri kümelerine sahip olunduğunda uygun olduğu düşünülmektedir (Kelleher, J. D., 2019).

Bazı derin öğrenme mimari örnekleri (Dargan S., 2019):

- Yapay Sinir Ağları (Artificial Neural Network, ANN): insan biyolojik nöronlarına dayalı olan karmaşık yapılardır. ANN'nin yapısı analitik olarak tanımlanamayan problemlere çözüm sağlar. Mimarisi, basit işleme birimleri olan nöronlardan ve bu nöronlar arasındaki ağırlaştırılmış bağlantılardan oluşur. ANN yapısı çok katmanlı algılayıcıya (Multilayer perceptron, MLP) benzemektedir. Karmaşık ve soyut veri problemlerini çözmek için kullanılmaktadır.

- Evrişimsel Sinir Ağları (Convolutional Neural Network, CNN): mimarisinin hayvan görsel korteksinden esinlendiği bir ağdır. Giriş katmanı, evrişimsel katman, havuz katman ve tamamen bağlı katman olan çoklu katmanlardan oluşur. Görüntü işleme, nesne algılama, el yazısı tanıma vb. alanlarda yaygın olarak kullanılmaktadır.

- Yinelenen Sinir Ağı (Recurrent Neural Network, RNN): sıralı bellek konseptine dayanan bir çeşit sinir ağıdır. RNN ile diğer modeller arasındaki ana saygı gizli katmanların çıktı bilgisini kendisine geri bildirmesidir. Sıralı verideki sonucu tahmin etmek için kullanılır. Yaygın olarak Doğal Dil İşleme ve konuşma işlemede kullanılmaktadır.

- Uzun Kısa Süreli Bellek (Long Short Term Memory, LSTM): RNN'lerin bir türüdür. Kaybolan ve patlayan gradyanlara karşı RNN hassaslığı probleminin üstesinden gelinmesi için tasarlandı. LSTM'nin mimarisi bilgiyi uzun bir süre hatırlamak için inşa edildi. Değerini yeterli süre tutabilen ve ona girdisinin bir işlevi olarak davranan bir bellek birimi kullanır. Böylece birimin önceden hesaplanmış değeri ezberlemesine yardım eder. Çoğunlukla konuşma metinleştirici uygulamalar için kullanılmaktadır.

- Üretken Çekişmeli Ağlar (Generative Adversarial Networks, GANs): eğitim veri setinde sunulmayan tamamen yeni ve farklı imgeler oluşturan özel bir ağ mimarisi çeşididir.

- Kısıtlı Boltzmann Makineleri (Restricted Boltzmann Machines, RBMs): bir giriş ile gizli katman arasında hiçbir bağlantının olmadığı bir modeldir. Gizli katmanın (görünür bir katman ve katmanlar arasındaki simetrik bağlantı) yönlü olmayan bir grafiği ve modellenmiş temsilidir. RBM'ler ANN'lerin stokastik modellerini oluşturmak için kullanılır. Girdileri ile ilgili olasılık dağılımını öğrenebilecek kabiliyettedir. Kısıtlı Boltzmann Makineleri uygulamaları özellik öğrenme, boyut indirgeme, işbirlikçi filtreleme, sınıflandırma ve konu modellemeleridir.

- Derin İnanç Ağı (Deep Belief Network, DBN): birden çok stokastik katmandan ve gizli değişkenlerden oluşur. Bayesçi olasılık üretken modelinin özel bir metodu olarak görülür. Eğitim metodunu her iki birbirine bağlı katmanın Kısıtlı Boltzmann Makinesi olduğu birçok gizli katmanla birleştirir. Ayrıca, Kısıtlı Boltzmann Makineleri yığını olarak da bilinmektedir. Derin İnanç Ağı etiketsiz verilere uygulandığında etkilidir.

- Otomatik Kodlayıcı: gözetimsiz öğrenmeye dayalı bir ağ türüdür. İlk olarak, ağ hedef sonuçların girdi değerlerine denk olmasını belirler ve birim fonksiyonuna denk bir yaklaşımı anlama arayışında olur. Otomatik kodlayıcının üç katmanı vardır. Bunlar giriş katmanı, kodlama katmanı ve kod çözme katmanıdır. Temel Bileşen Analizi ile yakından ilgilidir ancak temel bileşen analizinden (Principal Component Analysis, PCA) daha esnektir. Otomatik kodlayıcı, görüntülerin gürültüsünü azaltmak ve yapay veri örnekleri üretmek için kullanılır. RNN gibi diğer sinir ağları ile birleştirilebilmektedir.

Derin öğrenme ve makine öğrenimi arasında bulunan ana farklılıklardan bazıları şunlardır (Dargan S., 2019):

- Makine öğreniminde mevcut olan özellik mühendisliğinin karmaşık aşaması derin öğrenmede ortadan kaldırılır.
- Makine öğrenimi özellikleri kullanıcılar tarafından kesin bir şekilde belirlenirken derin öğrenme kendi işlemleriyle yeni özellikler yaratır.
- Derin öğrenmede büyük verileri işlemek için çok yüksek performanslı ve yüksek kaliteli GPU'lara sahip donanım gerekir.
- Makine öğrenimine kıyasla büyük miktarda verinin üstesinden gelebilir.
- Makine öğrenimi problemi daha büyük bir görevi daha küçük görevlere ayrıştırarak ve ardından sonuçları birleştirerek çözerken derin öğrenme problemi uçtan uca temellerinde çözebilir.
- Makine öğrenime kıyasla daha yüksek doğruluk oranı elde edebilir.
- Eğitim için makine öğrenimindekinden daha fazla zaman gerekir.
- Karmaşık ağ tasarımı ve hiperparametrelerden dolayı anlaşılması zordur.

3. LİTERATÜR TARAMASI

3.1. Yazılım Tanımlı Ağlarda Hizmet Kalitesi

OpenFlow uyumlu ağlarda hizmet kalitesinin ayrıntılı bir çalışması Karakus ve diğerleri (2017) tarafından sunulmuştur. Çalışmada, QoS'in YTA mimarisinde nasıl sağlanabileceği ele alınmış ve OpenFlow protokolünün QoS kabiliyetleri incelenmiştir. Çalışmanın göz önüne aldığı konular şunlardır: multimedya akışları yönlendirme, alanlar arası yönlendirme, kaynak ayırma protokolü, kuyruk yönetimi ve zamanlama, deneyim kalitesi farkındalığı, ağ izleme ve diğer QoS merkezli mekanizmalardır. Çalışma ayrıca YTA ağlarında daha iyi ve eksiksiz QoS kabiliyetleri sağlanması için aşılması gereken potansiyel zorluklara ve sorunlara değinmiştir.

Jeong ve diğerleri (2017) OpenFlow tabanlı ağlara uygulama farkındalığı sağlayan bir sistem geliştirmiştir. Sistem ile YTA'da trafik mühendisliği ve derin paket inceleme yönetilmiştir. Port sayısı ve derin paket inceleme tabanlı trafik sınıflandırması hizmet akışlarını belirlemek için kullanılmıştır. Sistemde belirlenen akışların hizmet kalitesini farklı önceliklere sahip çeşitli kuyruklara tahsis etme vasıtasıyla iyileştirmek amaçlanmıştır. Sistem, ağ yöneticilerinin belirlenen akış ve kuyruk önceliği arasında bir haritalama tablosu tanımlamasına olanak sağlamıştır. Kuyruklar paketleri çok yönlü yönlendirme kararı ve öncelikli kuyruk ataması içeren çeşitli kullanım yolları ile yönlendirmiştir. Sistemin fizibilitesini göstermek için bir kontrolcü uygulaması ve veri düzlemi varlıkları kurarak sistem tasarlanmış ve uygulanmıştır. Sonuçlar belirlenen akışların çıktılarında bir artış ve paket gecikmesinde azalma göstermiştir.

Li ve diğerleri (2017) meydana geldikten sonra trafik akışlarındaki sıkışıklığı tespit etmek ve kontrol etmek için YTA'ya dayalı CATS adlı bir trafik planlama algoritması önermiştir. Bir kümelenmiş fil akışı konsepti ve planlama için kümelenmiş akışı tespit edecek bir metot tanıtılmıştır. Çalışmada ayrıca bağlantı kullanımındaki varyansı en aza indirmek ve genel QoS'yi geliştirmek için bir trafik optimizasyon modeli ve optimizasyon problemini çözmek için kaos teorisine dayalı genetik algoritmanın kullanılmasını önerilmiştir. Sonuçlar, önerilen algoritmanın ağ tıkanıklıklarını verimli bir şekilde ortadan kaldırdığını ve daha düşük paket kaybı oranı ve dengeli bağlantı kullanımıyla QoS'yi geliştirdiğini göstermiştir.

Umadevi ve diğerleri (2017) tarafından YTA'ya dayalı çok seviyeli kuyruk metodu önerilmiştir. Önerilen sistem, birçok seviyeli anahtarlama kuyruğu oluşturmaya çalışarak gelen veri trafiğinin etkili bir şekilde üstesinden gelmeyi amaçlamıştır. Kuyruklar farklı öncelikler ile korunmuştur. Sonuçlar, önerilen yöntemin paket kaybında önemli bir azalma sağlayabildiğini göstermiştir.

Sun ve diğerleri (2018) tarafından YTA ve eski ağlara dayalı bir dinamik akış planlama sistemi önerilmiştir. Sistem bir bağlantı başladığında ve topoloji ve zamanda herhangi bir değişiklik olduğunda çalışmıştır. Çalışmada, atlama sayısını, bağlantı yükünü ve anahtar yükünü dikkate alarak bir ağırlıklı derecelendirme sistemi kullanılmıştır. Bir izleme modülü planlayıcının mevcut trafik akışı bilgisi hakkında sorgulama yapmasına ve bu bilgiyi trafik akışlarının yüklerini dinamik olarak dengelemek için kullanmasına izin vermek için geliştirilmiştir. Sonuçlar çıktıda yüzde 90'ın üstünde bir artış göstermiştir.

Xiao ve diğerleri (2017) veri merkezi ağlarındaki yük dengeleme sorunlarının üstesinden gelmek için çoklu trafik akışlarına sahip YTA tabanlı bir dinamik trafik planlama yaklaşımı geliştirmiştir. Akış dengesi, dinamik trafik planlamasının tetikleyicisi olarak tüm yol bant genişliği kullanımındaki değişikliklere göre tanımlanmıştır. Trafik aktarım işlemi boyunca akış monitörü etkinleştirilmiştir. Bu yüzden, genel akış dengesi seviyesi önceden tanımlanmış bir eşiğin altında olduğunda akış dinamik olarak alternatif bir rotaya değiştirilebilmiştir. Sonuçlar önerilen yaklaşımın dinamik yol değiştirme olmaksızın GLB ve ECMP algoritmalarına kıyasla daha iyi performans ürettiğini göstermiştir.

3.2. Makine Öğrenimi Tabanlı Trafik Sınıflandırma

Goli ve diğerleri (2018) port tabanlı, yük tabanlı ve makine öğrenimi tabanlı ve makine öğrenimi yaklaşımlarının uygulanması gibi ağ trafiği sınıflandırma teknikleri üzerine bir çalışma yürütmüştür. Çalışma trafik sınıflandırması için makine öğrenimi tekniklerinin uygulanması esnasında takip edilmesi gereken adımları ve trafik sınıflandırması için kullanılan farklı metrikleri tartışılmıştır. Ayrıca, yapılan bazı çalışmaları araştırarak trafik sınıflandırması için makine öğrenimi uygulamalarını incelemiştir.

Al Khater ve diğerleri (2018) (Goli ve diğerleri, 2018)'de bahsedilen trafik sınıflandırma tekniklerini araştırmıştır. Ek olarak istatistiksel sınıflandırma tekniğini (kullanılan

uygulamayı tanımlamak için trafik akışının istatistiksel özelliklerini kullanan bir tekniktir) incelemişlerdir. Paket süresi, paket arası varış zamanı, paket uzunlukları, trafik akışı boşta kalma süresi ve davranışsal teknik (hedef ana bilgisayardan üretilen trafik modellerini analiz ederek uygulama türünü belirlemeyi arayarak ana bilgisayar tarafından alınan tüm ağ trafiğini inceleyen bir tekniktir) gibi akış seviyesi ölçümlerinden yararlanmıştır. Bu teknik, sayılmış bağlı ana bilgisayarların sayısından yararlanmış, ayrıca bağlantı noktası sayısını ve taşıma katmanı protokolünü de dikkate alınmıştır. Ağ trafiğini istatistiksel ve davranışsal özelliklere dayalı olarak sınıflandırmak için makine öğrenimi uygulanmasının sebeplerini tartışılmıştır. Trafik sınıflandırma için ML tekniklerini uygulamanın bazı zorluklarına değinilmiştir. Trafik sınıflandırma için bir sistem önerilerek daha hızlı tanıma, zamanında tespit etme, tam trafik akışını bekleme ihtiyacını önleme ve bir tampon için gereken alanı minimize etme amacıyla tam akış yerine trafiğin alt akışının kullanılması tavsiye edilmiştir.

Archanaa ve diğerleri (2017) tarafından toplu öğrenme, karar ağacı ve trafik sınıflandırma için farklı denetimli makine öğrenme algoritmaları kullanımı hakkında ayrıntılı bir çalışma yapılmıştır. Özellik mühendisliği sarmalayıcı ve filtre metotları uygulanarak yapılmıştır. Algoritmaların performansını değerlendirmek için doğruluk, kesinlik, duyarlılık ve ROC alanı gibi metrikler kullanılmıştır. Deneysel sonuçlar topluluk sınıflandırıcıların diğer algoritmalardan daha iyi performans sağladığını göstermiştir.

Yan ve diğerleri (2018) trafik sınıflandırması için kullanılan tekniklerin bir incelemesini yapmış ve YTA üzerinde makine öğrenimine dayalı trafik sınıflandırmasının uygulanması ile ilgili çalışmaları incelemişlerdir. Çalışma, trafik sınıflandırması uygulanırken karşılaşılabilecek bazı zorlukları ana hatlarıyla belirlemiş ve bazı yaygın problemlerin üstesinden gelmek için tavsiyelerde bulunmuştur.

Amaral ve diğerleri (2016) tarafından topluluk öğrenmesi algoritmalarına dayalı trafik sınıflandırma sistemi geliştirilmiştir. Sistem YTA ağlarına dayalı bir kurumsal çevrede harekete geçirilmiş ve OpenFlow protokolü uygulayarak trafik akışlarını toplanmıştır. Sistem eski ağlara da uygulanabilmıştır. Ağ trafiğini sınıflandırmak için kullanılmış algoritmalar şunlardır: Rastgele Orman, Stokastik Gradyen Yükseltme ve Aşırı Gradyen Yükseltme. Veri, büyük ölçekli değerlere sahip özelliklerin kötü performansa neden olan daha küçük ölçekli olanlarla üst üste gelmesini önlemek için eğitimden önce standartlaştırılmıştır. Doğruluk metriği sadece sistem performansını değerlendirmek için

kullanılmıştır. Sonuçlar topluluk öğrenmesi algoritmalarının yüksek doğruluk seviyesine ulaştığını göstermiştir.

Fan ve diğerleri (2017) destek vektör makinesi ve k-means tabanlı trafik sınıflandırmasına odaklanmıştır. İki algoritma, akış parametrelerine göre trafik akışlarını çok çeşitli uygulama sınıflarına ayırmak için kullanılmıştır. Parametreler, segment boyutu ve paket arası varış zamanı gibi paket başlıklarından direkt olarak elde edilmiştir. SVM modelini eğitmek için sadece 13 özellik seçilmiştir. SVM ve k-means'in sınıflandırma doğruluğunun genel performansını değerlendirmek için bir karşılaştırma yönetilmiştir. SVM sonuçları, sınıflandırma modeli sekiz sınıfın büyük çoğunluğu için daha yüksek kesinlik elde ettiğini gösterilmiştir. Ayrıca, SVM modelinden elde edilen sonuçların k-means modelinden daha yüksek bir doğruluğa ulaştığı belirtilmiştir.

Lashkari ve diğerleri (2017) tarafından Tor trafiğini tespit etmek ve sınıflandırmak için trafik akışının zamana dayalı özelliklerine bağlı olan bir makine öğrenimi tabanlı trafik sınıflandırma sistemi önerilmiştir. Önerilen sistem, zaman dayalı istatistik özelliklerini sadece ağ akışlarını sınıflandırmak için kullanılmıştır. Özellikler trafiklerin gözlenmesinden türetilmiştir. Rastgele Orman, C4.5 ve KNN gibi farklı algoritmalar sınıflandırıcıyı yapmak için kullanılmıştır. Çalışma, zamana dayalı özelliklerin Tor trafiğini etkili bir şekilde tespit etmek ve ayrıca bu özelliklerin trafikleri karakterize etmek ve farklı türlerdeki trafik uygulamalarını tespit etmek için kullanılabileceğini göstermiştir. Ayrıca, sınıflandırma verimliliği üzerindeki etkisini araştırmak için farklı akış zaman aşımları incelenmiştir. Deneysel sonuçları daha kısa zaman aşımı değerleri kullanıldığında sınıflandırıcılar daha iyi performans sağladığını gösterilmiştir. Gerçekleştirilen deneylerde 15 saniyenin optimum uzunluk olduğu görülmüştür.

Mohammed ve diğerleri (2019) trafik ölçümü içim makine öğrenimi yaklaşımları ve YTA ağlarında trafik tahmini için derin öğrenme yaklaşımları üzerine yapılan araştırmaları incelemiştir. Çalışmada 15'ten fazla çalışma ayrıntılı olarak incelenmiş ve veri oranı, veri kümesi karakteristikleri, derin öğrenmeyi uygulama metodolojileri, YTA yapısı ile ilgili güvenlik sorunları ve akış şifreleme ile ilgili bazı sorunlara ve zorluklara değinmiştir.

3.3. Derin Öğrenme Tabanlı Trafik Sınıflandırması

Choubey ve diğerleri (2019) trafik akışlarının sınıflarını tahmin etmek için tamamen bağlı sinir ağları ve 1-B evrişimsel sinir ağlarına dayalı trafik sınıflandırma sistemi önermiştir. Deney için kullanılan veri kümesi 2005 yılında Andrew Moore tarafından toplanmıştır. Özellik mühendisliği, PCA algoritmasının uygulanması yolu ile yapılmıştır. Özellik sayısı 237'den 20'ye düşürülmüştür. Tamamen bağlı sinir ağları model mimarisi 1 giriş katmanı, 6 gizli katmandan oluşturulmuş ve çıktı katmanı her sınıf için bir tane olan 10 birim içermiştir. ReLU fonksiyonu gizli katmanlar için aktivasyon fonksiyonu olarak Sofmax ise çok sınıflı sınıflandırma sebebiyle çıktı katmanı için aktivasyon fonksiyonu olarak kullanılmıştır. 1-B evrişimsel sinir ağları tabanlı model mimarisi 2 evrişimsel katman, 2 maksimum havuz katmanı, 2 sinir ağı katmanı ve 10 birimlik çıktı katmanlarından oluşturulmuştur. Bu model için kullanılan aktivasyon fonksiyonları tamamen bağlı sinir ağları model aktivasyon fonksiyonları ile aynıdır. Sınıflandırma modellerinin performansı doğruluk, kesinlik, duyarlılık, f-puanı, değerlendirme zamanı ve eğitim zamanı gibi çeşitli metrikler kullanılarak değerlendirilmiştir. Ayrıca, derin öğrenme modellerinin sınıflandırma sonuçları ile makine öğrenimi modellerinin sınıflandırma sonuçları karşılaştırılmıştır. Sonuçlar, 1-B CNN modelinin uygulanan tüm modeller arasından en iyi sonucu verdiğini ve veride daha iyi örüntü tanıma ve eğitilebilir parametrelerin sayısının daha az olması sebebiyle aşırı uyma ihtimallerinin azaldığını göstermiştir.

Hartpence ve diğerleri (2019) sinir ağı yapılarını ve internet paketi ve trafik akışlarının sınıflandırılması için tasarımını inceleyen bir çalışma yaparak ağ trafiği için ön işleme teknikleri ve özellik seçimi üzerinde durmuştur. Çalışmada Choubey ve diğerleri (2019) tarafından kullanılan aynı veri kümesi kullanılmıştır. Modeli eğitmek için farklı sayıda özellikler seçilmiştir. Özellik sayıları 248'den 22 özellik setine düşürülmüştür. Özellik seçim işlemi boyunca PCA eğitim süresinde ya da sınıflandırma oranlarında otomatik bir gelişme sağlandığını fark edilmiştir. Ayrıca, her yapının etkisini incelemek için modeli eğitmek üzere farklı yapılar seçilmiştir. Trafik sınıflandırmasının değerlendirme sonuçlarına göre en iyi sonuçlar 150 özellik girişi, 128 parti boyutu ve aynı sayıda gizli düğümler ile çıktı sınıfı kullanıldığında elde edilmiştir. Genel trafik sınıflandırmasının sonucu %99, bireysel sınıfların sonucu ise %98.6'dır. Sinir ağı yapısının değiştirilmesi modelin tahmin doğruluğunu arttırmıştır.

Xu ve diğerleri (2018) tarafından sanallaştırılmış ağ fonksiyonuna (VNF) dağıtılan derin bir sinir ağına dayalı trafik sınıflandırma sistemi önerilmiştir. Sınıflandırma sisteminin VNF üzerinde dağıtılması kontrol kanalının bant genişliği meşguliyetini büyük miktarda veri ile çözmeyi amaçlanmıştır. Bant genişliği meşguliyetinin anahtar ve kontrolcü arasındaki iletişim üzerinde etkisi vardır. Ayrıca, VNF'nin başarısız olması ağ fonksiyonunu etkilenmemiştir. Önerilen sistem, farklı ağ kaynaklarına farklı uygulamalar atayarak QoS'yi geliştirmeye odaklanmıştır. Lashkari (2017)'de tanıtılan veri kümesini kullandılmıştır. Çalışma için zaman dayalı özellikler ve farklı akış zaman aşımı kullanılmıştır. DNN modelinin mimarisi bir giriş katmanı, üç gizli katman ve bir çıktı katmanından oluşturulmuştur. Bazı makine öğrenimi modelleri ile derin öğrenme modeli arasında bir karşılaştırma yapılmıştır. Sonuçlar, DNN modelinin 15 saniyelik akış zaman aşımı ile makine öğrenimi modellerine kıyasla en iyi kesinliğe sahip olduğunu göstermiştir.

Rezaei ve diğerleri (2019) derin öğrenmeye dayalı trafik sınıflandırma sistemi için genel bir çerçeve sunmuş ve veri toplama, temizlik, özellik seçimi ve model seçimi gibi sınıflandırma süreci için genel yönergeler sağlamıştır. Bu çalışmanın esas odak noktası şifreli trafiğin sınıflandırılmasıdır. Ayrıca, çalışma farklı derin öğrenme tekniklerini incelemiş ve trafik sınıflandırma için derin öğrenme metotlarının uygulanışını göstermiştir. Son olarak bazı açık problemleri, zorlukları ve trafik sınıflandırmanın gelecekteki yönlerini tartışmıştır.

4. ÖNERİLEN YÖNTEM

Bu tez çalışmasında, iki hedef üzerine odaklanılmıştır. Birincisi, trafik akışlarının sınıflandırılması için derin öğrenme tabanlı bir model geliştirilmesidir. İkincisi, trafik sınıflarına göre ağdaki trafik akışlarının hizmet kalitesi dikkate alanarak yönlendirilmesi için bir trafik mühendisliği modülünün YTA kontrolcü üzerinde gerçekleştirilmesidir. Bu bölümde öncelikle derin öğrenme sınıflandırıcısı ve trafik sınıflandırma sürecinde izlenen çerçeve hakkında bilgi verilmiştir. Sonrasında ise YTA'da uygulanan hizmet kalitesi odaklı trafik mühendisliği modülünün ana bileşenleri, OpenFlow ile paket yönlendirme yaklaşımı ve kullanılan OpenFlow uyumlu anahtar hakkında bilgi sağlanmıştır.

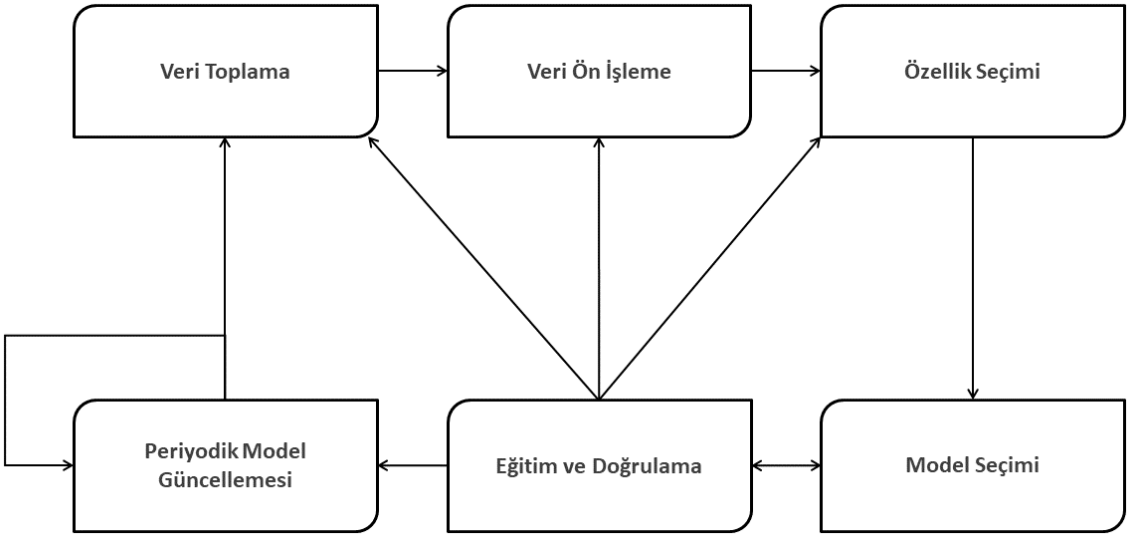
4.1. Derin Öğrenme Tabanlı Sınıflandırıcı

Derin öğrenme sınıflandırıcı, eğitim aşamalarına (bu aşamalar sınırsız sayıda girdi ve çıktı gerektirdiğinden) ve büyük miktarda bilgi işlem kaynağına yardımcı olmak için GPU gücü kullanılmıştır. Paketler önceden tanımlanmış sınıflara göre sınıflandırılmış ve bu sınıflar bir kuruluşden diğerine farklılık gösterebilmiştir. Önceden tanımlanmış sınıfların politikaları veya kuralları kuruluş ağ gereksinimlerine göre belirlenmiştir. Sonrası, sınıflandırılmış paketler IP paket başlığındaki Hizmet Türü (ToS) alanını değiştirme vasıtasıyla karşılık gelen sınıfın sonucu ile etiketlenmiştir.

4.1.1. Trafik sınıflandırma süreci

Trafik sınıflandırma sürecinde (Rezaei, S., and Liu, X., 2019) tarafından tanıtılan derin öğrenmeye dayalı trafik sınıflandırması için bir çerçeve takip edilmiştir. Çerçeve altı adımdan oluşmaktadır: veri toplama, ön işleme, özellik seçimi, model seçimi, eğitim ve tasdik ve periyodik model güncellemedir.

Şekil 4.1'de trafik sınıflandırması için kullanılan çerçeve gösterilmektedir.



Şekil 4.1. Trafik sınıflandırma çerçevesi

Veri toplama

Bir derin öğrenme modelinin eğitilmesi açısında büyük ve temsili veri kümeleri temel gereksinimlerdendir. Bir veri kümesinin toplanması sınıflandırma hedefine göre yapılmaktadır. Genellikle ağın merkezinde, istemci ya da sunucu tarafında, ağın kenarında veya aradaki herhangi bir yerde veri toplama toplanabilir. Veri kümesi toplama noktası mevcut özellikleri, güvenilir etiketlemeyi ve genellemeyi etkileyebilir.
Çevrimiçi olarak hazır birçok ağ trafiği veri kümesi vardır ancak bunların çoğu eskidir. Bu çalışma kapsamında kullanılan veri kümesi, (Lashkari, 2017) tarafından sağlanmış ve 2016'da toplanan bir Tor trafik veri kümesidir.

Veri kümesi 8 farklı tip trafik etiketi içermektedir:

- Ses Akışı: Bu etiket, Spotify gibi ses uygulamalarını tanımlar.
- Tarama: Bu etiket, Firefox ve Chrome gibi tarama uygulamalarını tanımlar.
- Sohbet: Bu etiket, ICQ, AIM, Skype, Facebook ve Hangouts gibi anlık mesajlaşma uygulamalarını tanımlar.
- Dosya Transferi: Bu etiket, Skype, SSH (SFTP) üzerinden FTP ve Filezilla ve harici bir hizmet kullanarak SSL üzerinden FTP (FTPS) gibi esas amacı dosyaları ve belgeleri göndermek ve almak olan trafik uygulamalarını tanımlar.

- E-Posta: Bu etiket, SMPTS, POP3S ve IMAPS gibi posta protokollerini tanımlar.
- P2P: Bu etiket, uTorrent ve Transmission (Bittorrent) gibi dosya paylaşım protokollerini tanımlar.
- Video Akışı: Bu etiket, Vimeo ve Youtube gibi video uygulamalarını tanımlar.
- VOIP: Bu etiket, Facebook Messenger, Skype ve Hangouts sesli aramaları gibi ses uygulamaları tanımlar.

Bu 8 trafik türü 18'den fazla temsili uygulamadan toplanmıştır ve 23 zamanla ilgili özellik içermektedir. Ayrıca, veri kümesi 5, 10, 15, 30, 60 ve 120 saniye farklı akış zaman aşımlarında yakalanan trafiği içerir. Farklı akış zaman aşımları kullanmanın amacı, akış zaman aşımının son sonuçlar üzerindeki etkisini incelemektir.

Çizelge 4.1'de veri setinde bulunan özelliklerin kısa bir açıklaması verilerek bunların çalışmanın amacına olan uygunluğu ortaya koyulmuştur.

Çizelge 4.1. Veri kümesi özelliklerinin açıklaması

Özelliğin ismi	Özellik türü	Açıklama
İleriye Varış Arası Zamanı (fiat) (ortalama, minimum, maksimum, standart sapma)	Gerçek	İleriye doğru gönderilen iki paket arasındaki zaman
Geriye Varış Arası Zamanı (biat) (ortalama, minimum, maksimum, standart sapma)	Gerçek	Geriye doğru gönderilen iki paket arasındaki zaman
Akış Arası Varış Zamanı (flowiat) (ortalama, minimum, maksimum, standart sapma)	Gerçek	Her iki yönde gönderilen iki paket arasındaki zaman
Aktif (ortalama, minimum, maksimum, standart sapma)	Gerçek	Boşta kalmadan önce akışın etkin olduğu süre
Boş (ortalama, minimum, maksimum, standart sapma)	Gerçek	Bir akışın aktif hâle gelmeden önce boşta kaldığı süre

Çizelge 4.1. (devam) Veri kümesi özelliklerinin açıklaması

Akış Baytları	Gerçek	Saniye başına akış baytları
Akış Paketleri	Gerçek	Saniye başına akış paketleri
Süre	Gerçek	Bir akışın süresi
Etiket (Hedef özellik)	Kategorik	Yukarıda bahsedilen 8 farklı etiket türünü içermektedir.

Veri ön işleme

Ön işleme, ham verilerin iyi biçimlendirilmiş bir veri kümesine dönüştürülmesini içeren bir süreçtir. Sınıflandırma performansını önemli derecede etkileyebilir. Ön işleme, veri temizleme, veri entegrasyonu, veri dönüşümü, veri azaltma ve veri ayrıştırmadan oluşmaktadır.

Eksik değerler veri temizleme aşamasında veri kümesinden çıkarılmıştır. Tüm özellikler sayısaldır, bu nedenle değerlerini yeniden ölçeklendirmekten başka dönüştürmeye gerek yoktur. Hedef özellik onun sayısal temsiline dönüştürülmüştür.

Benzer uygulama trafiğini tek bir sınıfta gruplamak, sınıflandırma sürecini daha basit hâle getirir. Bu yüzden 8 tür trafik, gecikme ve bant genişliğinin önemli kriterler olduğu yerlerdeki önceliğe dayanarak 3 farklı sınıfa gruplanmıştır. Çizelge 4.2'de yeni sınıflandırma kriteri göstermektedir.

Çizelge 4.2. Sınıflandırma kriteri

Sınıf	Uygulama	Öncelik
0	Sohbet, VOIP	Yüksek
1	Ses Akışı, Tarama, E-posta	Orta
2	Dosya Transferi, P2P, Video Akışı	Az

Gruplama işlemi tamamlandıktan sonra gruplandırılmış sınıfların dengesiz olduğu fark edilmiştir. Sınıf 1'deki örneklerin sayısı diğer iki sınıfa göre nispeten düşüktü. Bu problemi çözmek için Sentetik Azınlık Aşırı Örnekleme Tekniği (Synthetic Minority Over-sampling Technique, SMOTE) denilen bir aşırı örnekleme metodu sınıfları dengelemek için kullanılmıştır. SMOTE örneklerin yerini değiştirme vasıtasıyla aşırı örnekleme yapmak

yerine azınlık sınıfının sentetik örneklerini yaratmaya dayalı bir yöntemdir. Ayrıca, bir eğitim sürecine başlamadan önce modelin performansını geliştirmek için veriler ölçeklendirilmiştir.

Normalizasyon: Tüm özelliklerin değerleri 0 ile 1 arasında yeniden ölçeklendirilir. Bu, sinir ağları için kullanılan en yaygın ölçeklendirme yöntemidir. Normalizasyon yöntemi Eş. 4.1'de gösterilmiştir.

$$x_{norm} = \frac{x_i \_ min(x)}{max(x) - min(x)} \tag{4.1}$$

Standardizasyon: Tüm özelliklerin tüm değerlerini ortalamaları 0 ve standart sapmaları 1'e eşit olacak şekilde yeniden ölçeklendirir. Standardizasyon yöntemi Eş. 4.2'de gösterilmiştir.

$$x_{standardized} = \frac{x_i \_\ mean(x)}{std(x)} \tag{4.2}$$

Sentetik Azınlık Yüksek Hızla Örnekleme Tekniği

SMOTE, azınlık sınıfı için onları değiştirmek yerine sentetik örnekler oluşturmak için kullanılan bir yüksek hızda örnekleme tekniğidir. İlk olarak, el yazısı karakter tanımada kullanılan bir teknikten esinlenmiştir. Bu teknik, özellik alanında çalışarak daha az uygulamaya özgü bir şekilde sentetik numuneler üretmeye çalışır.

Azınlık sınıfı, her bir azınlık sınıfı örnek alınarak yüksek hızda örneklenir, ardından en yakın komşularından herhangi birini veya tümünü k azınlık sınıfına bağlayan çizgi segmentleri boyunca sentetik örnekleri sunar. En yakın k komşu, gerekli örnek sayısına göre rastgele seçilir.

Sentetik örnekler, dikkate alınan özellik vektörü ile en yakın komşusu arasındaki fark alınarak, farkın rastgele bir sayı 0 veya 1 ile çarpılması ve dikkate alınan özellik vektörüne eklenmesi ile oluşturulur. Bu, iki belirli özellik arasındaki çizgi parçası boyunca rastgele bir nokta seçmeye götürür. Sentetik numuneler, sınıflandırıcının küçük ve daha spesifik bölgeler yerine büyük ve daha az spesifik karar bölgeleri oluşturmasını sağlar (Chawla, 2002).

Özellik seçimi

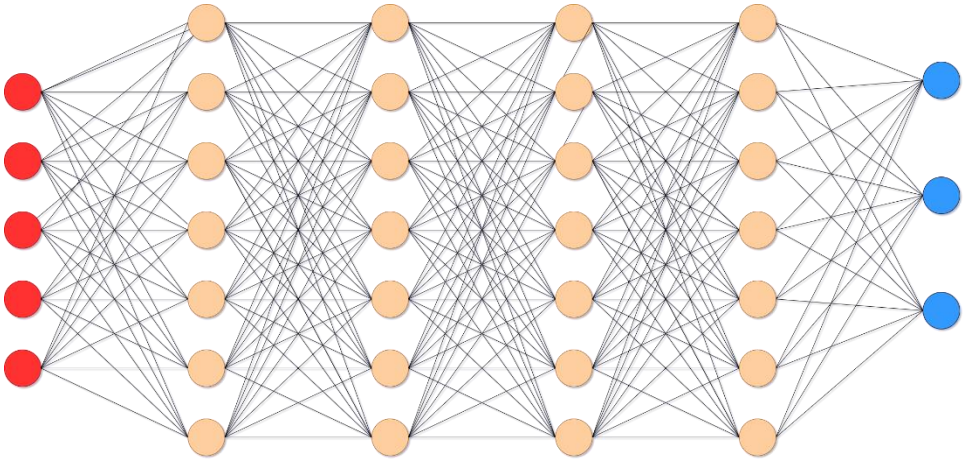
Özellik seçimi, ilgili olmayan ya da gereksiz özellikleri belirleyip onları kaldırma işlemidir. Gereksiz özelliklerin model performansı üzerinde etkisi olabilir. Zaman serisi, başlık verisi, yük verisi ve istatistiksel özellikler trafik sınıflandırmasında en yaygın kullanılan özelliklerdir.
23 özelliğin tümünün çalışmanın amacı ile ilgili olduğu düşünülmüştür. Daha önce de belirtildiği gibi bu özellikler istatistiksel özelliklerdir. İstatistiksel özellikleri kullanmanın avantajları şunlardır:

- Zaman serilerine veya trafik akışlarının istatistiklerine dayandığından hem şifrelenmiş hem de şifrelenmemiş trafiğin üstesinden gelebilir.
- Hesaplama karmaşıklığı düşüktür.

Model seçimi

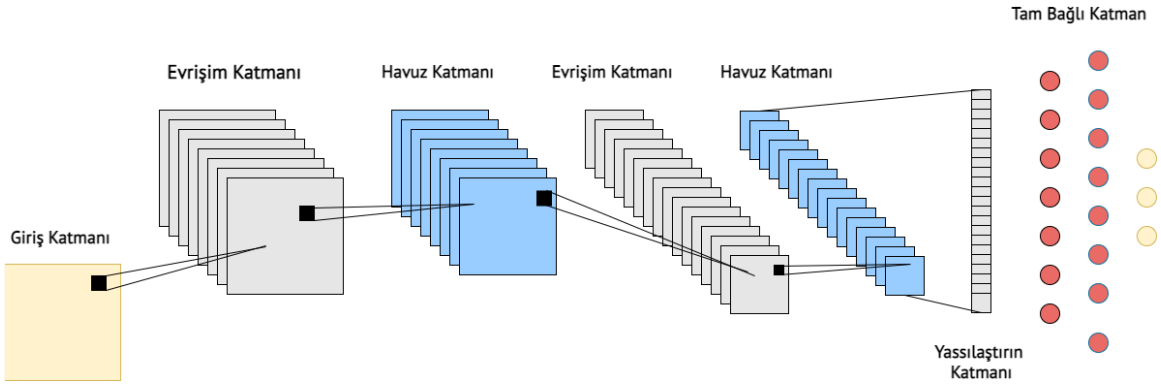
Birtakım koşullar trafik sınıflandırması için derin öğrenme modellerinin seçimini etkileyecektir. Önemli bir etken girdi özellikleridir ve bunlar sadece modelin doğruluğunu etkilemez, ayrıca hesaplama karmaşıklığı ve sınıflandırma için paket sayısını etkileyen girdi boyutunu da etkiler. Ayrıca, veri kümesinin boyutu model seçimini etkileyecektir. Derin öğrenme modelleri, daha büyük veri kümeleri için daha uygun olarak düşünülmüştür. Çalışma kapsamında seçilen modeller şunlardır: Yapay Sinir Ağları, Evrişimli Sinir Ağları.

- Yapay sinir ağları (Artificial Neural Network, ANN), çok katmanlı algılayıcılar olarak da bilinir ve birden çok algılayıcıdan oluşan bir koleksiyondur. Ağlar bir ya da daha fazla katmana sahip olabilirler. Dörtten fazla katmana sahip olan ağlar derin sinir ağları olarak düşünülmektedir ve yaygın olarak karmaşık ve soyut veri problemlerini çözmek için kullanılmaktadır.



Şekil 4.2. Derin sinir ağ mimarisi

- Evrişimsel sinir ağları (Convolutional Neural Network, CNN), bilgisayarlı görü alanında yaygın olarak kullanılmaktadır ve insan yeteneğini aşan yüksek seviyelerde doğruluğa ulaşabilme kabiliyetine sahiptir. Popülerliğinin artmasının sebebi budur. CNN' verilerin çeşitli yönlerini tanımak amacıyla farklı nöron gruplarını kullanan modeller yaratma arayışındadırlar. Bu grupların büyük resmi oluşturabilmeleri için birbirleriyle iletişim kurmaları gerekmektedir.



Şekil 4.3. Evrişimsel sinir ağı mimarisi

Eğitim ve doğrulama

Bu adımda, derin öğrenme modelinin hiperparametreleri en iyi doğruluğu elde etmek için ayarlanır. Genellikle, bir veri kümesi üç sete ayrılır. Bunlar eğitim, tasdik ve test setleridir. Bir model, eğitim setinde eğitilir ve tasdik setinin doğruluğu model hiperparametrelerini ayarlamak için kullanılır. Son olarak, tarafsız doğruluk elde etmek için test seti kullanılmaktadır.

Çalışma kapsamında modelleri değerlendirmek için kullanılan metrikler şunlardır:

- Doğruluk (Accuracy): modelin doğru olduğu tahminlerin bir parçasıdır. Doğruluk ölçüsü Eş. 4.3'te gösterilmiştir.

$$Doğruluk = \frac{Doğru\ tahmin\ sayısı}{Toplam\ tahmin\ sayısı} \quad (4.3)$$

- Karışıklık matrisi (Confusion matrix), sınıflandırıcının performansını özetleyen bir yöntemdir.

Çizelge 4.3. Karışıklık matrisi

		Gerçek	
		Pozitif	Negatif
Tahmin edilen	Pozitif	Doğru Pozitif (TP)	Yanlış Pozitif (FP)
	Negatif	Yanlış Negatif (FN)	Doğru Negatif (TN)

  - Doğru Pozitif (TP): Bu, aslında pozitif olan ve pozitif olarak tahmin edilen sonuçların sayısıdır.
  - Yanlış Pozitif (FP): Bu, aslında negatif olan ancak pozitif olarak tahmin edilen sonuçların sayısıdır.
  - Yanlış Negatif (FN): Bu, aslında pozitif olan ancak negatif olarak tahmin edilen sonuçların sayısıdır.
  - Doğru Negatif (TN): Bu, aslında negatif olan ve negatif olarak tahmin edilen sonuçların sayısıdır.

Amaç, Doğru Negatif ve Doğru Pozitif'teki değerleri maksimuma çıkarmak ve Yanlış Negatif ve Yanlış Pozitif'teki değerleri minimuma indirmektir.

- Duyarlılık (Sensitivity veya Recall), temel gerçekteki tüm pozitifler tarafından bölünmüş doğru pozitiflerin sayısıdır. Duyarlılık, gerçek değer pozitif iken tahminin ne sıklıkla doğru olduğunu ifade etmektedir. Duyarlılık ölçüsü Eş. 4.4'te gösterilmiştir.

$$Duyarlılık = \frac{TP}{TP+FN} \quad (4.4)$$

- Kesinlik (Precision), model tarafından tahmin edilmiş doğru pozitif ve yanlış pozitiflerin toplam sayısı tarafından bölünmüş olan doğru pozitiflerin sayısıdır. Kesinlik, tahmin edilen değer pozitif iken ne sıklıkla doğru olduğumuzu ifade etmektedir. Kesinlik ölçüsü Eş. 4.5'te gösterilmiştir.

$$Kesinlik = \frac{TP}{TP+FP} \quad (4.5)$$

- F-puanı (F-score) bir testin doğruluğunun ölçüsüdür. F-puanı ölçüsü Eş. 4.6'da gösterilmiştir.

$$F1 = 2 \times \frac{Kesinlik \times Duyarlılık}{Kesinlik + Duyarlılık} \quad (4.6)$$

Çapraz doğrulama

K-kat çapraz doğrulama olarak da bilinen çapraz doğrulama, eğitim ve test için veri kümesindeki örnek sayısı sınırlı olduğunda sıklıkla kullanılan eğitim ve test için alternatif bir yöntemdir. Veri seti N örnekten oluşuyorsa, bunlar k eşit parçaya bölünür ve k tipik olarak 5 veya 10 gibi küçük bir sayıdır. Eğitim ve test k kez gerçekleştirilir. Sınıflandırma problemi için, doğruluk tahmini, k yinelemelerinden gelen doğru sınıflandırmaların toplam sayısının orijinal verilerdeki toplam tuple sayısına bölünmesiyle hesaplanır. Sınıflandırma problemi için, doğruluk tahmini, k yinelemelerinden gelen doğru sınıflandırmaların toplam sayısının orijinal verilerdeki toplam tuple sayısına bölünmesiyle hesaplanır. Çapraz doğrulama, hiperparametrelerin ayarlanmasına yardımcı olmak için düzenleme ile birlikte kullanılır (Bramer M., 2020).

Periyodik model güncellemesi

Giriş veri dağıtımındaki değişimler ve veri hacmindeki artışlar sebebiyle modellerin kalitesi zamanla indirgenebilir. Bundan dolayı modelin kalitesini devam ettirmek ve sınıflandırma sürecini geliştirmek için modelin periyodik güncellemelere ihtiyacı vardır.

4.2. Hizmet Kalitesi Odaklı Trafik Mühendisliği Modülü

Sistem tasarımındaki hizmet kalitesi modülü üç ana bileşenden oluşmaktadır: kuyruğa ekleme, paket zamanlayıcısı ve trafik şekillendirme. Kuyruğa ekleme bileşeni OpenFlow akış tablosu mesajlarının yönetiminden ve akışları uyumlu kuyruklara eşleştirmekten sorumludur, ayrıca kontrolcünün içinde konumlanmıştır. Paket zamanlayıcısı kuyruklardaki paketleri yönetmekten, trafik şekillendirme ise kuyruklardaki bant genişliği hacmini yönetmekten sorumludur ve bu iki bileşen de anahtarın içinde konumlanmıştır. OpenFlow mevcut tanımlaması, kuyruk konfigürasyonu için destek sağlamamaktadır. Bu nedenle kuyruk konfigürasyonu, belirli OF-CONFIG ve OpenFlow vSwitch Veritabanı Yönetim Protokolü (OVSDB) protokolleri tarafından ele alınmaktadır. OVSDB protokolünün zaten Open vSwitch'te uygulanmış olmasına rağmen kuyrukların tutarlı bir yönetimini sunan hiçbir kontrolcü yoktur. Bu, kuyruk kurulum işleminin ve yapılandırmanın anahtarın içinde yapıldığı anlamına gelmektedir.

4.2.1. OpenFlow ile Paket Yönlendirme Yaklaşımı

Genellikle, OpenFlow ağları reaktif modda çalışır, yani bir paket OpenFlow anahtarına ulaştığında anahtar onu akış tablolarında aramaktadır. OpenFlow anahtarı, her akış tablosu birden çok akış girişi içeren bir veya daha fazla akış tablosu içerir. Bir OpenFlow anahtarının bir veya daha fazla sayıda akış tablosuna sahip olması gerekir. İlk akış tablosunda, paket ilk önce akış tablosunun akış girişleriyle eşleştirilmelidir. Trafik sınıflandırma için paket başlığındaki Hizmet Tipi (ToS) alanı gelen paket başlığıyla eşleştirmek için kullanılmaktadır. Bir akış girdisi bulunursa, bu akış girdisine ait olan komut seti yürütülür. Aksi takdirde, herhangi bir eşleşme bulunamazsa, yeni gelen paketin nasıl işleneceğini sormak için OpenFlow kanalı üzerinden kontrolcüye eşleşmeyen paket gönderilir. Bir OpenFlow kontrolcüsü, bir OpenFlow anahtarını bir veya daha fazla ağ üzerinden uzaktan yönetebilir.

Genellikle, her paketle bir eylem seti ilişkilendirilir. Kontrolcü paketi aldıktan sonra, bu akışla ilgili uygulanan herhangi bir politika veya kural olup olmadığını kontrol edecektir. Kontrolcüde bilgi yoksa paket atılır. Kontrolcü tarafından alınan kararlar anahtarlara gönderilir ve anahtar bunları yönlendirme tablolarına yükler, böylece aynı akışa ait olan sonraki paketler aynı şekilde ele alınmaktadır. Eylem setindeki çıktı eylemi en son yürütülür.

Kuyruk belirleme eylemi, bir paket için kuyruk kimliğini kurmaktadır. Paket, çıkış eylemi kullanılarak bir bağlantı noktasına iletildiğinde, kuyruk kimliği, paketi planlamak ve iletmek için bu bağlantı noktasına bağlı olan kuyruğun kullanılacağını belirler. Paket, bu kuyruk konfigürasyonuna göre işlenir. Yönlendirme davranışı, kuyruğun konfigürasyonu tarafından belirlenir.

OpenFlow kuyruk konfigürasyonu için destek sağlamadığından kurulum ve konfigürasyon anahtarda yapılır. Ayrıca, anahtar bir paket zamanlayıcısı türü belirlemeye ve akışları önceliklendirmeye izin verir. Paket zamanlayıcısı, kuyruklar için paketleri ekleme ve çıkarma üzerine çalışır ve akışın farklı bir şekilde ele alınmasını sağlayan ve gerekli iletişimi garanti eden her kuyruk için farklı öncelik değeri ayarlayarak akışların önceliklendirilmesine izin verir. Bu yüzden önce düşük öncelikli akışlar kurulur, sonra diğer akışlar kurulur. Bu çalışma kapsamında HTB paket zamanlayıcısı, her kuyruğa arzu edilen hızın karşılanması amacıyla paketleri geciktirmek için kullanılır ve minimum ve maksimum bant genişliği kapasitesinin yapılandırılmasına olanak sağlar. Bant genişliği kapasitesini yapılandırarak her uygulama türü için minimum bant genişliği garanti etmek ve bant genişliği açlığının önlenmesi amacıyla her kuyruk için bant genişliği tahsis edilmesine izin verir.

Akışlar ilgili kuyrukta sıraya dizildikten ve paket programlayıcı tarafından ele alındıktan sonra, trafik şekillendirme, tıkanıklığı önlemek için gelen akışların hızını ve hacmini kontrol ederek ağ bant genişliğini yönetmek için kullanılır. Trafik şekillendirme uygulamak, ağın performansını, bant genişliği kullanılabilirliğini artırır ve bant genişliği kullanımında beklenmedik bir artışı önlemektedir.

Kontrolcü bir yönlendirme kararı vermek için kaynak IP ve hedef IP'yi arar. Bu yüzden genel görünümdeki kontrolcü kaynak ve hedef arasında birden fazla rota hesaplar. Genellikle en az atlama sayısına sahip rotayı seçer ancak yoğun akışları iletmek için aynı rotayı seçmesi de mümkündür ve bu da bir tıkanıklığa neden olabilir. Aynı sayıda atlama sayısına sahip rotalar için port tabanlı algoritmalar, genel yük dengelemeyi yerine getirmek için her bağlantı noktasının kapasitesini göz önünde bulundurur. Trafik sıkışıklığını azaltmak, veri iletimini arttırmak ve mevcut fiziksel bağlantıları mümkün olduğunca verimli kullanmak amacıyla fiziksel bağlantıların sayısı üzerindeki iş yükünü bölmek üzerine çalışan bir yük dengeleme tekniği anahtar üzerine uygulanmaktadır.

4.2.2. Open vSwitch

Open vSwitch ya da Açık Sanal Anahtar, programlı ağ otomasyonu amacında olan bir açık kaynaklı sanal çok katmanlı bir anahtardır. OpenFlow protokolünün uygulanmasını destekler. Uygulamaları, her akış girişinin eşleşme koşullarına ve ilişkili eylemlere sahip olduğu akış tablolarından oluşmaktadır. Open vSwitch kontrolcüye güvenli bir kanal vasıtasıyla bağlanır ve ağ akışlarını kontrol etmek için OpenFlow protokolünü kullanır. Open vSwitch kuyruk disiplinlerini yönetmek, bağlantı noktası kuyruğunu değiştirmek, bağlantı noktası yansıtma kuralları vb. gibi geniş araç yelpazesini destekler (Pfaff B., 2013).

OpenFlow desteklediği teknolojiler şunlardır (Pfaff B., 2013):

- Güvenlik: VLAN yalıtım, Trafik Filtreleme.
- İzleme: NetFlow, sFlow , SPAN, RSPAN.
- Hizmet Kalitesi: Trafik Kuyruklama, Trafik Şekillendirme.
- Otomatik Kontrol: OpenFlow, OVSDB Yönetim Protokolü.

Kapsayan ağaç protokolü

Kapsayan ağaç protokolü (Spanning Tree Protocol, STP) anahtarlarda önemli bir protokoldür. STP protokolü, ağdaki herhangi bir döngü biçimini önlemek üzerine çalışır. Bu sayede ağdaki pakete boğma problemi önlenir. Bu çalışma kapsamında yaratılan ağ topolojisinde farklı kapalı döngüler olduğundan pakete boğma problemi ortaya çıkmıştır ve bu problem Kapsayan Ağaç Protokolü kullanılarak ortadan kaldırılmıştır (Chao Y., 2010).

Hiyerarşik Jeton kovası

Hiyerarşik Jeton Kovası (Hierarchical Token Bucket, HTB) bir paket zamanlayıcısıdır ve sınıf tabanlı kuyruklama (ağ yöneticilerinin, ne tür bir iletim olduğuna bağlı olarak her veri paketi setine belirli bir öncelik atamasına olanak tanıyan bir sistemdir) ile aynı genel işlevselliği desteklemektedir ama ortalama boş hesaplamaları belirteç kovası filtreleri ile değiştirir.

HTB, belirli bir bağlantının giden bant genişliğinin kullanımının kontrol edilmesinde

yardımcı olur. Birkaç daha yavaş bağlantıyı taklit etmek ve farklı taklit edilmiş bağlantılar üzerinden farklı türlerde trafik göndermek için tek bir fiziksel bağlantı kullanılmasına izin verir. Her iki durumda da fiziksel bağlantının taklit edilmiş bağlantılara nasıl bölüneceğini ve gönderilecek verilen bir paket için hangi taklit edilmiş bağlantının kullanılması gerektiğine nasıl karar verileceğini açıkça belirtmek zorundadır.

HTB, trafiği arayüz özelliklerine bağlı olmayan ve bu yüzden dışarı giden arayüzün altında yatan bant genişliğini bilmesine gerek olmayan Belirteç Kovası Filtresi algoritmasına dayalı olarak şekillendirir. Bir paketi kuyruğa eklerken HTB kökte başlar ve hangi sınıfın verileri alması gerektiğine karar vermek için farklı yaklaşımlar kullanır. Eğer hazırda yaygın olmayan konfigürasyon seçenekleri yoksa o zaman süreç oldukça kolaylaşır (Balan D. G., 2009).

4.2.3. Kuyruk yapısı

OpenFlow tabanlı anahtarlar, belirli bir çıkış bağlantı noktasına bağlı bir veya daha fazla kuyruğa sahip olabilir ve bu kuyruklar çıkış bağlantı noktasında veri yolu mevcut paketleri planlamak için paket zamanlayıcısı kullanabilir. Her kuyruk, kuyruk kimliği ve bağlantı noktası sayısı ile benzersiz şekilde tanımlanır. İki kuyruğun aynı kuyruk kimliğine sahip olması mümkündür. Paketler, paket çıkış bağlantı noktası ve kuyruk kimliğine dayanarak kuyruklardan birine yönlendirilir ve bu olay Çıktı ve Set Kuyruk eylemleri vasıtasıyla yapılır. Paket akışları, kuyruğa eklenerek belirli bir kuyruğa haritalanır ve akışlar bu kuyruğun konfigürasyonuna göre işlenir.

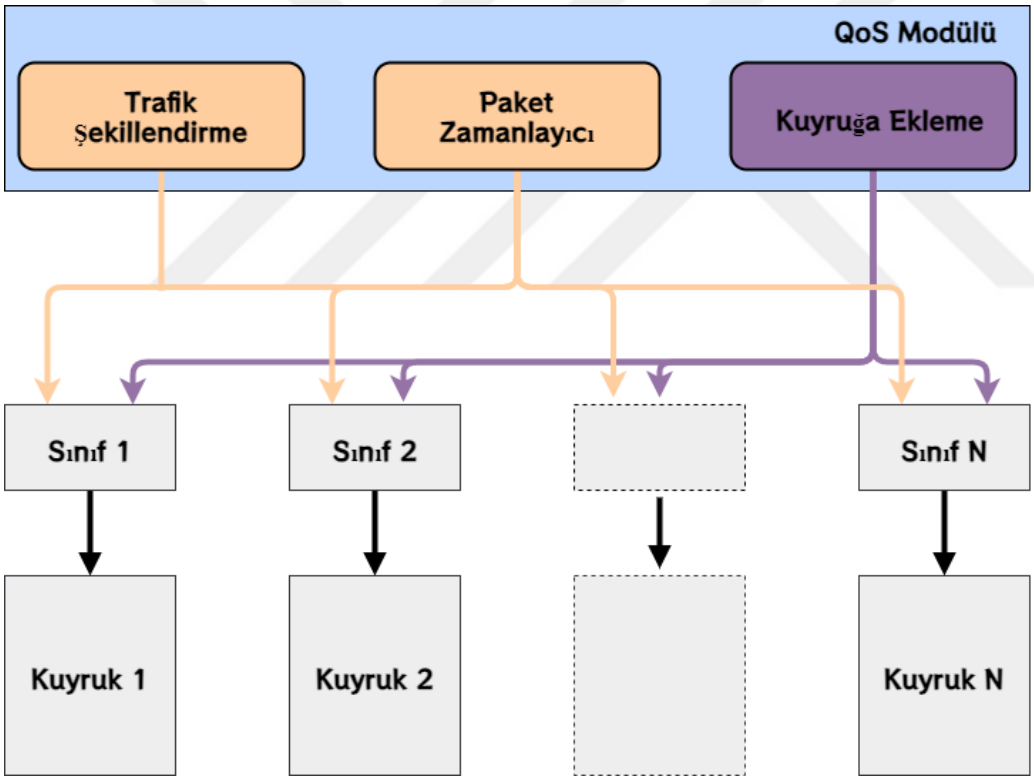
4.2.4. Kuyruğa ekleme

OpenFlow Protokolünün OFPT FLOW MOD mesajlarının çalıştırılmasından sorumludur. Bu mesaj, akış tablosunun durumunda değişiklik yapmak üzerine çalışır. Her paketin, akış paketlerini eşleştirmek için eylem, sayaç ve başlık alanları vardır. Kuyruğa ekleme, çekirdeğin SK BUFF adlı veri yapısının SKB PRIORITY'sini kullanarak paket akışlarını kuyruklara haritalar. Konfigürasyon, paketin SO PRIORITY opsiyonu kullanılarak tamamlanır.

4.2.5. Trafik şekillendirme ve paket zamanlayıcıları

Bu iki bileşen, bant genişliği hacmini paylaştırarak ve bu kuyruklar için paket zamanlayıcıları ekleyerek ya da çıkararak kontrol düzleminden alınan QoS mesajlarının çalıştırılmasından sorumludur. Ayrıca, bu bileşenler, OpenFlow protokolünde Hizmet Kalitesi mesajlarını temsil eden OFPT QOS QUEUEING DISCIPLINE mesajlarını manipüle eder. HTB paket zamanlayıcısı, her kuyruğa arzu edilen oranın karşılanması amacıyla paketleri geciktirmek için kullanılır ve minimum ve maksimum bant genişliği kapasitesinin yapılandırılmasına olanak sağlamaktadır.

Şekil 4.4 Hizmet Kalitesi modülü bileşenlerini göstermektedir.



Şekil 4.4. Hizmet kalitesi modülü bileşenleri

5. DENEYSEL SONUÇLAR

Bu bölümde, ilk önce çalışma kapsamında kullanılan araçlar tanıtılmaktadır, ardından test ortamları, kontrolcü seçimi ve topoloji tartışılmıştır. Daha sonra, derin öğrenme modellerinin sonuçları ve hizmet kalitesi odaklı trafik mühendisliğinin değerlendirmeleri detaylı olarak sunulmuştur.

5.1. Araştırmada Kullanılan Araçlar

5.1.1. Derin öğrenme ve makine öğrenimi kütüphaneleri

PyTorch

PyTorch açık kaynaklı bir derin öğrenme çerçevesidir ve Lua diline dayalıdır. Esnekliği, araştırma için modülerliği, kararlılığı ve üretim dağıtımına yönelik desteği ile bilinmektedir. PyTorch, GPU hızlandırma ve TorchScript (Bu, istekli ve grafik modu arasında kolay bir geçiş için serileştirilebilir ve optimize edilebilir modeller oluşturma tekniğidir) ile tensör hesaplama gibi üst düzey özellikler sağlar. PyTorch, sinir ağları, optimizasyon, görüntü işleme, grafik modeller ve enerji tabanlı modeller için paketlere sahiptir (Parvat, 2017).

PyTorch kullanmanın avantajları şunlardır:

- büyük veri kümeleri için uygundur,
- yüksek hızlıdır,
- yüksek performanslıdır,
- hata ayıklama daha kolay ve daha hızlıdır,
- ve dinamik hesaplama grafiğine sahiptir.

Keras

Keras, açık kaynaklı bir sinir ağı çerçevesidir ve Python'da yazılır. Keras, ana akım olmayan yüksek seviyeli bir sinir ağıdır ve TensorFlow ya da Theano gibi diğer çerçevelerin üzerinde çalıştırma kapasitesine sahiptir. Kolay kullanımı, modüleri ve hızlı prototip oluşturması ile bilinir (Parvat, 2017).

Keras kullanmanın avantajları şunlardır:

- yüksek seviyeli API,
- basit mimari,
- ve farklı arka uçlarda çalıştırma yeteneğine sahiptir.

Scikit-Learn

Scikit learn, açık kaynaklı bir makine öğrenimi kütüphanesidir. Orta ölçekli gözetimli ve gözetimsiz problemler için çok çeşitli gözetimli öğrenme ve gözetimsiz öğrenme algoritmaları sunar. Scikit-Learn iyi dokümantasyon ve tutarlı API tutar (Pedregosa, 2011).

Scikit-Learn kullanmanın avantajları şunlardır:

- kolay kullanım,
- makine öğrenimi algoritmaları için sabit arayüz,
- her algoritma için farklı ayar parametreleri sunar,
- makine öğrenimi görevleri için farklı işlevsellikler sunar.

5.1.2. Kullanılan araçlar

Netcat

Netcat, TCP veya UDP ağ bağlantılarını kullanarak IP paketleri göndermek veya almak için kullanılan bir ağ aracıdır. Doğrudan bir komut veya ascript olarak ya da diğer programlar tarafından çalıştırılabilir. Farklı bağlantı türlerini desteklediği için ağda hata ayıklama ve araştırma için kullanışlı bir araçtır. Netcat aracı, IP paketleri oluşturmak ve ağa göndermek için kullanılır. IP paketlerinin başlık alanlarını değiştirme yeteneği sunar. Böylece IP paketleri veri türlerine göre etiketlenebilir (Kurth M., 2020).

iPerf3

iPerf3, ağ performansını değerlendirmek ve ayarlamak için kullanılan bir ağ aracıdır. IP ağlarında mümkün olan maksimum bant genişliğinin canlı ölçümlerini sağlar. IPv4 ve IPv6 ile TCP, UDP, SCTP gibi zamanlama, tamponlar ve protokollerle ilgili farklı parametrelerin

ayarlanmasını destekler. Her test için bant genişliği, kayıp ve diğer parametreleri raporlar (Dugan, 2014).

5.2. Test Ortamları

5.2.1. Donanım bileşenleri

Sistem uygulaması sırasında kullanılan donanım bileşenlerinin özellikleri şunlardır:

- İşlemci: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 2208 Mhz, 6 Core(s), 12 Logical Processor(s).
- Yüklenmiş Hafıza (RAM): 32 GB
- Grafik İşlemci Birimi (GPU): 6G NVIDIA GeForce GTX 1060.

5.2.2. Google colaboratory

Google Colaboratory; öğrencilere, veri bilimcilerine ve araştırmacılara, güçlü GPU'ları kullanmak için hiçbir konfigürasyon olmadan ve ücretsiz erişim ile tarayıcıda Python kodu yazıp yürütmesine olanak tanıyan, Google tarafından sunulan bulut tabanlı bir hizmettir. 12 saate kadar kesintisiz kullanılabilen 12GB NVIDIA Tesla K80 GPU sunar (Bisong E., 2019).

5.2.3. Mininet

Mininet, YTA ağları için sanal bir test ortamı ve geliştirme ortamı sağlayan bir emülasyon aracıdır. Mininet, YTA geliştirme ağının herhangi bir bilgisayarda kullanılabilmesine ve yönetimi kolay olmasına izin verir. Ayrıca, YTA tasarımları Mininet ile gerçek donanım arasında sorunsuz bir şekilde taşınabilir.

Mininet, bir ana bilgisayar, anahtar, yönlendirici ya da kontrolcü gibi ağ cihazlarını sanallaştırmak için sanallaştırma kullanır. Oluşturulan tüm ağ öğelerinin aynı makinede çalışmasına rağmen onlar mantıksal olarak birbirinden tamamen ayrıdır ve her biri gerçek bir fiziksel aygıt olarak çalışır.

Mininet kullanmanın avantajları aşağıda listelenmektedir (Kaur K., 2014):

- Herhangi bir donanım yatırımı yapmadan mevcut kaynaklarla çeşitli ağ ortamlarının oluşturulmasını sağlar.
- Karmaşık ve büyük ağ projelerinin test edilmesini ve hataların üretime geçmeden önce tespit edilebilmesini garantiye alır.
- Ağa eklenen her düğüm bir SSH bağlantısı üzerinden bağlanır.
- Linux işletim sistemi programları her düğümde çalıştırılır.
- Bant genişliği ve gecikmeler, düğümler arasındaki her bağlantı için ayrı olarak kurulur.
- Mininet'in üzerinde çalıştığı makinenin donanım kaynakları olduğu sürece her çeşit ağ topolojisi oluşturulabilir.
- Mininet üzerinde yaratılan ve test edilen çözümler gerçek cihazlara aktarılır.

5.3. Kontrolcü Seçimi

Kontrolcü seçimi temel bir unsurdur. Literatürde kullanılan birçok kontrolcü versiyonu vardır. Bazıları ticari iken bazıları ise açık kaynaklı kontrolcülerdir.

Aşağıdaki noktalar kontrolcü seçerken önemli olarak görülmektedir:

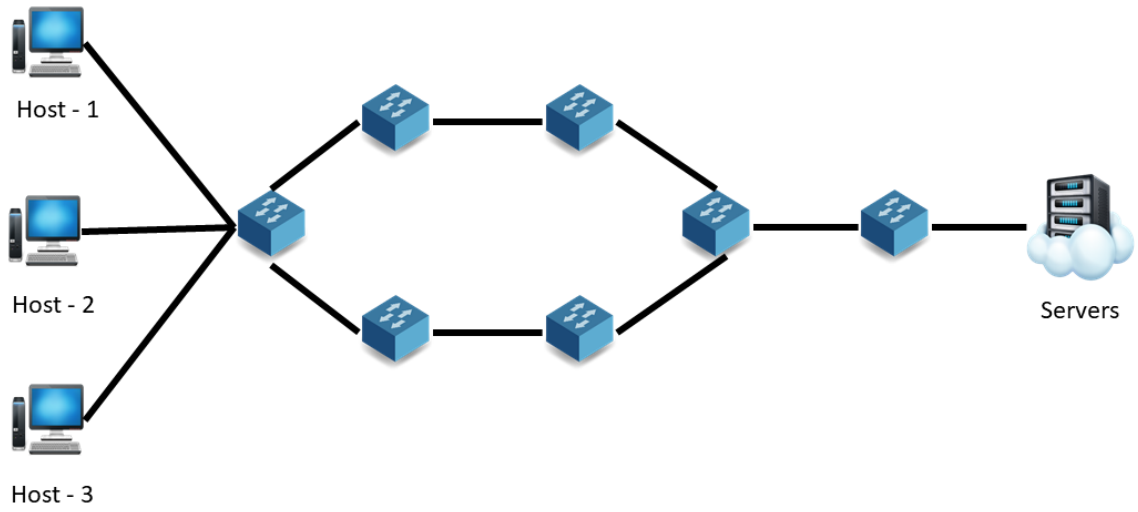
- Açık Kaynak
- Geliştirmede kullanılacak programlama dili
- Esnek ve hızlı

Daha önce bahsedilen noktalar dikkate alınarak Pox kontrolcünün uygun olduğu anlaşılmıştır.

POX, YTA uygulamaları geliştirmeye adanmış python tabanlı açık kaynaklı bir kontrolcüdür. Ağ uygulamalarının hızla büyümesi ve prototiplenmesi için bir platform olarak görülmektedir. POX'un temel amacı araştırmadır. Hub, anahtar, yönlendirici, yük dengeleyici, güvenlik duvarı ve daha fazlası gibi farklı uygulamaları çalıştırabilir. Deneyleme aşamasında, POX kontrolcünün başarılı olduğu görülür ve tez gereksinimlerini karşılamıştır (Kaur S., 2014).

5.4. Topoloji

Çalışmada kullanılan topoloji Mininet'te yaratılmıştır ve 10 anahtar ve altı düğümden oluşmaktadır. Mininet, oluşturulan topolojinin görsel bir çıktısını desteklemez. Yaratılan topolojinin gösterilebilmesi için görsel çıktıya ihtiyaç olduğundan topolojiyi çizmek için harici bir araç kullanılmıştır.



Şekil 5.1. Test ortamı topoloji

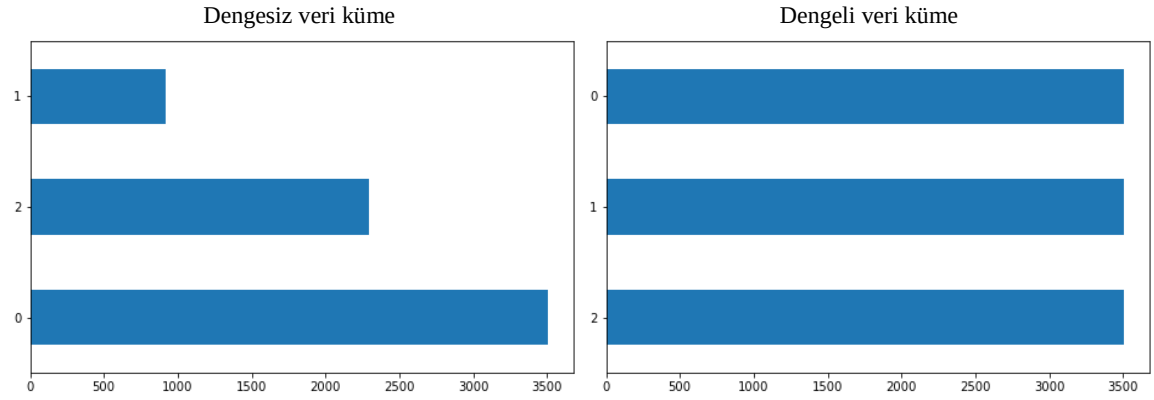
5.5. Derin Öğrenme Modellerinin Değerlendirilmesi

Bu bölümde, dengeli ve dengesiz veri setlerinin performansı değerlendirilir ve kullanılan aşırı örnekleme tekniğinin model doğruluğunu artırmadaki etkisini göstermek için karşılaştırılır. Ayrıca, derin öğrenme algoritmalarının performansı değerlendirilir ve çeşitli makine öğrenimi algoritmalarının performanslarıyla karşılaştırılır. K-En Yakın Komşu Algoritması, Destek Vektör Makinesi, Karar Ağacı ve Rastgele Orman algoritmaları derin öğrenme algoritmaları ile değerlendirme ve karşılaştırma için kullanılmıştır. Farklı akış zaman aşımı değerleri incelenmiş ve son sonuçlar üzerindeki etkisini göstermek için değerlendirilmiştir. 5, 10, 15, 30, 60 ve 120 saniyelik akış süreleri göz önüne alınmıştır.

5.5.1. Dengesiz ve dengeli veri kümeleri

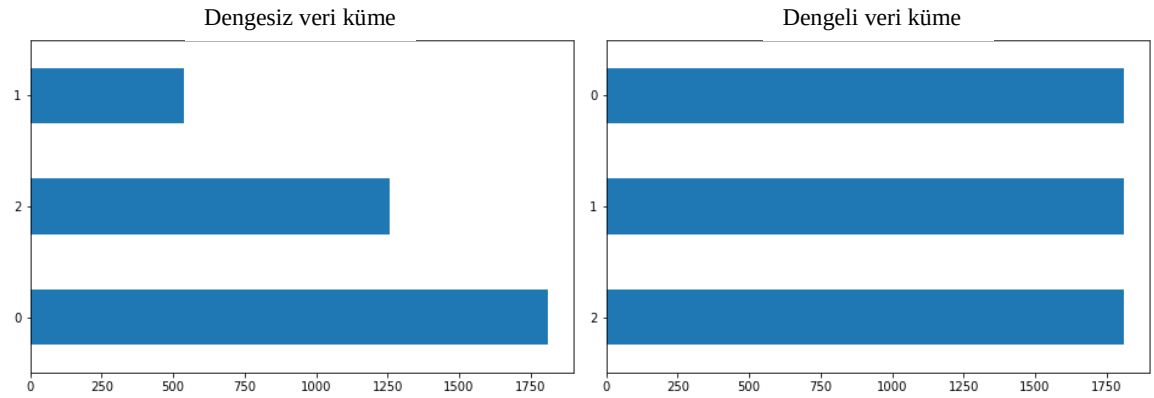
Sınıfların gruplanma süreci boyunca 15, 30, 60 ve 120 saniyelik veri setlerinin örnek sayısı dengesizdi. Sınıf 1'deki örneklerin sayısı diğer iki sınıfla karşılaştırıldığında nispeten düşüktü. Bu sorunu çözmek amacıyla sınıfları dengelemek için Sentetik Azınlık Aşırı Örnekleme Tekniği (SMOTE) adı verilen bir aşırı örnekleme metodu kullanılmıştır. SMOTE, azınlık sınıfını kopyalamak yerine sentetik örnekler oluşturarak aşırı örneklendirir. Sentetik numuneler, her bir azınlık sınıfı numunenin çizgi segmentlerinin yanında sunulmaktadır ve K-En Yakın Komşularının bazılarına ya da tamamına katılır. En yakın komşuların sayısı, aşırı örnekleme m setinin miktarına bağlı olarak seçilir. Örneğin; m'nin % 300 ve k'nin 5 olduğunu varsaydığımızda yöntem her bir azınlık numunesi için üç sentetik numune oluşturmak üzere en yakın üç komşuyu rastgele seçer.

Şekil 5.2'de 15 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu gösterilmektedir. Dengesiz veri setindeki örnek sayısı 6720 idi ve etiket 0 veri setinin % 52.14'ünü, etiket 1 veri setinin % 13.66'sını ve etiket iki veri setinin % 34'ünü temsil ediyordu. Aşırı örneklemeden sonra veri setine 3626 yeni örnek eklendi ve böylece örnek sayısı 10346 olmuştur.



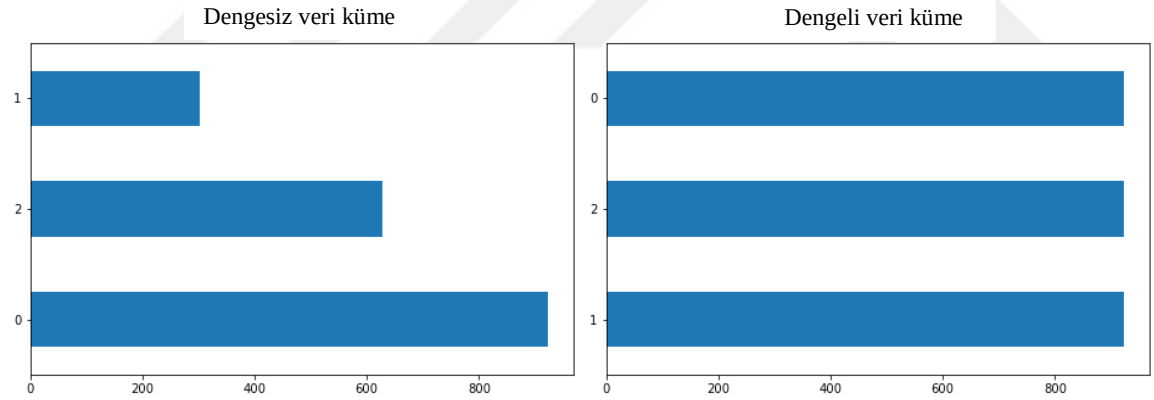
Şekil 5.2. 15 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu

Şekil 5.3'de 30 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu gösterilmektedir. Dengesiz veri kümesindeki örnek sayısı 3605'ti ve etiket 0 veri kümesinin % 50.18'ini, etiket 1 veri kümesinin % 14.92'sini ve etiket 2 veri kümesinin % 34.9'unu temsil ediyordu. Aşırı örneklemeden sonra veri setine 1749 yeni örnek eklendi ve böylece örnek sayısı 5354 olmuştur.



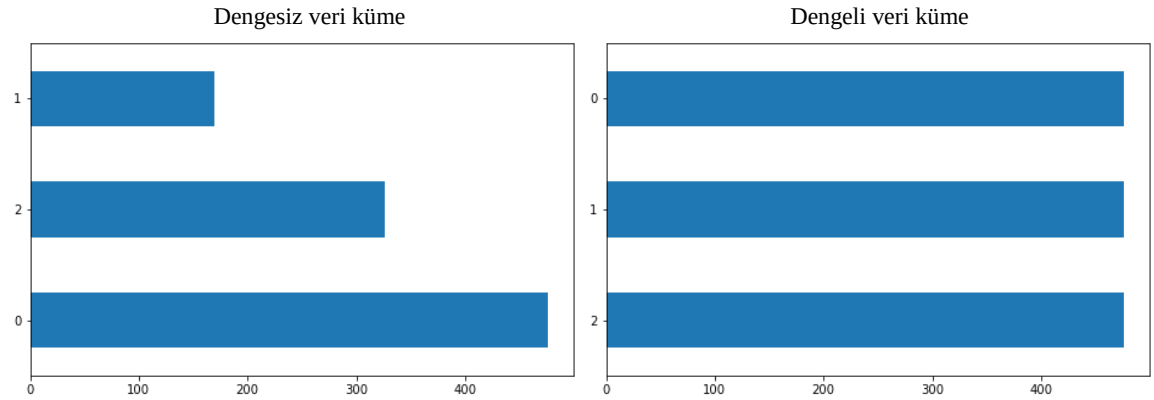
Şekil 5.3. 30 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu

Şekil 5.4'de 60 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu gösterilmektedir. Dengesiz veri setindeki örnek sayısı 1851 idi ve etiket 0 veri setinin % 49.81'ini, etiket 1 veri setinin % 16.32'sini ve etiket 2 veri setinin % 33.87'sini temsil ediyordu. Aşırı örneklemeden sonra veri setine 740 yeni örnek eklendi ve böylece örnek sayısı 2591 olmuştur.



Şekil 5.4. 60 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu

Şekil 5.5'de 120 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu gösterilmektedir. Dengesiz veri setindeki örnek sayısı 972 idi ve etiket 0 veri setinin % 48.97'sini, etiket 1 veri setinin % 17.49'unu ve etiket 2 veri setinin % 33.54'ünü temsil ediyordu. Aşırı örneklemeden sonra veri setine 440 yeni örnek eklendi ve böylece örnek sayısı 1412 olmuştur.



Şekil 5.5. 120 saniyelik veri kümesi için aşırı örnekleme tekniğinin sonucu

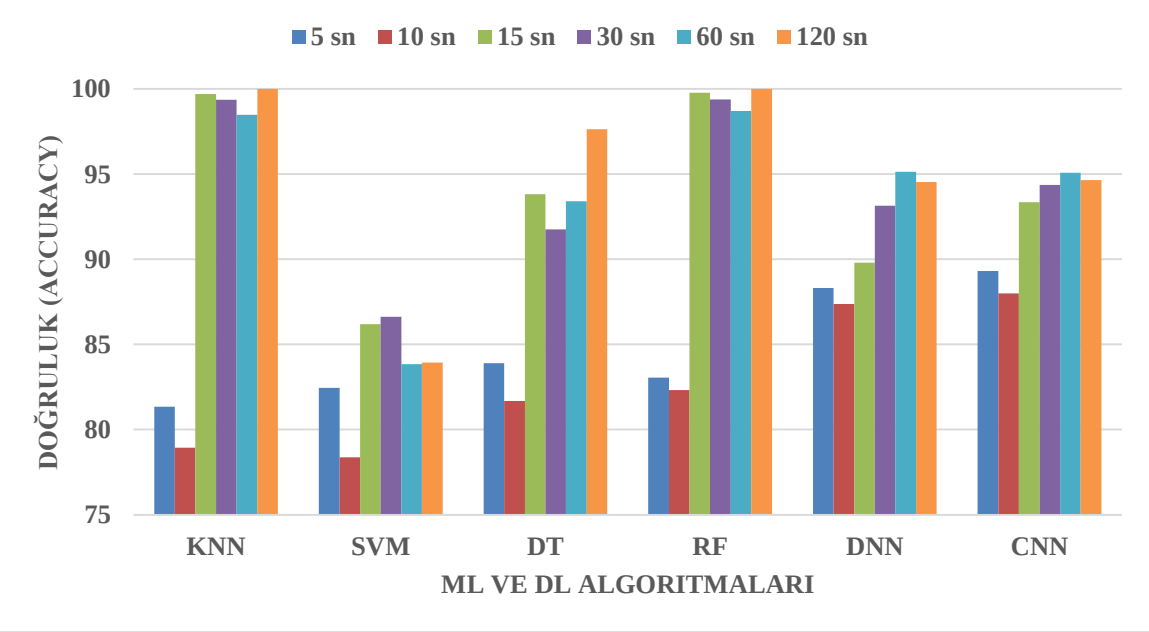
5.5.2. Değerlendirme sonuçları

Derin öğrenme algoritmalarının performansı, birçok makine öğrenme algoritması, k-en yakın komşu, destek vektör makinesi, karar ağacı ve rastgele orman algoritmalarının performansıyla karşılaştırılmıştır. Hem makine öğrenimi hem de derin öğrenme modellerinin performansını artırmak için makine öğrenimi modellerini eğitmek için kullanılan veri kümeleri standartlaştırıldı ve derin öğrenme modellerini eğitmek için kullanılan veri kümeleri normalleştirildi. Derin öğrenme ve makine öğrenimi modellerinin eğitimi sırasında 5 kat çapraz doğrulama kullanılmıştır. Çapraz doğrulama, hiperparametrelerin ayarlanmasına yardımcı olmak için düzenleme ile birlikte kullanılır. Çapraz doğrulama, sonunda test için kullanılacak bir test setini korurken modeli görünmeyen veriler üzerinde test etme yeteneği de sunar. Eğitim sırasında, hem derin öğrenme hem de makine öğrenimi modelleri için belirtilen parametre değerleri üzerinde kapsamlı bir araştırma yapılmıştır. Sınıflandırmanın genel doğruluğu (accuracy) sınıflandırıcı performansı hakkında iyi bir anlayış vermediğinden modellerin performansını belirlemek için kesinlik (precision), duyarlılık (sensitivity veya tecall) ve F-puanı (F-score) metrikleri kullanılmıştır. Ayrıca, aşırı örnekleme tekniğinin modellerin doğruluğu üzerindeki etkisini gözlerde canlandırmak için tüm modeller arasında dengeli ve dengesiz veri kümelerinin bir karşılaştırması gösterilmiştir. Çizelge 5.1'de hem makine öğrenimi hem de derin öğrenme modelleri için dengesiz veri kümelerinin doğruluğu gösterilmektedir.

Çizelge 5.1. Dengesiz veri kümelerinin doğruluğu

	5 sn.	**10 sn.**	**15 sn.**	**30 sn.**	**60 sn.**	**120 sn.**
KNN	81.34	78.94	99.70	99.36	98.48	**100.00**
SVM	82.45	78.38	86.2	86.62	83.84	83.94
DT	83.90	81.68	93.82	91.76	93.41	97.63
RF	83.06	82.32	**99.79**	**99.38**	**98.70**	**100.00**
DNN	88.31	87.38	89.81	93.15	95.14	94.54
CNN	**89.32**	**88.00**	93.35	94.37	95.08	94.65

Şekil 5.6, dengesiz veri kümelerinin doğruluk ölçüsü için Çizelge 5.1'deki verileri göstermektedir. DNN ve CNN, 5 ve 10 saniye için diğer makine öğrenimi algoritmalarından daha yüksek doğruluk elde etmiştir. KNN ve RF, 15, 30, 60 ve 120 saniye için diğer algoritmalardan daha yüksek doğruluk elde etmiştir. DT, 15 ve 120 saniye için DNN, CNN ve SVM'den daha yüksek doğruluk sağlamıştır. Ancak DNN ve CNN, 30 ve 60 saniye için DT ve SVM'den daha yüksek doğruluk elde etmiştir.

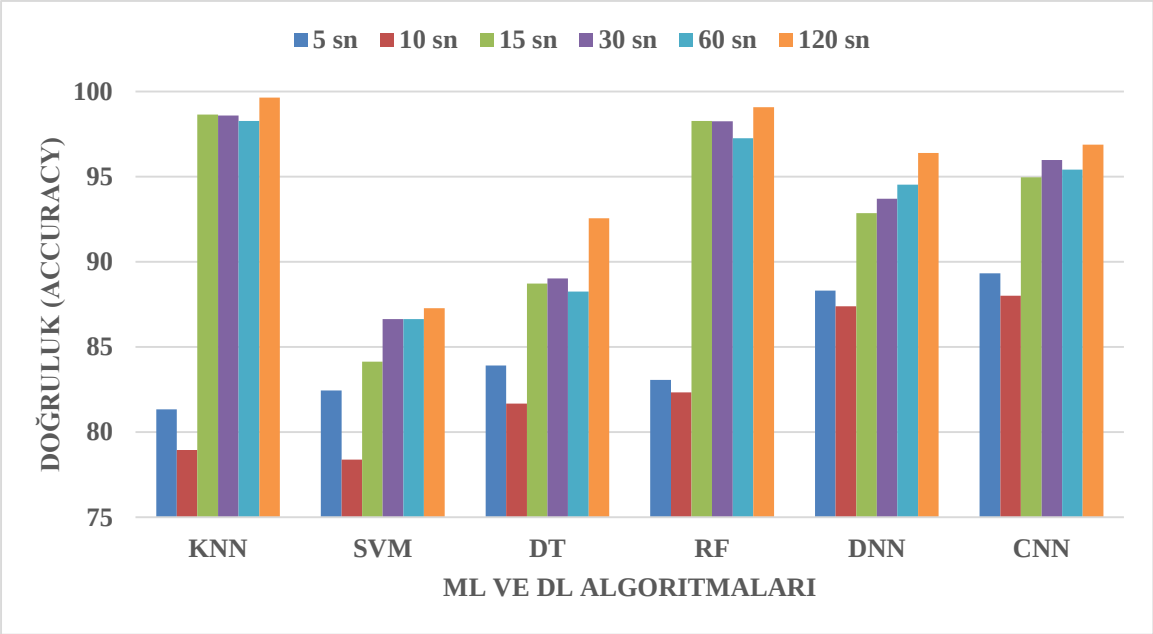


Şekil 5.6. Dengesiz veri kümelerinin doğruluğu

Çizelge 5.2'de dengeli veri kümelerinin doğruluğu gösterilmektedir. Şekil 5.7, dengeli veri kümelerinin doğruluk ölçüsü için Çizelge 5.2'deki verileri göstermektedir. DNN ve CNN, 5 ve 10 saniye için diğer algoritmalardan daha yüksek doğruluk elde edebilmiştir. KNN ve RF, 15, 30, 60 ve 120 saniye için diğer algoritmalardan daha yüksek doğruluk elde edebilmiştir. Yüksek hızda örnekleme uygulandıktan sonra KNN ve RF'nin doğruluğunun %2 azaldığı, DT'nin doğruluğunun %4 azaldığı, SVM'nin doğruluğunun %3 arttığı, DNN ve CNN'in doğruluğunun arttığı fark edilmiştir. DNN ve CNN, 15, 30, 60 ve 120 saniye için DT ve SVM'den daha iyi performans göstermiştir. CNN ise DNN'den daha iyi performans göstermiştir.

Çizelge 5.2. Dengeli veri kümelerinin doğruluğu

	5 sn.	**10 sn.**	**15 sn.**	**30 sn.**	**60 sn.**	**120 sn.**
KNN	81.34	78.94	**98.65**	**98.59**	**98.26**	**99.64**
SVM	82.45	78.38	84.13	86.64	86.64	87.28
DT	83.90	81.68	88.72	89.02	88.24	92.55
RF	83.06	82.32	98.27	98.24	97.26	99.07
DNN	88.31	87.38	92.85	93.70	94.52	96.38
CNN	**89.32**	**88.00**	94.96	95.98	95.41	96.88



Şekil 5.7. Dengeli veri kümelerinin doğruluğu

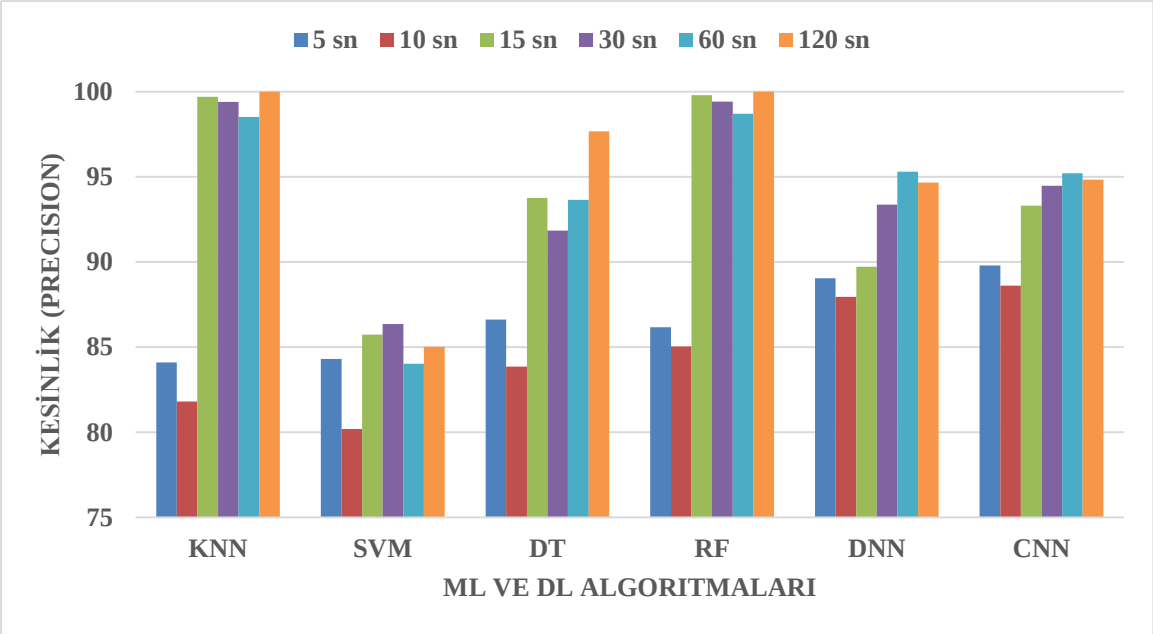
Sınıflandırma doğruluğu modellerin performansı hakkında tek başına yeterli bilgi vermediğinden, modelin performansını değerlendirmek için kesinlik, duyarlılık ve f-puanı ölçümleri kullanılmıştır. Çizelge 5.3'de dengesiz veri kümelerinin kesinliği gösterilmektedir. Şekil 5.8, dengesiz veri kümelerinin kesinlik ölçüsü için Çizelge 5.3'teki verileri göstermektedir. Sonuçlar, DNN ve CNN'nin 5 ve 10 saniye için diğer algoritmalardan daha yüksek hassasiyete ulaşabildiğini, KNN ve RF'nin 15, 30, 60 ve 120 saniye için yüksek hassasiyete ulaşabildiğini ve DT'nin CNN'den daha iyi performans gösterebildiğini göstermektedir. DT, 15 ve 120 saniye için CNN, DNN ve SVM'den daha iyi performans gösterirken, CNN ve DNN, 30 ve 60 saniye için DT ve SVM'den daha iyi performans göstermiştir. Ayrıca CNN, 15 saniye için DNN'den daha yüksek kesinlik elde etmiştir.

Çizelge 5.3. Dengesiz veri kümelerinin kesinliği

	5 sn.	**10 sn.**	**15 sn.**	**30 sn.**	**60 sn.**	**120 sn.**
KNN	84.10	81.81	99.70	99.39	98.51	**100.00**
SVM	84.30	80.18	85.74	86.35	84.03	85.01
DT	86.61	83.85	93.76	91.84	93.65	97.66
RF	86.16	85.03	**99.79**	**99.42**	**98.71**	**100.00**

Çizelge 5.3. (devam) Dengesiz veri kümelerinin kesinliği

DNN	89.05	87.96	89.72	93.37	95.30	94.66
CNN	**89.79**	**88.61**	93.31	94.48	95.21	94.83



Şekil 5.8. Dengesiz veri kümelerinin kesinliği

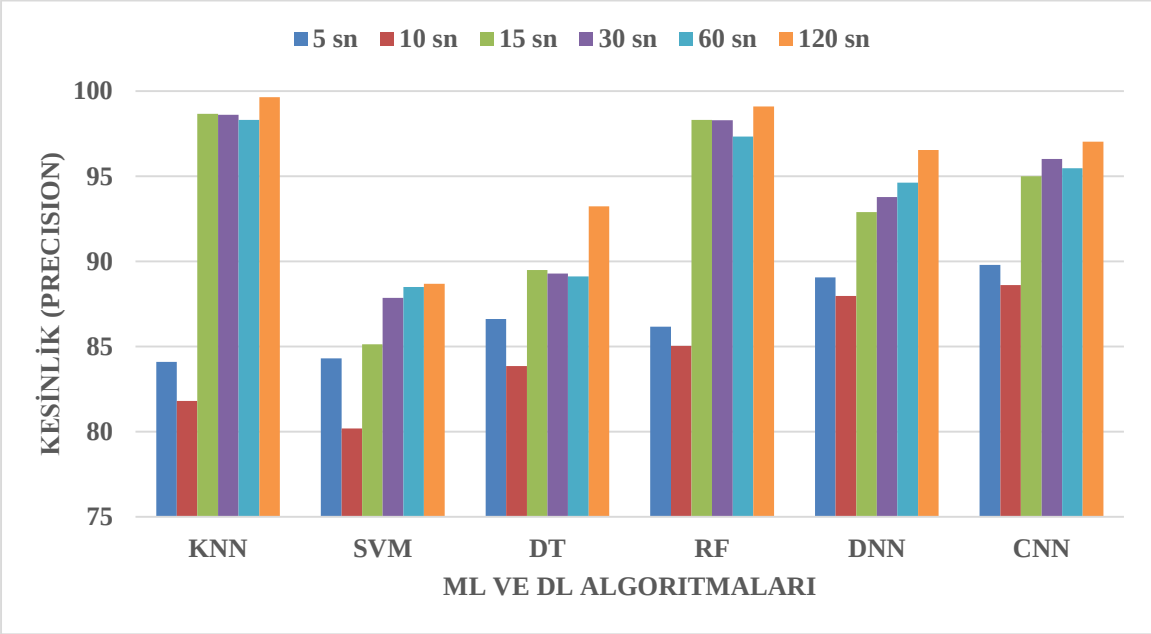
Çizelge 5.4'te dengeli veri kümelerinin kesinlik metrik sonuçları gösterilmektedir. Şekil 5.9, dengeli veri kümelerinin kesinlik ölçüsü için Çizelge 5.4'teki verileri göstermektedir. Sonuçlar, yüksek hızda örnekleme tekniğini uyguladıktan sonra KNN ve RF'nin kesinliğinde azalma olduğunu göstermektedir, ancak yine de 15, 30, 60 ve 120 saniye için diğer modellerden daha iyi kesinlik elde edebilmiştir, ayrıca DT kesinliği azalırken CNN, DNN ve SVM için kesinlik artmıştır. Buna göre, CNN ve DNN, yüksek hızda örnekleme tekniğini uyguladıktan sonra DT ve SVM'den daha iyi performans sağlamıştır.

Çizelge 5.4. Dengeli veri kümelerinin kesinliği

	5 sn.	**10 sn.**	**15 sn.**	**30 sn.**	**60 sn.**	**120 sn.**
KNN	84.10	81.81	**98.66**	**98.61**	**98.31**	**99.64**
SVM	84.30	80.18	85.13	87.86	88.49	88.69

Çizelge 5.4. (devam) Dengeli veri kümelerinin kesinliği

DT	86.61	83.85	89.49	89.28	89.12	93.23
RF	86.16	85.03	98.31	98.29	97.33	99.09
DNN	89.05	87.96	92.90	93.78	94.63	96.53
CNN	**89.79**	**88.61**	94.99	96.01	95.47	97.02

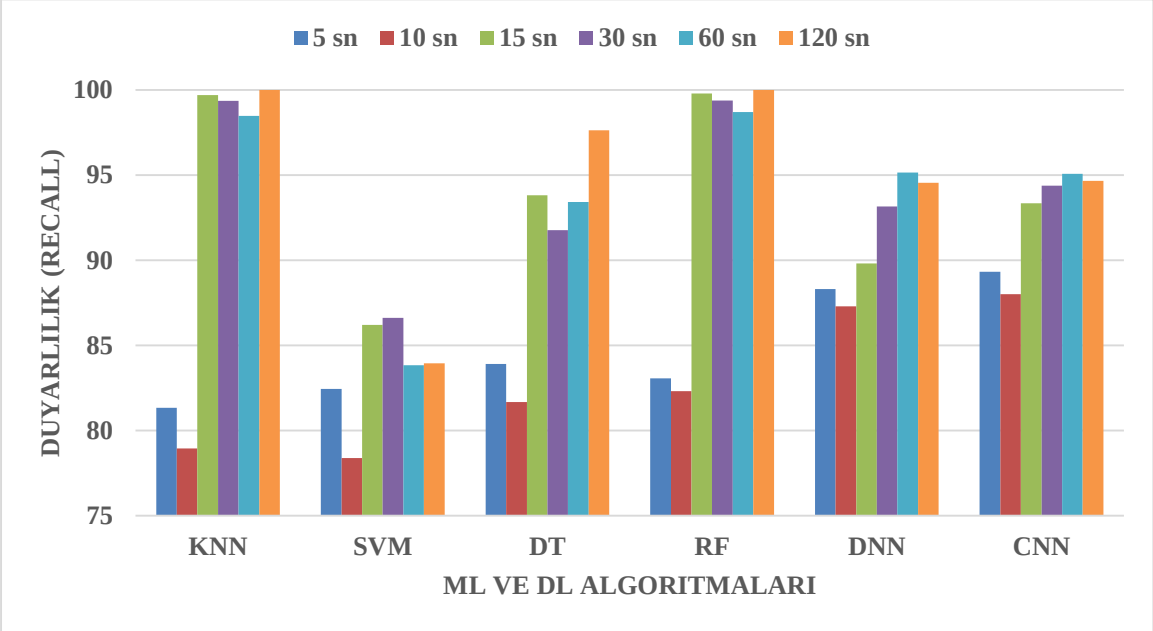


Şekil 5.9. Dengeli veri kümelerinin kesinliği

Çizelge 5.5'de dengesiz veri kümelerinin duyarlılık sonuçları gösterilmektedir. Şekil 5.10, dengesiz veri kümelerinin duyarlılık ölçüsü için Çizelge 5.5'teki verileri göstermektedir. Sonuçlar, DNN ve CNN'nin 5 ve 10 saniye için diğer algoritmalardan daha iyi performans gösterdiğini, KNN ve RF'nin 15, 30, 60 ve 120 saniye için yüksek duyarlılık seviyesine ulaşabildiğini göstermektedir. DT, 15 ve 120 saniye için CNN, DNN ve SVM'den daha iyi performans göstermiştir. Ancak CNN ve DNN, 30 ve 60 saniye için DT ve SVM'den daha iyi performans göstermiştir. Ayrıca CNN, 15 saniye için DNN'den daha iyi performans göstermiştir.

Çizelge 5.5. Dengesiz veri kümelerinin duyarlılığı

	5 sn.	10 sn.	15 sn.	30 sn.	60 sn.	120 sn.
KNN	81.34	78.94	99.70	99.36	98.48	**100.00**
SVM	82.45	78.38	86.20	86.62	83.84	83.94
DT	83.90	81.68	93.82	91.76	93.41	97.63
RF	83.06	82.32	**99.79**	**99.38**	**98.70**	**100**
DNN	88.31	87.30	89.81	93.15	95.14	94.54
CNN	**89.32**	**88.00**	93.35	94.37	95.08	94.65

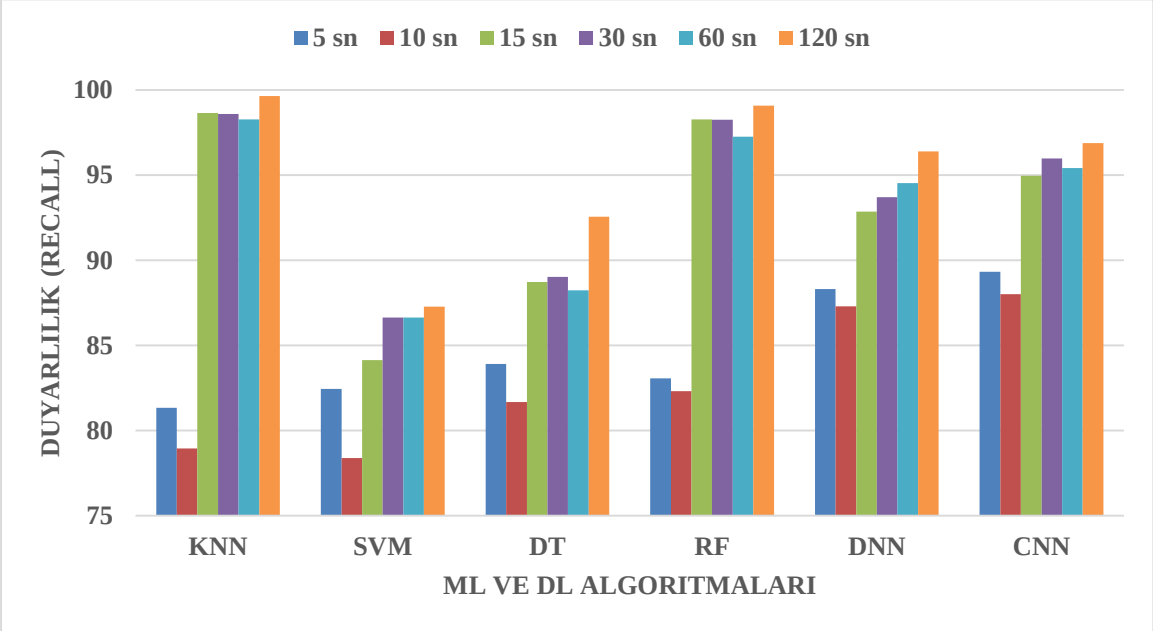


Şekil 5.10. Dengesiz veri kümelerinin duyarlılığı

Çizelge 5.6'da dengeli veri kümelerinin duyarlılığı gösterilmektedir. Şekil 5.11, dengeli veri kümelerinin duyarlılık ölçüsü için Çizelge 5.6'daki verileri göstermektedir. Sonuçlar, veri setlerini dengeledikten sonra KNN ve RF'nin duyarlılığında düşüş olduğunu, ancak yine de 15, 30, 60 ve 120 saniye için diğer modellerden daha iyi performans elde edebildiğini göstermektedir. Ayrıca, DT'nin duyarlılığı azalmış, CNN ve DNN'nin duyarlılığı artmıştır, SVM'nin duyarlılığı ise 15 saniye için azalırken, 30, 60 ve 120 saniye için artmıştır. Derin öğrenme modelleri bu açıdan DT ve SVM'den daha iyi performans göstermiştir.

Çizelge 5.6. Dengeli veri kümelerinin duyarlılığı

	5 sn.	10 sn.	15 sn.	30 sn.	60 sn.	120 sn.
KNN	81.34	78.94	**98.65**	**98.59**	**98.26**	**99.64**
SVM	82.45	78.38	84.13	86.64	86.64	87.28
DT	83.90	81.68	88.72	89.02	88.24	92.55
RF	83.06	82.32	98.27	98.24	97.26	99.07
DNN	88.31	87.30	92.85	93.70	94.52	96.38
CNN	**89.32**	**88.00**	94.96	95.98	95.41	96.88

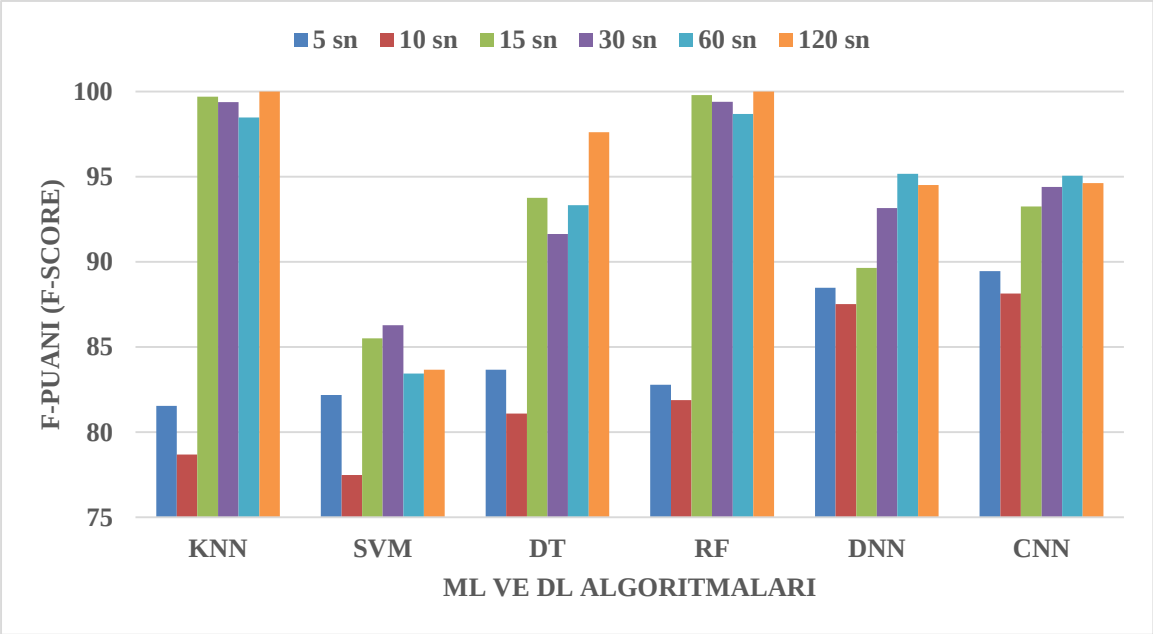


Şekil 5.11. Dengeli veri kümelerinin duyarlılığı

Çizelge 5.7'de dengesiz veri kümelerinin F-Puanı gösterilmektedir. Şekil 5.12, dengesiz veri kümelerinin f-puanı metriği için Çizelge 5.7'deki verileri göstermektedir. Sonuçlar, DNN ve CNN'nin 5 ve 10 saniye için diğer algoritmalardan daha iyi performans sunduğunu, KNN ve RF'nin ise 15, 30, 60 ve 120 saniye için diğer algoritmalardan daha iyi performans sunduğunu göstermektedir. CNN ve DNN, 30 ve 60 saniye için DT ve SVM'den daha iyi performans göstermiştir. DT, 15 ve 120 saniye için CNN, DNN ve SVM'den daha iyi performans göstermiştir.

Çizelge 5.7. Dengesiz veri kümelerinin F-Puanı

	5 sn.	**10 sn.**	**15 sn.**	**30 sn.**	**60 sn.**	**120 sn.**
KNN	81.54	78.68	99.70	99.37	98.48	**100.00**
SVM	82.19	77.48	85.51	86.28	83.45	83.67
DT	83.66	81.09	93.75	91.64	93.32	97.62
RF	82.78	81.88	**99.79**	**99.39**	**98.69**	**100.00**
DNN	88.48	87.52	89.65	93.16	95.17	94.52
CNN	**89.45**	**88.14**	93.26	94.4	95.06	94.62

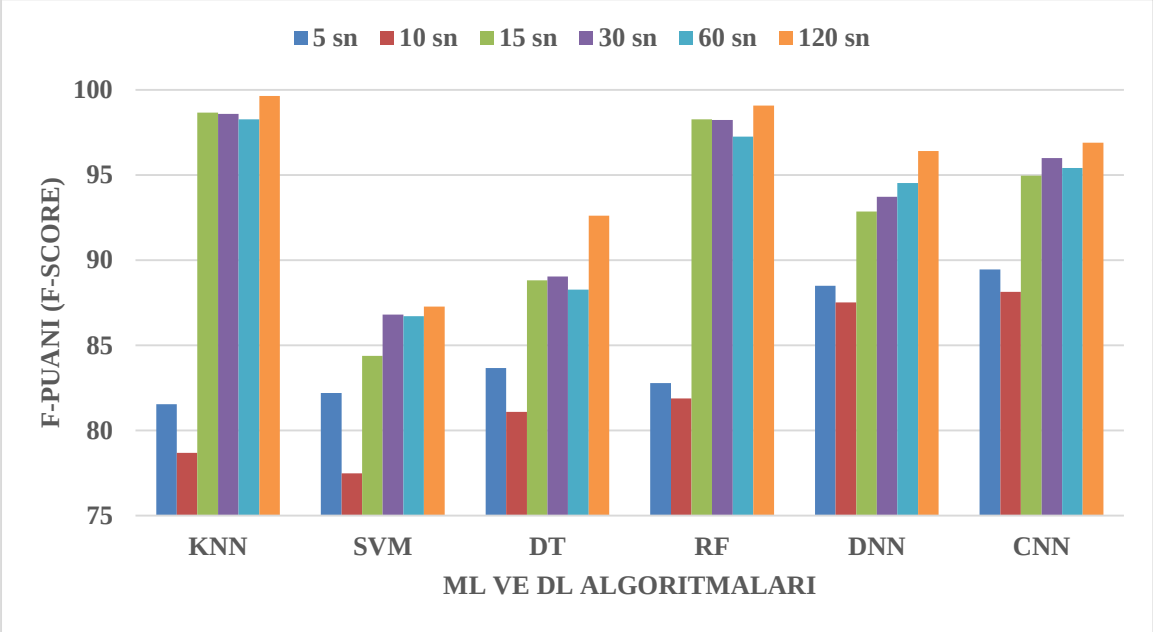


Şekil 5.12. Dengesiz veri kümelerinin F-Puanı

Çizelge 5.8'da dengeli veri kümelerinin F Puanı gösterilmektedir. Şekil 5.13, dengeli veri kümelerinin f-puanı metriği için Çizelge 5.8'deki verileri göstermektedir. Sonuçlar, yüksek hızda örnekleme tekniğini uyguladıktan sonra KNN ve RF'nin f-puanı doğruluğunda düşüş olduğunu, ancak yine de 15, 30, 60 ve 120 saniye için diğer modellerden daha iyi hassasiyet elde edebildiğini göstermektedir. DT'nin F skoru doğruluğu azalırken, CNN, DNN ve SVM'nin f skoru doğruluğu artmıştır. Bu nedenle, CNN ve DNN, yüksek hızda örnekleme tekniğini uyguladıktan sonra DT ve SVM'den daha iyi performans sağlamıştır.

Çizelge 5.8. Dengeli veri kümelerinin F-Puanı

	5 sn.	**10 sn.**	**15 sn.**	**30 sn.**	**60 sn.**	**120 sn.**
KNN	81.54	78.68	**98.65**	**98.59**	**98.27**	**99.64**
SVM	82.19	77.48	84.37	86.80	86.70	87.26
DT	83.66	81.09	88.80	89.03	88.27	92.61
RF	82.78	81.88	98.27	98.23	97.25	99.07
DNN	88.48	87.52	92.85	93.71	94.53	96.40
CNN	**89.45**	**88.14**	94.96	95.99	95.41	96.89



Şekil 5.13. Dengeli veri kümelerinin F-Puanı

Dengesiz veri kümeleri üzerinde yüksek hızda örnekleme tekniğini uygularken, modellerin performansında gözle görülür bir iyileşme gözlemlenmiştir. Bu, modellerin performansını artırmak için iyi bir uygulamadır, ancak bazı durumlarda doğru çözüm değildir. Sunulan sonuçlar ayrıca derin öğrenme modellerinin KNN ve RF dışında makine öğrenimi modellerine kıyasla nispeten daha iyi performans sunduğunu göstermiştir.

Derin öğrenme modellerini uygulamanın ana dezavantajlarından biri, büyük miktarda veri gerektirmesidir. Modellerin eğitimi için kullanılan veri kümelerindeki mevcut örneklerin miktarı yeterince büyük değildir ve bu tezdeki sınırlamalardan biridir. Bu nedenle, yüksek doğruluk bir zorluk olmaya devam etmiştir.

5.6. Trafik Mühendisliği Yaklaşımının Değerlendirmesi

Bu bölümde, farklı senaryolar uygulanarak uygulanan trafik mühendisliğinin genel performansı değerlendirilmiştir. Bu senaryolar, trafik aktarım hızındaki değişiklikleri gözlemlemek için farklı tekniklerin uygulanmasını içerir ve sunucu ve ana bilgisayarlar arasındaki trafik aktarım hızı, dosyalar gönderilerek ve gerçek trafiği taklit eden trafikler oluşturularak ölçülmüştür.

Sistem performansını değerlendirmek için kullanılan senaryolar şunlardır:

- İlk senaryo, normal bant genişliği tüketimini ve trafik aktarım hızını gözlemlemek için hiçbir şeyin uygulanmadığı temel senaryodur.
- İkinci senaryoda trafik akışına trafik şekillendirme uygulanmıştır.
- Üçüncü senaryoda ise trafik şekillendirme ve yük dengeleme teknikleri uygulanmıştır.

Mevcut OpenFlow kuyruk konfigürasyonunu desteklemez. Bu nedenle, konfigürasyon ve kurulum OpenFlow anahtarında yapılmıştır. Ayrıca, OpenFlow anahtarı, bir paket zamanlayıcı türü belirlememize ve akışlara öncelik vermemize izin verir. Paket zamanlayıcı, her bir kuyruk için farklı öncelik değerleri ayarlayarak akışların önceliklendirilmesine izin verir ve bu da akışın işlenmesini sağlar. Bu nedenle önce düşük öncelikli akışlar kurulur ve ardından diğer akışlar kurulmaktadır. HTB paket zamanlayıcı, paketleri her kuyruğa arzu edilen hızı karşılamak amacıyla geciktirmek için kullanılır ve minimum ve maksimum bant genişliği kapasitesini yapılandırmayı sağlar. Bant genişliği kapasitesinin yapılandırılması, o kuyruğun trafik akışları için minimum bant genişliğini garanti etmek amacıyla her kuyruk için trafik şekillendirme uygulanmasına izin verir ve bant genişliği açlığını önler. Yük dengeleme tekniği, trafik sıkışıklığını azaltmak, verimi artırmak ve mevcut fiziksel bağlantıları mümkün olduğunca etkili kullanmak için iş yükünü fiziksel bağlantıların sayısına bölme üzerine çalışan OpenFlow anahtarında uygulanmıştır.

Her bağlı cihaz arasındaki bağlantıların bant genişliği 100 Mbps olarak, gecikme ise 5ms ile 30ms arasında ayarlanmıştır. Tablo 5.9'da bant genişliği gereksinimleri ve öncelikleri ile uygulamaların türü gösterilmiştir. VOIP gibi gerçek zamanlı uygulamalar, gecikmenin bu tür uygulamalarda kritik bir öğe olması sebebiyle yüksek öncelikli olarak sınıflandırılmıştır ve çok fazla bant genişliği tüketmezler. Dolayısıyla bu tür uygulamalar için 10 Mbps tahsis edilmiştir. Ses Akışı, Tarama, E-posta gibi uygulamalar orta derecede öncelikli olarak görülmüştür çünkü gerçek zamanlı uygulamalar kadar düşük gecikme gerektirmezler. Son olarak, Dosya Aktarımı, P2P, Video Akışı gibi uygulamalar düşük öncelikli olarak tanımlanmıştır çünkü yüksek bant genişliği gerektirirler.

Çizelge 5.9. Kuyruk konfigürasyonu

Kuyruk	**Öncelik**	**Uygulama**	**Bant Genişliği**
0	Yüksek	Gerçek zamanlı trafik VOIP	10 Mbps
1	Orta	Ses Akışı, Tarama, E-posta	35 Mbps
2	Düşük	Dosya Aktarımı, P2P, Video Akışı	55 Mbps

Bu çalışma kapsamında, iki farklı ağ aracı (NetCat ve iPerf3) veri kümesindeki trafiği taklit eden trafiği oluşturmak için kullanılmaktadır. Ana bilgisayarlar ve sunucular arasında dosya gönderilirken TCP protokolü kullanılmıştır.

İlk senaryo, ana bilgisayarlar ve sunucular arasında düzenli bant genişliği tüketimini ve trafik aktarım hızını gözlemlemek için hiçbir kural veya politikanın uygulanmadığı temel senaryodur. Çizelge 5.10'de ilk senaryo için ana bilgisayarlar ve sunucular arasındaki veri aktarım hızı gösterilmektedir.

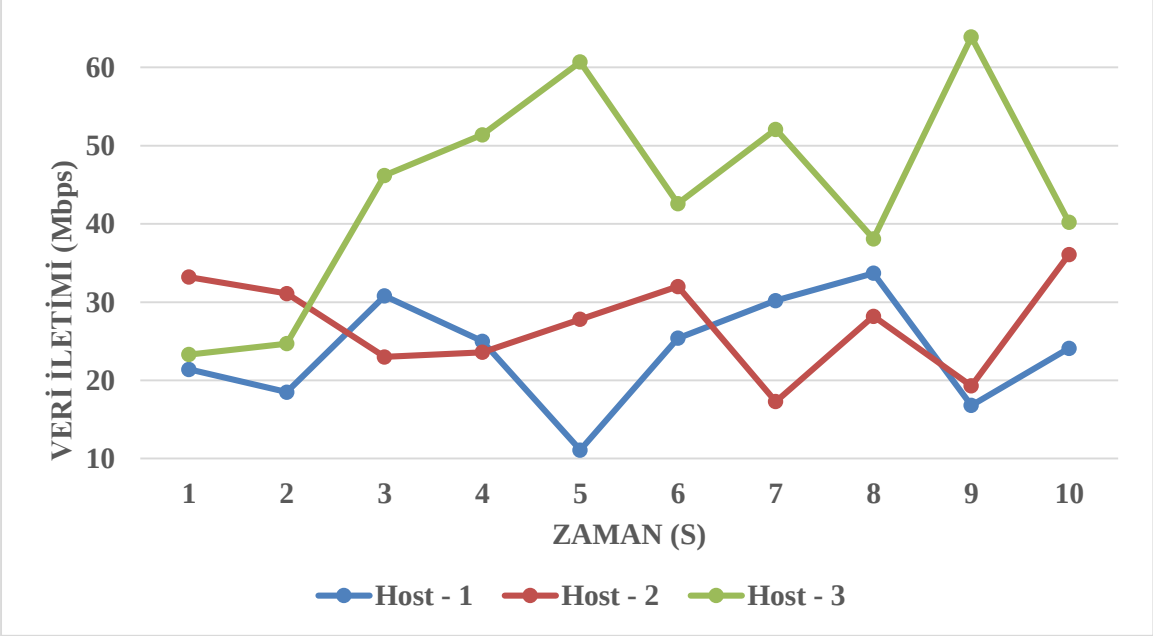
Çizelge 5.10. İlk senaryo için veri aktarım hızları (Mbps)

Zaman	**Host - 1**	**Host - 2**	**Host - 3**
1	21.4	33.2	23.3
2	18.5	31.1	24.7

Çizelge 5.10. İlk senaryo için veri aktarım hızları (Mbps)

3	30.8	23	46.2
4	25	23.6	51.4
5	11.1	27.8	60.7
6	25.4	32	42.6
7	30.2	17.3	52.1
8	33.7	28.2	38.1
9	16.8	19.3	63.9
10	24.1	36.1	40.2

Şekil 5.14'te Çizelge 5.10'da sunulan veriler gösterilmektedir. Sunulan verilerden, ana bilgisayarlar ve sunucular arasındaki veri aktarım hızı düzenlenmemiştir. Dolayısıyla, veri hızında beklenmedik bir artış görülebilmektedir. Akışları düzenlememe sorunu, ağın genel performansı üzerinde bir etkiye sahip olmaktadır. Bu yüzden bir ana bilgisayar aynı ağdaki diğer ana bilgisayarlardan daha fazla trafik oluşturduğunda diğer ana bilgisayarlar için bant genişliği eksikliğine neden olur ve kaotik bir durum yaratılmıştır. Böyle bir senaryo, yetersiz tasarlanmış ağlarda görülebilmiştir.



Şekil 5.14. İlk senaryonun sonucu

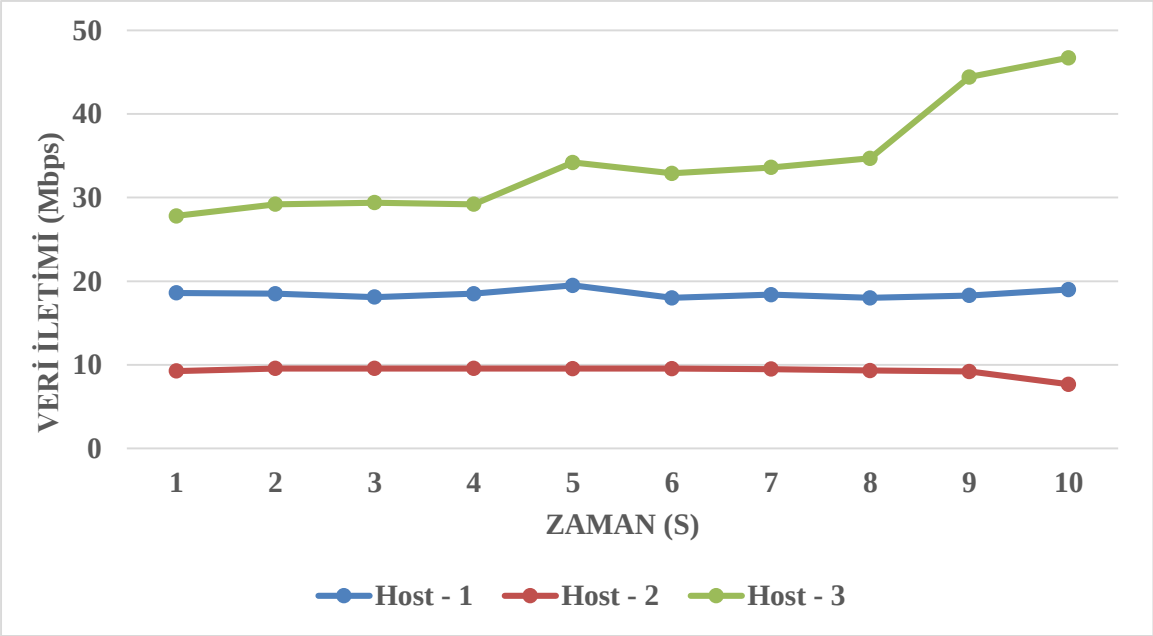
İkinci senaryoda, trafik akışlarında trafik şekillendirme uygulanmıştır. Öncelikle sıraların iletilmesini beklerken paketlerin tutucusu olarak çalışan OpenFlow switch (ayrıca paket

zamanlayıcı yardımıyla paket iletimini organize etme konusunda da çalışır) üzerine kuyruklar kurulmalıdır. Paket akışları sıraya konularak belirli bir kuyrukla eşleştirilir. Her akış, o sıranın konfigürasyonuna göre değerlendirilmektedir. Her bir kuyruğun minimum ve maksimum bant genişliği kapasitesinin yapılandırılması bant genişliğinin tahsis edilmesine izin verir ve bu da her uygulama türü için minimum bant genişliğini garanti eder ve bant genişliği açlığını önler. Bununla birlikte, akışlar üzerinde trafik şekillendirme uygulamak tıkanıklığı ve bant genişliği açlığını önlemek için akışların hızını ve hacmini kontrol ederek ağ mevcut bant genişliğini yönetmeye yardımcı olmaktadır. Çizelge 5.11'de ikinci senaryo için ana bilgisayarlar ve sunucular arasındaki veri aktarım hızı gösterilmektedir.

Çizelge 5.11. İkinci senaryo için veri aktarım hızları (Mbps)

Zaman	**Host - 1**	**Host - 2**	**Host - 3**
1	18.6	9.26	27.8
2	18.5	9.57	29.2
3	18.1	9.57	29.4
4	18.5	9.57	29.2
5	19.5	9.56	34.2
6	18	9.55	32.9
7	18.4	9.49	33.6
8	18	9.33	34.7
9	18.3	9.21	44.4
10	19	7.67	46.7

Şekil 5.15'te Çizelge 5.11’de sunulan veriler gösterilmektedir. Şekilde akışlara trafik şekillendirme uygulanması, ağ performansını ve bant genişliğinin kullanılabilirliğini artırmaya yardımcı olmaktadır. Ayrıca, bant genişliği kullanımındaki beklenmedik artışlar önlenerek bant genişliğinin düzenlenmesi her çeşit uygulama için kaynakların garantiye alınmasına yardımcı olmaktadır. İlk senaryo ile karşılaştırıldığında yapılandırılmış kapasite dahilinde olan bant genişliği tüketimindeki bir artışın diğer ana bilgisayarların bant genişliğini etkilemediği ve veri aktarım hızının daha istikrarlı olduğu açıktır.



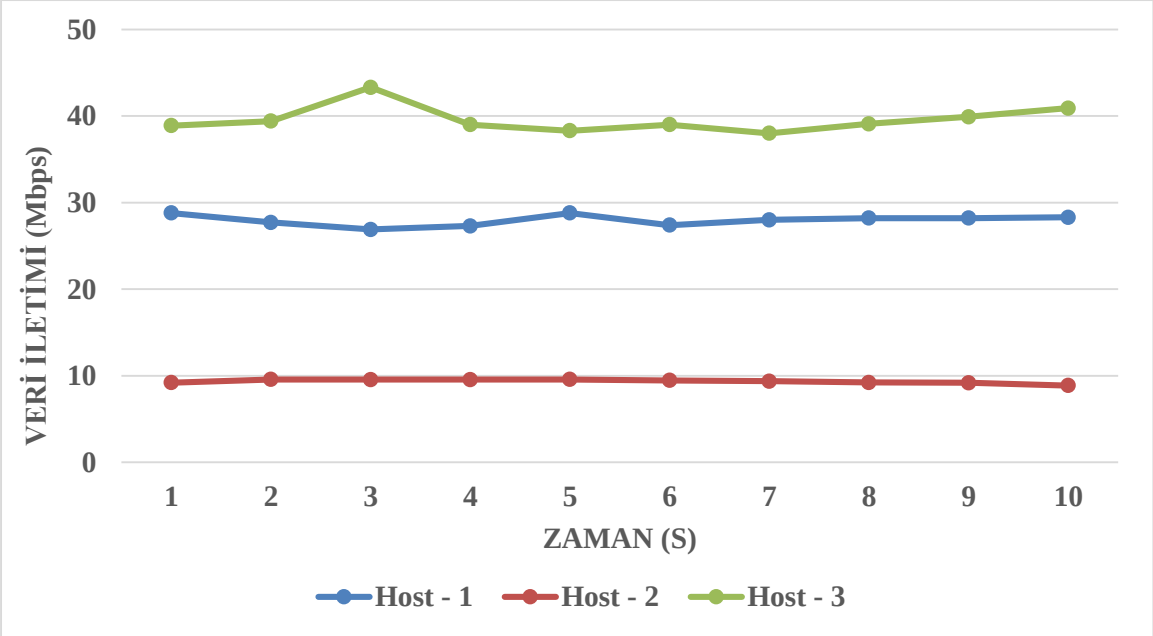
Şekil 5.15. İkinci senaryonun sonucu

Son senaryoda, ikinci senaryoda kullanılan aynı konfigürasyon uygulanmıştır. Ayrıca, OpenFlow anahtarında tıkanıklıkları azaltmak, verimi artırmak ve mevcut fiziksel bağlantıları mümkün olduğunca etkili kullanmak için iş yükünü fiziksel bağlantı sayısına bölmek üzerine çalışan bir yük dengeleme tekniği kurulmuştur. Çizelge 5.12'de üçüncü senaryo için ana bilgisayarlar ve sunucular arasındaki veri aktarım hızı gösterilmektedir.

Çizelge 5.12. Üçüncü senaryo için veri aktarım hızları (Mbps)

Zaman	Host - 1	Host - 2	Host - 3
1	28.8	9.2	38.9
2	27.7	9.57	39.4
3	26.9	9.56	43.3
4	27.3	9.56	39
5	28.8	9.57	38.3
6	27.4	9.46	39
7	28	9.38	38
8	28.2	9.22	39.1
9	28.2	9.19	39.9
10	28.3	8.87	40.9

Şekil 5.16'da Çizelge 5.12'de sunulan veriler gösterilmektedir. İkinci senaryo ile karşılaştırıldığında ana bilgisayar-1 ve ana bilgisayar-3'ün performansının arttığı fark edilebilir. Kontrolcü, kaynak ve hedef arasındaki birden fazla rotanın hesaplanmasına olanak tanıyan ağın küresel bir görünümüne sahiptir. Yoğun akışları iletmek için aynı rotanın seçilmesinin (tıkanıklığa neden olur) mümkün olmasına rağmen genel olarak en az atlama sayısına sahip yolu seçer. Aynı sayıda atlama sayısına sahip rotalar için port tabanlı yük dengeleme genel yük dengelemenin üstesinden gelmek amacıyla her bağlantı noktasının kapasitesini dikkate alır. Böylece, kurulu yük dengeleme tekniği verimi artırarak mevcut fiziksel bağlantıları mümkün olduğu kadar etkili kullanabilmektedir.



Şekil 5.16. Üçüncü senaryonun sonucu

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, çeşitli uygulamalar arasında bant genişliği tahsisini iyileştirmek için yazılım tanımlı ağlarda derin öğrenme tabanlı bir trafik mühendisliği yaklaşımı önerilmektedir. İşlenen ve müşterilere, çalışanlara ve iş ortaklarına sunulan veri miktarındaki büyük artış nedeniyle, ağ trafiğinin kaynak kullanımı ve planlaması ağ yönetiminin önemli süreçleri haline gelmiştir. Bu bağlamda tasarlanan sistem, trafik akışlarının sınıflandırılması için derin öğrenme tabanlı bir model geliştirilmeyi ve trafik sınıflarına göre ağdaki trafik akışlarının hizmet kalitesi dikkate alınarak yönlendirilmesi için bir trafik mühendisliği modülünü YTA kontrolcü üzerinde gerçekleştirilmeyi amaçlamıştır. Önerilen yaklaşım, derin öğrenme yardımıyla çeşitli uygulamalardan gelen ağ trafiğini tanımlayarak Hizmet Kalitesini iyileştirmeye ve belirlenen akışları her bir kuyruğun farklı bir önceliğe sahip olduğu çoklu kuyruklara dağıtmaya çalışmaktadır. Ağ bant genişliğini ve gelen trafiğin hacmini yönetmek için ağ trafiğine trafik şekillendirme uygulanmıştır. Derin sinir ağı modelleri ve 1-B evrişimli sinir ağı modeli gibi derin öğrenme modelleri, bir kurumsal ağ yöneticisi tarafından önceden belirlenebilecek kurallara göre trafik akışlarını tanımlamak ve sınıflandırmak için kullanılmıştır. Derin öğrenme ve makine öğrenmesi algoritmaları, trafik sınıflandırma için kullanılarak performansları karşılaştırılmıştır. Trafik mühendisliği ve Hizmet Kalitesi garantisi için doğru trafik sınıflandırması gereklidir. Hem şifreli hem şifresiz trafik ile kullanılabildikleri için, modellerin eğitimi için trafiğin zamana dayalı özellikleri seçilmiştir. Eğitim için kullanılan veri setinde, 18'den fazla temsili uygulamadan toplanan ve 23 zamanla ilgili özellik içeren sekiz tür trafik vardır. Ayrıca veri kümesi, nihai sonuçlar üzerindeki akış zaman aşımı etkisini incelemek için 5, 10, 15, 30, 60 ve 120 saniyelik farklı akış zaman aşımlarında toplanan trafiği içermektedir. Benzer uygulama trafiğini tek bir sınıfta gruplamak, sınıflandırma sürecini basitleştirir. Bu nedenle, sekiz trafik türü, gecikmenin ve bant genişliğinin önemli kriterler olduğu, önceliğe bağlı olarak üç farklı sınıfa ayrılmıştır.

Sınıf-1'deki örnek sayısı, diğer iki sınıfa kıyasla nispeten düşüktü. Bu nedenle, sınıfları dengelemek için SMOTE adlı bir yüksek hızda örnekleme yöntemi kullanılmıştır. Daha sonra modellerin performansını artırmak için veriler ölçeklendirilmiştir. Ayrıca, modelleri değerlendirmek için farklı ölçütler kullanılmış ve derin öğrenme modellerinin sonucu

değerlendirilmiş ve k-en yakın komşular, destek vektör makinesi, karar ağacı ve rastgele orman algoritmaları gibi çeşitli makine öğrenme modelleriyle karşılaştırılmıştır.

Mevcut OpenFlow kuyruk konfigürasyonunu desteklemediği için kuyruk konfigürasyonu ve kurulum OpenFlow anahtarında yapılmıştır. Kurallar, her trafik uygulama grubu için bant genişliğini sınırlamak ve garanti altına almak için her anahtar bağlantı noktasına yüklenmiştir. Her kuyruğun önceden tanımlanmış gereksinimlere göre önceliği vardır ve tanımlanan akışlar farklı önceliklere sahip birden çok kuyruğa dağıtılır. Trafik şekillendirme tekniği, tıkanıklığı önlemek için gelen akışların hızını ve hacmini kontrol ederek mevcut ağ bant genişliğini yönetmek için uygulanmıştır.

Ağın performansını artırmak için, OpenFlow anahtarına yüklenen bir yük dengeleme tekniği, tıkanıklıkları azaltmak, verimi artırmak ve mevcut fiziksel bağlantıları olabildiğince verimli kullanmak için iş yükünü fiziksel bağlantı sayısına bölmeye çalışmaktadır. Her bir çıkış bağlantı noktasının kapasitesini göz önünde bulundurarak genel yük dengelemesi gerçekleştirilmektedir.

Derin öğrenme modellerinin değerlendirilme aşamasında, DNN ve 1-B CNN'nin performansı, KNN, SVM, DT ve RF gibi bazı makine öğrenimi modelleriyle karşılaştırılmış ve değerlendirilmiştir. Sonuçlar, 1-B CNN ve DNN'nin 5 s ve 10 s zaman aşımlarında yakalanan trafikte yüksek doğruluk elde edebildiğini, KNN ve RF ise 15 s, 30 s, 60 s ve 120 s zaman aşımlarında yakalanan trafikte yüksek doğruluk elde edebildiğini göstermiştir. CNN ve DNN, 15 s, 30 s, 60 s ve 120 s zaman aşımlarında DT ve SVM'den daha iyi doğruluk elde etmeyi başarmıştır.

Trafik mühendisliği yaklaşımının değerlendirmesi için üç farklı senaryo uygulanmıştır. İlk senaryo, normal bant genişliği tüketimini ve trafik aktarım hızını gözlemlemek için kullanılmıştır. İkinci senaryoda, trafik akışına trafik şekillendirme uygulanmıştır. Üçüncü senaryoda trafik şekillendirme ve yük dengeleme teknikleri uygulanmıştır. Sonuç, tanımlanan akışa trafik şekillendirmenin uygulanmasının, gelen akışların hızını ve hacmini kontrol ederek ağın performansını, bant genişliği kullanılabilirliğini artırdığını gösterilmiştir. Ayrıca, her uygulama türü için bant genişliğinin garanti edilebildiği ve yük dengeleme tekniği ile trafik şekillendirme uygulanmasının yanı sıra her bir kuyruğun veriminde bir artış sağlanabildiği gösterilmiştir.

Karmaşık topolojilere sahip gerçek YTA ortamlarında çalışmak ve farklı ağ uygulamalarından gelen trafik hacmini artırmak için sistem tasarımını iyileştirmek ve geliştirmek için atılabilecek bazı adımlar vardır. Gelecekteki çalışmalarda, sistemi daha geniş ağ topolojileri üzerinde test etmek ve veri aktarımına daha fazla ana bilgisayarın dahil olmasının etkilerini incelemek planlanmaktadır. Başka bir ilginç araştırma fikri, ana bilgisayar başına daha yüksek trafik hacminin ve trafiği oluşturan uygulamalardaki daha fazla çeşitliliğin etkilerini incelemektir.

KAYNAKLAR

Al Khater, N., and Overill, R. E. (2015, October). Network traffic classification techniques and challenges. *Tenth International Conference on Digital Information Management (ICDIM)*. IEEE, 43-48.

Amaral, P., Dinis, J., Pinto, P., Bernardo, L., Tavares, J., and Mamede, H. S. (2016, November). Machine learning in software defined networks: Data collection and traffic classification. *IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 1-5.

Archanaa, R., Athulya, V., Rajasundari, T., and Kiran, M. V. K. (2017, January). A comparative performance analysis on network traffic classification using supervised learning algorithms. *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 1-5.

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. (2002). *Rfc3272: Overview and principles of internet traffic engineering*, 4-8, 21-26.

Balan, D. G., and Potorac, D. A. (2009, July). Linux HTB queuing discipline implementations. *First International Conference on Networked Digital Technologies*. IEEE,122-126.

Bisong, E. (2019). Google Colaboratory. *In Building Machine Learning and Deep Learning Models on Google Cloud Platform.* Apress, Berkeley, CA, 59-64.

Bramer M. (2020) *Estimating the Predictive Accuracy of a Classifier*. Principles of Data Mining. Undergraduate Topics in Computer Science. Springer, London, 82.

Chao, Y., and Hongxia, W. (2010, June). Developed Dijkstra shortest path search algorithm and simulation. *International Conference on Computer Design and Applications*. IEEE. Vol. 1, V1-116.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321-357.

Choubey, R. N., Amar, L., Khare, S., and Venkanna, U. (2019, December). Internet Traffic Classifier Using Artificial Neural Network and 1D-CNN. *International Conference on Information Technology (ICIT).* IEEE, 291-296.

Dargan, S., Kumar, M., Ayyagari, M. R., and Kumar, G. (2019). A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 1-22.

Doherty, J. (2016). *SDN and NFV simplified: a visual guide to understanding software defined networks and network function virtualization.* Addison-Wesley Professional, 149-155.

Fan, Z., and Liu, R. (2017, August). *Investigation of machine learning based network traffic classification*. International Symposium on Wireless Communication Systems (ISWCS). IEEE, 1-6.

Goli, Y. D., and Ambika, R. (2018, December). Network Traffic Classification Techniques-A Review. *International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE, 219-222.

Goransson, P., Black, C., and Culver, T. (2016). *Software defined networks: a comprehensive approach*. Morgan Kaufmann, 61-66, 72.

Hartpence, B., and Kwasinski, A. (2019, April). Fast internet packet and flow classification based on artificial neural networks. *SoutheastCon*. IEEE, 1-8.

Huang, D., Chowdhary, A., and Pisharody, S. (2018). *Software-Defined networking and security: from theory to practice*. CRC Press, 90-95.

İnternet: Dugan, J. O. N., Elliott, S., Mah, B., Poskanzer, J., & Prabhu, K. (2014). Iperf—the network bandwidth measurement tool. Web: https://iperf.fr adresinden 11 Nisan 2020'da alınmıştır.

Jeong, S., Lee, D., Hyun, J., Li, J., and Hong, J. W. K. (2017, September). *Application-aware traffic engineering in software-defined network.* 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 315-318.

Karakus, M., and Durresi, A. (2017). Quality of service (QoS) in software defined networking (SDN): A survey. *Journal of Network and Computer Applications*, 80, 200-218.

Kaur, K., Singh, J., and Ghumman, N. S. (2014, February). Mininet as software defined networking testing platform. *International Conference on Communication, Computing and Systems (ICCCS)*, 139-42.

Kaur, S., Singh, J., and Ghumman, N. S. (2014, August). Network programmability using POX controller. *ICCCS International Conference on Communication, Computing and Systems*, IEEE, Vol. 138, 70.

Kelleher, J. D. (2019). *Deep Learning*. Mit Press, 1-5.

Kurth, M., Gras, B., Andriesse, D., Giuffrida, C., Bos, H., and Razavi, K. (2020, May). NetCAT: Practical cache attacks from the network. *41st IEEE Symposium on Security and Privacy (SandP).*

Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., and Ghorbani, A. A. (2017, February). Characterization of Tor Traffic using Time based Features. *International Conference on Information Systems Security and Privacy (ICISSP)*, 253-262.

Li, W., Meng, W., and Kwok, L. F. (2016). A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, 68, 126-139.

Li, X., Yan, J., and Ren, H. (2017). Software defined traffic engineering for improving Quality of Service. *China communications*, 14(10), 12-25.

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

Mendiola, A., Astorga, J., Jacob, E., and Higuero, M. (2016). A survey on the contributions of software-defined networking to traffic engineering. *IEEE Communications Surveys and Tutorials*, 19(2), 918-953.

Mir, N. F. (2015). *Computer and communication networks.* Pearson Education, 607-613.

Mohammed, A. R., Mohammed, S. A., and Shirmohammadi, S. (2019, July). Machine Learning and Deep Learning Based Traffic Classification and Prediction in Software Defined Networking. *IEEE International Symposium on Measurements and Networking (MandN).* IEEE, 1-6.

Neghabi, A. A., Navimipour, N. J., Hosseinzadeh, M., and Rezaee, A. (2018). Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature. *IEEE Access*, 6, 14159-14178.

Parvat, A., Chavan, J., Kadam, S., Dev, S., and Pathak, V. (2017, January). A survey of deep-learning frameworks. *International Conference on Inventive Systems and Control (ICISC).* IEEE, 1-7.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., and Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *The Journal of machine Learning research*, 12, 2825-2830.

Pfaff, B., and Davie, B. (2013). The open vswitch database management protocol. *Internet Requests for Comments*, RFC Editor, RFC, 7047. 3-5.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., ... and Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5), 1-36.

Rebala, G., Ravi, A., and Churiwala, S. (2019). *An Introduction to Machine Learning.* Springer, 1-4, 19-22.

Rezaei, S., and Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine*, 57(5), 76-81.

Robertazzi, T. G., and Shi, L. (2020). *Networking and Computation*. Springer International Publishing, 166-175.

Shalev-Shwartz, S., and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms.* Cambridge university press, 167-177.

Singh, S., and Jha, R. K. (2017). A survey on software defined networking: Architecture for next generation network. *Journal of Network and Systems Management*, 25(2), 321-374.

Stallings, W. (2015). *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud.* Addison-Wesley Professional, 80-85, 93, 268-273.

Sun, X., and Mendez, F. (2018, December). Dynamic Flow Scheduling in a Software-Defined Network Environment. *IEEE 4th International Conference on Computer and Communications (ICCC).* IEEE, 2405-2409.

Szigeti, T., Hattingh, C., Barton, R., and Briley Jr, K. (2013). *End-to-End QoS Network Design: Quality of Service for Rich-Media and Cloud Networks.* Cisco press, 1-13.

Umadevi, K. S., Pranay, M. S. S., and Rachana, K. (2017, April). Multilevel queue scheduling in software defined networks. *Innovations in Power and Advanced Computing Technologies (i-PACT).* IEEE, 1-4.

Xiao, G., Wenjun, W., Jiaming, Z., Chao, F., and Yanhua, Z. (2017, December). An OpenFlow based Dynamic Traffic Scheduling strategy for load balancing. *3rd IEEE International Conference on Computer and Communications (ICCC).* IEEE, 531-535.

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., and Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(1), 393-430.

Xu, J., Wang, J., Qi, Q., Sun, H., and He, B. (2018, September). Deep neural networks for application awareness in YTA-based network. *IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP).* IEEE, 1-6.

Yan, J., and Yuan, J. (2018, August). A survey of traffic classification in software defined networks. *1st IEEE International Conference on Hot Information-Centric Networking (HotICN). IEEE*, 200-206.

EKLER

EK-1. Derin Öğrenme ve Makine Öğrenimi Modellerini Eğitmek için Kullanılan Parametreler

Çizelge 1.1. Karar ağacı modelini eğitmek için kullanılan parametreler.

Veri Kümesi	**criterion**	**max_depth**	**min_samples_leaf**	**min_samples_split**	**Tür**
5 sn	entropy	7	2	15	**Dengeli**
10 sn	entropy	4	1	2	**Dengeli**
15 sn	entropy	10	1	2	**Dengesiz**
15 sn	gini	10	1	4	**Dengeli**
30 sn	gini	10	1	2	**Dengesiz**
30 sn	gini	10	1	4	**Dengeli**
60 sn	gini	10	1	2	**Dengesiz**
60 sn	entropy	10	1	3	**Dengeli**
120 sn	entropy	10	1	2	**Dengesiz**
120 sn	gini	10	1	2	**Dengeli**

Çizelge 1.2. K-en yakın komşu modelini eğitmek için kullanılan parametreler.

Veri Kümesi	**n_neighbors**	**p**	**weights**	**Tür**
5 sn	16	1	uniform	**Dengeli**
10 sn	15	1	distance	**Dengeli**
15 sn	15	1	distance	**Dengesiz**
15 sn	6	2	distance	**Dengeli**
30 sn	9	1	distance	**Dengesiz**

EK-1. (devam) Derin Öğrenme ve Makine Öğrenimi Modellerini Eğitmek için Kullanılan Parametreler

Çizelge 1.2. (devam) K-en yakın komşu modelini eğitmek için kullanılan parametreler.

30 sn	8	1	distance	**Dengeli**
60 sn	10	1	distance	**Dengesiz**
60 sn	6	1	distance	**Dengeli**
120 sn	9	1	distance	**Dengesiz**
120 sn	5	2	distance	**Dengeli**

Çizelge 1.3. Destek vektör makinesi modelini eğitmek için kullanılan parametreler.

Veri Kümesi	**C**	**tol**	**Tür**
5 sn	25.0	0.01	**Dengeli**
10 sn	25.0	0.1	**Dengeli**
15 sn	25.0	0.01	**Dengesiz**
15 sn	25.0	0.1	**Dengeli**
30 sn	25.0	0.01	**Dengesiz**
30 sn	25.0	0.00001	**Dengeli**
60 sn	25.0	0.1	**Dengesiz**
60 sn	25.0	0.00001	**Dengeli**
120 sn	25.0	0.00001	**Dengesiz**
120 sn	25.0	0.00001	**Dengeli**

EK-1. (devam) Derin Öğrenme ve Makine Öğrenimi Modellerini Eğitmek için Kullanılan Parametreler

Çizelge 1.4. Rastgele Orman modelini eğitmek için kullanılan parametreler.

Veri Kümesi	bootstrap	criterion	min_samples_leaf	min_samples_split	n_estimators	Tür
5 sn	FALSE	entropy	2	9	100	**Dengeli**
10 sn	FALSE	entropy	1	5	100	**Dengeli**
15 sn	FALSE	entropy	1	5	100	**Dengesiz**
15 sn	FALSE	entropy	1	2	100	**Dengeli**
30 sn	FALSE	entropy	1	4	100	**Dengesiz**
30 sn	FALSE	entropy	2	3	100	**Dengeli**
60 sn	FALSE	entropy	1	8	100	**Dengesiz**
60 sn	FALSE	entropy	1	2	100	**Dengeli**
120 sn	FALSE	entropy	1	3	100	**Dengesiz**
120 sn	FALSE	entropy	1	2	100	**Dengeli**

Çizelge 1.5. Derin sinir ağı modelini eğitmek için kullanılan parametreler.

Veri Kümesi	Input layer	Hidden layer	Dropout layer	Hidden layer 2	Hidden layer 3	Output layer	Tür
5 sn	23	32		32	32	3	**Dengeli**
10 sn	23	64		32	64	3	**Dengeli**
15 sn	23	128		32	128	3	**Dengesiz**
15 sn	23	128		96	128	3	**Dengeli**

EK-1. (devam) Derin Öğrenme ve Makine Öğrenimi Modellerini Eğitmek için Kullanılan Parametreler

Çizelge 1.5. (devam) Derin sinir ağı modelini eğitmek için kullanılan parametreler.

30 sn	23	128		64	128	3	**Dengesiz**
30 sn	23	128		32	128	3	**Dengeli**
60 sn	23	128		64	128	3	**Dengesiz**
60 sn	23	128		32	128	3	**Dengeli**
120 sn	23	128		64	128	3	**Dengesiz**
120 sn	23	128	0.2	64	128	3	**Dengeli**

Derin sinir ağı modelini eğitmek ReLU, gizli katmanlar için kullanılan aktivasyon fonksiyonudur ve Softmax, çıktı katmanı için kullanılan aktivasyon fonksiyonudur.

Çizelge 1.6. Evrişim sinir ağı modelini eğitmek için kullanılan parametreler.

Veri Kümesi	**Conv 1D (filter size)**	**MaxPooling 1D (pool size)**	**Conv 1D 2 (filter size)**	**MaxPooling 1D 2 (pool size)**	**Dropout layer**	**Hidden layer**	**Dropout layer 2**	**Hidden layer 2**	**Tür**
5 sn	64	2	128	2	0.2	64	0.2	32	**Dengeli**
10 sn	64	2	128	2	0.2	64	0.2	32	**Dengeli**
15 sn	32	2	64	2	0.2	128		64	**Dengesiz**

EK-1. (devam) Derin Öğrenme ve Makine Öğrenimi Modellerini Eğitmek için Kullanılan Parametreler

Çizelge 1.6. (devam) Evrişim sinir ağı modelini eğitmek için kullanılan parametreler.

15 sn	32	2	128	2		128		32	**Dengeli**
30 sn	64	2	128	2	0.2	128		64	**Dengesiz**
30 sn	32	2	64	2	0.2	128		64	**Dengeli**
60 sn	32	2	64	2	0.2	128		64	**Dengesiz**
60 sn	32	2	64	2	0.2	128		64	**Dengeli**
120 sn	64	2	128	2	0.2	128	0.2	64	**Dengesiz**
120 sn	64	2	128	2	0.2	128		64	**Dengeli**

Evrişim sinir ağı modelini eğitmek aşamasında ReLU, gizli katmanlar için kullanılan aktivasyon fonksiyonudur ve Softmax, çıktı katmanı için kullanılan aktivasyon fonksiyonudur. (3, 3), Conv1D katmanında kullanılan çekirdek boyutudur.

GAZİLİ OLMAK AYRICALIKTIR.