T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK FPGA TABANLI GERÇEK ZAMANLI YENİ BİR TRAFİK SINIFLANDIRMA MİMARİSİ TASARIMI

TUNCAY SOYLU

DOKTORA TEZİ

HESAPLAMALI BİLİMLER ANABİLİM DALI

TEZ DANIŞMANI: DOÇ. DR. OĞUZHAN ERDEM

EDİRNE, 2018

Tuncay SOYLU'nun hazırladığı **"Makine Öğrenmesi Yöntemleri Kullanılarak FPGA Tabanlı Gerçek Zamanlı Yeni Bir Trafik Sınıflandırma Mimarisi Tasarımı"** başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından **Hesaplamalı Bilimler Anabilim Dalı**nda bir **Doktora Tezi** olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad):	İmza
Prof. Dr. Mustafa ÖZCAN	
Doç. Dr. Cüneyt BAZLAMAÇCI	
Doç. Dr. Tansu FİLİK	
Doç. Dr. Oğuzhan ERDEM	
Dr. Öğr. Üyesi Aydın ÇARUŞ	

Tez Savunma Tarihi: 15 / 08 / 2018

Bu tezin Doktora tezi olarak gerekli şartları sağladığını onaylarım.

İmza

DOÇ. DR. OĞUZHAN ERDEM

Tez Danışmanı

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Murat YURTCAN

Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ

HESAPLAMALI BİLİMLER ANABİLİM DALI DOKTORA PROGRAMI

DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

15 / 08 / 2018

Tuncay SOYLU

İmza

Doktora Tezi

MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK FPGA TABANLI GERÇEK ZAMANLI YENİ BİR TRAFİK SINIFLANDIRMA MİMARİSİ TASARIMI

T.Ü. Fen Bilimleri Enstitüsü

HESAPLAMALI BİLİMLER

ÖZET

İnternetin kullanımının artması birçok konuda olduğu gibi internet trafik sınıflandırmaya olan ihtiyacı giderek arttırmaktadır.

İnternet trafik sınıflandırma internet servis sağlayıcılar (ISS), kamu kurumları veya özel şirketler için oldukça önemli bir kullanım alanı oluşturmaktadır. Bunun nedeni, her bir kuruluşun kendi internet trafiğini izlemek istemesidir. Buna ilaveten internet trafik sınıflandırma trafik önceliklendirme, trafik şekillendirme ve bant genişliği paylaşımı sağlama gibi *ağ yönetim görevleri* için kullanılmaktadır. Ayrıca, ağ güvenliği, dinamik erişim kontrolü ve saldırı tespiti gibi *internet ağ güvenliği sağlama* ve *yeni nesil internet ağ mimarilerinin tasarımı*n da internet trafik sınıflandırmaya ihtiyaç vardır.

İnternet trafik sınıflandırma mimarisi tasarımında önemli kriterlerden biri yüksek hızlarda yüksek sınıflandırma doğruluğunu destekleyebiliyor olmasıdır. Özellikle gerçek zamanlı trafik sınıflandırma yapabilmek için 100+ Gpbs hızlara ulaşabilen trafik sınıflandırma mimarisine ihtiyaç vardır. Önerilen mevcut sınıflandırma yöntemlerinin çoğu yazılım tabanlı çözümler olmakla birlikte bu çözümlerin bu hızlara ulaşabilmesi oldukça güçtür. Dolayısıyla yüksek hızlarda internet trafik sınıflandırma yapabilmek için yazılım tabanlı çözümler yerine yüksek hızlarda trafik sınıflandırma yapabilen donanım tabanlı mimariler tercih edilmektedir. Donanım tabanlı mimariler, yüksek hızlara

ulaşmalarının yanı sıra bellek verimliliği, çıkan yüksek iş oranı (throughput), dinamik güncelleme ve düşük gecikme gibi trafik sınıflandırma kriterlerinde de yüksek başarım sağlamaktadırlar.

Önerilen mevcut internet trafik sınıflandırma çözümlerinden port tabanlı, DPI tabanlı ve sezgisel tabanlı yöntemler şifreli trafik altında ve dinamik port atamalarında düşük performans göstermektedir. Son yıllarda, özellikle şifreli trafik ve dinamik port atamaları altında yüksek doğruluk elde eden makine öğrenmesi (machine learning - ML) tabanlı yöntemler tercih edilmektedir. ML tabanlı yöntemler, trafik akışının (traffic flow) sadece istatistiksel özelliklerine bakarak trafik sınıflandırma yapmaktadır. Yüksek hızlı ve gerçek zamanlı ML tabanlı trafik sınıflandırma için donanım mimarileri tasarımına ihtiyaç vardır.

Bu tezde gerçek zamanlı, yüksek hızlarda ve doğrulukta trafik sınıflandırma yapabilmek için makine öğrenmesi tabanlı ve donanım üzerinde gerçeklenen trafik sınıflandırma yöntemleri incelendi. Bu tezin ana katkısı olarak paralel boru hatlı (pipeline) mimariler üzerinde uygulanan yüksek hızlarda ve doğrulukta sınıflandırma yapabilen makine öğrenmesi tabanlı *Genişletilmiş Simple CART* (E-SC) mimarisi önerilmiştir. Aynı zamanda, benzer katkıları sağlamak amacıyla her bir uygulama sınıfından bir ağaç elde edilen ve ağaçları bitmaplerle zenginleştirilmiş iki aşamalı hibrit bir yapı olan *Simple CART Ormanları* (SCF*)* mimarisi önerilmiştir. Son olarak yüksek doğrulukta ve oldukça düşük sınıflandırma gecikmesi sağlayan tek adımlı *Bitmap Kodlu Simple CART* (BC-SC) veri yapısı önerilmiştir. *Alanda Programlanabilir Kapı Dizilimleri* (FPGA) tabanlı paralel ve boru hattı üzerinde tasarlanmıştır.

Yıl : 2018

Sayfa Sayısı : 98

Anahtar Kelimeler : İnternet Trafik Sınıflandırma, Makine Öğrenmesi, Gerçek Zamanlı, FPGA tabanlı

Ph.D. Thesis

DESIGNING A NEW FPGA-BASED REAL-TIME TRAFFIC CLASSIFICATION ENGINE USING MACHINE LEARNING TECHNIQUES

Trakya University Institute of Sciences

Department of Computational Sciences

ABSTRACT

Increasing use of the internet is growing the need for internet traffic classification as for many subjects.

Internet traffic classification covers a very important usage area for private companies, public institutions, governments and Internet Service Providers (ISPs). The reason for this is that each institution wants to monitor its own internet traffic. In addition, internet traffic classification is used for network management tasks such as traffic prioritization, traffic shaping and bandwidth sharing provisioning. Additionally, internet networking security such as dynamic access control and intrusion detection, and the design of next-generation internet network architectures also require internet traffic classification.

One of the important criteria in designing internet traffic classification architecture is that it can support high classification accuracy at high speeds. Especially, in order to make real time traffic classification, reaching up to 100+ Gbps speed internet traffic classification architecture is needed. While most of the proposed classification methods are software based solutions, it is very difficult for these solutions to reach such speeds. Thus, in order to make internet traffic classification faster, hardware-based architectures that can make traffic classification in high speeds are preferred on behalf of software-based solutions. Hardware-based architectures are highly successful on traffic

classification criterion such as low latency, dynamic update, high throughput, memory efficiency besides reaching higher speeds.

Among the proposed Internet traffic classification solutions, port based, DPI based and intuitive based methods show poor performance under encrypted traffic and dynamic port assignments. In recent years, machine learning (ML) based methods, which obtain high accuracy especially under encrypted traffic and dynamic port assignments, have been preferred. ML based methods perform traffic classification by only examining the statistical features of traffic flow. There is a need for designing hardware architectures for high-speed and real-time ML based traffic classification.

In this thesis, machine-learning based hardware traffic classification methods were examined in order to achieve real time, high speed and accuracy traffic classification. As the main contribution on this dissertation, *Extended-Simple CART* (E-SC) architecture is proposed that can achieve high accuracy and speed on parallel pipelined architectures. In the meantime, in order to make the similar contributions, *Simple CART Forests* (SCF) architecture is proposed, which is a two-staged hybrid structure and whose trees are enriched with bitmaps. Finally, a single-step *Bitmap Coded Simple CART* (BC-SC) data structure is recommended that provides high accuracy and fairly low classification latency. The BC-SC data structure is designed on *Field Programmable Gate Arrays* (FPGAs) based parallel and pipeline.

Year : 2018

Number of Pages : 98

Keywords : Internet Traffic Classification, Machine Learning, Real-Time, on FPGA

Her şeyden çok sevdiğim sevgili kızım Deniz Naz'a…

ÖNSÖZ

Doktora Tez danışmanlığımı üstlenerek çalışmalarımın yürütülmesi sırasında yardımlarını esirgemeyen danışman hocam Sayın Doç. Dr. Oğuzhan ERDEM'e sonsuz teşekkür ederim.

Doktora sürecinde yaptığım akademik çalışmalarımın tümünde desteğini esirgemeyen değerli hocam Sayın Dr. Öğr. Üyesi Aydın CARUS'a çok teşekkür ederim.

Tez izleme komisyonunda yer alarak değerli önerileri için Sayın Prof. Dr. Mustafa ÖZCAN'a ve Sayın Dr. Öğr. Üyesi E. Serdar GÜNER'e teşekkür ederim.

Hesaplamalı Bilimler Anabilim Dalı'nı öneren ve sonrasında tüm doktora ders ve tez aşamalarında gösterdiği anlayışla desteğini esirgemeyen Sayın Dr. Öğr. Üyesi Hayati ARDA'ya çok teşekkür ederim. Ayrıca, eğitim hayatıma katkıda bulunan tüm değerli hocalarıma teşekkür ederim.

Eğitim hayatım boyunca yanımda yer alan ve maddi ve manevi desteklerini esirgemeyen ağabeyim Durmuş SOYLU'ya, ayrıca Enes Asansör Mühendislik İnşaat ve Taahhüt San. Tic. Ltd. Şti. ortaklarına ve çalışanlarına teşekkür ederim.

Ayrıca, bana her zaman destek olan ve sabır gösteren annem Saniye SOYLU, babam Hüseyin SOYLU ve eşim Yıldız SOYLU'ya çok teşekkür ederim.

Tuncay SOYLU

İÇİNDEKİLER

SEMBOLLER VE KISALTMALAR

ACT	Application Class Trees	Uygulama Sınıfı Ağaçları
AVG	Average Packet Size	Ortalama Paket Boyutu
BC-SC	Bit Vector Coded Simple CART	Bit Vektör Kodlu Simple CART
BRAM	Block Random Access Memory	Blok Rasgele Erişimli Bellek
CART	Classification and Regression Tree	Sınıflandırma ve Regresyon Ağacı
DP	Destination Port Number	Hedef Port Numarası
DPI	Deep Packet Inspection	Derin Paket İnceleme
E-SC	Extended Simple CART	Genişletilmiş Basit CART
FPGA	Field Programmable Gate Array	Alanda Programlanabilir Kapı Dizilimleri
FT	Feature Tree	Feature Ağacı
Gbps	Giga Bit Per Second	Milyar Bit/Saniye
HDL	Hardware Description Language	Donanım Tanımlama Dili
IANA	Internet Assigned Numbers Authority	İnternet Tahsisli Sayılar ve İsimler Kurumu
IP	Internet Protocol	İnternet Protokolü
ISS	Internet Service Provider	İnternet Servis Sağlayıcı

k-NN	k-Nearest Neighborhood	k-En Yakın Komşu
MAX	Maximum Packet Size	Maksimum Paket Boyutu
MIN	Minimum Packet Size	Minimum Paket Boyutu
ML	Machine Learning	Makine Öğrenmesi
MCPS	Million Classification Per Second	Milyon Sınıflandırma Paketi/Saniye
MHz	Mega Hertz	Milyon Hertz
ns	Nano Second	Nano Saniye
PaR	Place And Route	Yer Ve Rota
PE	Processing Element	İşleme Elemanı
Prtcl	Protocol	Protokol
SCF	Simple CART Forest	Simple CART Ormanı
SP	Source Port Number	Kaynak Port Numarası
SRAM	Static Random Access Memory	Statik Rasgele Erişimli Bellek
SVM	Support Vector Machine	Destek Vektör Makineleri
VAR	Variance of Packet Size	Paket Boyutu Varyansı
YSA	Artificial Neural Network	Yapay Sinir Ağları

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

BÖLÜM 1

GİRİŞ

1.1. Trafik Sınıflandırma

Trafik sınıflandırma, trafik şekillendirme, akış önceliklendirme, dinamik erişim kontrolü ve izinsiz giriş tespiti gibi birçok önemli ağ yönetimi görevleri için temel oluşturmaktadır (Tong, vd., 2013). Trafik sınıflandırma, gelen ağ trafiğini, her biri önceden tanımlanmış anlaşmalar temelinde kullanıcılara farklılaştırılmış hizmetler sağlamak için farklı şekilde işlenebilen bir dizi uygulama sınıfına ayırma işlemidir (Soylu, vd., 2018). Trafik sınıflandırma ile ilgili bir başka tanım, internet trafiğinin gerçek zamanlı akışı sırasında internet yönlendiricilerinde (router, switch vb.) uygulama türünün belirlenmesidir. Bir diğer tanım, trafik sınıflandırma, internet trafik akışının belirli özelliklerine bakılarak bu trafiğin daha önceden bilinen uygulama sınıflarının (Http, SSH, Skype, MSN, PPLive vb.) hangisine ait olduğuna karar verilmesi işlemidir. Gerçek zamanlı (real-time) trafik sınıflandırma ise, ağ yönetimi ve güvenlik uygulamaları için internet trafiği akarken (gerçek zamanlı) internet trafiğinin sınıflandırılması işlemidir (Soylu, vd., 2017; Soylu, vd., 2018).

Trafik sınıflandırma için genellikle internet paketlerinin başlık (header) bilgileri kullanılır. Bir internet trafik akışı (traffic flow), aynı 5 *çokuzlu* (tuple) başlık alanlarını (SA, DA, SP, DP ve Protokol) paylaşan bir dizi olarak temsil edilir (Soylu, vd., 2017; Soylu, vd., 2018). İnternet paketlerinin klasik başlık bilgileri yanında bir akışın ilk n paketinin ortalama, maksimum, minimum, varyans paket büyüklüğü ve akış seviyesi gibi

istatistiksel özellikleri de trafik sınıflandırmada kullanılmaktadır (Tong, vd., 2013; Soylu, vd., 2017; Soylu, vd., 2018).

1.1.1. Performans Kriterleri

İnternet yönlendiricileri (router veya switch) ile trafik sınıflandırmada öne çıkan performans kriterleri; *doğruluk* (accuracy), *çıkan iş oranı* (throughput), *gecikme* (latency), *bellek kullanımı* (memory usage)'dır.

- ***Doğruluk (Accuracy):*** Makine Öğrenmesi tabanlı internet trafik sınıflandırma yöntemlerinde en önemli kriterlerden biri doğruluktur. Doğruluk,

$$Doğruluk = \frac{doğru\ sınıflandırılan\ paket\ sayısı}{toplam\ paket\ sayısı} (\%)$$

  denklemi ile bulunur. Literatürde, doğruluk oranının %90'ın üzerinde yer alması kullanılan algoritmanın o sınıflandırma veri seti için uygun algoritma olduğu anlamına gelmektedir. Bu oran %100'e ne kadar yakın ise o kadar iyi bir sınıflandırıcı tasarlanmış demektir. Fakat doğruluk tek başına yeterli bir kriter değildir.
- ***Çıkan İş Oranı (Throughput):*** Gerçek zamanlı trafik sınıflandırma için $100 + Gbps$ (Giga bit per second – Giga bit/saniye) üzerinde paket işleyebilen bir sınıflandırıcı tasarımına ihtiyaç duyulur. Ayrıca iş oranı $MCPS$ (Milyon Sınıflandırma Paketi/Saniye – Million Classification Per Second) olarak da hesaplanmaktadır. Sınıflandırma süresi ne kadar düşük olursa saniyedeki sınıflandırılan paket sayısı da o oranda artar. Paket sayısının artması ise tasarlanan sınıflandırıcıdan çıkan iş oranını artırır. Sınıflandırma süresi, veri yapısı ve tasarlanan mimari ile doğrudan ilişkilidir.
- ***Gecikme (Latency):*** İnternet trafik sınıflandırmada gecikmenin minimum olması istenmektedir. Gecikme, gelen paket değerinin tasarlanan mimaride sınıflandırıcıya girişinden itibaren sınıflandırıcıdan çıkana kadar geçirdiği toplam süre olarak tanımlanır.

- ***Bellek Kullanımı (Memory Usage):*** Bellek kullanımı, tasarlanan veri yapısı ile doğrudan ilişkili olup ağaç tipi veri yapılarında ağaç derinliği arttıkça kullanılan bellek miktarı da üstel olarak artmaktadır.

1.2. Motivasyon

İnternet hızlarındaki artış, saniyede yüzlerce gigabit hızlarda paket işleyebilecek gerçek zamanlı internet trafik sınıflandırıcı mimarilerine olan gereksinimi arttırmaktadır. Ancak, günümüzde kullanılan trafik sınıflandırma araçları saniyede birkaç gigabiti destekleyebilmektedirler. SSH ve Skype gibi şifrelenmiş trafiğin sınıflandırılması oldukça zor ve başarım oranı düşüktür (Jiang ve Gokhale, 2010). Ayrıca günümüzde birçok uygulama aynı port numarasını kullanmakta olup (80 port numarası gibi) bu uygulamaların klasik trafik sınıflandırıcılar ile sınıflandırılmasında da düşük doğruluk oranı elde edilmektedir. Bu tip uygulamaların, internet paketlerinin istatistiksel özellikleri ile sınıflandırma yapan makine öğrenmesi yöntemleri ile sınıflandırılmasında yüksek doğruluk oranı elde edilmektedir (Jiang ve Gokhale, 2010; Qu ve Prasanna, 2014). Bu nedenle yüksek hızlarda sınıflandırma için donanım tabanlı sınıflandırıcılara olan ihtiyacı arttırmaktadır (Tong, vd., 2013; Qu ve Prasanna, 2014; Gandhi, vd., 2014). Bu tez çalışmasında yüksek hızlarda, internet trafik sınıflandırma işlemini gerçekleştirecek paralel boru hattı yapısını kullanan donanım mimarileri önerilmektedir.

1.3. Katkılar

Bu tez çalışmasında internet trafik sınıflandırma için üç farklı veri yapısı ve bu veri yapılarını destekleyen sınıflandırma mimarileri önerilmektedir. Önerilen veri yapılarında *makine öğrenmesi* (ML) tabanlı teknikler kullanılmaktadır. 8 uygulama sınıfı ile Simple CART makine öğrenmesi algoritması kullanılarak analizler yapılmaktadır. Bu yönü ile de bu tez literatürdeki en çok uygulama sınıfı ile gerçek zamanlı sınıflandırma yapabilen çalışmalar arasında yer almaktadır. Seçilen uygulama sınıfları, geleneksel mevcut sınıflandırıcılar tarafından sınıflandırılması zor olan P2P servislerini, şifreli trafik ve anında mesajlaşma (instant messenger) uygulamalarını da içermektedir. Önerilen veri

yapılarını destekleyen sınıflandırma mimarileri güncel FPGA'lar üzerinde işlenmektedir. Önerilen *veri yapıları ve sınıflandırma mimarileri* aşağıda özetlenmiştir;

1. Bir makine öğrenmesi algoritması olan Simple CART karar ağacı donanıma uygun hale gelebilmesi için iki aşamalı bir veri yapısına dönüştürülmüştür. Geliştirilen veri yapısı, birinci aşamada her özellikten bir ağaç barındıran ve ikinci aşamada ise Simple CART karar ağacını *bitmap* ile zenginleştirilen hibrit veri yapısıdır. Önerilen veri yapısı *Genişletilmiş Simple CART* (Extended–Simple CART (*E-SC*)) veri yapısı olarak adlandırılmaktadır. *E-SC* veri yapısında birinci aşamada elde edilen *arama anahtarı*, ikinci aşamada aranmaktadır (Soylu, vd., 2017).
   - FPGA aygıtı kullanılarak elde edilen sonuçlara göre tasarım saniyede 1741 milyon paket ($MCPS$) veya 557 $Gbps$ (minimum 40 baytlık paket boyutu için) hızda sınıflandırma işlemi gerçekleştirmektedir.
   - Önerilen *E-SC* veri yapısının doğruluk oranı %96.8125 olarak elde edilmiştir.
   - Önerilen *E-SC* veri yapısı için gecikme ise 22 (Aşama 1'de 6 ve Aşama 2'de 16) saat döngüsü (clcock cycle) olarak bulunmaktadır.
   - Veri yapısının saat süresi (clock time) 4.60 ns ve sınıflandırma frekansı ise 217 MHz'dir.
   - Veri yapısının toplam bellek gereksinimi 3632 $Byte$'tır.
2. Simple CART karar ağacı, birinci aşamada herhangi bir değişiklik yapılmadan, ikinci aşamada ise uygulama sınıflarından oluşan *uygulama sınıfı ağaçları* içeren iki aşamalı veri yapısına dönüştürülmüştür. Önerilen veri yapısı *Simple CART Ormanı* (Simple CART Forest – *SCF*) veri yapısı olarak adlandırılmaktadır. *SCF* veri yapısında birinci aşamada elde edilen *arama anahtarı*, ikinci aşamada aranmaktadır (Soylu, vd., 2018).
   - FPGA aygıtı kullanarak elde edilen sonuçlara göre, n ormanda bulunan uygulama sınıfı sayısını göstermek üzere, $n = 1, 2, 4$ ve 8 için sırasıyla $2616, 2669, 2247$ ve 1741 milyon paket ($MCPS$) veya $837, 854, 735$ ve 557 $Gbps$ (minimum 40 baytlık paket boyutu için) iş oranı elde edilmiştir.

- Önerilen *SCF* veri yapısı $n = 1, 2, 4$ ve 8 için sırasıyla %94.0625, %96.6719, %95.0781 ve %96.8125 doğruluk ile sınıflandırma yapmaktadır.
- Önerilen *SCF* veri yapısı için gecikme $n = 1, 2, 4$ ve 8 için sırasıyla 15, 15, 18 ve 22 (Aşama 1'de 6 ve Aşama 2'de sırasıyla 9, 9, 12 ve 16) saat döngüsü olarak bulunmaktadır.
- Önerilen *SCF* veri yapısı için saat süresi (clock time) $n = 1, 2, 4$ ve 8 için sırasıyla 3.06, 3.00, 3.48 ve 4.60 ns ve sınıflandırma frekansı ise sırasıyla 322.5, 333.3, 285.7 ve 217.0 MHz'dir.
- Veri yapısının toplam bellek gereksinimi $n = 1, 2, 4$ ve 8 için sırasıyla 2275, 2284, 2767 ve 3632 Bayt'tır.

3. Simple CART algoritmasının doğruluk oranı sabit kalacak şekilde önerilen yeni yapı Simple CART algoritmasını doğrudan tek adıma düşürmektedir. Önerilen yeni veri yapısı *Bit Vektör Kodlu Simple CART* (Bit Vector Coded Simple CART – (*BC-SC)*) veri yapısı olarak adlandırılmıştır.
   - FPGA aygıtı ile elde edilen sonuçlara göre, (*BC-SC* ve *BC-SC*^{opt}) sırasıyla 2034 ve 2857 milyon paket veya 650 ve 914 $Gbps$ (minimum 40 baytlık paket boyutu için) iş oranı elde edilmiştir.
   - Önerilen *BC-SC* veri yapıları için gecikme (*BC-SC* ve *BC-SC*^{opt}) sırasıyla 3.93 ns ve 2.80 ns olarak elde edilmiştir.
   - Önerilen *BC-SC* veri yapıları için saat süresi (*BC-SC* ve *BC-SC*^{opt}) sırasıyla 3.93 ns ve 2.80 ns ve sınıflandırma frekansı ise sırasıyla 253.2 ve 357.1 MHz'dir.
   - *BC-SC* veri yapılarının toplam bellek gereksinimi *(BC-SC* ve *BC-SC*^{opt}) sırasıyla 4.07 ve 2.02 $Kbyte$'tır.

Bu tez çalışmasında önerilen veri yapıları farklı uygulamalar için de genişletilebilir. Makine öğrenmesi algoritmalarının kullanıldığı ve gerçek zamanlı veya çok hızlı sonuçlar üretmeyi gerektiren uygulamalar için önerilen veri yapıları kullanılabilirdir.

1.4. Tez Organizasyonu

Bu tezin kalan bölümleri şu şekilde düzenlenmiştir. *Bölüm 2*'de *İnternet Trafik Sınıflandırma Yaklaşımları* anlatılmaktadır. *Bölüm 3*'de *Gerçek Zamanlı İnternet Trafik Sınıflandırma Mimarisi Tasarım Süreci* ayrıntılı olarak anlatılmaktadır. *Bölüm 4*'de Simple CART algoritması için önerilen ilk veri yapısı olan *E-SC* veri yapısı ve FPGA uygulaması anlatılmaktadır. Simple CART Ormanı ve FPGA uygulaması *Bölüm 5*'de anlatılmaktadır. Simple CART algoritması için önerilen tek adımlı veri yapısı *BC-SC* ve FPGA uygulaması *Bölüm 6*'da anlatılmaktadır. *Bölüm 7*'de ise tez sonuçlarının değerlendirildiği ve gelecek çalışma önerilerinin sunulduğu *Sonuç* bölümü yer almaktadır.

BÖLÜM 2

İNTERNET TRAFİK SINIFLANDIRMA YAKLAŞIMLARI

2.1. İnternet Trafik Sınıflandırma Yaklaşımları

Günümüzde internet trafik sınıflandırma için kullanılan yöntemler 4 ana sınıfta toplanabilir; (i) *port numarası tabanlı yaklaşımlar*, (ii) *derin paket inceleme (DPI) tabanlı yaklaşımlar*, (iii) *sezgisel tabanlı yaklaşımlar* ve (iv) *makine öğrenmesi tabanlı yaklaşımlar* (Qu ve Prasanna, 2014).

2.1.1. Port Numarası Tabanlı Yaklaşımlar

Port numarası tabanlı trafik sınıflandırma, internet paketlerinin sadece TCP veya UDP port numaraları kontrol edilerek gerçekleştirilen bir yaklaşımdır (Karagiannis, vd., 2004a; Karagiannis, vd., 2004b; Karagiannis, vd., 2005). TCP veya UDP port numarasına göre sınıflandırmada, uygulama sonrası *İnternet Tahsisli Sayılar ve İsimler Kurumu* (Internet Assigned Numbers Authority – IANA)'da kayıtlı hedef port numarasına bakılarak uygulama sınıfı belirlenir. Özellikle bazı uygulamaların (Kazaa, Napster) IANA'da kayıtlı kendi port numaraları olmayabilir. Bununla birlikte, yeni uygulamalar kendilerini gizlemek için öngörülemeyen port numaralarını kullanabilmektedirler (Karagiannis, vd., 2004a). Dolayısıyla mevcut sınıflandırıcılar port numaralarına bakarak yanlış sınıflandırma yapabilmektedirler. Bu yanlış sınıflandırma oranı belirtilen sorunlar nedeniyle her geçen gün artmaktadır. Bu nedenle port numarası tabanlı yaklaşımlar, güvenirliliğini giderek kaybetmektedir. Literatür incelendiğinde son zamanlarda daha kompleks sınıflandırma yöntemlerinin tercih edildiği görülmektedir (Harthi, 2015).

2.1.2. Derin Paket İnceleme (DPI) Tabanlı Yaklaşımlar

Port numarası tabanlı tekniklerinin yanlış tahminlere neden olması ve 2000'li yıllarda virüslerin iyice yaygınlaşması paketin içeriğinin derin analiz edilmesini gerektirmiştir. Dolayısıyla yapılan çalışmaların yönünün de *Saldırı Tespit Sistemleri* (Intrusion Detection Systems – IDS)'ne doğru yöneltilmesine neden olmuştur. Bu nedenle de *derin paket inceleme* (Deep Packet Inspection – *DPI*) tabanlı yaklaşımlar ortaya çıkmıştır (Harthi, 2015). *DPI tabanlı yaklaşımlar*, her bir uygulamanın imzalarını tanımlamak için *imza analizi* (Signature Analysis) yöntemini uygulamaktadır. Bu imzalar daha sonra belirli bir trafiği karşılaştırmak için kullanılan bir referans veritabanında birleştirilmektedir. Bu yöntem, sınıflandırma motorunun o uygulamayı tanımlayacağı şekilde gerçekleştirilmektedir. Daha sonra veri tabanında, referans güncellemeleri sık sık yapılmalıdır. Böylece son gelişmeler, uygulamalarla birlikte mevcut protokollerle birleştirilmektedir (Karagiannis, vd., 2004a; Moore ve Papagiannaki, 2005; Haffner, vd., 2005; Chen ve Wasikowski, 2008).

DPI tabanlı yaklaşımların en önemli sorunu, kullanıcıların paket içeriklerini gizlemek için şifreli trafik kullanmasıdır. Ayrıca hükümetler gizlilik yönetmelikleri ile üçüncü şahısların paket içeriklerini meşru bir şekilde incelemelerini de kısıtlamaktadır. Bu nedenle DPI tabanlı yaklaşımların popülaritesi de giderek azalmaktadır (Nguyen ve Armitage, 2008).

2.1.3. Sezgisel Tabanlı Yaklaşımlar

Sezgisel (Heuristic) tabanlı yaklaşımlar, bilinmeyen internet trafiğini sınıflandırmak için makine öğrenmesi algoritmalarını kullanan istatistiksel tabanlı bir sınıflandırma tekniğidir. İnternet trafiğinin örüntülerine dayalı olarak internet trafiğini sınıflandırmaktadır. Sezgisel tabanlı yaklaşımlar örüntü kalıpları depolamak için geniş bellek gereksinimine ihtiyaç duymaktadırlar (Tong, vd., 2013). Ayrıca düşük sınıflandırma doğruluğuna sahiptirler. Belirtilen nedenlerle sezgisel tabanlı yaklaşımların kullanımı da azalmaktadır.

2.1.4. Makine Öğrenmesi Tabanlı Yaklaşımlar

Araştırmacılar, *Port tabanlı*, *DPI tabanlı* ve *Sezgisel tabanlı yaklaşımlar*ın yukarıda anlatılan sorunları nedeniyle internet trafik sınıflandırma için yeni arayışlara yönelmişlerdir. *Makine öğrenmesi* (*ML*) *tabanlı yaklaşımlar*, trafik akışlarının istatistiksel özelliklerini tanıyarak internet trafiğini sınıflandırmaktadırlar (Moore ve Papagiannaki, 2005; Sen, vd., 2004; Qu ve Prasanna, 2015). ML tabanlı yaklaşımlarda, sınıflandırma kuralları kümesi önceden sınıflandırılmış veya etiketli akış kümesinden çıkarılır. Ardından, yeni bilinmeyen akışlar bu kurallara veya oluşturulan modellere göre sınıflandırılır. Yüksek doğruluğu, şifreli trafik koşulları altında çalışabilmesi ve son zamanlardaki dinamik internet trafik ortamında sağlamlığı nedeniyle ML tabanlı sınıflandırma yaklaşımları son zamanlarda araştırmacıların dikkatini çekmiştir (Tong, vd., 2013; Tong, vd., 2014; Qu, ve Prasanna, 2014; Qu, ve Prasanna, 2015; Soylu, vd., 2017; Soylu, vd., 2018).

Makine öğrenmesi tabanlı öğrenme, *sınıflandırma* (denetimli öğrenme), *kümeleme* (denetimsiz öğrenme), *ilişki* ve *sayısal tahmin* olmak üzere 4 ana öğrenme grubuna ayrılmaktadır. İnternet trafik sınıflandırmada kullanılan çoğu makine öğrenmesi teknikleri *sınıflandırma* (denetimli öğrenme*)* ve *kümeleme* (denetimsiz öğrenme*)* yöntemlerine odaklanmaktadır. *Sınıflandırma,* bilinmeyen örnekleri sınıflandırmak için sınıflandırma kurallarının kümesini oluşturur. *Kümeleme,* önceden rehberlik olmadan benzer özelliklere sahip örneklerin gruplanmasıdır (Witten ve Frank, 2005). Bu tezde sadece *denetimli öğrenme* (sınıflandırma) yöntemlerine odaklanılmıştır.

2.1.4.1. Denetimli Öğrenme

Önceden tanımlı olan sınıfa yeni örneklerin eşleme yöntemi ile sınıflandırılmasına dayanan öğrenme yöntemidir (Reich ve Fenves, 1991). Denetimli öğrenmede *eğitim* ve *test fazı* olmak üzere iki faz yer almaktadır. *Eğitim aşamasında*, öğrenme aşamasında sağlanan veriler incelenir ve bir sınıflandırma modeli oluşturulur. *Test aşaması,* eğitim aşamasında inşa edilen sınıflandırma modeli bilinmeyen örnekleri sınıflandırmak için kullanılır (Nguyen ve Armitage, 2008).

2.1.4.2. Denetimli Öğrenme Algoritmalarının Değerlendirilmesi

ML algoritmalarının daha doğru bir sonuç elde edebilmesi için değerlendirilmesi gerekmektedir. ML algoritmalarını değerlendirmek için mevcut uygulamalarda yaygın olarak *çapraz doğrulama* (cross-validation) adı verilen bir yöntem kullanılır. Bu yöntemde eğitim için, önceden etiketlenmiş örneğin bir kısmı ve test için de kalanı kullanılır. Veri kümesi, N adet eşit bölüme ayrılır. Her bölümde etiketlenmiş örneğin $1/N$ oranı test için kullanılırken $(N-1)/N$ oranı eğitim için kullanılır. Sonra test için kullanılan küme değiştirilerek aynı işlem tekrarlanır ve sonuçta her örnek, test için sadece bir kez kullanılmış ve böylece prosedür N kez tekrar edilmiştir. Literatürde N değeri genellikle 10 olarak alınmakta olup bunun da *çapraz doğrulama* yönteminin doğruluğunu arttırdığı görülmektedir (Witten ve Frank, 2005).

Tasarımı yapılacak sınıflandırıcı için tercih edilecek algoritmaların değerlendirilmesinde yüksek doğruluk mu yoksa düşük maliyet mi olduğuna karar verilmelidir. Karar verme sürecinde uygulama sınıfının kullanım alanı etkili olmaktadır. Ayrıca ticari veya operasyonel öncelikler de karar verme sürecini etkilemektedir (Nguyen ve Armitage, 2008).

2.1.4.3. Özellik Seçimi Algoritmaları

Özellik seçimi olarak bilinen süreç, bir makine öğrenmesi tabanlı sınıflandırıcı inşa etmede anahtar doğruluk hedeflerine ulaşmak için gerekli olan özelliklerin en küçük alt kümesinin belirlenmesi işlemidir. Özellik seti kalitesi, makine öğrenmesi algoritmasının performansı için çok önemlidir. İlgisiz ve gereğinden fazla özellik kullanmak, çoğu makine öğrenmesi algoritmalarının doğruluğunda olumsuz etkilere yol açmaktadır (Nguyen ve Armitage, 2008).

2.1.4.4. Operasyonel Zorluklar

İnternet trafik sınıflandırma yapabilmenin operasyonel zorlukları vardır ve kullanılacak algoritmaların bu zorlukları aşabilmesi istenmektedir.

Zamanında ve sürekli sınıflandırma, trafik sınıflandırmanın, internet trafiği akarken (real-time) yapılması istenir. Zamanında sınıflandırma yapabilmek için sınıflandırıcı her akışın mümkün olduğunca az paketini kullanarak karara ulaşması

gerekir. Ancak, az sayıda paket ve özellik kümesi kullanımı istenmeyen hatalara neden olabilir. Dolayısıyla tasarlanacak olan sınıflandırıcının kendi istatistiki bilgilerini sürekli güncellemesi gerekmektedir (Nguyen ve Armitage, 2006a; Nguyen ve Armitage, 2006b).

Bellek ve işlemci etkin kullanım, tasarlanan sınıflandırıcının sınırsız kaynak kullanımı mümkün olmadığı için ve fazla kaynak kullanımının sınıflandırıcı maliyetini aşırı derecede arttırdığı için hesaplama kaynakları verimli kullanılmalıdır. Özellikle büyük ölçekli trafik sınıflandırma yaklaşımlarında bu önem daha da artmaktadır (Nguyen ve Armitage, 2006a; Nguyen ve Armitage, 2006b).

Taşınabilirlik ve sağlamlık, önerilen trafik sınıflandırma modeli değişik ağ alanlarında kullanılabiliyorsa *taşınabilir* sayılmaktadır. Paket kaybı, trafik şekillendirme, paket parçalanması ve dengesizlik durumları gibi ağ katmanı karşısında tutarlılık sağlayabiliyorsa *sağlam* kabul edilmektedir. Aynı zamanda bir sınıflandırıcı yeni trafik uygulamalarını tespit edebilmesi durumunda da *sağlam* olarak kabul edilmektedir (Nguyen ve Armitage, 2006a; Nguyen ve Armitage, 2006b).

2.2. İnternet Trafik Sınıflandırma Yaklaşımlarının Karşılaştırılması

İnternet trafik sınıflandırma yaklaşımlarının karşılaştırılması *Çizelge 2.1*'de gösterilmektedir (Wang, 2013). Bir sınıflandırıcı tasarımı yapılırken çizelgede belirtilen özellikler dikkate alınarak planlama yapılmalıdır. Yani tasarımı gerçekleştirilecek olan sınıflandırma mimarisinin beklentilerinin ne olduğu önceden belirlenmesi gerekmektedir. Eğer tasarlanan bir mimarinin yüksek doğruluk ile bilinmeyen trafik şartlarında da çalışması ve bunlara ek olarak sınıflandırıcı karmaşıklığının da az olması isteniyorsa makine öğrenmesi tabanlı yaklaşımlarının kullanılması zorunlu hale gelecektir.

Çizelge 2.1'den de anlaşılacağı gibi hangi yaklaşımın kullanılacağı tasarımı yapılacak olan yapının önceliğinin ne olacağına bağlıdır.

Çizelge 2.1 İnternet Trafik Sınıflandırma Yaklaşımlarının Karşılaştırılması

	Port Tabanlı	**DPI Tabanlı**	**Sezgisel Tabanlı**	**ML Tabanlı**
Doğruluk	Düşük	Yüksek	Yüksek	Yüksek
Özellik Çıkarım Zorluğu	Düşük	Yüksek	Yüksek	Özellik Setine Bağlı
Karmaşıklık	Düşük	Yüksek	Orta	Düşük - Orta
Şifreli Trafik	Evet	Hayır	Evet	Evet
Ön Bilgi	Port Listesi	İmzalar	Etiketli Veri	Etiketli Veri
Bilinmeyen Trafik Reddetme	Evet	Evet	Hayır	Hayır

2.3. Literatür Taraması

2.3.1. Makine Öğrenmesi Tabanlı Çalışmalar

Makine öğrenmesi tabanlı teknikler kullanılarak bir ağ trafiği kontrolü, devre anahtarlamalı telekomünikasyon şebekesinde çağrı tamamlamayı maksimize etmek amacıyla 1990 yılında önerilmiştir (Silver, 1990). Makine öğrenmesi tabanlı teknikler ilk olarak 1994 yılında saldırı tespiti için internet trafik akış sınıflandırmada kullanılmıştır (Frank, 1994). Bu çalışma, internet trafiğinin sınıflandırmasında makine öğrenmesi tekniklerini kullanan birçok çalışma için başlangıç noktası olmuştur (Nguyen ve G. Armitage, 2008).

Roughan ve arkadaşları, internet trafik sınıflandırma yapabilmek için *k-NN*, *LDA* ve *QDA* makine öğrenmesi algoritmalarını önerdiler (Roughan, vd., 2004). Önerilen sistem özellik sayısı olarak *paket seviyesi* (ortalama, varyans vb.), *akış seviyesi* (akış süresi, akış başına veri hacmi), *bağlantı seviyesi* (TCP, veri dağılımı), *iç akış/bağlantı özellikleri* (akışlardaki paketler arası varış süresi) ve *çoklu akış özellikleri* (uç sistemlerdeki aynı set arasındaki çoklu eşzamanlı bağlantı) olmak üzere 5 kategoride sınıflandırma yapmaktadır. Düşünülen özelliklerin çoğunda akış süresi ve paket uzunluğu vardır. Sınıflandırma yaparken ftp, Telnet ve RealMedia olarak 3 uygulama sınıfı; Telnet, ftp, RealMedia ve DNS olarak 4 uygulama sınıfı; DNS, Ftp, Https, Kazaa, RealMedia, Telnet ve www olarak ise 7 uygulama sınıfı incelenmektedir. Sınıflandırma işlemi 10 kez *çapraz doğrulama yöntemi* kullanılarak değerlendirilmektedir. Hata oranına göre analiz yapılmakta olup en düşük hata oranı ilk tanımlanan 3 uygulama sınıflı yapıda olduğu görülmektedir.

Moore ve Zuev, internet trafik sınıflandırma yapabilmek için denetimli *Naive Bayes* makine öğrenmesi tekniğini kullanmayı önerdiler (Moore ve Zuev, 2005). Manuel olarak sınıflandırılan (akış içeriğine dayalı) 248 tam akış tabanlı özellik, sınıflandırıcıyı eğitmek için kullanıldı. Sınıflandırıcıda toplu veri aktarımı, veritabanı, interaktif, mail, servisler, www, P2P, saldırı, oyunlar ve multimedya gibi uygulama sınıfları kullanıldı. Sınıflandırıcı %65 akış doğruluğu göstermektedir. Moore ve Zuev, *Naive Bayes Kernel Tahmini* (Naive Bayes Kernel Estimation – *NBKE*) ve *Hızlı Korelasyon Tabanlı Filtre* (Fast Correlation-Based Filter – *FCBF*) sınıflandırma yöntemlerini kullanarak %95'in üzerinde akış doğruluğu elde etmişlerdir.

Haffner ve arkadaşları, trafik sınıflandırma için, özellik olarak veri akışında ilk kez $n-byte$ kullanan *denetimli makine öğrenmesi* tekniğini önerdiler (Haffner, vd., 2005). Ftp kontrol, smtp, pop3, imap, https, http ve ssh uygulama sınıfları için *Naive Bayes, AdaBoost* ve *Maksimum Entropi* algoritmaları önerilmiştir. Algoritmalardan *AdaBoost* ve *Maksimum Entropi* tüm akışları %99'dan fazla doğru sınıflandırma ile en iyi sonuçları sağlamaktadır.

Nguyen ve Armitage, internet trafik sınıflandırma yapabilmek için, çoklu alt akış kullanarak gerçek zamanlı sınıflandırma yapısını önerdiler (Nguyen ve Armitage, 2006a). Çalışmada ilk olarak, gelecekte tespit etmek istediği trafiği temsil etmek için her akış iki ya da daha çok alt akışa ayrılmaktadır. Her akış N paketten elde edilen özellik değerleridir. Orjinal akış yerine bu alt akışların kombinasyonu kullanılmaktadır. Algoritma olarak makine öğrenmesi algoritmalarından *Naive Bayes* algoritması kullanılmıştır. Uygulama sınıfı olarak web, DNS, ftp, Smtp, SSH, Telnet ve P2P uygulama sınıfları kullanılmıştır. Sonraki çalışmalarında araştırmacılar (Nguyen ve Armitage, 2006b), akış ters yöndeymiş gibi kendi kopya görüntüleri ve hedef uygulama tarafından oluşturulan, tam akıştan çıkarılan ve çoklu alt akışlar üzerinden hesaplanan istatistiksel özellikler kullanılarak makine öğrenmesi tabanlı sınıflandırıcı eğitimi önermişlerdir. Oyun trafiğini belirlemek için *Naive Bayes* ve *Karar Ağaçları Algoritmaları*nı kullanmışlardır.

Park ve arkadaşları, trafik sınıflandırma için, *Genetik Algoritmalara* (GA) dayalı özellik seçimi tekniğini kullandılar (Park, vd., 2006). www, Telnet, Chat (Messenger), ftp, P2P (Kazaa, Gnutella), Multimedia, SMTP, POP, IMAP, NDS, Oracle, X11

uygulama sınıflarını *NBKE*, *J48* (*C4.5*) ve *RepTree Karar Ağaçları* algoritmaları kullanarak sınıfladılar. Karar ağacı algoritmalarının *NBKE* algoritmasından daha iyi sonuçlar verdiğini gözlemlediler.

Williams ve arkadaşları, trafik sınıflandırma için *NBD* (Naive Bayes with Discretisation), *NBKE* (Naive Bayes with Kernel Density Estimation) , *C4.5 Karar Ağacı*, *Bayesian Network* ve *Naive Bayes* karar ağacı algoritmalarını kullanmışlardır. Analiz için kullanılan özellikler *Korelasyon Tabanlı Özellik Seçimi* ve *Tutarlılık Tabanlı Özellik Seçimi* algoritmaları tarafından seçilen 2 en iyi set ve 22 tam özellik setini içermektedir (Williams, vd., 2006). 22 özellik setinin tamamıyla en yüksek akış doğruluğu elde edilmektedir. 8 veya 9 özellik seti seçilerek *çapraz doğrulama* yapılmış ve seçilen özellik seti ile neredeyse aynı sonuç elde edilmiştir. Her bir özellik kümesinde *C4.5* algoritmasının hızlı olduğu görülmektedir. Algoritma hızları bakımından sıralama *C4.5*, *NBD*, *Bayes Ağlar*, *Naive Bayes Ağacı* ve *NBKE* şeklindedir.

Crotti ve arkadaşları, paket uzunluğu, iç geliş süresi ve paket varış süresine dayalı bir trafik sınıflandırma yöntemi önerdiler (Crotti, vd., 2007). Bu yöntem eğitim ve sınıflandırma aşamalarından oluşmakta olup eğitim aşamasında, uygulama sınıflarının ön etiketli akışları protokol parmak izi (tahmin edilen *PDF* vektörü) oluşturmak için analiz edildi. Analiz "0" – "1" arasında değerlerle ifade edildi ve akış o protokole aitse olasılık yüksek olacak şekilde ayarlandı. Paketin ilk birkaç özelliklerine bakarak ve http, Smtp, POP3 uygulama sınıfları kullanılarak %91'in üzerinde doğruluk edilmiştir. Bu çalışmanın dezavantajı ise sınıflandırıcı, gelen paketin ilk birkaçını kaçırdığı zaman sınıflandırma yapamamaktadır.

Erman ve arkadaşları, denetimsiz öğrenme ve *yarı denetimli öğrenme* yöntemlerini birleştirerek *yarı denetimli öğrenme* yöntemi olarak sunulan bir trafik sınıflandırma yaklaşımı önerdiler (Erman, vd., 2007). İlk olarak, etiketli ve etiketsiz akışlar bir kümeleme algoritmasını beslemektedir. İkinci olarak, mevcut etiketli akışlar bilinen farklı sınıflara kümeleme algoritmasından gelen akış ile eşlemek için kullanılmaktadır. Etiketli akışlarla küme eşleştirmek için bir olasılık ataması yapılmakta ve seçilen mesafeye en yakın kümeye atama yapılmaktadır.

Bonfiglio ve arkadaşları, gerçek zamanlı Skype trafiğini ayrıştırmak için çerçeve (framework) tabanlı iki teknik önermişlerdir (Bonfiglio, vd., 2007). Birincisi, şifreleme işlemi tarafından tanıtılan mesaj içeriğini rastgele analizi yoluyla Skype parmak izi tespit eden *Person's Chi-Square Testi*; ikincisi mesaj boyutu ve varış oranı karakteristiğinden Skype trafiğini algılayan *Naive Bayes* tekniğidir. Derin paket inceleme (DPI) tabanlı yöntem kullanılarak, sınıflandırılan tekniğin performansı karşılaştırılmıştır. Kaynak uygulamasına bakılmaksızın IP üzerinden ses trafiğini belirlemede *Naive Bayes* tekniğinin etkili olduğu gösterilmiştir. *Person's Chi-Square Testi,* UDP ve TCP akışlar için tüm sıkıştırılmış ve şifrelenmiş trafik üzerinde Skype trafiğini ayrıştırmışlardır.

Angevine ve Zincir-Heywood, 10 akış istatistiksel özellik kullanılarak *C4.5* ve *AdaBoost* algoritmaları ile Skype trafiğini sınıflandırma işlemi yapmışlardır (Angevine ve Zincir-Heywood, 2008). Her iki sınıflandırıcı tarafından da yüksek doğruluk elde edildiği gözlenmiştir.

Kim ve arkadaşları, 9 *akış istatistiksel özelliği* (paket boyutu vb.) ile birden fazla uygulama sınıfı kullanmışlar ve en yüksek doğruluğu *SVM* ve *C4.5* algoritmaları ile elde etmişlerdir (Kim, vd., 2008).

Alshammari ve Zincir-Heywood, internet trafik sınıflandırma yapabilmek için *AdaBoost, SVM* (Support Vector Machine), *Naive Bayes, RIPPER* ve *C4.5* algoritmalarını incelemişlerdir (Alshammari ve Zincir-Heywood, 2009). Çalışmada sınıfları inşa etmek için *paket seviyesi özellikleri* ve *akış seviyesi özellikleri* (paketler arası varış süresi vb.) olmak üzere 22 adet özellik içeren set kullanılmıştır. *C4.5* algoritması tüm algoritmalar arasında en iyi sonucu vermiş olup Skype trafiğini tanımlamada %97,8 ve SSH trafiğini tanımlamada %83 doğruluk elde edilmiştir.

Lim ve arkadaşları, *Naive Bayes, SVM, K-NN* ve *C4.5* karar ağacı algoritmalarını kullanılarak 9 akış istatistiksel özellik seti, internet trafik sınıflandırma gerçekleştirmişlerdir (Lim, vd., 2010). *C4.5* algoritması tüm özellik setleri için en doğru sonucu vermiş olup *ayrıklaştırma işlemi*nin algoritma doğruluğunu artırdığı gözlemlenmiştir.

Este ve arkadaşları, *SVM* algoritması kullanarak 3 farklı internet trafik verisi için http, smtp, POP3, Ftp gibi birçok uygulama sınıfını sınıflandırmışlardır (Este, vd., 2009).

SVM algoritmalarının eğitim aşamasının oldukça karmaşık olduğu görülmekle birlikte sınıflandırıcılarla %90 üzerinde doğruluk elde edilmiştir.

Qu ve Prasanna, MSN, Skype, P2PTV, http, QQ_IM, Skype IM, Thunder Yahoo IM uygulama paketlerini sınıflandırılmak için *C4.5* karar ağacı algoritması ile yazılım tabanlı bir mimari önermişlerdir (Qu ve Prasanna, 2014). Çalışmada *C4.5* Karar Ağacı, Kompakt Hash Tablolarına dönüştürülerek %98,15 üzerinde doğruluk ve 134,15 MLPS (Million Lookups Per Second) iş oranı elde edilmiştir.

2.3.2. FPGA Tabanlı Makine Öğrenmesi Kullanılan Çalışmalar

Luo ve arkadaşları, FPGA tabanlı *C4.5* karar ağacı algoritması kullanarak internet trafik sınıflandırma yapmışlardır (Luo, vd., 2008). Donanım üzerinde sınıflandırıcı etkili kullanılmış olup kullanılan bellek miktarı azaltılmıştır. Ayrıca paralelleştirme işlemi de yapıldığı belirtilmiş olup mimarinin FPGA sonuçları paylaşılmamıştır.

Jiang ve Gokhale, *paket başlık bilgileri* (port numarası, protokol vb.) kullanarak multimedya trafiğini (Skpye, Instant Messaging (IM), IPTV) sınıflandırmak için *k-NN* algoritmasını önermişlerdir (Jiang ve Gokhale, 2010). Çalışmada, donanım üzerinde uygulama ve yerleştirme sonuçlarına göre 40 $Gbps$ çıkan iş oranı elde edilmiştir. Ancak, 3 paket özelliği dikkate aldığından uygulama sınıfının küçük sayısıyla sınıflandırılma yapılmış olup genel sınıflandırıcı için uygun değildir.

Groleat ve arkadaşları, *SVM* algoritması kullanılarak trafik sınıflandırma yaparak ve yazılım uygulamasının, donanım uygulamasının gerisinde kaldığını göstermişlerdir (Groleat, vd., 2012).

Monemi ve arkadaşları, karar ağacı tabanlı sınıflandırıcı NetFPGA üzerinde donanım tasarımı önermişlerdir (Monemi, vd., 2013). Çalışmanın saat hızı 67 Mhz olup gerçek zamanlı trafik sınıflandırma için yetersizdir.

Tong ve arkadaşları, yüksek iş oranı veya düşük maliyetli yaklaşım elde etmek için farklı iki trafik sınıflandırma mimarisi önererek *C4.5* karar ağacı algoritması kullanmışlardır (Tong, vd., 2013). FPGA sonuçlarına göre önerilen mimari 520 $MCPS$ iş oranı elde etmekte ve dengesiz karar ağaçları için iş oranı negatif olarak etkilenmektedir.

Gandhi ve arkadaşları, sınıflandırma yapmak için *C4.5* karar ağacı algoritmasını *Hash Tablolarına* dönüştürülerek yüksek iş oranına sahip bir mimari önermişlerdir (Gandhi, vd., 2014). Özellik seti olarak protokol, kaynak port, hedef port, ortalama paket boyutu, maksimum paket boyutu, varyans kullanılarak *C4.5* karar ağacı sınıflandırması yapılmaktadır ve ağaç çoklu hash tablolarına dönüştürülerek Skype, MSN, Thunder ve http uygulama sınıfları ayrıştırılmaktadır. Çalışma güncel FPGA Virtex-7 üzerinde modüler işleme elemanı (Process Element – PE) olarak tasarlanarak $1654\ MCPS$ iş oranı elde edilmiştir.

Qu ve Prasanna, önerilen trafik sınıflandırma yönteminde *C4.5* karar ağacını, kural seti tablosuna (Rule Set Table) dönüştürerek sınıflandırma yapmışlardır (Qu ve Prasanna, 2015). Çalışmada MSN, Skype, P2PTV, http, QQ_IM, Skype IM, Thunder, Yahoo IM uygulama sınıfları kullanılarak internet trafiğinin akış istatistiksel özellikleri; ulaşım katmanı protokolü, kaynak port, hedef port, ortalama paket boyutu, maksimum paket boyutu, minimum paket boyutu olarak kullanılmıştır. Güncel FPGA üzerinde yapılan son yerleştirme ve rota sonuçlarına göre $645\ MCPS$ (Million Classifications Per Second) iş oranı elde edilmiştir.

BÖLÜM 3

MAKİNE ÖĞRENMESİ TEMELLİ İNTERNET TRAFİK SINIFLANDIRMA MİMARİSİ TASARIM SÜRECİ

Literatür incelendiğinde gerçek zamanlı (real-time) trafik sınıflandırma için 100 $Gbps$ hızlarında paket işleyebilen sınıflandırma mimarilerine ihtiyaç vardır. Yüksek hızlarda internet trafik sınıflandırmanın mevcut yaklaşımlardan makine öğrenmesi algoritmalarının donanım üzerinde gerçeklenmesi ile mümkün olduğu gözlenmiştir. Makine öğrenmesi algoritmalarının donanım üzerinde gerçeklenerek sınıflandırma mimarisi tasarımı birbirine bağlı çok sayıda adımdan oluşmaktadır.

Bu bölümde tez çalışmasında önerilen makine öğrenmesi temelli donanım mimarilerinin tasarım adımları anlatılacaktır.

3.1. Uygulama Sınıflarının Belirlenmesi Ve Eğitim Setinin Hazırlanması

İnternet trafik sınıflandırma için kullanılacak eğitim *veri seti* (data set), kamuya açık ücretsiz olarak internet sitelerinde yer alan tasnif edilmiş internet trafik verilerinden (Tstat, 2016) veya profesyonel olarak bu işin ticaretini yapan firmalardan elde edilmektedir. Bir başka alternatif kaynak ise üniversite ağları veya diğer kurum/şirket ağlarında akan trafik üzerinden veri alınarak, bu verilerden eğitim veri seti oluşturmaktır. Fakat alternatif yöntemlerden elde edilen trafik, sınırlı içerik kullanıyor olabilir. Dolayısıyla gerçek zamanlı trafik sınıflandırmanın gerçek dünyadaki daha genel trafik verilerine yakın benzerlikte olması ve tercih edilecek veri setinin de bu duruma uygun seçilmesi önemlidir.

Bu tez çalışmasında eğitim seti olarak uluslararası geçerliliği olan Tstat (Tstat, 2016) tarafından sağlanan internet trafik paketleri toplanmıştır. Toplanan paketler öncelikle *veri temizleme* ve *veri hazırlama* aşamasından geçirilmiştir. Belirtilen *veri temizleme* ve *veri hazırlama* aşamalarına *ön işleme aşaması* adı verilmektedir.

Tstat verileri ön işleme aşamasından geçirildikten sonra, internet trafik sınıflandırma için referans olarak kullanılacak en uygun *özellik seti* çıkarılmaktadır (feature set extraction). Trafik akış istatistiklerinin ve özellik setinin belirlenmesinde mevcut çözümlerde de olduğu gibi her bir trafik akışının ilk dört paketi kullanılmaktadır (Tong, vd., 2013; Qu ve Prasanna, 2015). Bu tez çalışmasında kullanılan Tstat trafik verisi içerisinde yer alan uygulama sınıfları *Çizelge 3.1*'de gösterilmektedir.

Çizelge 3.1 Uygulama Sınıfları

Uygulama	**Açıklama**
Skype TCP	Skype Ses ve Video Araması
Skype UDP	Skype Ses ve Video Araması
Yahoo IM	Yahoo Anlık Mesajlaşma
Joost	P2P TV Programı
PPLive	P2P TV Programı
Sopcast	P2P TV Programı
TVAnts	P2P TV Programı
MSN	MSN Microsoft Messenger

8 uygulama sınıfı ile bu çalışma literatürde en çok uygulama sınıfı kullanan çalışmalar arasında yer almaktadır. Görüldüğü üzere seçilen uygulama sınıfları arasında özellikle geleneksel sınıflandırıcılar tarafından sınıflandırılamayan P2P servisler ve anlık mesajlaşma uygulamaları da bu çalışmada yer almaktadır. Her bir uygulama sınıfından belirli sayıda (aşırı öğrenme/yetersiz öğrenme (overfit/underfit) durumlarından kaçınılarak) veri alınarak *eğitim seti* oluşturulur ve aynı zamanda bu veriler sınıflandırma *doğruluk testleri* için kullanılır.

3.2. Özellik Seti (Feature Set) Seçimi/Çıkarımı

Özellik seti (feature set) seçimi makine öğrenmesi algoritmasının performansı ile yakından ilişkilidir. Bilgi kazancı sağlamayan özellikler kullanmak makine öğrenmesi algoritmalarının doğruluğunu olumsuz etkilemektedir. Aynı zamanda daha fazla bilgi saklama ve veri miktarındaki artış, sistemin maliyetini arttırmaktadır. Dolayısıyla mümkün olan en ideal özellik alt kümesi seçilmelidir (Witten ve Frank, 2005; Goldberg, vd., 1989; Kohavi ve G. John, 1997; Winston, 1984; Hall ve Holmes, 2003). Bu seçimi yaparken test aşamasında mümkün olduğu kadar fazla özellik kullanılmalı ve bu özellikler arasından yapılacak sınıflandırmaya en uygun *alt özellik set* seçilmelidir.

Literatürde *paket başlık bilgileri*nin (protokol, kaynak port numarası, hedef port numarası vb.) yanı sıra yüksek doğruluk elde edebilmek için *akış seviyesi özellikleri* (maksimum paket boyutu, minimum paket boyutu vb.) de kullanılmaktadır (Tong, vd., 2013).

Çizelge 3.2 Aday Özellik Listesi

Özellik	**Tür**	**Açıklama**
Prtcl	Paket Başlık Bilgileri	Protokol
SP	Paket Başlık Bilgileri	Kaynak Port Numarası
DP	Paket Başlık Bilgileri	Hedef Port Numarası
Avg	Akış İstatistiksel Özellikleri	Ortalama Paket Boyutu (bayt)
Max	Akış İstatistiksel Özellikleri	Maksimum Paket Boyutu (bayt)
Min	Akış İstatistiksel Özellikleri	Minimum Paket Boyutu (bayt)
Var	Akış İstatistiksel Özellikleri	Paket Boyutu Varyansı

Bu tez kapsamında *paket başlık bilgileri* olarak *protokol (Prtcl), kaynak port numarası(SP), hedef port numarası (DP)* olmak üzere 3 özellik ve *akış istatistiksel özellikleri* olarak *ortalama paket boyutu (Avg), maksimum paket boyutu (Max), minimum paket boyutu (Min)* ve *paket boyutu varyansı (Var)* olmak üzere 4 özellik kullanılmıştır. Seçilen *Özellik Seti Çizelge 3.2* 'de gösterilmektedir.

3.2.1. Aday Özellik Seti

Literatürde yer alan çalışmalar genellikle, *özellik seti* seçiminde doğruluk oranına odaklanmaktadır. İnternet trafik sınıflandırma için doğruluk oranı önemli bir kriter olmakla birlikte tek başına yeterli bir kriter değildir. Bu tezde doğruluk oranı yanında donanıma uygun veri yapısı oluşturmada kullanılabilecek *özellik*lerin de belirlenmesi gerekmektedir. Bu tez çalışmasında, önerilen donanım mimarilerinde çarpma işlemleri vb. bazı işlemler maliyeti arttırdığı için *özellik* seçiminde bu gibi durumlar dikkate alınmıştır. Literatürdeki çalışmalarda kullanılan özellikler arasında varyans özelliği de yer almaktadır. *Varyans özelliği, "Alınan ilk n paketin hesaplanan standart sapmasının karesi"* alınarak hesaplanmaktadır. Donanım üzerinde standart sapma hesaplama, kare alma gibi çarpım işlemleri maliyetlidir. Dolayısıyla varyans özelliğinin etkisi analiz edilerek *Çizelge 3.3* 'de verilen *Aday Özellik Seti* belirlenmiştir.

Çizelge 3.3 Aday Özellik Set

No	Set Adı	Prt.	SPN	DPN	Avg	Max.	Min	Var.
1	Set 100 A	✓	✓	✓	✓	✓	✓	✓
	Set 100 B	✓	✓	✓	✓	✓	✓	
2	Set 200 A	✓	✓	✓	✓	✓	✓	✓
	Set 200 B	✓	✓	✓	✓	✓	✓	
3	Set 300 A	✓	✓	✓	✓	✓	✓	✓
	Set 300 B	✓	✓	✓	✓	✓	✓	
4	Set 400 A	✓	✓	✓	✓	✓	✓	✓
	Set 400 B	✓	✓	✓	✓	✓	✓	
5	Set 500 A	✓	✓	✓	✓	✓	✓	✓
	Set 500 B	✓	✓	✓	✓	✓	✓	
6	Set 600 A	✓	✓	✓	✓	✓	✓	✓
	Set 600 B	✓	✓	✓	✓	✓	✓	
7	Set 700 A	✓	✓	✓	✓	✓	✓	✓
	Set 700 B	✓	✓	✓	✓	✓	✓	
8	Set 800 A	✓	✓	✓	✓	✓	✓	✓
	Set 800 B	✓	✓	✓	✓	✓	✓	

Çizelge 3.3 'de görüldüğü üzere 8 uygulama sınıfı içeren farklı büyüklüklerde (akış sayısı 100, 200, ..., 800) eğitim setleri oluşturuldu. Örneğin, Set $100A$ bütün özellikler (Prt., SPN, DPN, Avg., Max., Min. ve Var.) eklenmiş ve Set $100B$ ise (Prt., SPN, DPN, Avg., Max. ve Min.) ise varyans özelliği eklenmemiş olarak oluşturuldu. Özellik seti analizi için Simple CART algoritması kullanılarak ayrıklaştırma (discretization) ve *çapraz doğrulama* işlemleri yapılmıştır.

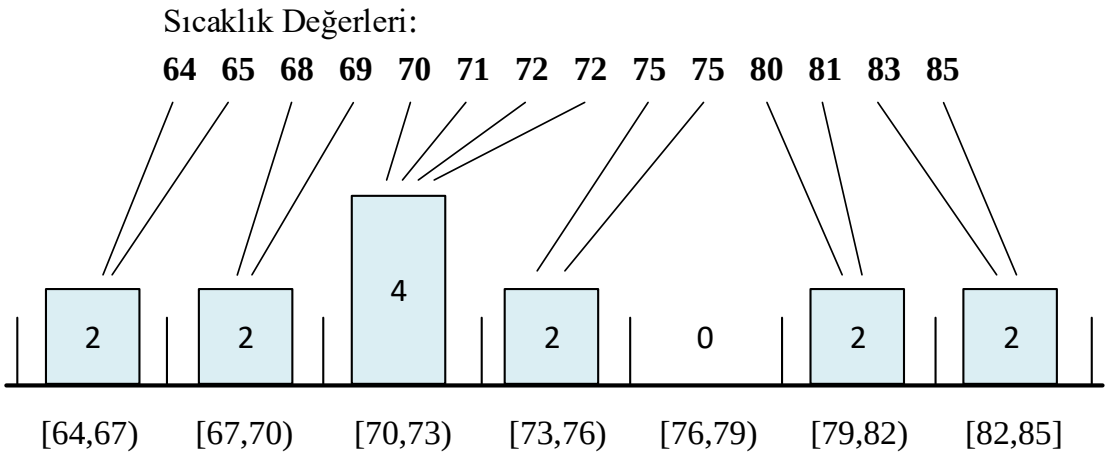
3.2.2. Ayrıklaştırma Süreci

Ayrıklaştırma işlemi (discretization process), yüksek doğruluk elde etmek için kullanılan bir tür gruplama işlemidir (Tong, vd., 2013; Qu, ve Prasanna, 2014; Kim, vd., 2008). *Ayrıklaştırma İşlemi*, literatürde ML algoritmalarının iyileştirilmesi için özellikle internet trafik sınıflandırmada sıkça kullanılan bir yöntemdir (Tong, vd., 2013; Tong, vd., 2014; Qu, ve Prasanna, 2014; Qu, ve Prasanna, 2015).

Bir veri setinde ayrıklaştırma işlemi, *gruplandırılmış frekans dağılımı* (grouped frequency distribution) kavramı yardımıyla yapılmaktadır. Veri setinin değerleri küçükten büyüğe dizildikten sonra, elde edilen sayı dizisinin en büyük değeri ve en küçük değeri arasındaki farka *değişim aralığı* ya da *açıklık* denir. *Açıklık* betimsel istatistikte bütün veri dizisini içinde kapsayan en küçük aralıktır. Veri setindeki verilerin gruplandırılma işlemi için açıklığın elde edilecek eşit genişlikli aralıkların genişliğine (interval width) bölümüyle *aralık sayısı* (interval numbers) elde edilir.

$$Aralık\ sayısı = \frac{Açıklık}{Aralık\ genişliği}$$

Şekil 3.1'de sıcakları (°C) ifade eden bir veri setinin yukarıda anlatılan yöntem yardımıyla ayrıklaştırma işlemi gösterilmektedir. Sıcaklık veri setinin sıralanmış değerleri: $64, 65, 68, 69, 70, 71, 72, 72, 75, 75, 80, 81, 83, 85$ biçimindedir. Bu veri seti $[64,85]$ aralığının bir alt setidir. Normal şartlarda bu sette en kötü durum araması 21 olacaktır. Bu sette açıklık 21 (85-64)'dir ve dolayısıyla aralık sayısı bulunurken göz önüne alınması gereken en önemli nokta, arama işlemini en aza indirgeyecek değerlere karar vermektir. Açıklık 21 olacak şekilde formülü sağlayan toplam fonksiyonun minimum değeri 9,16 olarak bulunur. Bunum anlamı en az 10 işlem ile arama yapılacağını göstermektedir. Bu örnekte $aralık\ sayısı\ +\ aralık\ genişliği$ değeri 10 olacaktır. En ideal değerler verilerek minimum arama sayısı elde edilmelidir ve bu değerler $aralık\ genişliği = 3, aralık\ sayısı = 7\ (\frac{21}{3} = 7)$ alındığında, istenilen değer elde edilir $(3 + 7 = 10)$. *Şekil 3.1*, $[64,85]$ aralığının genişlikleri 3 olan aralıklara ayrıştırılmış şekilde verilmiştir. Dolayısıyla bu örnekte en kötü durum arama sayısı 21'den $7 + 3 = 10$'a inmiş olacaktır.



Şekil 3.1 Ayrıklaştırma İşlemi İçin Sıcaklık Değerleri Örneği

Bu tezde ayrıklaştırma algoritması olarak, özellik değerlerini ayrık değerlere dönüştürmek için en çok kullanılan *Entropi-MDL Algoritması* (Fayyad ve Irani, 1991) kullanılmıştır.

3.2.3. Çapraz Doğrulama

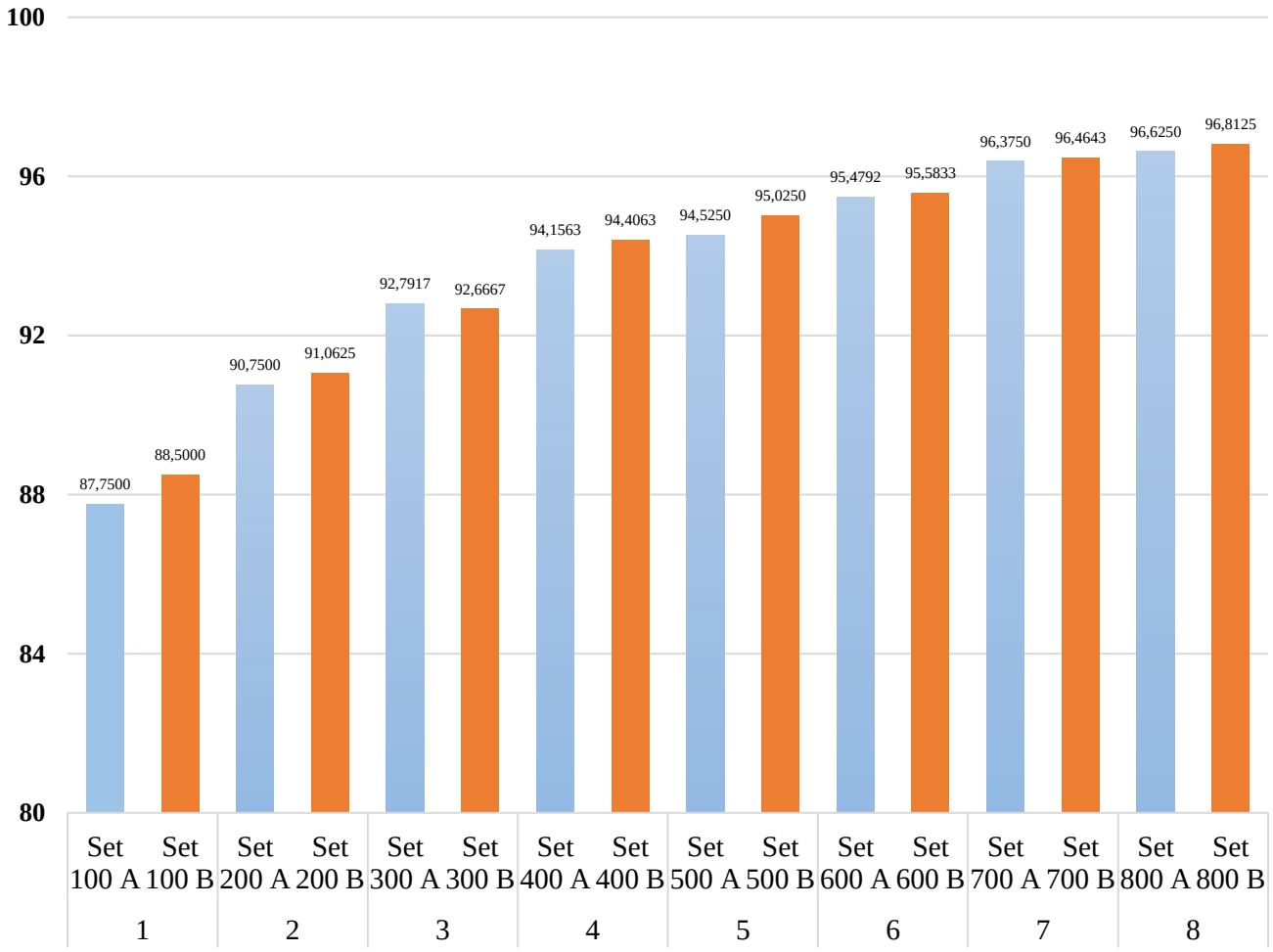
ML ile yapılan sınıflandırma işleminin test edilebilmesi için literatürde birçok yöntem uygulanmaktadır. Bu yöntemlerden en çok kullanılanı ise *çapraz doğrulama* (cross-validation) (Kohavi, 1995) yöntemidir. 10 katlı (10-fold) *çapraz doğrulama* yönteminde veri seti 10 eşit parçaya ayrılmaktadır. Ayrılan veri setinin 9 parçası *eğitim verisi*, kalan 1 parçası ise *test verisi* olarak kullanılmaktadır. Daha sonra kullanılan test verisi değiştirilerek aynı işlem 10 kez tekrarlanmaktadır. *Çapraz doğrulama* yöntemi veri setinin doğruluğunu test etmekte ve 10 iterasyon yaparak hesaplanan değerlerin ortalamasını almaktadır. Bulunan *ortalama değer* ise o algoritmanın *doğruluk* (accuracy) değeri olarak elde edilmektedir.

3.2.4. Özellik Seti Seçimi

Makine öğrenmesi sınıflandırma algoritmaları arasında yer alan Simple CART algoritması ikili karar ağacı üreten bir algoritmadır. Simple CART kullanılarak *ayrıklaştırma işlemi* uygulanan ve *çapraz doğrulama işlemi* ile de test edilerek elde edilen sonuçlar *Şekil 3.2*'de gösterilmektedir. *Şekil 3.2* incelendiğinde $300A$ ve $300B$ hariç

diğer setlerde, varyans değeri çıkarıldıktan sonra daha yüksek bir sınıflandırma doğruluğu elde edildiği görülmektedir. Dolayısıyla kullanılan veri setine göre varyans değerini kullanmanın bize avantaj yerine dezavantaj sağlayacağı ortadadır. *Bölüm 3.2.1'de* de belirtildiği gibi varyans özelliği kullanmak avantaj sağlamadığından donanım maliyetlerini düşürmek için özellik setinden çıkarılabilir.

Bu tez çalışmasında kullanılan özellik setinde varyans özelliği çıkarılarak 7 özellik yerine 6 özellik kullanılmıştır (Prt., SPN, DPN, Avg., Max. ve Min.).



Şekil 3.2 Aday Özellik Set Analizi

3.3. Makine Öğrenmesi Algoritma(ları)sının Belirlenmesi

Veri seti ve *özellik seti*ne uygun algoritma seçimi, tasarım sürecinde en kritik adımlardan biridir. Literatürde donanıma en uygun algoritmaların *sınıflandırma tabanlı algoritmalar* olduğu bilinmektedir ve dolayısıyla bu çalışmada makine öğrenmesi algoritmalarından sadece *sınıflandırma tabanlı algoritmala*ra odaklanılmıştır. Bu tez kapsamında, en uygun algoritmayı seçmek için *sınıflandırma tabanlı algoritmala*rın analizi yapılmıştır. Literatürde en çok tercih edilen *sınıflandırma tabanlı algoritmala*rdan 10 tanesi seçilmiştir. Analiz sonuçlarına göre donanıma en uygun algoritma seçimi gerçekleştirilmiştir.

3.3.1. Makine Öğrenmesi Algoritmaları

Sınıflandırma tabanlı algoritmalar hakkında özet bilgiler aşağıda yer almaktadır;

1. **Simple CART,** Sınıflandırma ve Regresyon Ağaçları (Classification and Regression Trees – *CART*) algoritmalarından biridir. 1984 yılında Breiman tarafından ortaya atılmış bir yöntemdir. Bu algoritma *Bölüm 3.3.3*'de ayrıntılı olarak anlatılacaktır (Breiman, 1984).
2. **BFTree (Best-first Decision Tree)**, *C4.5* ve *CART* algoritmalarından farklı olarak en iyi düğümü erken belirlemektedir. Algoritmada en iyi düğüm, bölünmesi için ayrılmış tüm düğümler arasında en fazla benzerliğin bozulmasının azalmasına neden olan düğüm olarak belirlenir. BFTree ikili ağaç oluşturmaktadır (Shi, 1992).
3. **C4.5 Karar Ağacı (C4.5 Decision Tree Algorithm)**, 1993 yılında Ross Quinlan tarafından yayınlanan ve *ID3* algoritmasının uzantısı olarak kabul edilen karar ağacı algoritmasıdır. En iyi bölen özelliği seçiminde *entropi* ve *enformasyon kazancı* tekniklerini kullanmaktadır (Quinlan, 1993).
4. **BayesNET (BayesNET Algorithm – Bayesian Network)**, elde var olan sınıflanmış verileri kullanarak yeni bir verinin mevcut sınıflardan herhangi birine girme olasılığını hesaplayan istatistiksel bir yöntemdir. Olasılık kuramındaki *bayes kuralı*nı kullanmaktadır (Heckerman, vd., 1995).

5. **k-NN (k-Nearest Neighbours Algorithm)**, makine öğrenmesi algoritmaları içinde en basit sınıflandırma algoritmalarından biridir. Sınıflandırma yapılırken veritabanındaki her bir kaydın diğer kayıtlarla olan uzaklığı hesaplanır. Bir kayıt için sadece *k* değeri kadar değer göz önüne alınarak hesaba katılmaktadır (Aha, vd., 1991).
6. **Naive Bayes** (**Naive Bayes Algorithm**), olasılık bilgisini temsil etmek, kullanmak ve öğrenmek için açık semantiği ile basit bir yaklaşım sağlamaktadır. Bu yöntem, performans hedefinin test örneklerinin sınıfını doğru bir şekilde tahmin etmesi ve eğitim örneklerinin sınıf bilgisini içerdiği *denetimli tümevarım* görevlerinde kullanılmak üzere tasarlanmıştır (John ve Langley, 1995).
7. **REPTree (Reduces Error Pruning Tree)**, *modellerin birleştirilmesi* (ensemble learning) yöntemi olarak bilinir. Bu yöntem genel olarak elde edilecek sonuçların performansının arttırılması amacıyla farklı öğrenme algoritmalarının bir arada kullanılmasıdır. REPTree bölen kriteri olarak *enformasyon kazancı*nı, budama için *hata azaltma yöntemi*ni kullanan bir karar/regresyon ağacı algoritmasıdır (Quinlan, 1987).
8. **SVM (Support Vector Machine – SVM)**, ilk olarak 1960'larda ortaya atılan, hesaplanabilir öğrenme teorisinin bir bileşeni olan öğrenmenin temel teorisi olarak bilinen algoritma, 1992 yılında sunulmuştur. Diğer yöntemlerle karşılaştırıldığında eğitim süresinin oldukça uzun olmasına rağmen *yüksek güvenirliliği*, *ezbere öğrenmeye dayanıklılığı* ve *lineer olmayan sınıflardaki başarısı* ile sıkça tercih edilen bir yöntemdir (Boser, 1992).
9. **RIPPER (Repeated Incremental Pruning to Produce Error Reduction)**, IREP algoritmasının optimize edilmiş bir versiyonu olarak önermeli kural öğrenicisi olan algoritmadır. Karar ağaçlarında da kullanılan *azaltılmış hata budama* ve *birliktelik kuralları*nı kullanmaktadır. RIPPER algoritmasında veri seti *büyüyen sete* ve *budama sete* bölünür. Sezgisel yöntemlerle büyüyen set üzerinde kural seti oluşturulur. Bu üste çıkan set budama operatörleri ile art arda basitleştirilir (Cohen, 1995).
10. **Yapay Sinir Ağları (YSA) (Artificial Neural Network – ANN)**, biyolojik sinir ağlarından esinlenerek geliştirilmiş bir makine öğrenmesi algoritmasıdır.

Yapay sinir hücrelerinin birbirleriyle çeşitli şekilde bağlanmasıyla oluşur ve katmanlar şeklinde düzenlenir. Donanımda uygulanması zor algoritmalardan biridir çünkü katmanların içlerindeki matematiksel yapı hala bilinmemektedir (Hopfield, 1982).

3.3.2. Algoritma Seçimi

Bu tez çalışmasında, literatürde özellikle akademik çalışmalarda sıklıkla kullanılan makine öğrenmesi algoritmalarının analizi için Waikato Üniversitesi'nden araştırmacılar tarafından geliştirilen WEKA (Waikato Environment for Knowledge Analysis) programı kullanılmaktadır (Hall, vd., 2009).

Seçilen *veri seti* (*Bölüm 3.1*) ve *özellik set* (*Bölüm 3.2*) ile WEKA'da yapılan analiz sonuçları *Çizelge 3.4* 'de gösterilmektedir. Analiz yapılırken ön aşamada *ayrıklaştırma işlemi* ve *çapraz doğrulama* testi gerçekleştirilmiştir.

Çizelge 3.4 'de yer alan algoritmalardan Simple CART, BFTree, C4.5 ve REPTree algoritmaları *karar ağacı algoritmaları*dır. BayesNet ve Naive Bayes algoritmaları *istatiksel* sınıflandırma algoritmalarıdır. k-NN algoritması *mesafeye dayalı* veya *bellek tabanlı* sınıflandırma algoritmasıdır. RIPPER algoritması *kural tabanlı* sınıflandırma algoritmasıdır. SVM ve YSA algoritmaları ise *fonksiyon tabanlı* sınıflandırma algoritmalarıdır.

Çizelge 3.4 incelendiğinde en iyi sonuçların *SVM* algoritması dışında *karar ağacı tabanlı algoritmaları* ile elde edildiği görülmektedir. *SVM* algoritmasının donanım üzerinde uygulanması pratik değildir. Bunun yanında karar ağacı tabanlı algoritmaların donanım üzerinde gerçeklenmesi daha kolaydır. Dolayısıyla donanım çözümlerinde karar ağacı tabanlı algoritmalar tercih edilmektedir.

Çizelge 3.4 'de yer alan karar ağacı tabanlı algoritmalardan (Simple CART, BFTree, C4.5 ve REPTree) en iyi sonucu %97.1876 ile *C4.5* algoritmasının elde ettiği görülmektedir. Fakat *C4.5* algoritmasının ağaç derinliği incelendiğinde 44 seviye derinlik ile derinliği 16 seviye olan Simple CART algoritmasından 28 seviye (%275) daha derin olduğu görülmektedir. Bu iki algoritmanın doğruluk değerleri arasındaki fark ise sadece %0.375'dir. Ayrıca *C4.5* algoritması ağacı incelendiğinde dengesiz düğüm dağılımına

sahip olduğu gözlenmiştir. Buna karşın Simple CART ağacı *C4.5* ağacına göre daha dengeli dağılımlı yapıdadır. *C4.5* ağacının bu dengesizliği ve artan ağaç derinliği hem sınıflandırma maliyeti artışına hem de donanım kaynaklarının verimsiz kullanımına neden olmaktadır. Bu tez çalışmasında önerilen sınıflandırma mimarileri için en uygun algoritma olarak Simple CART algoritması seçilmiştir.

Çizelge 3.4 Makine Öğrenmesi Sınıflandırma Tabanlı Algoritma Analizleri

No	Algoritma	Algoritma Türü	Çapraz Doğrulama	Toplam Doğruluk (%)	Derinlik	Kappa	F-Measure	Yaprak Düğüm Sayısı	Toplam Düğüm Sayısı
1	Simple CART Algoritması	Karar Ağacı	10-kez	96.8125	16	0.9636	0.9680	112	223
2	BFTree Algoritması	Karar Ağacı	10-kez	96.7656	17	0.9630	0.9680	127	253
3	C4.5 Algoritması	Karar Ağacı	10-kez	97.1875	44	0.9679	0.9720	103	205
4	REPTree Algoritması	Karar Ağacı	10-kez	94.5781	NA	0.9380	0.9460	NA	NA
5	BayesNET Algoritması	İstatiksel	10-kez	95.1094	NA	0.9441	0.9510	NA	NA
6	Naive Bayes Algoritması	İstatiksel	10-kez	94.6875	NA	0.9393	0.9470	NA	NA
7	k-NN Algoritması	Mesafe Tabanlı	10-kez	96.0781	NA	0.9552	0.9610	NA	NA
8	RIPPER Algoritması	Kural Tabanlı	10-kez	94.9375	NA	0.9421	0.9490	NA	NA
9	SVM Algoritması	Fonksiyon Tabanlı	10-kez	98.2188	NA	0.9796	0.9820	NA	NA
10	YSA Algoritması	Fonksiyon Tabanlı	10-kez	75.0469	NA	0.7148	0.7570	NA	NA
		Fonksiyon Tabanlı	10-kez	23.7592	NA	0.9796	0.9820	NA	NA
		Fonksiyon Tabanlı	10-kez	20.0469	NA	0.0863	0.1790	NA	NA

3.3.3. Simple CART

CART algoritmaları en iyi dallara ayırma kriterini seçmek için *entropi algoritması*nı kullanır (Dunham, 2003) ve en iyi bölme kriterini seçmek için *C4.5* algoritmasından farklı bir yol izler (Yohannes ve Webb, 1999). Ayrıca *CART* algoritmaları ikili ağaç üretir ve hangi düğümün kök düğüm ve alt düğüm olacağı hesaplanır. Hesaplama yapılırken düğümün hangi noktasından kesilmesi gerektiği de belirlenmelidir. Algoritma dallara ayırmada en uygun değişken kriterlerini üretir ve bu değişken kriterleri ikiden fazla değer içeriyorsa hangi şekilde iki gruba ayrılacağını da belirlemektedir. *CART* algoritmaları bu işlemler için *Gini Kazancı*nı kullanmaktadır. *Gini kazancı*, hedef niteliklerin değerlerinin olasılık dağılımları arasındaki farklılıkları ölçen benzer olmayan değerlere dayalı bir ölçüttür (Mitchell, 1997).

3.3.3.1. Gini İndeks

S bir eğitim seti olsun, hedef özellik k farklı sınıf üzerinde değer alsın, S'nin gini indeksi;

$$Gini(S) = 1 - \sum_i^k p_i^2 \tag{3.1}$$

dir. Burada i sınıfına ait S'nin olasılığı p_i olmak üzere; olasılık dağılımı $P = (p_1, p_2, \dots, p_k)$ ve $\sum_i^k p_i = 1$'dir. Bir veri kümesinde sadece bir sınıf varsa gini indeksi "$1 - 1^2 = 0$" olmaktadır ve buna *saflık veri kümesi* denir. Bir olasılık dağılımı üniform $P = (\frac{1}{k}, \frac{1}{k}, \dots, \frac{1}{k})$ ise o dağılımın gini indeksi maksimuma ulaşır. Burada S_i, A özellik değeri tarafından indirgenen S'nin parçalanışıdır.

CART algoritmasında özellik setindeki özellik değerleri numerik değerler de alabilir. Bu numerik değerler artan sırada olacak şekilde numaralandırılır.

Sıralama İşlemi (Sort)

$$A_i < A_{i+1}, \quad i = 1, \dots, n \tag{3.2}$$

Sıralanmış A_i özellikleri için ardışık herhangi iki özelliğin orta noktası bulunur, bu değerlere *kesme pozisyonu* (split position) denir. Kesme pozisyonu sayısı ilk ve son A_i özelliği içermesi gerektirdiğinden A_i özellik sayısından 1 fazla olmalıdır $(n + 1)$.

Kesme değerleri; $(A_0 = 0)$

$$k_i = \begin{cases} \frac{A_{i-1} + A_i}{2} & , \quad i = 1, \dots, n \ \ ise \\ A_{i-1} + \frac{A_{i-2} - A_{i-3}}{2} & , \quad i = n + 1 \ \ ise \end{cases} \tag{3.3}$$

3.3.3.2. Gini Kazancı

Gini Kazancı (Gini Gain), A özelliklerini seçen değerlendirme kriteridir. Her bir A_i özelliği için *kesme pozisyonları*na karşılık gelen gini kazanç değerleri aşağıdaki şekilde bulunur;

$$GiniGain(A, S) = Gini(S) - Gini(A, S) = Gini(S) - \sum_{i=1}^{n+1} \frac{S_i}{S} Gini(S_i) \tag{3.4}$$

Tüm özelliklerin *gini kazancı* hesaplanır. En yüksek *gini kazancı* değerine sahip özellik seçilir ve o özellik değerinden düğüm iki dala ayrılır. Bu işlem bütün düğümlerde benzer şekilde devam ettirilir.

3.3.3.3. Örnek Simple CART Ağacı Oluşturma

Literatürde makine öğrenmesi algoritmalarını açıklamak için en çok kullanılan *İris Çiçeği* (Fisher, 1936) örneğini Simple CART algoritmasını açıklamak için kullanacağız. İris çiçeği *alt yaprak uzunluğu*, *alt yaprak genişliği*, *üst yaprak uzunluğu* ve *üst yaprak genişliği* olmak üzere 4 özellik değerine sahip ve *iris-versicolor*, *iris-setosa* ve *iris-virginica* olmak üzere 3 uygulama sınıfına sahip bir veri setidir. Uygulamaları doğru sınıflandırabilmek için her sınıftan eşit miktarda özellik alınmıştır. *Çizelge 3.5* İris çiçeği veri setini göstermektedir ve bu set ile Simple CART algoritması kullanılarak bir karar ağacının nasıl oluşturulduğu anlatılacaktır.

Çizelge 3.5 İris Çiçeği Veri Seti

Alt Yaprak Uzunluğu	Alt Yaprak Genişliği	Üst Yaprak Uzunluğu	Üst Yaprak Genişliği	Uygulama Sınıfı
5.0	2.0	3.5	1.0	Iris versicolor
6.0	2.2	4.0	1.0	Iris versicolor
5.0	2.3	3.3	1.0	Iris versicolor
5.0	3.0	1.6	0.2	Iris setosa
4.3	3.0	1.1	0.1	Iris setosa
5.8	4.0	1.2	0.2	Iris setosa
6.0	2.2	5.0	1.5	Iris virginica
6.0	3.0	4.8	1.8	Iris virginica
6.5	3.0	5.2	2.0	Iris virginica

Örnek olarak *Çizelge 3.5* 'den *üst yaprak uzunluğu* değerleri göz önüne alalım,

1. **Adım:** *Üst yaprak uzunluğu* özelliğine ait değerler numerik değerler olduğu için öncelikle *üst yaprak uzunluğu* özellik setindeki değerleri (3.5, 4.0, 3.3, 1.6, 1.1, 1.2, 5.0, 4.8, 5.2) artan şekilde sıralanır (1.1, 1.2, 1.6, 3.3, 3.5, 4.0, 4.8, 5.0, 5.2). Sonra her özelliğin orta noktasını bulunup (0.55, 1.15, 1.4, 2.45, 3.4, 3.75, 4.4, 4.9, 5.1, 5.3) kesme pozisyonlarını belirlenir. Yapılan işlemler *Çizelge 3.6*'da gösterilmektedir.

Çizelge 3.6 Kesme Pozisyonları

Kesme Numarası (i)	1	2	3	4	5	6	7	8	9	10
Kesme pozisyonları (k_i)	0.55	1.15	1.4	2.45	3.4	3.75	4.4	4.9	5.1	5.3

2. **Adım:** *Çizelge 3.6*'daki uygulama sınıfı C olarak gösterilsin. *Çizelge 3.5*'deki İris çiçeği her uygulama sınıfından 3 adet olmak üzere toplam 9 adet değer içermektedir. *Denklem 3.4* ile C'nin gini indeksi hesaplanır;

$$Gini(C) = 1 - \left[\left(\frac{3}{9}\right)^2 + \left(\frac{3}{9}\right)^2 + \left(\frac{3}{9}\right)^2\right] = \frac{2}{3}$$

3. **Adım:** *Üst yaprak uzunluğu* özelliklerinin *Çizelge 3.6*'da hesaplanan her kesim noktası için gini indeks ve gini kazanç değerleri birlikte hesaplanır;

$$GiniGain(S_1) = \mathbf{0}$$

$$GiniGain(S_2) = \frac{2}{3} - \left[\frac{1}{9}\left[1 - \left[\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2 + \left(\frac{0}{1}\right)^2\right]\right] + \frac{8}{9}\left[1 - \left[\left(\frac{2}{8}\right)^2 + \left(\frac{3}{8}\right)^2 + \left(\frac{3}{8}\right)^2\right]\right]\right] = \mathbf{0,083333}$$

$$GiniGain(S_3) = \frac{2}{3} - \left[\frac{2}{9}\left[1 - \left[\left(\frac{2}{2}\right)^2 + \left(\frac{0}{2}\right)^2 + \left(\frac{0}{2}\right)^2\right]\right] + \frac{7}{9}\left[1 - \left[\left(\frac{1}{7}\right)^2 + \left(\frac{3}{7}\right)^2 + \left(\frac{3}{7}\right)^2\right]\right]\right] = \mathbf{0,1904}$$

$$GiniGain(S_4) = \frac{2}{3} - \left[\frac{3}{9}\left[1 - \left[\left(\frac{3}{3}\right)^2 + \left(\frac{0}{3}\right)^2 + \left(\frac{0}{3}\right)^2\right]\right] + \frac{6}{9}\left[1 - \left[\left(\frac{0}{6}\right)^2 + \left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2\right]\right]\right] = \mathbf{0,533333}$$

$$GiniGain(S_5) = \frac{2}{3} - \left[\frac{4}{9}\left[1 - \left[\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{0}{4}\right)^2\right]\right] + \frac{5}{9}\left[1 - \left[\left(\frac{0}{5}\right)^2 + \left(\frac{2}{5}\right)^2 + \left(\frac{3}{5}\right)^2\right]\right]\right] = \mathbf{0,233333}$$

$$GiniGain(S_6) = \frac{2}{3} - \left[\frac{5}{9}\left[1 - \left[\left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2 + \left(\frac{0}{5}\right)^2\right]\right] + \frac{4}{9}\left[1 - \left[\left(\frac{0}{4}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right]\right]\right] = \mathbf{0,233333}$$

$$GiniGain(S_7) = \frac{2}{3} - \left[\frac{6}{9}\left[1 - \left[\left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2 + \left(\frac{0}{6}\right)^2\right]\right] + \frac{3}{9}\left[1 - \left[\left(\frac{0}{3}\right)^2 + \left(\frac{0}{3}\right)^2 + \left(\frac{3}{3}\right)^2\right]\right]\right] = \mathbf{0,333333}$$

$$GiniGain(S_8) = \frac{2}{3} - \left[\frac{7}{9}\left[1 - \left[\left(\frac{3}{7}\right)^2 + \left(\frac{3}{7}\right)^2 + \left(\frac{1}{7}\right)^2\right]\right] + \frac{2}{9}\left[1 - \left[\left(\frac{0}{2}\right)^2 + \left(\frac{0}{2}\right)^2 + \left(\frac{2}{2}\right)^2\right]\right]\right] = \mathbf{0,1904}$$

$$GiniGain(S_9) = \frac{2}{3} - \left[\frac{8}{9}\left[1 - \left[\left(\frac{3}{8}\right)^2 + \left(\frac{3}{8}\right)^2 + \left(\frac{2}{8}\right)^2\right]\right] + \frac{1}{9}\left[1 - \left[\left(\frac{0}{1}\right)^2 + \left(\frac{0}{1}\right)^2 + \left(\frac{1}{1}\right)^2\right]\right]\right] = \mathbf{0,08333}$$

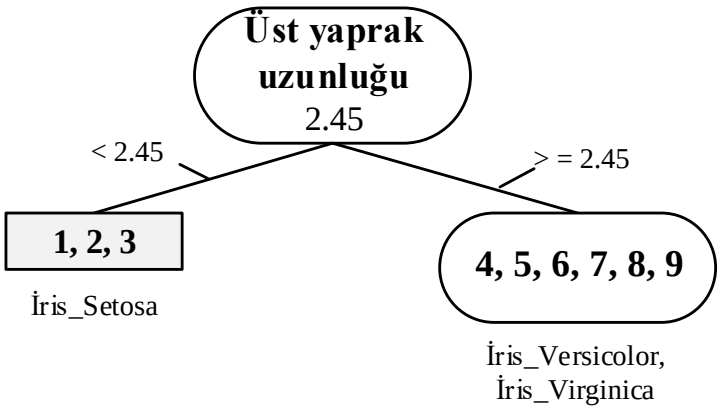
$$GiniGain(S_{10}) = \mathbf{0}$$

4. **Adım:** Hesaplama sonuçları *Çizelge 3.7*'de gösterilmiştir;

Çizelge 3.7 İlk Gini Kazanç Değerleri

Kesme No (i)	1	2	3	4	5	6	7	8	9	10
Kesme Pozisyonları (k_i)	0.55	1.15	1.4	2.45	3.4	3.75	4.4	4.9	5.1	5.3
Gini Kazancı	0	0.08	0.19	**0.53**	0.23	0.23	0.33	0.19	0.083	0

Gini kazancı en yüksek değer "0.53" olduğu için elde edilen sonuçlara göre ağaç 2.45 kesme pozisyon değerinden bölünecektir.



Şekil 3.3 İris Çiçeği Kök Düğüm Kesimi

Elde edilen sonuçlara göre ağaç iki dala ayrılmış olup kök düğüm belirlenmiştir. Sol düğümdeki kesme numaralarından 1, 2 ve 3 değerleri sadece *İris_Setosa* uygulama sınıfını içerdiği için sol düğüm yaprak düğüm olarak elde edilmiş ve *İris_Setosa* olarak etiketlenmiştir. Sağ düğümde henüz yaprak düğüme ulaşılamadığı için tekrar bölme işlemine tabi tutulacaktır.

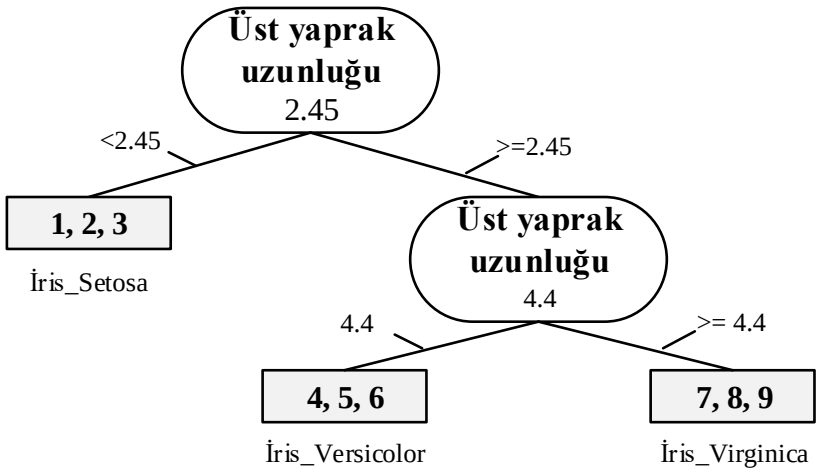
5. **Adım:** Kesme pozisyonları $4, 5, 6, 7, 8, 9$ için aynı işlemler tekrarlanarak ağacın sonraki kesim noktası elde edilmektedir. Bu işlem için bir önceki aşamada kesilen düğüm değerleri tablodan silinerek bütün adımlar tekrarlanmaktadır.
6. **Adım:** Bütün adımlar tekrarlanıp hesaplamalar yapıldığında *Çizelge 3.8*'deki değerler elde edilmektedir.

Çizelge 3.8 İris Çiçeği Düğüm Kesim Pozisyonları

Kesme No (i)	4	5	6	7	8	9	10
Kesme Pozisyonları (k_i)	2.45	3.4	3.75	4.4	4.9	5.1	5.3
Gini Kazancı	0	0,1	0,25	**0,5**	0,25	0,1	0

Gini kazancı en yüksek değer kesme numarası 7 (4.4) olan değer olduğu için elde edilen sonuçlara göre ağaç 4.4 kesme pozisyonundan bölünecektir.

Elde edilen sonuçlara göre *Şekil 3.3*'deki düğüm (4, 5, 6, 7, 8, 9) 2 dala ayrılmış ve *Şekil 3.4* ağacı elde edilmiştir. Sol düğümdeki 4, 5, 6 değerleri sadece *İris_Versicolor* uygulama sınıfına ait olduğu için yaprak düğüm olarak ayrılacak ve *İris_Versicolor* olarak etiketlenecek ve aynı şekilde sağ düğümdeki 7, 8, 9 değerleri sadece *İris_Virginica* uygulama sınıfına ait olduğu için yaprak düğüm olarak ayrılacak ve *İris_Virginica* olarak etiketlenecektir.



Şekil 3.4 İris Çiçeği İkinci Düğüm Kesimi

Ayrıca tüm değerler yaprak düğümlere ulaştığı için hesaplama ve kesme işlemleri tamamlanmıştır. Dolayısıyla Simple CART algoritması ile elde edilen ağaç *Şekil 3.4*'de gösterildiği gibi elde edilmiştir.

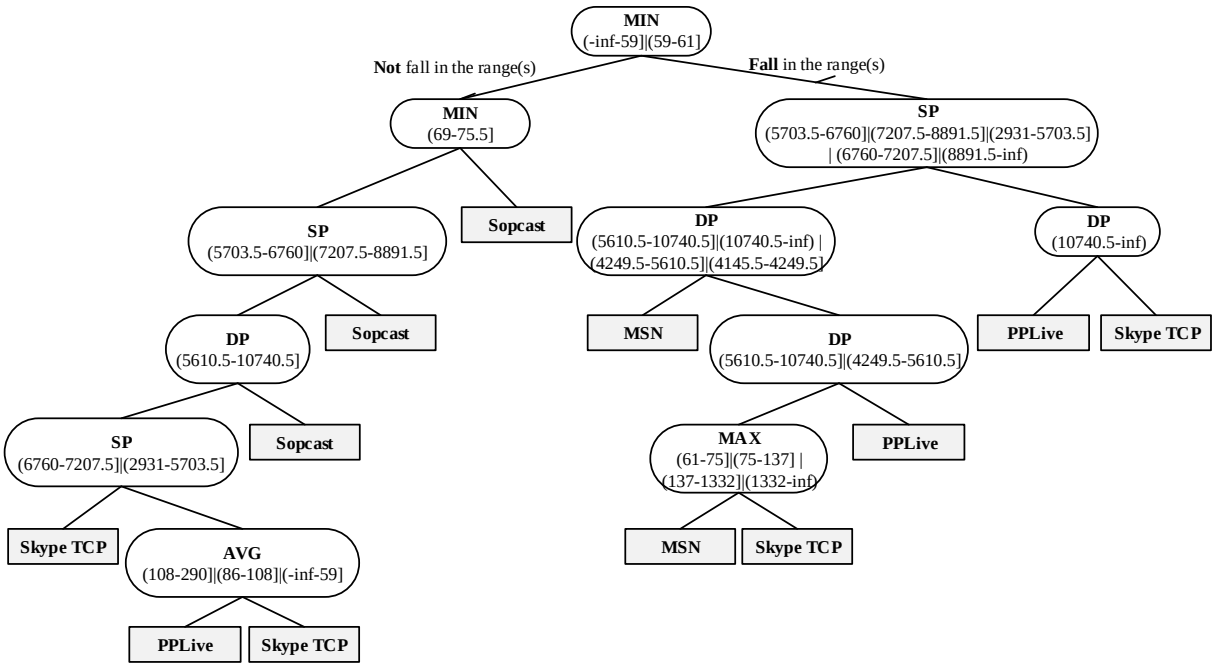
3.4. Veri Yapısı

Bu tez çalışmasında önerilen trafik sınıflandırma mimarileri için Simple CART karar ağacı algoritması kullanılacaktır. Simple CART karar ağacının en önemli sorunu düğümlerinde çoklu ve birbirine eşit olmayan aralıklar üretmesidir. Simple CART karar ağacını farklı büyüklüklerde düğümleri ile donanımda gerçeklemek donanım kaynaklarının oldukça verimsiz kullanımına neden olacaktır ve pratik değildir. Simple CART karar ağacı algoritmasını donanıma uygun veri yapılarına dönüştürmek bu tez çalışmasının önemli bir katkısıdır. Bu dönüşümün ana işlemi, aralık değeri bitmap dönüşümüdür.

3.4.1. Aralık Değeri-Bitmap Dönüşümü

Şekil 3.5'de gerçek veri setinden örneklenmiş alt veri seti ile elde edilen örnek bir Simple CART karar ağacı görülmektedir. Ağaç incelendiğinde bazı düğümlerin "1"; bazı düğümlerin "2" ve bazı düğümlerin ise "3" aralık değeri içerdiği görülmektedir. Dolayısıyla bu kadar farklı tipte ve eşit olmayan aralık değerlerine sahip bir yapıyı donanımda gerçeklenmek istendiğinde donanım kaynaklarının tahsisi en büyük düğüme göre hesaplanır (örnekte her düğüm için 3 aralık değeri ayrılır) ve bu durumda kaynakların verimsiz kullanımına neden olur. Bu dengesizliği ortadan kaldırmak ve her düğümde eşit aralık değerleri atayabilmek için aralık değerlerinin sabit uzunlukta bitmap dizilerine dönüştürülmesi önerilmiştir.

Bitmap dönüşümü için ilk adım her özellik setine ait tüm aralık değerleri ağaçtan okunur. Daha sonra bu değerler artan sırada bir tabloya yazılır ve bu değerlere ilk değerden başlanarak sıralı şekilde numara verilir. Bu numaralara *pozisyon numarası* adı verilir. Oluşturulan tablonun ardından o tabloya ait ağaçtaki özelliklere gidilir ve özellik değerleri okunmaya başlanır. Ağaçtan okunan değer oluşturulan tablo değerlerinden hangi aralığa giriyorsa tabloda o değerin karşısına "1" yazılır ve o aralığa girmiyorsa karşısına "0" yazılır. Bütün değer okumaları bu şekilde devam ettirilerek tablo tamamlanır ve elde edilen tabloya *"Özellik-Bitmap Tablosu"* adı verilir: MAX-Bitmap tablosu, MIN-Bitmap tablosu, SP-Bitmap Tablosu gibi.



Şekil 3.5 Ayrıklaştırılmış Simple CART Karar Ağacı

Örneğin *Şekil 3.5*'deki ayrıklaştırılmış Simple CART karar ağacında SP özelliğini göz önüne alalım. Ağaçtaki SP özelliğine ait aralık değerleri okunur: $5703.5, 6760, 7207.5, 8891.5, 2931$ (aynı değerlerden sadece bir tanesi yeterlidir). Ardından aralık değerleri bir tabloda artan sırada sıralanır ve ilk değer $-\infty$ ve son değer $+\infty$ olarak eklenir. Trafik sınıflandırma için tabloya $-\infty$ değeri yazılmasına rağmen eğitim aşamasında eksi değer gelmeyecektir. $-\infty$ olarak ifade edilen değer, uygulamada değişkenin alabileceği en küçük değer olarak ifade edilmektedir. Oluşturulan değerler *Çizelge 3.9*'da gösterilmektedir.

Çizelge 3.9 SP Aralık Değerleri

Aralık Değerleri	$-\infty$	2931	5703.5	6760	7207.5	8891.5	$+\infty$

Daha sonra SP özelliğinin yer aldığı her düğüm için SP_0, SP_1, SP_2 (örnek ağaçta 3 adet SP düğümü mevcut) gibi bir numara verilerek *Çizelge 3.10*'da gösterilen SP-Bitmap Tablosu oluşturulur. Sonra *Şekil 3.5*'deki Simple CART karar ağacından SP özellik değerleri okunarak başlanır. SP_0 düğümü değerleri okunur ($6760 - 7207.5$ ve $2931 - 5703.5$) ve bu değerlerin tablodaki karşılığına "1" ve diğer durumlara "0" değeri

yazılır. Benzer şekilde SP_1ve SP_2 düğümü için de aralık bit değeri ("0","1") ataması yapılır ve düğüm için bitmap dizisi elde edilir. SP özelliğinin tüm düğümlerinin aralık değerleri tabloda yazılmış ve SP-Bitmap Tablosu oluşturulmuştur. Benzer şekilde diğer özellikler için de bitmap tabloları oluşturulur. Bu dönüşüm ile düğümlerde bulunan farklı sayılardaki aralık değerleri, sabit uzunlukta 0 ve 1 ler ile temsil edilir. Düğümde aralık değerinin bulunması "1", aralık değerinin bulunmaması ise "0" ile temsil edilir. Bit dizisinde toplam bit sayısı veya dizi uzunluğu ağaçtaki o özelliğe ait farklı aralık değerlerinin sayısı kadardır.

Çizelge 3.10 SP-Bitmap Tablosu

Pozisyon No	1	2	3	4	5	6
Aralık Değerleri	-∞ 2931	5703.5	6760	7207.5	8891.5	∞
SP_0	**0**	**1**	**0**	**1**	**0**	**0**
SP_1	**0**	**0**	**1**	**0**	**1**	**0**
SP_2	**0**	**1**	**1**	**1**	**1**	**1**

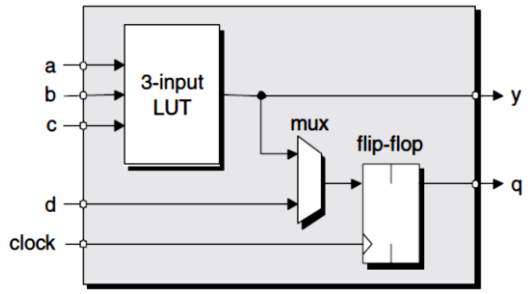
3.5. Veri Yapısı Donanım Gerçeklemesi

Ağaç veri yapılarının donanımda gerçeklenmesi için literatürde en çok tercih edilen platform *Alanda Programlanabilir Kapı Dizilimleri* (Field Programmable Gate Arrays, FPGA's)'dir. FPGA'lar blokları arasında yapılandırılabilir (programlanabilir) bağlantılar ve mantık blokları içeren dijital entegre devrelerdir (Integrated Circuits – ICs). Bu cihazlar farklı görevleri yerine getirmek için yeniden yapılandırılabilir (Maxfield, 2004).

FPGA, ilk kez 1980'lerin ortasında ortaya çıkmış ve ilk kullanım alanı telekomünikasyon ve ağ sektörü olmuştur. 1990'ların sonuna doğru ise diğer sektörlerde kullanılmaya başlanmıştır. Günümüzde nerede ise her alanda kullanılmaktadır (Maxfield, 2004). Bazı ana uygulama alanları ve sektörler şu şekildedir: 5G teknolojileri, uzay ve savunma sanayi, otomotiv, gerçek zamanlı TV, yayıncılık, tüketici elektroniği, veri merkezi, prototipleme, yüksek performanslı bilgi işlem merkezleri, sanayi, tıp, test ve ölçüm, kablolu ve kablosuz iletişim, gömülü görüş sistemleri, endüstriyel yapay zeka ve bulut bilişim (Xilinx, 2018).

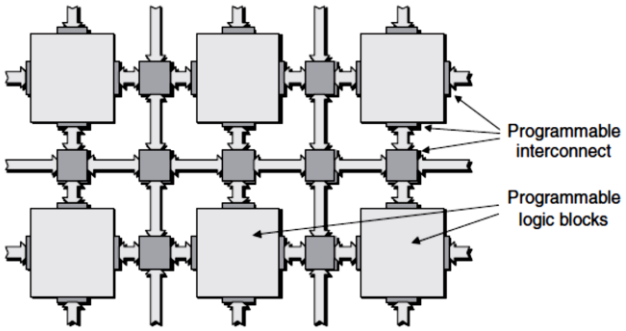
3.5.1. FPGA Mimarisi

1980'lerin başında *Uygulamaya Özel Entegre Devre* (The Application Specific Integrated Circuit – ASIC)'ye rakip olabilecek ara elemanlar arayışı sürerken Xilinx firması tarafından FPGA mimarisi ortaya çıkarıldı. *Şekil 3.6*'de 3 girişli LUT, 1 adet MUX ve 1 adet Flip-Flop içeren basit bir FPGA mimarisi gösterilmektedir (Maxfield, 2004).



Şekil 3.6 Basit Bir FPGA Mimarisi

Şekil 3.7'da genel bir FPGA mimarisi gösterilmektedir. FPGA'ların diğer dijital devrelerden farklı olarak ara bağlantıları da programlanabilmektedir (Maxfield, 2004).



Şekil 3.7 Genel FPGA Mimarisi

3.5.2. Donanım Tanımlama Dilleri (HDL)

Herkes kendi Donanım Tanımlama Dili (Hardware Description Language – HDL) yazmaya başlamasıyla birçok HDL dil ortaya çıkmıştır. Günümüzde en çok kullanılan Verilog HDL ve VHDL (Very High Speed Integrated Circuit - HDL) dilleridir (Maxfield, 2004).

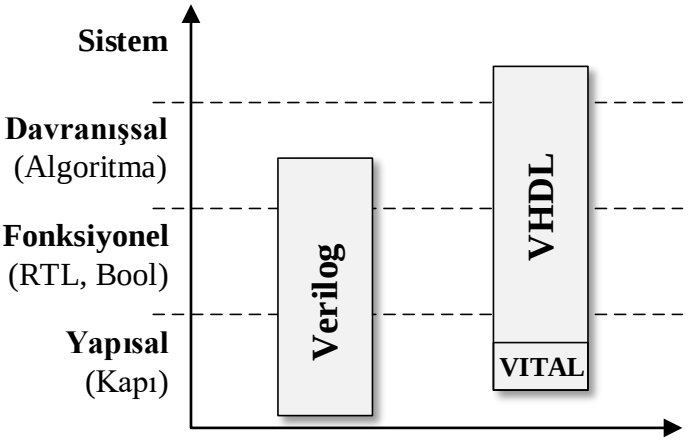
3.5.2.1. Verilog HDL

1980'lerin başlarında Phil Moorby (ünlü HILO lojik simülatörü oluşturan ekibin üyesi) tarafından Verilog adında yeni bir HDL tasarlanmıştır. Kullanım olarak C diline benzerliği nedeniyle oldukça kolay kullanıma sahiptir (Maxfield, 2004).

3.5.2.2. VHDL

1980'lerde ABD Savunma Bakanlığı'nın başlatmış olduğu program sonucunda VHDL dili ortaya çıkmıştır (Maxfield, 2004).

Şekil 3.8'de VHDL ve Verilog arasındaki farklar gösterilmektedir (Maxfield, 2004).



Öğrenmesi nispeten kolay	Öğrenmesi nispeten zor
Sabit veri türleri	Soyut veri tipleri
Yorumlanan yapılar	Derleyen yapılar
İyi kapı seviyesi zamanlaması	Az iyi kapı seviyesi zamanlaması
Sınırlı tasarım yeniden kullanılabilirliği	İyi bir tasarım tekrar kullanılabilirliği
Sınırlı tasarım yönetimi	İyi bir tasarım yönetimi
Yapı çoğaltması yok	Yapı çoğaltma desteği

Şekil 3.8 Soyutlama Düzeyleri (VHDL & Verilog)

Şekil 3.8'den de anlaşılacağı üzere her iki dil de aynı alanlarda kullanılabilir. İki dil arasında önemli bir fark söz konusu değildir. En büyük fark ise sistem tarafında VHDL dilinin daha iyi cevap verebilmesidir.

BÖLÜM 4

SIMPLE CART TABANLI GERÇEK ZAMANLI TRAFİK SINIFLANDIRMA MİMARİSİ

4.1. Giriş

Bu bölümde gerçek zamanlı internet trafik sınıflandırma için ML karar ağacı algoritmalarından *Basit Sınıflandırma ve Regresyon Ağaçları* (Simple Classification and Regression Trees – (Simple CART)) algoritması kullanılarak gerçekleştirilen internet trafik sınıflandırma mimarisi tasarımı anlatılacaktır. Bu algoritma, internet trafik sınıflandırma için literatürde ilk kez bu çalışmada kullanılmıştır.

Bu bölümde aşağıdaki özetlenen katkılar anlatılacaktır:

- *Genişletilmiş Simple CART* (Extended–Simple CART – (E-Simple CART (*E-SC*))) adı verilen iki aşamalı bir hibrid veri yapısı. Bu hibrid yapının 1. Aşaması her bir özellikten ayrı ayrı elde edilen *özellik ağaçları*ndan (Feature Tree - FT), 2. aşaması ise düğümleri bitmaplerle zenginleştirilmiş (aralık değeri bitmap dönüşümü yapılarak) Simple CART karar ağacından oluşmaktadır (*Bölüm 4.2*). *E–SC* veri yapısı, sınıflandırma doğruluğunu kaybetmeden, ayrıklaştırılmış Simple CART karar ağacının alternatif bir temsilidir, aynı zamanda bu yapı ağaçta meydana gelen çoklu ve çeşitli aralık sorunlarını da çözmektedir (*Bölüm 4.2.3*).
- Önerilen *E-SC* veri yapısını destekleyen ve 8 uygulama sınıfı içeren trafiği izlemek için $557\ Gbps$ (veya $1741\ MCPS$) (minimum 40 baytlık paket

boyutu için) iş oranı ve %96.8125 doğruluk oranı sağlayan donanım olarak FPGA'lar üzerinde yüksek iş oranlı iki seviyeli bir mimari (*E-SC* mimarisi – *Bölüm 4.3*) önerilmiştir.

4.2. Algoritma Ve Veri Yapısı

4.2.1. Tanımlar

4.2.1.1. Özellik (Feature)

Özellik (feature), bir makine öğrenmesinde resmi olarak gözlemlenen bir olgunun ölçülebilir bir niteliği olarak tanımlanır. Örneğin, İnternet trafik sınıflandırmada, trafik sınıfını (Skype, MSN vb.) tahmin ederken kullanılan, giriş özellikler *kaynak portu* (SP), *hedef portu* (DP), *ortalama paket boyutu* (AVG), *minimum paket boyutu* (MIN)) vb.'dir (Soylu, vd., 2017).

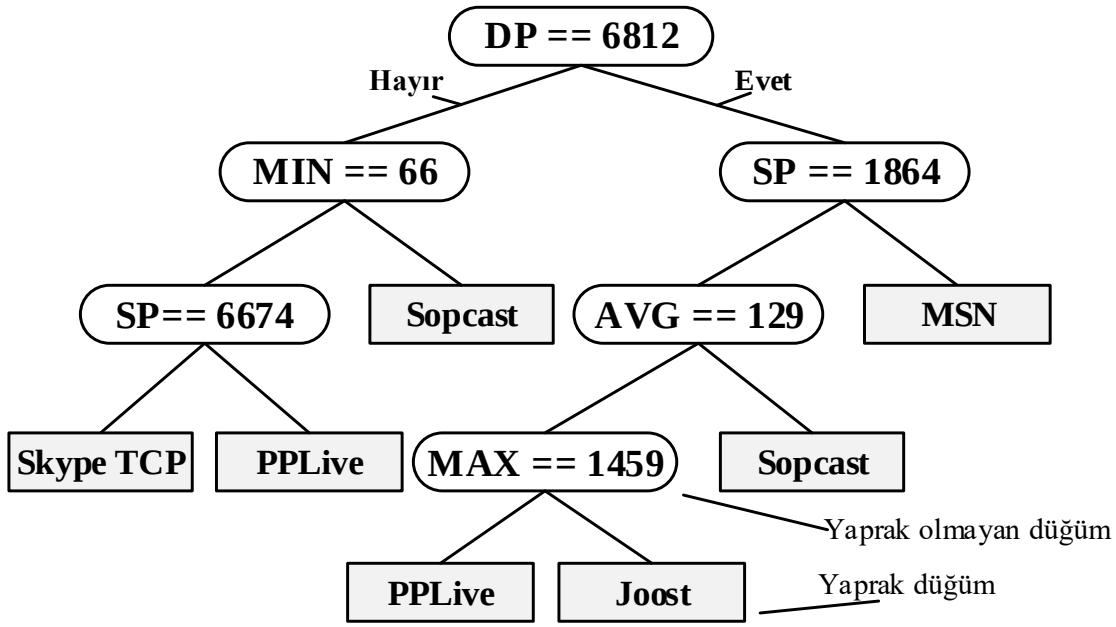
4.2.1.2. Trafik Akışı

Trafik akışı (traffic flow), aynı 5 *çokuzlu* (tuple) başlık alanlarını (SA, DA, SP, DP ve Protokol) paylaşan bir dizi internet paketini temsil eder (Soylu, vd., 2017).

4.2.2. Motivasyon

Literatürde yer alan mevcut çalışmalarda (Alshammari ve Zincir-Heywood, 2009; Kim, vd., 20008) yapılan analizlere dayanarak, karar ağacı tabanlı algoritmalardan özellikle *C4.5* algoritmasının, yüksek doğruluk sağlama açısından *SVM* (Este, vd., 2009; Papadonikolakis ve Bouganis, 2012), K-means (Jiang ve M. Gokhale, 2010; Lin, vd., 2012) ve *NB* (Moore ve Zuev, 2005) gibi diğer tüm ML algoritmalarını geride bıraktığı görülmektedir. Bir karar ağacı, her bir düğümün belirli bir özellik üzerinde bir test gerçekleştirdiği akış şemasına benzer bir veri yapısıdır. *Şekil 4.1* giriş trafiğini Skype, PPLive, Sopcast, Joost ve MSN gibi uygulama sınıflarına ayrıştırmak için *kaynak portu* (SP), *hedef portu* (DP), *ortalama paket büyüklüğü* (AVG), *minimum paket boyutu* (MIN) ve *maksimum paket boyutu* (MAX) özelliklerini kullanan örnek bir ikili karar ağacını gösterir. Ağaçtaki bir *yaprak düğüm (leaf node)*, arama sonucu olan uygulama sınıfını

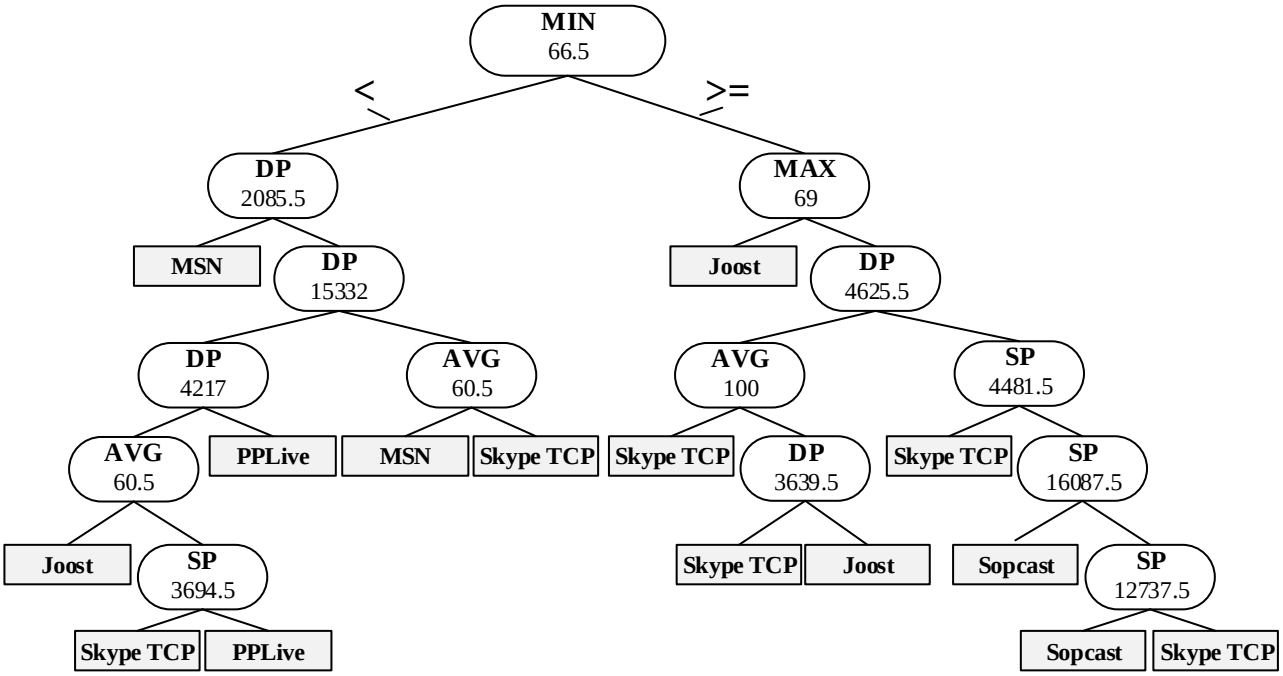
(sınıf kimliği) temsil eder ve *yaprak olmayan düğüm (non-leaf node),* ilgili özellik temelinde bir kararı temsil eder.



Şekil 4.1 Örnek Bir İkili Karar Ağacı

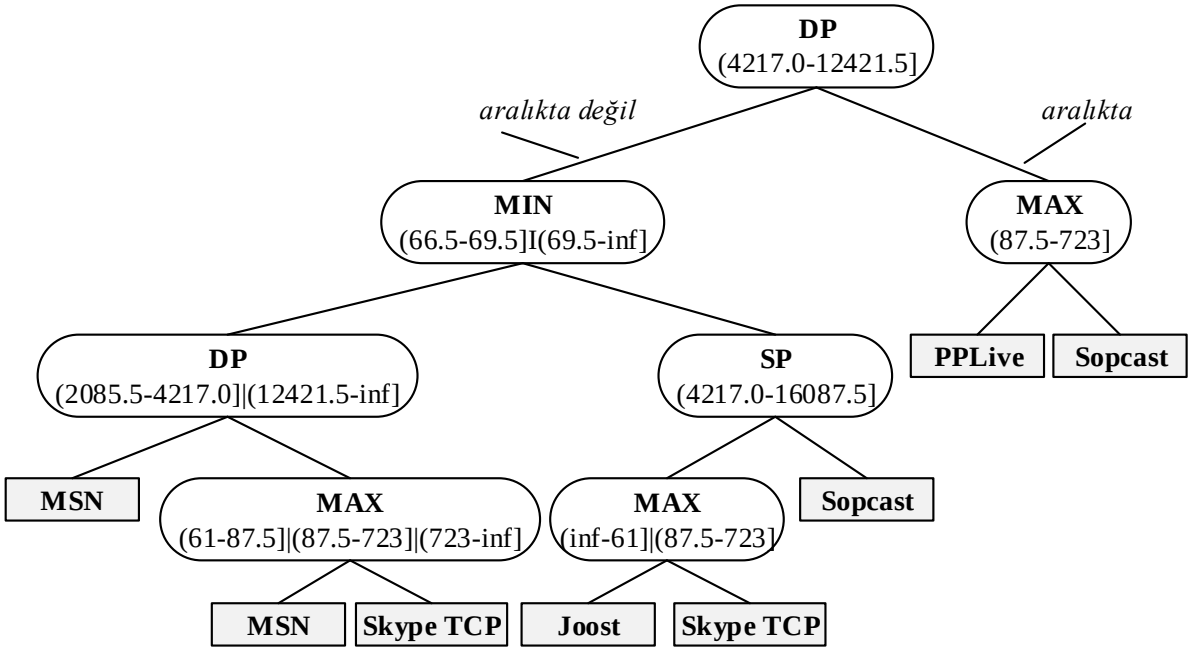
Son zamanlarda yapılan çalışmalar, genel karar ağacı tabanlı trafik sınıflandırmayı hızlandırmak için yüksek performanslı boru hatlı (pipeline) donanım mimarilerini kullanmayı önermektedirler (Tong, vd., 2013; Gandhi, vd., 2014; Qu ve Prasanna, 2015). Buradaki çözümlerde genellikle klasik *C4.5* karar ağacı kullanılmaktadır. Bir *C4.5* karar ağacı nispeten az sayıda yaprak düğümleri içermesine rağmen, arama işlemlerinde uzun gecikmelere neden olan oldukça dengesiz bir yapıya sahiptir. Ayrıca, dengesiz bir ağaç yapısının uygulama sınıflarının sayısı bakımından ölçeklenebilirlik sorunu olasıdır. Gelecekte sınıf sayısındaki potansiyel artış nedeniyle, *C4.5* karar ağacı yapısının arama performansı da buna göre düşme eğiliminde olacaktır. Diğer bir deyişle, en kötü durum performansı, (en kötü durum arama süresi) dengesiz bir ağaçtaki ağaç düğümleri sayısı ile doğrusal olarak değişir (Tong, vd., 2014; Gandhi, vd., 2014; Qu, vd., 2014; Qu, vd., 2015). Ayrıca, karar ağacı boru hattı donanımına eşlendiğinden, bir ağacın derinliği dengesiz yapısından dolayı artar ve bu artış kaynak ve bellek verimliliğini de olumsuz yönde etkiler.

Basit Sınıflandırma ve Regresyon Ağaçları (The Simple Classification and Regression Trees – Simple CART) algoritması metodolojisi literatürde ilk defa Breiman (Breiman, vd., 1984) tarafından tanıtılmıştır. Simple CART algoritması ikili bir karar ağacı oluşturmaktadır. Ancak, Simple CART algoritmasını, *C4.5* algoritmasından farklılaştıran en önemli unsur ise en iyi bölünme ölçütünü seçmek için *entropi ölçümü*nü kullanmasıdır. Simple CART algoritması en iyi ayırma özelliğini seçmek için, tüm özellikler için olası tüm ayırmaları denetlemektedir (tüm olasılıkların ayrıntılı araştırılması). Bu işlem, her düğüm için daha fazla bölünme imkansız hale gelene kadar sürekli olarak sürdürülmektedir. Kategorik veya sayısal veri kümesine dayanarak, sınıflandırma veya regresyon ağaçları sırasıyla oluşturulur. Simple CART, budama işleminde en iyi ağacı seçmek için *çapraz doğrulama* yöntemini kullanmaktadır. Gözlemlerimize dayanarak, Simple CART karar ağacının belirli bir veri kümesine bağımlı olmadığını (veya diğer algoritmalara göre çok daha az bağımlı olduğunu) ve giriş trafiğindeki aykırı değerlerden önemli ölçüde etkilenmediğini görülür. Dahası, *C4.5* karar ağacıyla kıyaslandığında, Simple CART karar ağacı daha dengelidir. *Şekil 4.2* bir Simple CART karar ağacı örneğini göstermektedir.



Şekil 4.2 Örnek Bir Simple CART Karar Ağacı

Trafik sınıflandırmada yüksek doğruluk elde etmek için, mevcut sınıflandırıcıların çoğu, giriş trafik özelliklerinin sürekli değerlerini ayrık değerlere dönüştüren *ayrıklaştırma süreci* (discretization process) (Tong, vd., 2013; Qu ve Prasanna, 2014; Lim, vd., 2010; Kim, vd., 2008) kullanmaktadır. Ayrıca, *ayrıklaştırma süreci*nin Simple CART algoritmasının doğruluğunu, mevcut algoritmalara benzer şekilde, önemli ölçüde arttırmaktadır. Buna ilaveten, *ayrıklaştırma süreci*, karar ağacının büyüklüğünü, iç ve yaprak düğüm sayısı bakımından yaklaşık olarak yarıya indirmektedir. *Şekil 4.3*, *Şekil 4.2*'de verilen Simple CART ağacının ayrıklaştırılmış versiyonunu göstermektedir. Bu iki ağacı karşılaştırdığımızda, iç düğümlerin sayısının ve yaprak düğümlerin sayısının sırasıyla 14 düğümden 7 düğüme ve 15 düğümden 8 düğüme azaldığı görülmektedir. Diğer taraftan, *ayrıklaştırma işlemi*, *Şekil 4.3*'te görülebileceği gibi, Simple CART karar ağacında düğüm büyüklüğü çeşitliliğine (farklı büyüklükte düğümler) yol açmaktadır. Ağaç veri yapıları boru hattı donanım mimarilerine eşlenirken, her bir düğüm için ayrılan kaynak miktarı ağacın en büyük düğümü üzerinden hesaplanır. Sonuç olarak, ayrıklaştırılmış Simple CART karar ağacı, boru hattı donanım yapısı üzerinde uygulandığında düğümlerindeki aralıkların değişken sayısı nedeniyle fazladan bellek kullanımına neden olmaktadır. Benzer şekilde, düğüm büyüklüklerindeki çeşitlilik, lojik kaynakların verimli kullanımını da olumsuz yönde etkilemektedir.



Şekil 4.3. Ayrıklaştırılmış Örnek Bir Simple CART Karar Ağacı

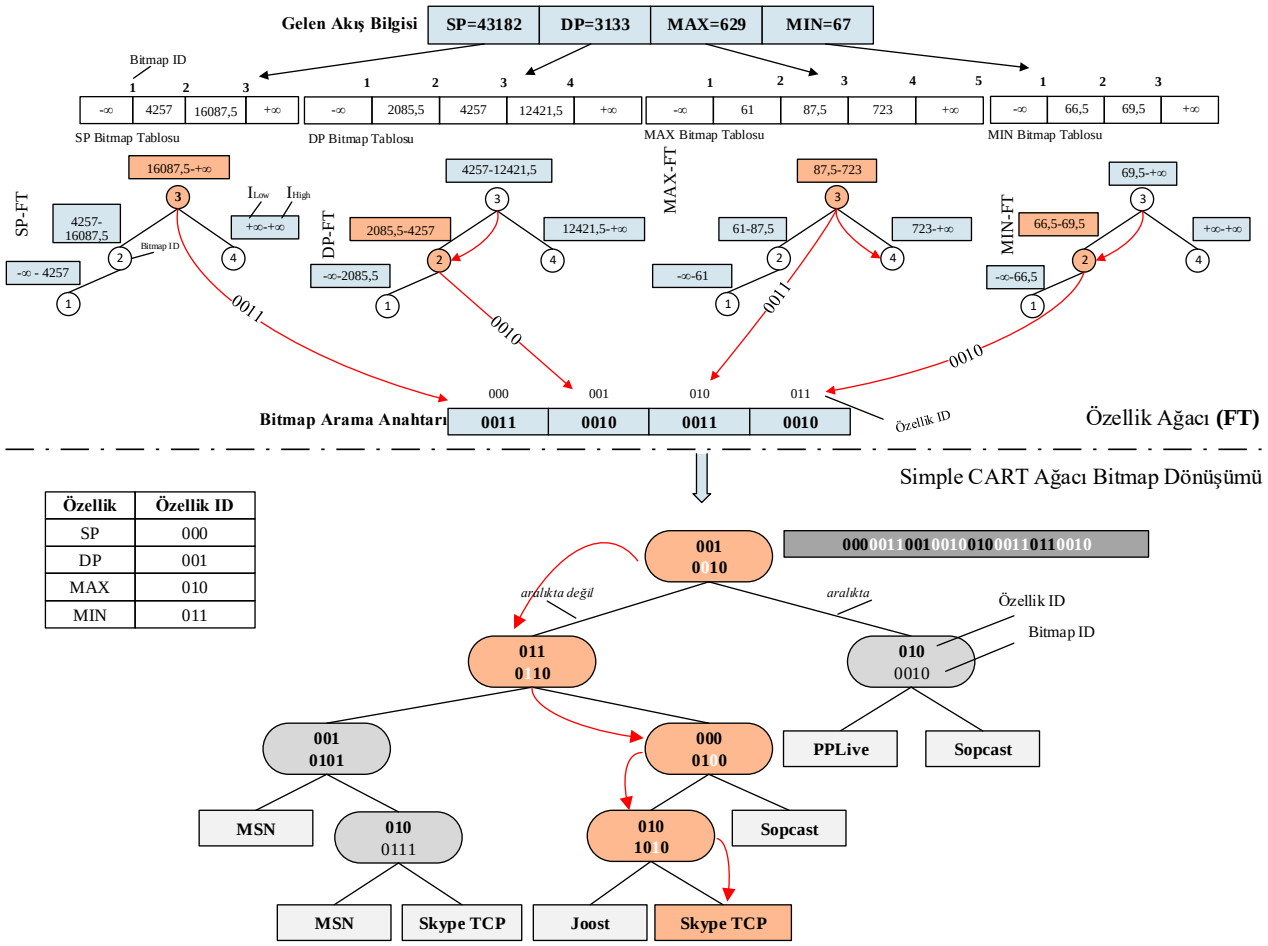
Yukarıda özetlenen sorunları çözmek için, ayrıklaştırılmış Simple CART karar ağacı iki aşamalı hibrid veri yapısına (Genişletilmiş Simple CART) dönüştürülmesi önerilmiştir. İki aşamalı yapının *1. aşama*sında her bir özellik değerinden elde edilen *Özellik Ağaçları* (Feature Tree – FT) ve *2. aşama*da ise düğümleri bitmapler ile zenginleştirilmiş Simple CART karar ağacından oluşmaktadır. Sonuç olarak, önerilen şema yüksek iş oranı elde etmek için Alanda Programlanabilir Kapı Dizilimleri (Field Programmable Gate Arrays–FPGA) kullanılarak paralel ve boru hattı (pipeline) mimari üzerinde uygulanacaktır.

4.2.3. Genişletilmiş Simple CART (*E-SC*)

Önerilen *Genişletilmiş Simple CART* (E-SC) veri yapısı 2 aşamadan oluşmaktadır:

4.2.3.1. Aşama 1 Özellik Ağacı (Feature Tree – *FT*)

Kullanılan her bir özellik (DP, SP, AVG, MIN vb.) için bir *özellik ağacı* (Feature Tree – *FT*) oluşturulmaktadır. *Şekil 4.4*, *Şekil 4.3*'te verilen ayrıklaştırılmış Simple CART karar ağacının her bir özelliğe ait aralık değerlerini ayrı tablolar şeklinde ve ayrıca yatay tabloda göstermektedir. *FT* ağaçlarının oluşturulması iki adımdan oluşur: i) tüm düğümlerdeki ayrık aralık değerlerinin üst ve alt sınırlarını bir sayı çizgisi üzerinde işaretlemek (Şekil 4.4 görüldüğü gibi) ve (ii) sayı çizgisi üzerinde bulunan aralık değerlerinden kök olarak doğru aralığı (pivot) seçmek ve sol ve sağ alt ağaçları yinelemeli olarak oluşturmak. Oluşturulan ağaç yapısında, herhangi bir düğümün sol alt ağacı, üst sınırları o düğüme dâhil olan alt sınıra eşit veya daha küçük olan aralıklar içerir. Benzer şekilde, sağ alt sınır, alt sınırları bu düğümün üst sınırından daha büyük olan aralıkları tutar. Özetle, aralık ağacı arama alanını ayrık aralıklara ayırır ve ağacın her bir düğümü tek bir aralık değerine karşılık gelir. Bir *FT* ağacındaki her düğüm beş alanı saklar; (1) aralığın alt sınırı (I_{low}), (2) aralığın üst sınırı (I_{high}), (3) sol çocuk işaretçisi (Ch_L), (4) sağ çocuk işaretçisi (Ch_R) ve (5) Bitmap Kimliği (B_{ID}).



Şekil 4.4 *Şekil 4.3*'te Verilen Ağacın E-Simple CART (E-SC) Versiyonu

FT ağaçlarında arama işlemine başlamadan önce, gelen trafiğin tüm paket başlık bilgileri ayıklanır ve ilgili paket başlık bilgisi veya özellik değeri bağımsız olarak ilgili *FT* ağacında aranır. Farklı *FT* ağaçlarında her paralel arama işleminin iki sonucu vardır; (i) *Özellik Kimliği* (F_{ID}) ve (ii) *Bitmap Kimliği* (B_{ID}). Her özellik, *özellik kimliği* olarak adlandırılan benzersiz bir tanımlayıcıya sahiptir. Örneğin, *Şekil 4.4*'de gösterildiği gibi, DP ve SP Özellik-ID değerleri sırasıyla "001" ve "000" olan özellik kimlik değerlerine sahiptir. Bir *FT* ağacının her düğümünde yer alan tablodaki her bir aralık değerine karşılık gelen bir sayı değeri vardır. Düğümlerdeki aralık değerleri sıralanır ve tablodaki karşılık gelen ilk değer 1 ve son değer ise aralık sayısı adedi olan değer olmak üzere değerler atanır (*Bitmap Kimliği* – B_{ID}). Bitmap dönüşümü *Bölüm 3.4.1*'de ayrıntılı olarak anlatılmıştır. Bir *FT* ağacında arama sırasında bir eşleşme bulunduğunda, eşleşen düğümde depolanan B_{ID} değeri alınır ve sonraki aşamadaki arama işleminde kullanılır.

Sonuç olarak, *Şekil 4.4*'te gösterildiği gibi çoklu *FT* ağaçlarında arama sonuçlarından (*Bitmap ID* değerleri) bir *bitmap arama anahtarı* (bitmap search key) oluşturulmaktadır. *Şekil 4.4*'deki *bitmap arama anahtarı*, her bir ağaçtan gelen B_{ID} değerleri için 4 slot içerir ve bir slotun bit uzunluğu, "$logN_{max} + 1 = 3$" bit olarak hesaplanır. Burada N_{max} en büyük aralık tablosundaki aralıkların sayısı (veya en büyük FT ağacındaki düğüm sayısı) ile tanımlanır.

4.2.3.2. Aşama 2 Bitmap İşlenmiş Simple CART Ağacı (SC-B)

Bitmap işlenmiş Simple CART Ağacı (Simple CART Tree with Bitmaps – (*SC-B*)), klasik Simple CART ağacı ile aynıdır. Buradaki tek fark düğümlerde bulunan aralık değerleri, bu aralıkları bitler ile temsil eden sabit uzunlukta bit dizilerine (veya bitmap) dönüştürülmüştür. Ağaç düğümlerinin büyüklüklerini eşitlemek için, her bir düğümde bulunan bit dizilerinin uzunluğu, bir önceki aşamada verilen bitmap tablolarındaki en fazla sayıdaki aralık sayısı ile sabitlenir. Örneğimizde, bitmap uzunluğu, aralık tablosundaki (en fazla aralık değeri içeren özellik) girişlerin sayısı olan 4'dür. *SC-B* her bir ağaç düğümü, bir bitmape ek olarak, o düğümde saklanan özellik bilgisini taşıyan *özellik kimlik* değerini de saklar. Örneğin *Şekil 4.3*'deki, ayrıklaştırılmış Simple CART karar ağacının kök düğümündeki DP özelliğine ait düğümdeki, "$4217.0 - 12421.5$" aralık değeri yeni yapıda "001", "0010" ile değiştirilir; burada "001" değeri DP özelliğinin bir *özellik kimliği* (F_{ID}) ve "0010" değeri ise DP aralık tablosunda sadece bir aralık değeri olan, DP_2'nin bu düğümde depolandığını tanımlayan bitmaptir. Bir *SC-B* ağacı düğümü beş alanı saklar: (1) sadece yaprak olmayan düğümlerinde özellik tanımlayıcısı (F_{ID}), (2) bitmap veya bit vektörü, (3) sol alt çocuk işaretçisi (SCh_L), (4) sağ alt çocuk işaretçi (SCh_R) ve (5) sadece yaprak düğümlerde sınıf kimliği (C_{ID}).

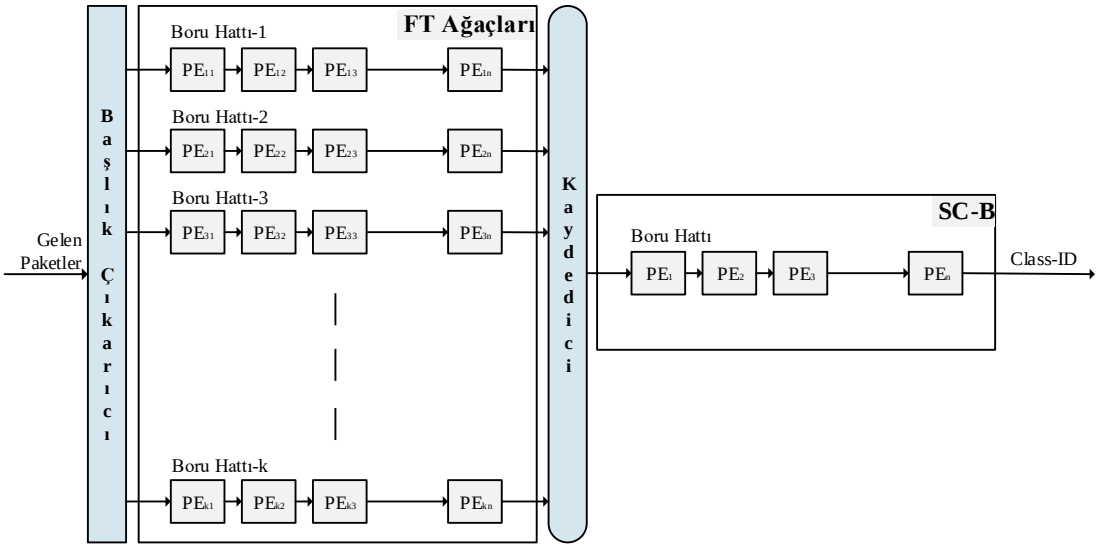
Bir *SC-B* ağacında aramada, *Aşama 1*'de elde edilen *bitmap arama anahtarı Aşama 2*'deki karar ağacına aktarılır. Her düğümde, saklanan F_{ID} değerine karşılık gelen tek bir B_{ID} değeri, arama anahtarından çıkarılır. Daha sonra, düğümde kayıtlı bit dizisinde B_{ID} değerinin işaret ettiği (B_{ID}. sıradaki bit) bit değeri "1" ise, aramalar sağ düğüme; aksi halde sol düğüme geçer. Bir bitmap içindeki "1" biti, giriş özellik değerinin o düğümde depolanan, belirtilen aralığa düştüğünü ifade eder.

Giriş internet trafik akış bilgisinin "$SP = 43182$", "$DP = 3133$", "$MAX = 629$" ve "$MIN = 67$" olarak verildiğini varsayalım. *Aşama 1*'de, gelen "$SP = 43182$" değeri kendi *SP-FT* ağacına girer ve "$16087.5 - \infty$ " aralığında olduğundan "3" değerini alıp ağaçtan çıkar; "$DP = 3133$" değeri de kendi *DP-FT* ağacına girer, kökteki değerden küçük olduğu için sol düğüme geçer ve bu düğümdeki "$2085.5 - 4217$" aralığına girdiği için "2" değerini alarak ağaçtan çıkar; "$MAX = 629$" değeri *MAX-FT* ağacına girer, kök düğümdeki "$87.5 - 723$" aralığına girdiği için "3" değerini alarak ağaçtan çıkar; "$MIN = 67$" değeri *MIN-FT* ağacına girer, kökteki değerden küçük olduğu için sol düğüme geçer ve bu düğümdeki "$66.5 - 69.5$" aralığına girdiği için "2" değerini alarak ağaçtan çıkar. *Şekil 4.4*'te gösterildiği gibi, *Aşama 1*'de, arama sonucu *bitmap arama anahtarı* "3232" (veya ikili "0011 0010 0011 0010") olarak elde edilir. *Aşama 2*'de, *SC-B* ağacında arama kök düğümden başlar. Kök düğümdeki "$F_{ID} = 001$" değeri, arama anahtarı "001" olan ikinci B_{ID} değerini ("0011 **0010** 0011 0010") gösterir. Bitmapin karşılık gelen bit konumu ("0**0**10") "0" olduğundan, arama sol düğüme iletilir. Bir sonraki düğümde, "$F_{ID} = 011$" değeri, "011" olan üçüncü B_{ID} değerini ("0011 0010 **0011** 0010") işaret eder. Bitmapin ("0**1**10") ikinci biti "1" olduğundan, arama sağ çocuk düğümüne yönlendirilir. Arama, yaprak düğümüne ulaşılana kadar aynı şekilde aramaya devam edilir. *Şekil 4.4*'teki kırmızı çizgiler, verilen arama örneği için hem *Aşama 1* hem de *Aşama 2*'de arama yollarını göstermektedir. Sonunda, Skype TCP verilen akışın *Sınıf Kimliği* (Class ID – (C_{ID})) olarak bulunur.

4.3. Donanım Mimarisi Ve FPGA'da Uygulaması

Şekil 4.5, önerilen *E-SC* arama mimarisinin blok diyagramını göstermektedir. İş oranını artırmak için boru hattı (pipeline) mimarisini kullanılmıştır. *FT* boru hatları *FT* ağaçlarını depolar ve *SC-B* boru hattı ise *SC-B* ağacını saklar. Ağaç veri yapısı boru hattı üzerine işlenirken, her boru hattı aşaması (pipeline stage) ağacın bir seviyesine karşılık gelecek şekilde boru hattına eşlenir. *FT* boru hatlarının toplam sayısı (K), makine öğrenmesinde kullanılan özelliklerin sayısına eşittir. Tek bir boru hattındaki aşama sayısı, bu boru hattına eşlenen ağaç yapısının derinliğine bağlıdır. Bir *FT* ağacının derinliği, karşılık gelen aralık tablosundaki ayrık aralıkların sayısına bağlıdır ve bu nedenle *FT*

ağaçlarının derinliği eşit olmak zorunda değildir. Arama işlemleri tüm *FT* boru hatlarında paralel olarak işlendiğinden, paralel aramaların gecikme sürelerini dengelemek için daha kısa olan *FT* boru hatlarının sonuna ek gecikme blokları eklenebilir.

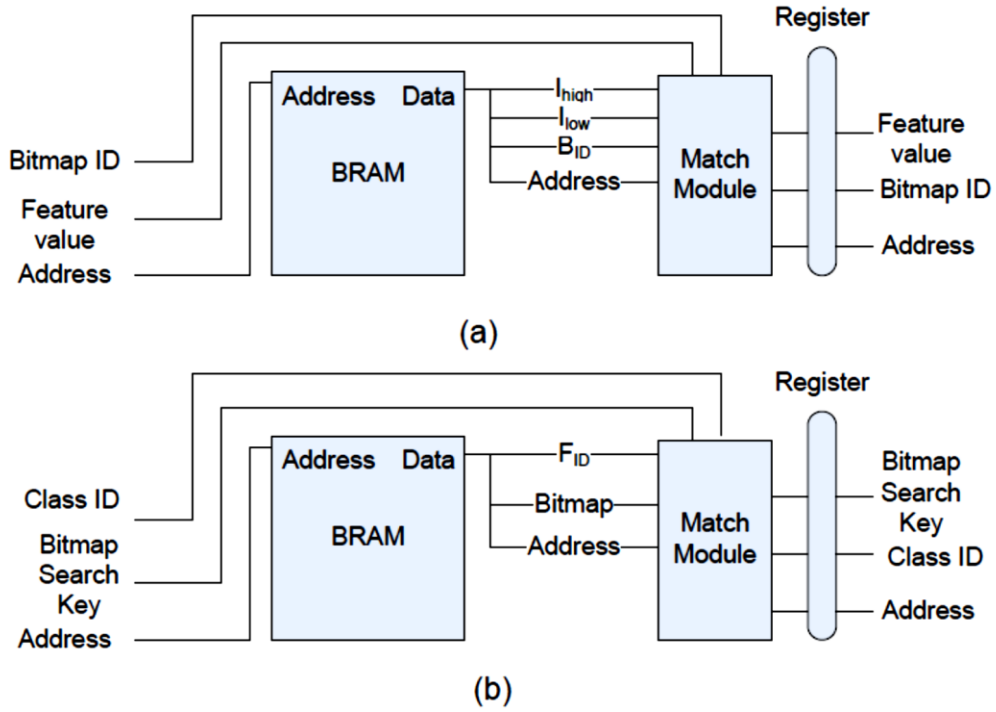


Şekil 4.5 E-Simple CART Boru Hattı Mimarisi

Şekil 4.6a ve *Şekil 4.6b*, sırasıyla *FT* ve *SC-B* boru hatlarının tek bir aşamasını göstermektedir. FPGA üzerinde gerçeklenen mimaride, tek bir boru hattı aşaması bir *Blok RAM* (BRAM) ve bir *eşleme modülü*nü içermektedir. *BRAM*, ağaç düğümlerini depolamak için kullanılır ve *eşleme modülü*, aralık karşılaştırmaları veya bit inceleme işlemlerini uygulayan ve adresi bir sonraki aşama için hazırlayan bir mantık devresidir. Güncel FPGA'ların BRAM'leri çift okuma/yazma bağlantı noktalarına sahip olduğundan, her saat döngüsünde iki arama aynı anda gerçekleştirilebilir. Bu durumda, *E-SC* mimarisi, iş oranını (throughput) ikiye katlayan çift doğrusal boru hatları olarak yapılandırılmıştır.

Başlık çıkarıcı modülü, paketlerin akışını depolar ve daha sonra *protokol*, *SP* ve *DP* özellik değerleri gibi *klasik özellik* değerlerine ek olarak, *ortalama*, *maksimum*, *minimum paket büyüklüğü* gibi gerekli *akış seviyesi* istatistiksel özellik değerlerini hesaplar. İstatistiksel özellik değerleri, ilk birkaç paket trafik akışlarının sınıflandırılmasında kullanılabilir (Tong, vd., 2013).

Mimarilerimiz uygulama sınıflarının sayısı bakımından ölçeklenebilirdir, çünkü sınıf sayısındaki değişim boru hatlarının sayısında artışa neden olmaz.



Şekil 4.6 (a) FT Boru Hattı (b) SC-B Boru Hattı Tek Bir Aşaması (Soylu, 2017)

4.4. Performans Değerlendirmesi

4.4.1. Deney Düzeneği

Tstat (Tstat, 2016)'dan alınan veri seti, *Çizelge 4.1*'de listelenen 8 uygulama sınıfını içermektedir. Bu uygulama sınıfları arasında, geleneksel port numarası tabanlı sınıflandırıcılar tarafından sınıflandırılamayan P2P servislerini ve anlık mesajlaşma uygulamaları da mevcuttur. Veri seti, her bir uygulama için 800 olmak üzere toplamda 6400 trafik akışından oluşmaktadır. Veri seti oluşturma ve uygulama sınıfı detayları *Bölüm 3.1*'de ayrıntılı olarak anlatılmıştır.

Çizelge 4.1 E-SC Uygulama Sınıfları

Uygulama	**Açıklama**
Skype TCP	Skype Ses ve Video Araması
Skype UDP	Skype Ses ve Video Araması
Yahoo IM	Yahoo Anlık Mesajlaşma
Joost	P2P TV Programı
PPLive	P2P TV Programı
Sopcast	P2P TV Programı
TVAnts	P2P TV Programı
MSN	MSN Microsoft Messenger

4.4.2. Özellik Setinin Yapısı

Çizelge 4.2, bu bölümde kullanılacak aday özellik listesini göstermektedir. Akış seviyesi özellikleri, akıştaki ilk *R* paket dikkate alınarak hesaplanan değerleri temsil etmektedir. *R* değeri, mevcut çalışmalara benzer şekilde 4 olarak seçilmiştir (Tong, vd., 2013). Örneğin *MIN* özelliği, akıştaki ilk 4 paket arasında minimum paket boyutu anlamına gelir. Önerilen *E-SC* şeması, donanım üzerinde uygulanacağı için, seçilen özellikler de donanımla uyumlu olmalıdır. Varyans dışındaki tüm özellikler, uygun bir donanım tarafından kolayca çıkartılabilir veya hesaplanabilir. Varyans değeri, *ilk R paketin boyutunun standart sapmasının karesi* alınarak hesaplanır. Donanımda standart sapma ve kare hesaplama gibi çarpma işlemlerini uygulamak maliyetlidir. Dolayısıyla varyans değerinin etkisini ölçmek için bir dizi analiz yapılmış olup ayrıntılar *Bölüm 3.2*'de verilmiştir. Analiz sonrası elde edilen sonuçlara göre varyans değeri setten çıkarılmıştır. Özetle, önerilen trafik sınıflandırma mimarisi için 6 özellik değeri seçilmiştir.

Çizelge 4.2 Aday Özellik Listesi

Özellik	**Açıklama**
Prtcl	Protokol
SP	Kaynak Port Numarası
DP	Hedef Port Numarası
Avg	Ortalama Paket Boyutu (bayt)
Max	Maksimum Paket Boyutu (bayt)
Min	Minimum Paket Boyutu (bayt)
~~Var~~	~~Paket Boyutu Varyansı~~

4.4.3. Performans Karşılaştırması

Çizelge 4.3 mevcut ML algoritmalarından sınıflandırma tabanlı algoritmaların performansını *Doğruluk* (Sütun 4), *Kappa* (Sütun 6) ve *F-Ölçümü* değerleri (Sütun 7) açısından karşılaştırmaktadır. Ayrıca, karar ağacı tabanlı algoritmalar, *ağaç derinliklerine* (Sütun 5), *yaprak düğümlerinin sayısına* (Sütun 8) ve *ağaç düğümlerinin toplam sayısına* (Sütun 9) göre de karşılaştırılmaktadır.

Çizelge 4.3 Makina Öğrenmesi Algoritmalarının Performans Karşılaştırması

No	Algoritma	Tür	Doğruluk (%)	Derinlik	Kappa	F-Ölçümü	Yaprak Düğüm Sayısı	Toplam Düğüm Sayısı
1	C4.5	Karar Ağacı	97,1875	44	0,9679	0,9720	103	205
2	Simple CART (E-SC)	Karar Ağacı	96,8125	16	0,9636	0,9680	112	223
3	Simple CART (A-SC)	Karar Ağacı	93.9219	18	0.9305	0.9390	223	445
4	BFTree	Karar Ağacı	96,7656	17	0,9630	0,9680	127	253
5	k-NN	Mesafe Tabanlı	96,0781	NA	0,9552	0,9610	NA	NA
6	BayesNET	İstatiksel	95,1094	NA	0,9441	0,9510	NA	NA
7	Naive Bayes	İstatiksel	94,6875	NA	0,9393	0,9470	NA	NA
8	RIPPER	Kural Tabanlı	94,9375	NA	0,9421	0,9490	NA	NA
9	YSA	Fonksiyon Tabanlı	75,0469	NA	0,7148	0,7570	NA	NA

Sonuçlar karar ağacı tabanlı yaklaşımların mevcut tüm algoritmalardan daha iyi performans gösterdiğini kanıtlamaktadır. Karar ağacına dayalı çözümler arasında *C4.5* algoritması en iyi doğruluk sonucunu elde etmektedir. Ancak, Simple CART ağacı *C4.5* ağacından daha dengeli ve derinliği daha düşüktür. 3 numaralı satırda ayrıklaştırma yapılmamış Simple CART (A-SC) algoritması sonuçları gösterilmektedir.

4.4.4. Bellek Kullanımı

Çizelge 4.4, önerilen *E-SC* veri yapısının ayrıntılı bellek kullanımı sonuçlarını vermektedir. Son üç sütun, aynı zamanda, *toplam E-SC, orijinal ayrıklaştırılmış Simple CART (SC)* ve *C4.5* ağaçlarının sonuçlarını göstermektedir. Ayrıca, tablo, her bir ağacın (Satır 2) derinliğini de temsil etmekte ve bu da gecikmeleri ve donanımdaki gerekli olan boru hattı aşama sayısını göstermektedir. Sonuç olarak, önerilen *E-SC* veri yapısı 3.55 $Kbyte$'lık bir bellek kullanımına sahiptir. En büyük gecikme "$6 + 16 = 22$" saat döngüsü

(clock cycle) olmakla birlikte, burada "6" ve "22" sırasıyla *Aşama 1*'deki en büyük FT ağacı derinliği ve *Aşama 2*'deki *SC-B* ağacı derinliği anlamına gelmektedir. Ayrıklaştırılmış *SC* (54.36 $Kbytes$) veri yapısının bellek gereksinimi ile karşılaştırıldığında, önemli ölçüde bellek tasarrufu sağlamaktadır. Ayrıca, önerilen yapıda toplam gecikme (22 döngü), *C4.5* ağacındaki gecikmenin (44 döngü) yarısı kadardır.

Çizelge 4.4 E-Simple CART Veri Yapısının Bellek Gereksinimi

	E-SC							**E-SC**	**SC**	**C4.5**
	Özellik Ağaçları (FT)						SC-B			
	Prtcl	SP	DP	Avg	Max	Min	16	**22**	**16**	**44**
Derinlik	1	6	6	6	6	6				
Düğüm Sayısı	1	59	59	46	62	40	223	**490**	**223**	**205**
Bellek (Bayt)	1	354	354	218.5	295.5	180	2230	**3632**	**55666.1**	**1230**

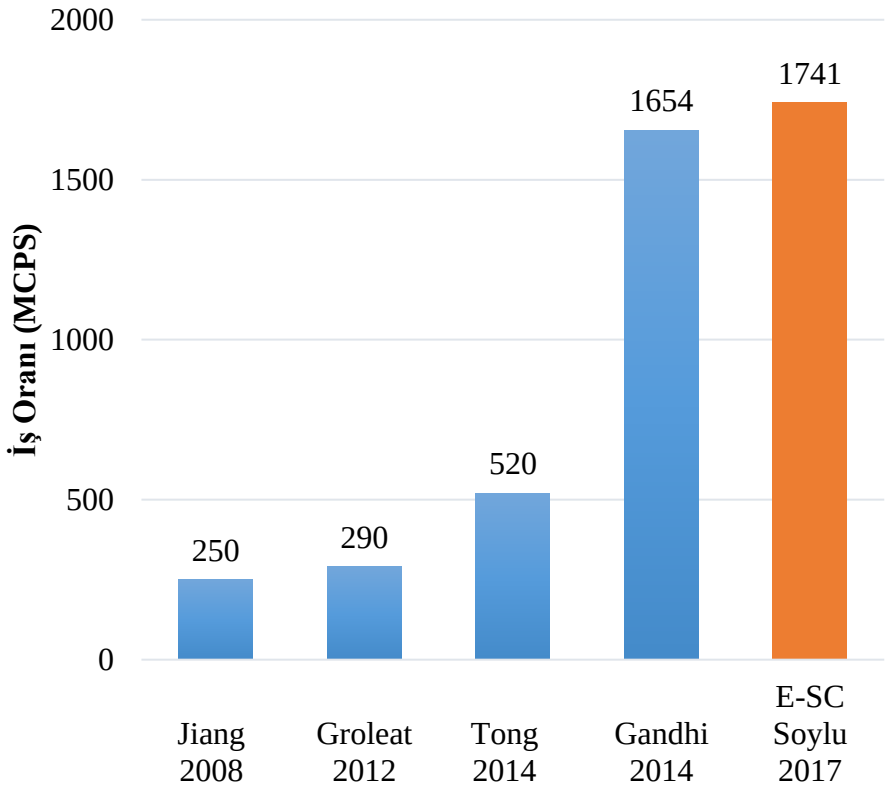
4.4.5. İş Oranı (Throughput)

E-SC tasarımı Xilinx Vivado 2017.2 programı ve Verilog HDL dili kullanılarak, Xilinx Virtex-7 XC7VX485T kartı ile hedef olarak speed-2 hız ile programlanmıştır. FPGA sonuçları *Çizelge 4.5*'te gösterilmektedir. İlk 4 paketi kullanan akışları sınıflandıran *E-SC*, 217 MHz'lik bir saat hızı ile çift portlu bellek kullanarak 557 $Gbps$ veya 1741 $MCPS$'yi desteklemektedir. Bu çalışma boyunca $Gbps$ cinsinden iş oranı, minimum 40 bayt (veya 320 bit) paket boyutuna göre hesaplanmaktadır.

Çizelge 4.5 Uygulama Sonuçları

Saat (ns)	**Frekans** (MHz)	**LUT Sayısı**	**Slice Sayısı**	**BRAM** (36-Kb block)	**BRAM** (18-Kb block)	**Bağlı** IOB
4.60	217	3933 (1.3%)	1351 (1.8%)	130 (12.6%)	6 (0.3%)	201 (33.5%)

Mevcut FPGA tabanlı tasarımlarının iş oranı ($MCPS$) karşılaştırması *Şekil 4.7*'de gösterilmektedir. Tablodaki sonuçlara göre önerilen mimari 1741 MCPS ile en yüksek iş oranını elde etmektedir.



Şekil 4.7 FPGA Tabanlı Tasarımların İş Oranı (MCPS) Karşılaştırması (*E-SC*)

4.5. Sonuç

Bu bölümde, donanım tabanlı internet trafik sınıflandırma için Simple CART makine öğrenmesi algoritması kullanımı önerildi. Algoritmanın doğruluğunu arttırmak için, mevcut çalışmalarda olduğu gibi, ayrıklaştırma ön süreci de kullanıldı. Öte yandan, ayrıklaştırma işlemi, Simple CART ağacının donanım uygulamasında bellek ve kaynak verimsizliğine neden olmaktadır. Bu bölümde, ayrıklaştırılmış Simple CART ağacını temsil etmek için iki aşamalı karma veri yapısı (*E-SC*) ile bu problemler ortadan kaldırıldı. Önerilen veri yapısı, güncel FPGA'ları kullanan donanımlar üzerinde uygulandı. Deneysel sonuçlar, *E-SC* mimarisinin mevcut yaklaşımlara kıyasla doğruluk, gecikme ve iş oranı açısından daha iyi performans gösterdiğini kanıtlamaktadır. Ayrıca, *E-SC* uygulanan veri kümesinin türüne daha az duyarlıdır.

BÖLÜM 5

SIMPLE CART ORMANI (SCF) KULLANARAK FPGA ÜZERİNDE GERÇEK ZAMANLI TRAFİK SINIFLANDIRMA MİMARİSİ TASARIMI

5.1. Giriş

E-SC (*Bölüm 4*) veri yapısı, *Aşama 1*'de özellik ağaçlarından ve *Aşama 2*'de bitmap dönüşümü yapılmış Simple CART ağacından oluşan iki aşamalı bir yapıdır. Karar ağacı tabanlı çözümlerde, ağacın boyutu, ilgili trafik uygulama sınıflarının sayısına bağlıdır. Ayrıca, bir ağacın derinliği ve buna bağlı olarak arama gecikmesi, gelecekte yeni ortaya çıkacak trafik uygulama sınıfları ile artabilir. Gelecekte uygulama sınıfı sayısının artması ile *E-SC* veri yapısının özellikle *Aşama 2*'deki ağaç boyutunun olası artışı ölçeklenebilirlik sorunu ortaya çıkaracaktır. Dolayısıyla *E-SC* veri yapısının artan uygulama sınıfları altında etkin çalışması için alternatif çözümlere ihtiyaç vardır.

E-SC veri yapısının *Aşama 2*'deki ağaç yapısını belirli kurallar çerçevesinde daha küçük ağaç yapılarına parçalayarak (en basit hali ile her uygulama sınıfından bir ağaç elde etmek), ağaç derinliği ve düğüm sayısı bakımından daha etkin bir çözüme dönüştürülebilir. *Basit Sınıflandırma ve Regresyon Ağaçları Ormanı* (Simple Classification and Regression Trees Forest – *SCF*) olarak adlandırılan yeni veri yapısı 2. aşamada, çoklu paralel Simple CART ağaç yapıları ile daha ölçeklenebilir bir çözüm olacaktır. Bu bölümde aşağıdaki özetlenen katkılar detaylıca anlatılacaktır;

- *Aşama 1, Özellik Ağaçları* (Feature Trees - *FTs*)'ndan oluşan ve *Aşama 2*, her uygulama sınıfından bir ağaç elde edilen çoklu *Uygulama Sınıfı Ağaçları* (Application Class Trees – *ACTs*)'ndan oluşan ve *Basit Sınıflandırma ve Regresyon Ağaçları Ormanı* (Simple Classification and Regression Trees Forest – *SCF*) olarak adlandırılan iki aşamalı bir veri yapısı önerilmiştir (*Bölüm 5.2*).
- Her bir uygulama sınıfı için ayrı ağaç içeren *SCF* veri yapısı, bellek ve arama gecikmesi açısından uygulama sınıflarındaki artışa göre ölçeklenebilir bir çözümdür. Gelecekte uygulama sınıflarının sayısındaki olası artış, klasik karar ağacı tabanlı veri yapılarının boyutunu ve derinliğini aşırı dallanma nedeniyle arttıracak ve sınıfları birbirinden ayırmak için çok sayıda düğüm ihtiyacı olacaktır (*Bölüm 5.2.1*).
- *SCF* veri yapısı için önerilen optimizasyon ile, uygulama sınıflarının bazıları gruplandırılarak *SCF* ormanındaki ağaç sayısı azaltılabilir, bu sayede donanımda kaynaklar daha etkin kullanılabilecektir (*Bölüm 5.2.2*).
- 8 uygulama sınıfı içeren Tstat internet trafik eğitim seti ile %96.6719 doğruluk oranı ve 854 $Gbps$ (veya 2669 $MCPS$) (en az 40 baytlık paket boyutu için) iş oranı başarımı sağlayan *SCF* mimarisi FPGA üzerinde gerçeklenmiştir (*Bölüm 5.3*).

5.2.Algoritma Ve Veri Yapısı

Tüm karar ağaçlarında olduğu gibi *E-SC* veri yapısında da, bir ağacın büyüklüğü kullanılan uygulama sınıflarının sayısına göre değişir. Bu durumda, ağacın derinliği ve nihayetinde arama gecikmesi, gelecekte muhtemel çok sayıda uygulama sınıfı ile artacaktır. Bu artışın nedeni, kullanılan algoritmanın sınıfları birbirinden ayırabilmek için aşırı dallanmaya ve çok sayıda düğüme ihtiyacı olmasıdır. Ayrıca, derinlik arttıkça, her ağaç seviyesindeki düğüm sayısı üstel (2^n) olarak artar. Bu nedenle, ağaç yapısı donanıma uygulandığında ihtiyaç duyulan bellek bloklarının miktarı çok büyük olacaktır. Bu durumda, artan bellek boyutları, mimarinin veya donanımın hızını da olumsuz yönde etkileyecektir.

Bu bölümde, çoklu paralel ağaç yapılarından oluşan yeni bir veri yapısı (*SCF*) öneriyoruz. Birbirinden ayrık küçük ağaç yapılarında, yüksek performanslı sınıflandırma için paralel aramalar gerçekleştirilir. Bu yeni tasarım, temel olarak her bir uygulama sınıfı için ayrı bir karar ağacı oluşturmaya dayanmaktadır. *SCF* mimarisi, tek bir *E-SC* mimarisine kıyasla arama süresini daha da azaltmaktadır.

5.2.1. Basit Sınıflandırma ve Regresyon Ağaçları Ormanı (*SCF*)

SCF veri yapısı *Şekil 5.1*'te gösterildiği gibi iki aşamadan oluşur: (i) *Özellik Ağaçları* (Feature Tree – *FTs*) ve (ii) *Uygulama Sınıfı Ağaçları* (Application Class Tree – *ACT*). *FT* ağaçları *Bölüm 0* 'de açıklandığı gibi internet trafiği verilerinin her bir özellik değeri için oluşturulur. Oluşturulan ağaçlar, *DP-FT, SP-FT* vb. gibi o özelliğin adını alır. *FT* ağaçlarının sayısı, özelliklerin sayısına eşittir ve uygulama sınıf sayısından bağımsızdır.

ACT ağaçları her biri iki sınıflı Simple CART karar ağacıdır. *E-SC* de olduğu gibi, düğüm büyüklüğü çeşitliliğini ortadan kaldırmak için bitmap dönüşümü uygulanmıştır. Düğümlerinde sabit büyüklükte bitmapler ile zenginleştirilmiş olan *ACT*, bitmap ile genişletilmiş *ACT* olarak adlandırılmaktadır. Ölçeklenebilirlik problemini çözmek için her uygulama sınıfı bir *ACT* ağacı ile temsil edilerek çoklu *ACT* ağaçları ile bir *ACT* ormanı (*ACT-Forest*) oluşturulmaktadır.

5.2.1.1. ACT Ağaçlarının Organizasyonu

ACT ağaçlarının organizasyonu, her bir uygulama sınıfının ön doğruluk değerlendirmesiyle başlar. Simple CART algoritması (bir ayrıklaştırma işlemi ile) ilk olarak *Algoritma 5.1*'de açıklandığı gibi N-sınıfı olan tüm veri kümesi kullanılarak analiz edilir. Daha sonra, *hata matrisi*nden (confusion matrix) elde edilen en yüksek doğruluğa sahip uygulama sınıfı, en yüksek öncelikli sınıf olarak tanımlanır. Yeni veri seti şimdi iki örnek sınıfı ile oluşturulmaya hazırdır: Birincisi, en yüksek öncelikli (highest priority) sınıf (C_{High}) verisidir; ikincisi, iki kümenin büyüklüğünün eşit olması için "$Others$" olarak adlandırılan ve kalan sınıflardan eşit sayıda uygulama alınması ile oluşturulan veridir. İki kümenin boyutunu eşitlemek için, kalan her bir veri sınıfından aynı miktarda örnek alınmalıdır. Simple CART algoritması bu yeni sete uygulanır ve en yüksek önceliğe

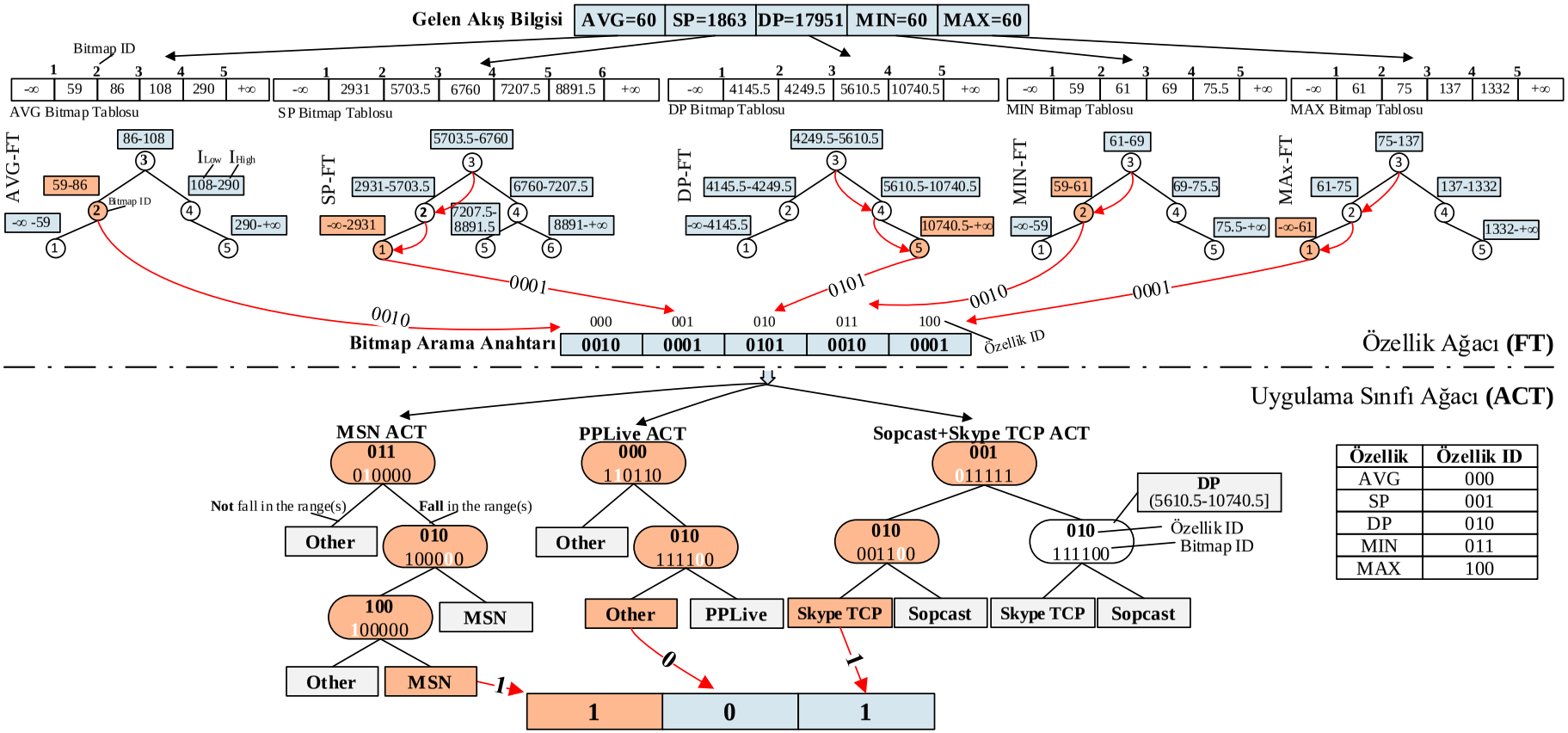
sahip ilk *ACT* (ACT^1) ağacı elde edilir. Sonra, en yüksek öncelikli sınıf örnekleri setten çıkarılır ve böylece sınıf sayısı N'den $N-1$'e indirilir. Bu işlem, *Şekil 5.1*'te gösterildiği gibi $N-1$ farklı *ACT* oluşturuluncaya kadar tekrarlanır. Yapılan işleme *önceliklendirme süreci* adı verilir ve hangi ağacın ilk sırada olacağına karar vermede kullanılır. Ağaçların sırası önemlidir çünkü daha sonra arama sonuçlarını birleştirip nihai sonucu elde etmek için bu sıra kullanılacaktır.

Algoritma 5.1 *SCF* Organizasyon Algoritması

Giriş : Sınıf sayısı N olan ayrıklaştırılmış veri seti (*VS*)
Çıkış : *N-1* sayılı ACT yapısı.

1. Geçici boş kümeleri, C^{High} ve *Others*, tanımla
2. *ACT* indeksini $i = 1$ olarak tanımla
3. **while** $N > 2$ **do**
4. Simple CART algoritmasını *VS* ile test et ve çalıştır ve hata matrisini (*CM*) oluştur
5. *CM* bağlı olarak, *VS*'den doğruluğu en yüksek sınıf verisini C^{high}'ye kopyala
6. Kalan sınıflardan, aynı veri miktarını *Others*'a kopyala, (*Others*) = (C^{High})
7. Simple CART'ı çalıştır ve C^{High} ve *Others*'ın birleştirilmiş verisini kullanarak ACT^i'yi oluştur.
8. $i = i + 1$
9. $N = N - 1$
10. C^{High} ve *Others*'ı *temizle*
11. *VS*'den C^{High}'ı çıkartarak *VS*'yi güncelle
12. **end while**
13. *VS* ile Simple CART çalıştır ve ACT^{N-1} oluştur
14. **return** *ACT* Ormanı

Örnek Bir Önceliklendirme Süreci; Dört uygulama sınıfı (MSN, PPLive, Skype TCP ve Sopcast) içeren bir örnek set *Çizelge 5.1*'de gösterilmektedir. İlk iterasyonda tüm uygulama sınıfları Simple CART algoritması ile analiz edilir ve en az sayıda hataya sahip uygulama sınıfı ("*" ile işaretlenen en yüksek doğruluk oranına karşılık gelen) MSN uygulama sınıfı olarak gözlenir. İkinci iterasyonda, verilerden MSN uygulama sınıfı kaldırılır ve kalan 3 uygulama sınıfı aynı algoritma ile yeniden analiz edilir. Bu kez PPLive uygulama sınıfı en az sayıda hataya sahiptir. Üçüncü iterasyonda PPLive



Özellik	Özellik ID
AVG	000
SP	001
DP	010
MIN	011
MAX	100

Şekil 5.1 SCF Veri Yapısı

uygulama sınıfı setten çıkarılır ve kalan 2 uygulama sınıfı test edilir ve Sopcast uygulama sınıfının en az hataya sahip olduğu gözlemlenir. Sonuç olarak, tüm uygulama sınıfları artan hata sayısına göre sıralanırsa, MSN, PPLive, Sopcast ve Skype TCP şeklinde olacaktır. Dolayısıyla ilk sırada *MSN ACT* ağacı yer alacaktır ve diğer ACT ağaçları da belirtilen sırada yer alacaktır.

Çizelge 5.1 Uygulama Sınıfı Analizi (SCF)

Uygulama Sınıfı	**Örnek**	**Hata Sayısı**		
		İterasyon-1	İterasyon-2	İterasyon-3
MSN	600	3*	–	–
Sopcast	600	8	4	2*
Skype TCP	600	10	7	4
PPLive	600	7	2*	–

Örnek Bir Veri Seti Düzenleme Süreci; Örnek bir veri seti düzenlemesi süreci *Çizelge 5.2*'de verilmiştir. ACT^1 ağacı oluşturmak için, uygulama sınıfına ait en az sayıda hata oluşturan 600 örnek, sınıf adı (*MSN*) ile işaretlenir ve etiketlenir. Aynı sayıya (600) sahip ikinci sınıf (*Others*), kalan 3 uygulama sınıfından (her uygulama sınıfından 200 olmak üzere toplam 600 örnek) rasgele seçim ile derlenir. Amaç, aşırı uyumu önlemek için sınıflandırıcıyı her sınıftaki aynı miktardaki örneklerle eğitmektir. Bu yeni setten yeni bir Simple CART karar ağacı elde edilmektedir. Ortaya çıkan ilk ağaç iki sınıflı (MSN ve Others) ACT^1(*MSN-ACT*) (İterasyon-1) olarak adlandırılır ve veri setinden MSN uygulama sınıfı örnekleri çıkartılarak veri seti güncellenir.

Çizelge 5.2 ACT Ağacı Eğitim Seti Boyutları

İterasyon – 1				**Ağaç Adı**
Etiket	**U. Sınıfı**	**Örnek**	**Toplam**	
MSN	MSN	600	600	MSN-ACT
Others	Skype TCP	200	600	
	PPLive	200		
	Sopcast	200		
İterasyon – 2				
PPLive	PPLive	600	600	PPLive-ACT
Others	Skype TCP	300	600	
	Sopcast	300		
İterasyon – 3				
Skype TCP	Skype TCP	600	600	Skype TCP-Sopcast ACT
Sopcast	Sopcast	600	600	

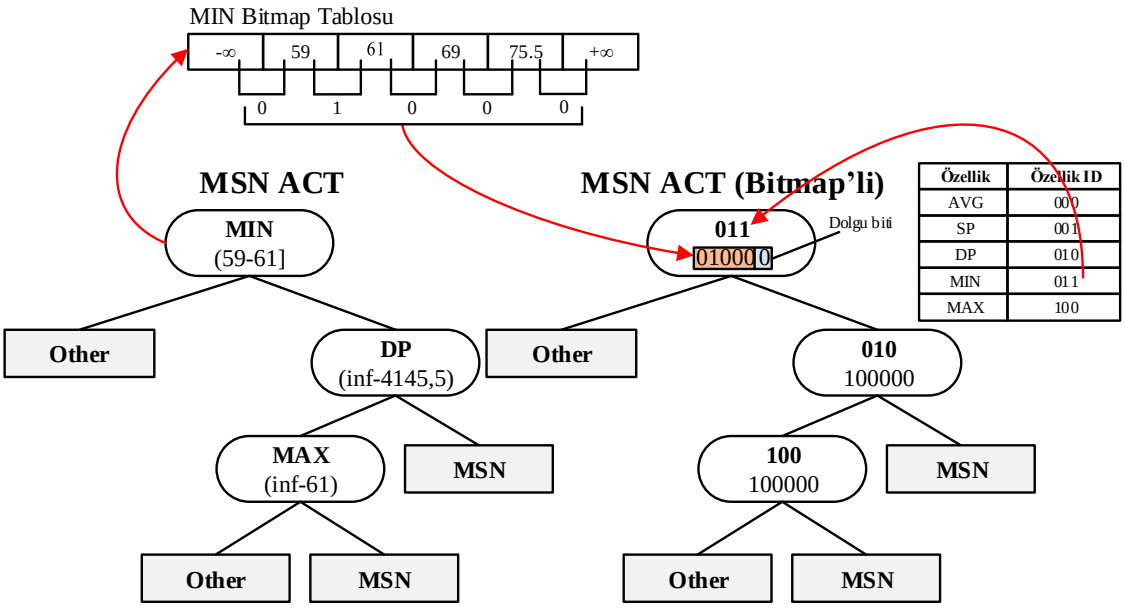
ACT^2 ağacı oluşturmak için, tüm sınıflar arasında en yüksek doğruluk oranına sahip sınıfın (PPLive) 600 örneği güncellenmiş veri kümesinden alınır (MSN önceden kaldırılmıştı). Daha sonra, toplam 600 örneğe sahip kalan diğer uygulama sınıflarından (Skype TCP ve Sopcast) 300 örnek rasgele olarak seçilir (Others) (İterasyon-2). Derlenen veri kümesi daha sonra ACT^2 (PPLive ACT) ağacı oluşturmak için kullanılır ve sonraki adım için PPLive uygulama sınıfı örneklerini veri setinden çıkararak veri seti güncellenir. Üçüncü iterasyonda bir başka sınıf oluşturmaya gerek yoktur, çünkü sadece iki uygulama sınıfı kalmıştır. Bu adımda, her sınıftan (Skype TCP ve Sopcast) 600 örnek alınır ve ACT^3 (Sopcast-Skype TCP ACT) (İterasyon-3) ağacı oluşturulur.

5.2.1.2. Özellik Ağaçları (FTs)

Özellik Ağaçları (Feature Tree – *FTs*), (*Bölüm 4.2.3.1*'de detaylıca anlatıldığı gibi) farklı özellik değerlerinden oluşturulan aralık ağacıdır. Ağaç isimlendirmeleri her özellik için, *DP-FT, SP-FT* vb. gibi "*Özellik Adı-FT*" olarak adlandırılır.

5.2.1.3. Bitmap ile Genişletilmiş ACT

Her bir *ACT* ağacı, *Algoritma 5.1*'de açıklandığı gibi oluşturulur. Sabit büyüklükteki bit dizileri ile çoklu aralıkları temsil etmek için, *Şekil 5.2'*te gösterildiği gibi bitmap dönüşümü gerçekleştirilir (*Bölüm 3.4.1*'de detaylı bir şekilde anlatılmıştı). *Şekil 5.2'*te sadece MIN *bitmap tablosu* gösterilmektedir. Bir bitmapin uzunluğu, en uzun bitmap tablosuyla tanımlanır (*Şekil 5.1*'teki 6 uzunluğuna sahip SP *bitmap tablosu*). Dolgu bit(ler)inin kısa bitmapleri belirlenen sabit bir genişliğe uzatmak için kullanılır. Örneğimizde (*Şekil 5.2*), MIN özelliğine ait bitmap tablosu boyutunu 6'ya uzatmak için tek bir "0" dolgu biti eklenmiştir. Her bir *ACT* ağaç düğümü, *Şekil 5.1*'te gösterildiği gibi, o düğümde depolanan bir özelliğin bir tanımlayıcısı olan bir *özellik kimliği*ni (F_{ID}) de depolar.



Şekil 5.2 Bitmap ile Genişletilmiş MSN ACT

5.2.1.4. *SCF*'de Arama Süreci

Gelen trafik başlangıçta *FT* ağaçlarında paralel olarak aranır ve her ağaç iki sonuç üretir: (i) *Özellik Kimliği* (F_{ID}) ve (ii) *Bitmap Kimliği* (B_{ID}). Bu değerler *bitmap arama anahtarı*nda birleştirilir (*Şekil 5.1*'teki örneğimizde F_{ID} değeri "000" olan *AVG-FT* çıktısı "0010" olan aramanın bitmap değeri "0010 0 001 0101 0010 0001") ve bu değer bir ara belleğe (register) kaydedilir. İkinci aşamada, *bitmap arama anahtarı*, paralel olarak *bitmap ile genişletilmiş ACT* ağaçlarında aranmaktadır. Buradaki her bir *ACT*'de gerçekleşen arama işlemi daha önce *E-SC* de anlatılan arama ile aynıdır. Farklı olarak, her bir ACT den gelen sonuçlar değerlendirilirken, nihai sonuç en sola dayalı (önceliği en yüksek) *ACT* ağacının sonucu olarak alınır.

Her bir *ACT* ağacında, gerçekleşen arama işlemi aşağıdaki örnekte açıklanmıştır; Gelen trafik akış bilgisinin *Şekil 5.1*'te gibi "$MAX = 60$", "$DP = 17951$", "$AVG = 60$", "$MIN = 60$" ve "$SP = 1863$" olarak verildiğini varsayalım. Öncelikle *FT* ağaçlarına paralel olarak giren arama her özellik için kendi *FT* ağacında aranmaktadır (MIN değeri MIN-FT ağacında gibi). *Bitmap arama anahtarı* "21521" (veya ikili olarak "0010 0001 0101 0010 0001"), *Şekil 5.1*'te gösterildiği gibi *FT* ağaçlarında yapılan arama sonucu olarak elde edilmektedir. Arama, *ACT* ağaçlarının kök düğümünden

başlamaktadır. Örneğin, *MSN-ACT* ağacı kök düğümündeki F_{ID} = "011" değeri, *bitmap arama anahtarı*nın ("0010") dördüncü yuvasını gösterir. Bir bitmap değerine ("0**1**0000") karşılık gelen bit konumu "1" olduğundan (örneğin bir bit vektörde "0001" birinci bit'e karşılık gelir), arama sağ düğüme iletilir. Bir sonraki düğümde F_{ID} = "010" değeri üçüncü yuvayı işaret eder ("0101"). Bitmap değerinin beşinci biti ("1000**0**0") "0" olduğundan, arama sol alt düğüme yönlendirilir. Arama, yaprak düğümüne ulaşılana kadar aynı şekilde işlemler devam etmektedir. *Şekil 5.1'*te, verilen arama örneğinin arama yolları gösterilmektedir. Sonuçta, MSN gelen akışın *Sınıf Kimliği* (Class ID – C_{ID}) olarak bulunur. Benzer şekilde, aramalar diğer tüm *ACT* ağaçlarında paralel olarak aynı zamanda gerçekleşir. En son arama çıktısı (MSN, PPLive, Skype TCP + Sopcast) = ("**1**01") olacaktır. Karar verme kuralına göre, en soldaki öncelikli bit değerine bağlı olarak çıktı, doğru bir sınıflandırma olan MSN uygulama sınıfı olacaktır.

5.2.2. Optimizasyon

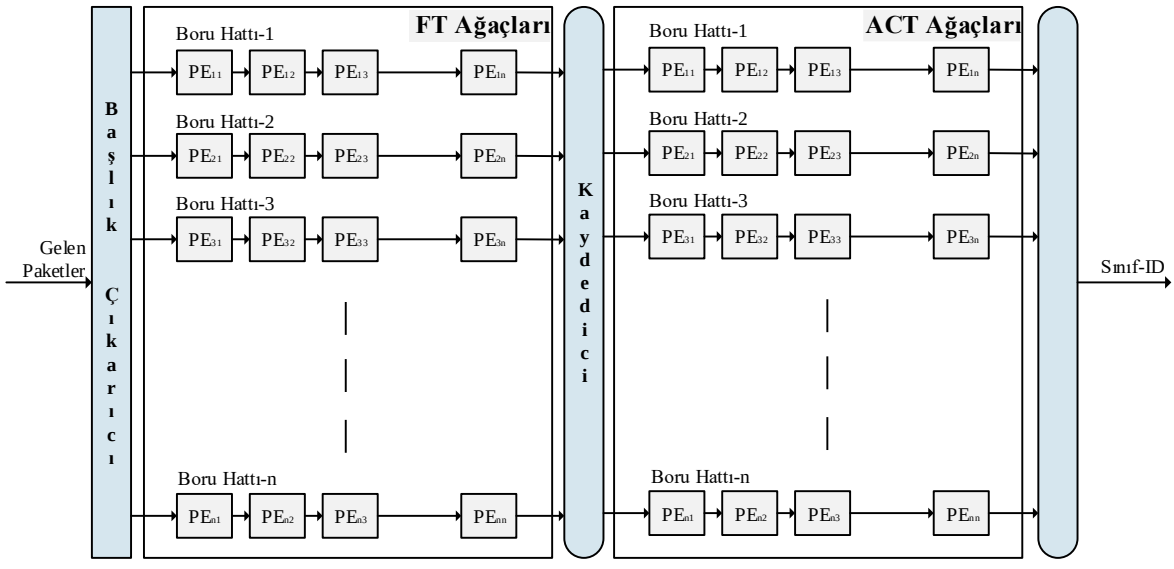
SCF veri yapısında, *ACT* ağaç sayısı, sınıf sayısına paralel olarak artar. Bu durum yapının uygulama sınıfına göre ölçeklenebilirliği ile tezat oluşturmaktadır. Ancak sınıfların sayısı arttıkça, yeni sınıf için yeni bir *ACT* ağacı oluşturmak yerine, birkaç sınıfı tek bir *ACT* ağacında birleştirmek mümkündür. Bunu yapmanın bir yolu, sınıf sayısının pozitif tam bölenleri olan ağaçlar yaratmaktır. Örneğin, bu çalışmada 8 sınıf vardır ve 8 sayısının pozitif tam bölenleri $1, 2, 4$ ve 8'dir. Yani, oluşturulan *ACT* ağaçları $1, 2, 4$ veya 8 uygulama sınıfı içerecektir. 8 uygulama sınıfı içeren ağacın, *Bölüm 0'*deki (Soylu, vd., 2017) tasarıma karşılık gelen herhangi bir parçalanma ($n = 8$) içermeyen orjinal *E-SC* ağacı olduğu açıktır. Sadece bir uygulama sınıfı içeren ağaç yapısı ($n = 1$) *Bölüm 5.2.1'*de açıklanmıştır. $n = 2$ veya $n = 4$ uygulama sınıfı içeren bir veri yapısı oluşturmak, tek sınıflı bir ağaç oluşturma sürecine benzer. Tek fark, en yüksek doğrulukta 1 uygulama sınıfı yerine 2 veya 4 uygulama sınıfı kullanmaktır. Aynı şekilde ağaç oluşturulurken elde edilen model de 1 yerine 2 veya 4 uygulama sınıfı kullanılacaktır. Öte yandan, *ACT* ağacındaki uygulama sınıflarının sayısı arttıkça, ağacın derinlik değeri de orantılı olarak artacaktır. Bu nedenle, her zaman *ACT* ağaç sayısı ve ağaçların derinlikleri arasında bir ilişki vardır. Benzer şekilde, bir ağaçtaki sınıfların sayısı arttıkça o *ACT* ağaçlarındaki bellek gereksinimi de artar. Bu aslında, birbirinden sınıfları ayırabilmek için düğüm sayısındaki artış ihtiyacından kaynaklanmaktadır. Arama hızı, donanım haritalama

organizasyonu ile yakından ilgilidir. Boru hatlarının sayısı ve dolayısıyla lojik kaynak gereksiniminin düşmesine rağmen, boru hattı aşamalarında kullanılan belleklerin büyüklüğü ve tiplerine bağlı olarak çıkan iş oranı artabilir.

Optimizasyon sonucu oluşturulan 2 veya 4 uygulama sınıfı içeren ağaçların performans değerlendirmeleri *Bölüm 5.4*'de ayrıntılı şekilde verilmektedir.

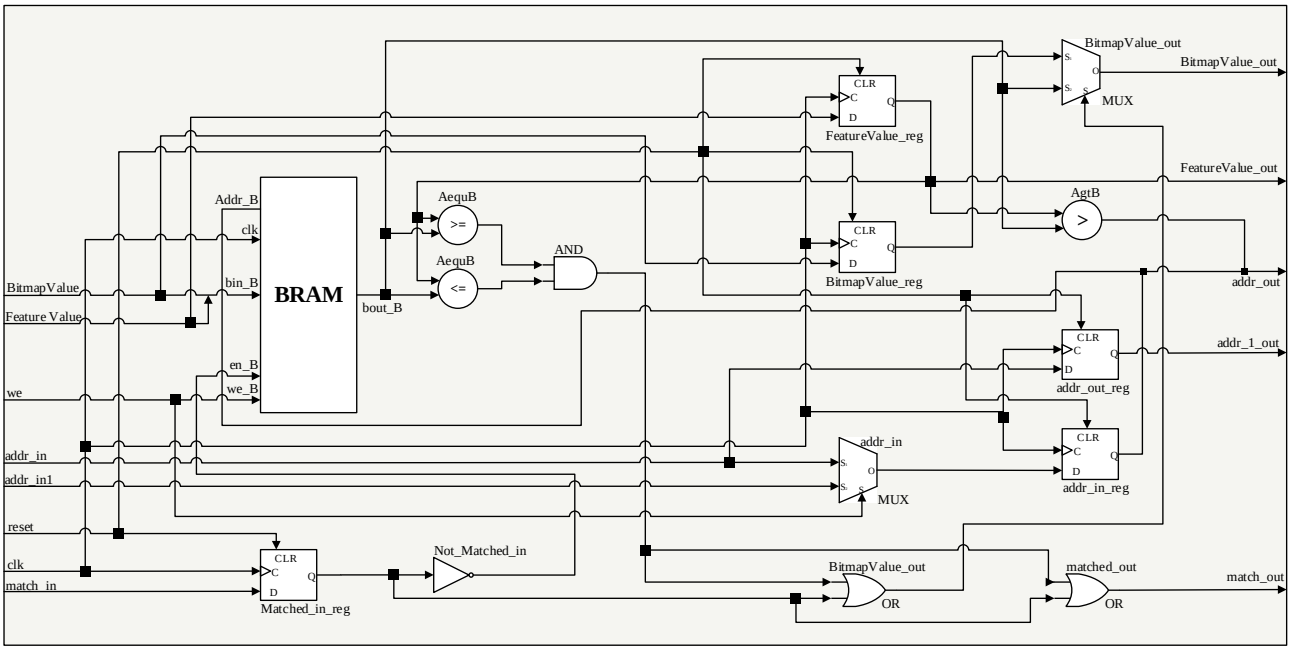
5.3. SCF Mimarisi Ve FPGA Uygulaması

SCF veri yapısı için önerilen mimari *Şekil 5.3*'te gösterilmektedir. Yüksek iş oranı elde etmek için FPGA'lar üzerinde bir boru hattı (pipeline) kullanılır ve uygulanır. *FT* ve *ACT* boru hattı yapıları sırasıyla FT ve ACT ağaçlarını depolamaktadır. Bir ağacın her bir seviyesi boru hattında bir aşamaya karşılık gelir. Toplam *FT* boru hattı sayısı, kullanılan özelliklerin sayısına eşittir. Bir *FT* ağacının derinliği, o ağaca karşı gelen bitmap tablosuna bağlıdır. Ancak, arama işlemleri paralel olarak gerçekleştirildiği için, sonuçları eşzamanlı olarak elde etmek amacıyla, kısa boru hatlarının sonunda ek boş aşamalar eklenebilir. *ACT* ağaçlarının sayısı, sınıfların sayısına ve bu sınıfların birleştirilip birleştirilmediğine bağlıdır. *ACT* ağacının derinliği, ML algoritmasından elde edilen sonuçlara bağlıdır ve uygulama sınıflarının kombinasyonuna bağlı olarak da değişebilir.



Şekil 5.3 SCF Boru Hattı (Pipeline) Mimarisi

Tek bir boru hattı aşaması (İşleme Elemanı (Processing Element – *PE*)), *Şekil 5.4*'da gösterildiği gibi bir FPGA üzerinde Blok RAM (BRAM) ve bir *haritalama modülü* içerir. BRAM, FPGA'larda ağaç düğümlerini depolamak için kullanılır. Haritalama modülü, aralık karşılaştırmalarını veya bit kontrol işlemlerini uygulayan ve bir sonraki aşama adresini hazırlayan devredir. Arama, kök düğümü depolayan ilk boru hattı aşamasından başlar. Arama son aşamaya ulaşmadan önce biterse, en yeni arama sonucu, herhangi bir BRAM erişimi olmaksızın ve bir boru hattının kalan kısmı boyunca eşleştirme işlemi gerçekleştirmeden tüm aşamaları geçer. Güncel FPGA'ların BRAM'leri çift okuma/yazma bağlantı noktalarına sahip olduğundan, her bir arama döngüsünde aynı anda iki arama işlenebilmektedir. *SCF* mimarisi böylece iki kat daha fazla iş oranı (throughput) sağlayan çift doğrusal boru hatları olarak yapılandırılmıştır.



Şekil 5.4 *SCF* Tek Bir PE Dizaynı

5.4. Performans Değerlendirmesi

5.4.1. Deney Düzeneği

Bu bölümde kullandığımız veri seti *Bölüm 0*'teki *E-SC* veri yapısının veri seti ile aynıdır.

5.4.2. Özellik Setinin Yapısı

Aday özellik seti Bölüm 0'teki *E-SC* mimarisindeki aday özellik seti ile aynıdır.

5.4.3. Performans Karşılaştırması

Çizelge 5.3, mevcut sınıflandırma tabanlı ML algoritmalarının performansını, bir makine öğrenme aracı olan WEKA (Hall, vd., 2009) kullanılarak elde edilen *Doğruluk* (Sütun 4), *Kappa* (Sütun 6) ve *F-Ölçümü* değerleri (Sütun 7) açısından karşılaştırmaktadır. Ayrıca, karar ağacı tabanlı algoritmalar, *ağaç derinliklerine* (Sütun 5), *yaprak düğümlerinin sayısına* (Sütun 8) ve *ağaç düğümlerinin toplam sayısına* (Sütun 9) göre de karşılaştırılmaktadır.

Çizelge 5.3 Makine Öğrenmesi Algoritmalarının Performans Karşılaştırması

No	Algoritma	Tür	Toplam Doğruluk (%)	Derinlik	Kappa Değeri	F-Ölçümü	Yaprak Düğüm Sayısı	Toplam Düğüm Sayısı
1	SCF (n=1)	Karar Ağacı	94.0625	15	0.9320	NA	52	97
2	SCF^{Opt} (n=2)	Karar Ağacı	96.6719	15	0.9620	NA	51	98
3	SCF^{Opt} (n=4)	Karar Ağacı	95.0781	18	0.9440	NA	71	140
4	Simple CART (E-SC)	Karar Ağacı	96.8125	16	0.9636	0.9680	112	223
5	Simple CART (A-SC)	Karar Ağacı	93.9219	18	0.9305	0.9390	223	445
6	C4.5	Karar Ağacı	97.1875	44	0.9679	0.9720	103	205
7	BFTree	Karar Ağacı	96.7656	17	0.9630	0.9680	127	253
8	k-NN	Mesafe Tabanlı	96.0781	NA	0.9552	0.9610	NA	NA
9	BayesNET	İstatiksel	95.1094	NA	0.9441	0.9510	NA	NA
10	Naive Bayes	İstatiksel	94.6875	NA	0.9393	0.9470	NA	NA
11	RIPPER	Kural Tabanlı	94.9375	NA	0.9421	0.9490	NA	NA
12	YSA	Fonksiyon Tabanlı	75.0469	NA	0.7148	0.7570	NA	NA

Sonuçlar, karar ağacı tabanlı yaklaşımların mevcut tüm algoritmalardan daha iyi olduğunu ve *C4.5* algoritmasının diğer ağaçlara kıyasla en iyi doğruluk sonucunu elde ettiğini kanıtlamaktadır. Ancak, bu mimarileri gerçek dünya senaryolarına uyarlarken, yalnızca girdi verilerinden elde edilen doğruluklara dayanarak karar vermek, yanıltıcı olsa bile, literatürdeki farklı sistemlerin performanslarını karşılaştırmak için doğruluk değerleri en sık kullanılan metrik olarak değerlendirilmektedir. Elde edilen doğruluk değerleri, verilerin yapısına (yani, özellik ve uygulama sınıflarının çeşitliliğine) özeldir ve sınıf etiketlerinin sayısındaki bir artışla büyük ölçüde değişebilir. Bunun yerine, algoritma performans analizinden elde edilen *Kappa* değeri, önerilen mimarinin performansını ölçmek için kullanılabilir. (Kappa skoru her zaman -1 ile $+1$ arasındadır (Landis ve Koch, 1977). Kappa değeri $+1$'e yaklaştıkça, algoritmanın ve veri yapısının gerçek uygulamaya yaklaştığını gösterir). Önerilen mimarilerin yüksek olan Kappa skorları açısından önemli farklılıklar göstermediği tabloda görülmektedir.

5.4.4. Bellek Kullanımı

SCF veri yapısı, *Aşama 1*'deki çoklu *FT* ağaçlarını ve *Aşama 2*'deki *ACT* ağaçlarını içermektedir. Her *FT* ağaç düğümü, boyutları her bir özellik için farklı olan I_{low} ve I_{high} alanlarını taşır. Örneğin, DP ve SP özellik ağaçları için, sınırların minimum ve maksimum değerleri sırasıyla "0" ve "65536"′dır. Bu durumda, alt ve üst sınırları saklamak için "16 + 16 = 32" bit gerekir. Sol ve sağ çocuk işaretçisinin büyüklüğü, düğüm sayısı bakımından en kalabalık seviyeye göre hesaplanmaktadır. 6 seviyeli *FT* ağacı için, muhtemelen en kalabalık olanı, 32 kadar olabilir ve bu da en fazla $log32 =$ "5" bit işaretçiye ihtiyaç duymaktadır. Son olarak, *Bitmap kimliği*nin boyutu, en geniş aralık tablosundaki ayrık aralıkların sayısına bağlıdır. Tabloda en büyük 62 giriş varsa, Simple CART ağacındaki 62 bit uzunluğundaki bitmapi adreslemek için "6" bit yeterlidir. Diğer yandan, her bir *ACT* ağaç düğümleri F_{ID} ve *bitmap tablolarını* depolamaktadır. Önerilen veri yapısında 6 özellik kullanılmaktadır ve her özelliği benzersiz bir şekilde tanımlamak için "3" bit yeterlidir. Bir bitmap tablosu boyutu, daha önce belirtildiği gibi en geniş aralık değerine göre belirlenir. Her bir ağaç düğümü, yalnızca 8 uygulama sınıfı içerdiği için boyutu "3" bit değerinde bir *Sınıf Kimliği* tutar. Son olarak, sol ve sağ çocuk işaretçi boyutları *FT* cinsinden hesaplanır.

Çizelge 5.4 SCF Veri Yapasının Bellek Gereksinimi (*n=1*)

	Özellik Ağaçları (FTs)						
	Prtcl	SP	DP	Avg	Max	Min	
Derinlik	1	6	6	6	6	6	
Düğüm Sayısı	1	59	59	46	62	40	
Bellek (Bayt)	1	354	354	218.5	295.5	180	
	Uygulama Sınıfı Ağaçları (ACTs)						
	MSN	Yahoo	S.UDP	S.TCP	Joost	TVAnts	PPLive+Sopcast
Derinlik	6	9	6	4	4	4	4
Düğüm Sayısı	15	27	19	7	9	11	9
Bellek (Bayt)	135	243	171	63	81	99	81

Çizelge 5.4, önerilen *SCF* veri yapısının ayrıntılı bellek sonuçlarını göstermektedir. Ayrıca, çizelge, her bir *FT* ve *ACT* ağaçlarının (Satır 2) derinliğini göstermektedir. Ek olarak, *Çizelge 5.5* önerilen mimarilerin performansını karşılaştırmaktadır. Sonuçlara göre, 2.22 Kbayt'lık toplam bellek gereksinimine sahip *SCF* yapısı, orjinal (*SC*) ve *E-SC* veri yapısına kıyasla önemli ölçüde bellek tasarrufu sağlamaktadır. Orjinal olarak ayrıklaştırılmış Simple CART (*SC*) veri yapısının, diğerlerine göre en az sayıda düğüm içermesine rağmen, düğüm büyüklüğünün, sabit boyutlu bitmapler yerine her düğümde depolanan farklı sayıdaki aralıklar nedeniyle daha büyük olduğu bilinmektedir. Ayrıca, düğüm boyutu çeşitliliği, *SC* veri yapısının doğrudan donanım uygulamasını pratik olmaktan çıkarır. *Çizelge 5.5*de görüldüğü üzere *SCF*'de toplam gecikme (en büyük *FT* ve *ACT* toplamı) $(6 + 9 = 15$ saat döngüsü) *E-SC* ağacından (22 saat döngüsü) daha azdır.

Çizelge 5.5 Simple CART Yapılarının Bellek Kullanımı

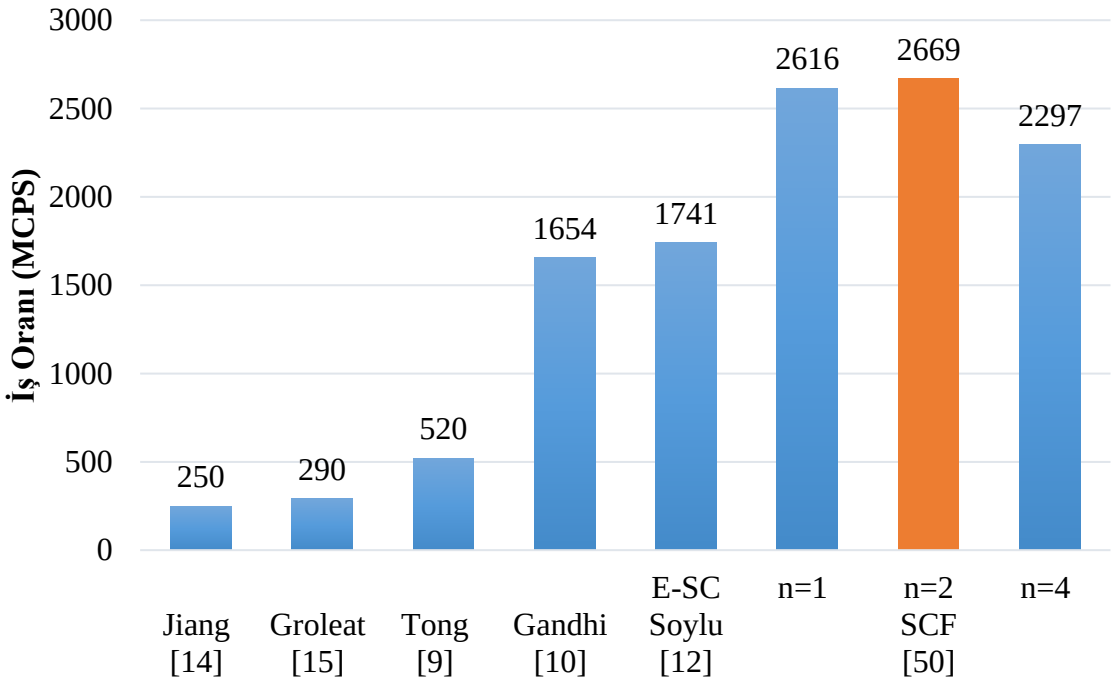
Mimari	**Derinlik**	**Düğüm Sayısı (Aşama 1)**	**Düğüm Sayısı (Aşama 2)**	**Düğüm Sayısı (Toplam)**	**Bellek (Bayt)**
SCF (n=1)	15	267	97	364	2275.0
SCF^{Opt} (n=2)	15	267	98	365	2284.0
SCF^{Opt} (n=4)	18	267	140	407	2767.0
E-SC	22	267	223	490	3632.0
SC	16	NA	NA	223	55666.1

5.4.5. Çıkan İş Oranı

Önerilen *SCF* tasarımı Xilinx Vivado 2017.3 programı kullanılarak Verilog donanım dili ile Xilinx Virtex-7 XC7VX485T aygıtı üzerinde hedef olarak -2 hız derecesi kullanılarak simüle edilmiştir. FPGA kaynak kullanım sonuçları *Çizelge 5.6*'de sunulmaktadır. $Gbps$'deki sonuçlar, minimum 40 bayt (veya 320 bit) paket boyutuna göre hesaplanmıştır. Çizelge ayrıca, bölümde önerilerin optimize mimarilerin iş oranı değerlerini *E-SC* mimarisi ile karşılaştırmaktadır. Sonuçlara göre optimize edilmiş *SCF* mimarisinin ($n = 2$) diğer tüm mimarilerden daha iyi performans gösterdiğini ve 333.3 MHz'lik bir saat hızına ulaştığını ve saniyede 854 $Gbps$ veya 2669 milyon sınıflandırmayı ($MCPS$) desteklediğini göstermektedir. Ayrıca, *Şekil 5.5 SCF* tasarımının mevcut FPGA tasarımlarının en iyi iş oranına sahip olduğunu göstermektedir.

Çizelge 5.6 Uygulama Sonuçları (SCF ve E-SC)

Mimari	**İş Oranı** (Gbps)	**İş Oranı** (MCPS)	**Süre** (ns)	**Frekans** (MHz)	**LUT** Sayısı	**Slice** Sayısı	**BRAM** (36-Kb)	**BRAM** (18-Kb)	**Bağlı** IOB
SCF (n=1)	837	2616	3.06	322.5	10106(3.3%)	2894(3.8%)	0(0.0%)	0(0.0%)	139(23.2%)
SCF^{Opt} (n=2)	854	2669	3.00	333.3	6560(2.2%)	1812(2.4%)	0(0.0%)	0(0.0%)	130(21.7%)
SCF^{Opt} (n=4)	735	2297	3.48	285.7	5615(1.8%)	1650(2.2%)	4(0.4%)	4(0.2%)	130(21.7%)
E-SC	557	1741	4.60	217.0	3933(1.3%)	1351(1.8%)	130(12.6%)	6(0.3%)	201(33.5%)



Şekil 5.5 FPGA Tabanlı Tasarımların İş Oranı (MCPS) Karşılaştırması (SCF)

5.5. Sonuç

Bu bölümde, yüksek performanslı trafik sınıflandırması için iki aşamalı *SCF* veri yapısı önerilmiş ve FPGA'lar tarafından sağlanan bol paralellik avantajları kullanılmıştır. Klasik tek yapılı karar ağaçlarında, büyüklük ve dolayısıyla ağaçların yüksekliği, aşırı dallanma nedeniyle sınıf sayısındaki artışla birlikte artar ve sınıfları birbirinden ayırmak için çok sayıda düğüm gerekli hale gelir. Klasik ağaç yapılarından ayrı olarak, *SCF* mimarisinde yeni eklenen bir uygulama sınıfı sadece kendi *ACT* karar ağacını üretir ve diğer *ACT* ağaçlarının boyutunu etkilemez. *SCF* mimarisinde, arama gecikme performansı, tek bir büyük karar ağacından ziyade en büyük *FT* ve *ACT* ağaç yapılarına bağlıdır. Ayrıca, paralel ve dağıtılmış yapı, *SCF* mimarisinin modüler ve uygulanan veri setinin tipine daha az duyarlı olmasını sağlar. Önerilen tasarım, birden çok uygulama sınıfını tek bir yapıda birleştirerek donanım kaynaklarının etkin kullanımı sağlanabilir. Deneysel sonuçlar, *SCF* veri yapısının boru hattı üzerinde kolaylıkla eşleştirildiğini ve tüm mevcut yaklaşımları çıkan iş oranı, arama gecikmesi ve bellek gereksinimi açısından geride bıraktığını kanıtlamaktadır.

BÖLÜM 6

BİT VEKTÖR KODLU SIMPLE CART (BC-SC) YAPISI İLE DÜŞÜK GECİKMELİ TRAFİK SINIFLANDIRMA MİMARİSİ TASARIMI

6.1. Giriş

ML karar ağacı tabanlı yapılarda ağaç derinliği, sınıflandırma yapılacak trafik uygulama sayısı ile ilişkilidir. Uygulama sınıf sayısı arttıkça, bu sınıfları ayırabilmek için algoritmalar ağaçta daha çok dallanma ile ağaç derinliğini arttıracaktır. Ağaç derinliğinin artması doğrudan arama gecikmesinin artmasına neden olur. Aynı zamanda sınıflandırıcının hızı veya çıkan iş oranı da olumsuz etkilenir. Gelecekte yeni uygulama sınıfları ortaya çıkması tasarlanacak olan yeni mimarilerin bu artan sınıf sayısına performansını koruyarak cevap vermesi beklenir.

Bu bölümde Simple CART karar ağacını bit vektörlere dönüştürerek ağaç yapısını ortadan kaldıran; dolayısıyla uygulama sınıfı sayısı artışından doğrudan etkilenmeyen; ağaç yapısında dönüşüm yaparken sınıflandırma doğruluk oranını koruyan; tek adımda sınıflandırma yapabilen *Bit Vektör Kodlu Simple CART* (Bit Vector-Coded Simple Cart (*BC-SC*)) yapısı önerilmektedir. Bu bölümde sunulan katkılar özetle:

- Her özellik için bir *Özellik-Set* ve özellik setteki aralık değerlerini bit vektörlere dönüştürülen *Bit Vektör Kodlu Simple CART* (Bit Vector – Coded Simple CART (*BC-SC*)) olarak adlandırılan yeni bir veri yapısı önerilmiştir. Yüksek sınıflandırma iş oranı elde etmek için bütün özellik setler ve özellik setlerdeki

aralıklar paralel arama işlemi gerçekleştirecek şekilde FPGA donanım mimarisi tasarlanmıştır (*Bölüm 6.2*).

- *BC-SC* veri yapısı (i) uygun olan aralık değerleri birleştirilerek aralık değer sayısı azaltılması ve (ii) uygun uygulama sınıfları birleştirilerek bit vektör uzunluğunu azaltılması gibi iyileştirmeler ile performansı arttırılabilir. Bu sayede donanım bakımından kaynak tasarrufu sağlanır ve sınıflandırıcının üretmiş olduğu iş oranı miktarı da arttırılabilir (*Bölüm 6.2.3*).
- 8 uygulama sınıfı içeren trafik verisi ile %96,8125 doğruluk ve 914 $Gbps$ (2857 $MCPS$) (en az 40 baytlık paket boyutu için) iş oranı başarımı elde edebilen FPGA platformuna uygun donanım mimarisi tasarlanmıştır (*Bölüm 6.3*).

6.2. Algoritma Ve Veri Yapısı

Bu bölümde, ölçeklenebilirlik sorununu gidermek ve doğruluk oranı değerini korumak için tek bir adımda sınıflandırma yapabilen yeni bir veri yapısı öneriyoruz. Bu tek adımlı *Bit Vektör Kodlu Simple CART* (Bit Vector Coded Simple CART (*BC-SC)*) olarak adlandırılan veri yapısı, yüksek performanslı sınıflandırma için paralel olarak araştırılırken, güncel FPGA'lar tarafından sağlanan bol paralellikten yararlanır. Bu *BC-SC* mimari tasarım temel olarak her özellik için tek adımlı yapılar üretir. *BC-SC* mimarisi, arama süresini de önemli ölçüde azaltmaktadır.

6.2.1. Bit Vektör Kodlu Simple CART (BC-SC)

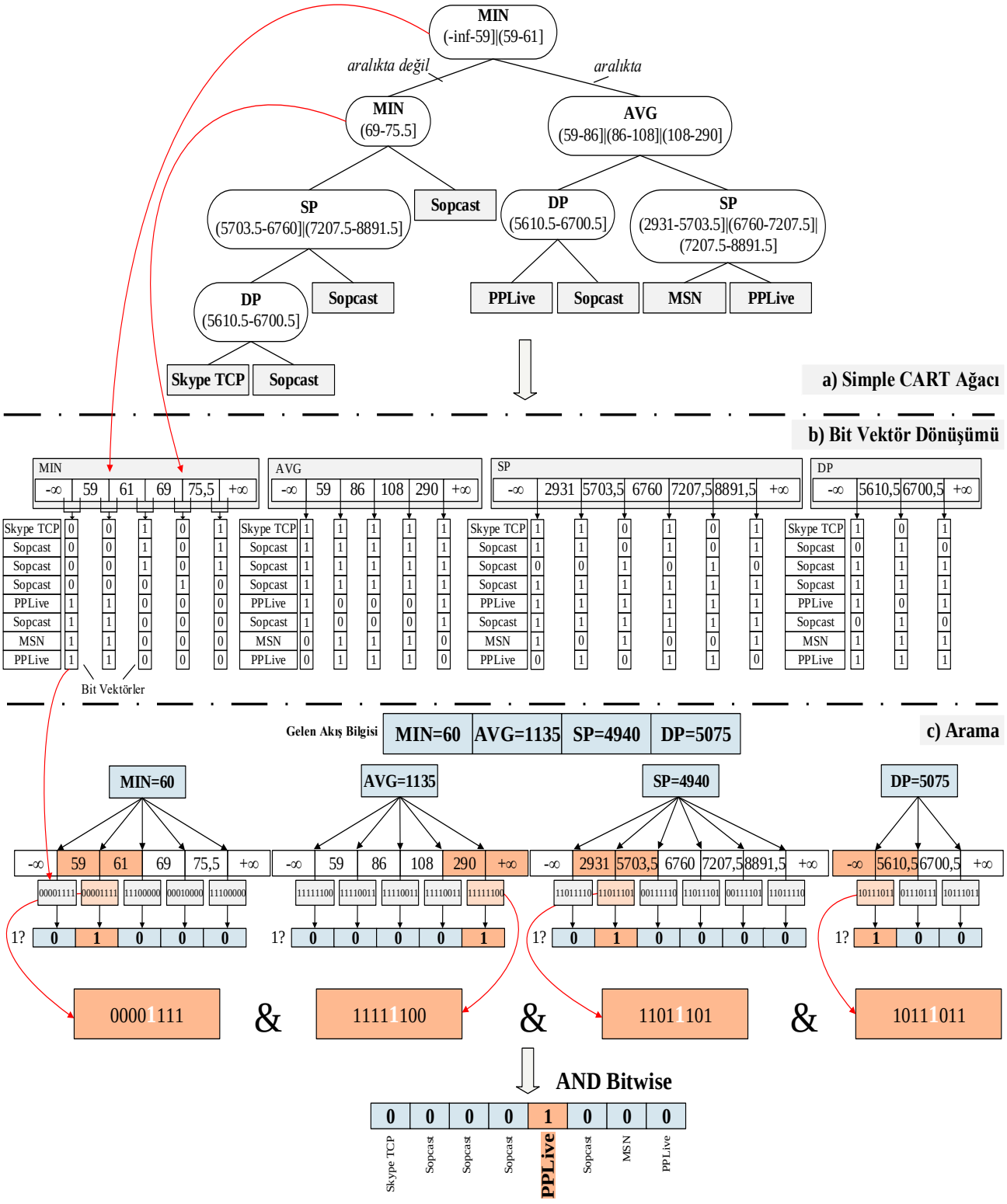
BC-SC oluştururken, Simple CART karar ağacının her bir özelliğinin düğümlerindeki *aralık* değerleri alınarak sıralanır ve bir aralık değeri ile bir sonraki aralık değeri arasında bir *iç aralık* oluşturur ve bu iç aralığa bir bit vektör değeri atanır. İç aralık değerleri üzerinde paralel arama işlemi tek adımda gerçekleştirilir.

Önerilen *BC-SC* veri yapısı, *Şekil 6.1*'te gösterilmektedir. Simple CART karar ağacında, her bir özellik için aralık kümesi *"Özellik Adı-Set"* olarak adlandırılır: MIN-Set, AVG-Set, SP-Set, DP-Set. Örneğin, *Şekil 6.1*'teki Simple CART ağacı incelendiğinde, ağacın içerdiği MIN-Özelliği için aralık değerleri $(-\infty - 59) - (59 -$

61) ve $(69 - 75.5)$ şeklindedir. Bu aralık değerleri sıralanırsa, $" - \infty, 59, 61, 69$ ve $75.5"$ elde edilir ve aynı işlem diğer özellikler aralık değerleri için de yapılır. Ardından ağacın her bir yaprak düğümündeki her uygulama sınıfı değeri soldan başlanarak okunur (Skype TCP, Sopcast, Sopcast, ...) ve tabloya işlenir ve *Uygulama Sınıfı Listesi* (Application Class List) olarak adlandırılır. *Uygulama Sınıfı Listesi* değerleri yazılırken, her özellik kümesinin her bir aralığı için; uygulama sınıfı değeri, o aralıkta ise "1" aksi durumda "0" değeri verilerek tablo oluşturulur. Bitlerden meydana gelen yeni oluşturulan bu tabloya *Bit Vektör Tablosu* adı verilir. Tüm *uygulama sınıfı listesi* için, benzer işlem her özellik kümesinin her bir aralığı için gerçekleştirilir ve *bit vektör tabloları* elde edilir. Örneğin, MIN-Set $" - \infty - 59"$ aralığı için bit vektör tablosu oluşturulursa: Simple CART ağacına bakılarak soldaki ilk uygulama sınıfı olan Skype TCP uygulama sınıfı MIN özellik $" - \infty - 59"$ aralığına girmemesi nedeniyle "0" değerini alır. Aynı şekilde, Sopcast değeri (soldan 2.) "0", Sopcast (soldan 3.) "0", Sopcast (soldan 4.) "0", PPLive (soldan 5.) "1" ($" - \infty - 59"$ aralığında olduğu için), Sopcast (soldan 6.) "1", MSN (soldan 7.), "1" ve PPLive (soldan 8.) "1" olarak bit vektör oluşturulur (Gandhi, vd., 2014). Benzer şekilde, bit vektör tabloları, her özellik kümesinin tüm aralık değerleri için oluşturulur.

Önerilen *BC-SC* yapısında, her özellik kümesinin her bir aralığı için bir *bit vektör tablosu* oluşturulmuştur. Örneğin, *Şekil 6.1b*'deki MIN-Set $" - \infty - 59"$ aralığının bit vektörü, *uygulama sınıf listesine* karşılık gelen "00001111" değeridir (yukarıdan aşağıya). Benzer şekilde, SP-Set $"6760 - 7207.5"$ aralığının bit vektör değeri "11011101" değeridir. Diğer aralıklara karşılık gelen bit vektör değerleri *Şekil 6.1b*'de gösterilmektedir.

Son olarak, BC-SC yapısı her aralık için dört alanı saklar; (1) Bitmap veya bit vektörü, (2) aralık değeri işaretçisi (R_p), (3) özellik set tanımlayıcı (FS_{ID}) her özellik seti için bir değer ve (4) Sınıf kimliği (C_{ID}) sadece son aşamada.



Şekil 6.1 a. Simple CART Ağacı b. Bit Vektör Dönüşümü c. Arama

6.2.2. *BC-SC*'de Arama Süreci

Gelen trafik değeri (paket bilgileri), her özellik grubuna (MIN-Set, AVG-Set, SP-Set ve DP-Set) paralel olarak girer ve o sete ait bir bit vektör değeri alarak çıkar. Özellik kümesine giren trafik, kümedeki tüm aralıklarda paralel olarak aralığın alt ve üst sınırları ile karşılaştırılır. Gelen trafik aralığa giriyor ise "1"; aksi halde "0" değeri alınır. Bu değerlerden "1" değerine karşılık gelen bit vektörü, özellik kümesinin çıktısı olarak sonuç *bit vektör* değeri olarak atanır. Benzer şekilde, aynı işlem diğer özellik kümelerinde de gerçekleştirilir ve tümü için ayrı ayrı bit vektör değerleri elde edilir. Tüm özellik kümelerinin elde edilen *bit vektör* değerlerine *bit bit çarpma işlemi* olarak tanımlanan *AND-Bitwise* işlemi uygulanır ve nihai *bit vektör* değeri elde edilir. Bu son değer de, sadece bir değer "1"'dir ve karşılık gelen Uygulama Sınıfı, sonuç uygulama sınıfı olarak belirlenir (*Şekil 6.1c*).

Örneğin, *Şekil 6.1c*'de gösterildiği gibi, gelen trafiğin özellik değerleri "$MIN = 60$", "$AVG = 1135$", "$SP = 4940$" ve "$DP = 5075$" olarak verilsin. Gelen trafiğin "$MIN = 60$" özellik değeri MIN-Set'e girer ve özellik kümesindeki tüm aralıklarda paralel olarak aranır. MIN özellik değeri olan "60" değeri "$-\infty - 59$" aralığına girmediğinden, bu aralıkta "0" değerini alır ve "$59 - 61$" aralığına girdiği için "1" değerini alır. Benzer şekilde sırasıyla "$61 - 69$", "$69 - 75.5$" ve "$75.5 - +\infty$" aralıkları için "0", "0" ve "0" değerlerini alır. Benzer işlem diğer özellikler (AVG, SP ve DP) için de uygulanır. "$AVG = 1135$" özellik değeri AVG-Set'e girer ve tüm aralıklarda paralel olarak aranır ve "$290 - +\infty$" aralığında "1" değerini alır ve diğer aralıklarda "0" değerini alır. "$SP = 4940$" özellik değeri SP-Set'e girer ve "$2931 - 5703.5$" aralığında "1" değerini alır ve diğer aralıklarda "0" değerini alır. "$DP = 5075$" özellik değeri de DP-Set'e girer ve "$-\infty - 5610.5$" aralığında "1" değerini alır ve diğer aralıklarda "0" değerini alır. Bu değerlerden "1" değerine karşılık gelen bit vektörü, özellik kümesinin bit vektörü olarak atandığından, her bir özellik sete ait *bit vektör* değerleri: MIN-Set'in *bit vektör* değeri: "00001111"; AVG-Set'in *bit vektör* değeri: "11111100", SP-Set'in *bit vektör* değeri: "11011101"; DP-Set'in *bit vektör* değeri: "10111011" olarak elde edilir. Tüm özellik kümelerinin elde edilen bit vektörlerine AND-Bitwise işlemi gerçekleştirilir ve son *bit vektör* değeri "00001000" olarak elde edilir. Son olarak, *uygulama sınıf*

listesinde, elde edilen *bit vektör* değerindeki ilgili sınıf "1" değerine karşılık gelen PPLive uygulama sınıfı olarak belirlenir.

6.2.3. Optimizasyon

BC-SC, tek adımlı bir veri yapısıdır. Ancak aralık sayıları ve bit vektör değerlerinin büyüklüğü donanımda sınıflandırma süresini de etkilemektedir. Bu değerlerin sayısının sınırlandırılması sınıflandırma süresini azaltacaktır. Dolayısıyla önerilen *BC-SC* veri yapısında *(i) aralık sayıları* ve *(ii) bit vektör uzunluğu* iyileştirilebilir.

Şekil 6.1c'de görüldüğü gibi, bazı bit vektör değerleri birbirine eşittir ve bu nedenle bu bit vektör değerleri birleştirilebilir. Bit vektör birleştirme işlemi, özellik kümesinin boyutunu küçültecek ve donanım üzerinde daha hızlı arama yapılmasını sağlayacaktır. Birleştirme işlemindeki kural: yalnızca ardışık aralıkların birleştirilebilir olmasıdır. Sıralı olmayan (art arda gelmeyen) bit vektör değerleri birleştirilirse, sınıflandırmada hataya neden olur. Örneğin, *Şekil 6.1c*'deki MIN-Set düğümlerinden "$-\infty - 59$" ve "$59 - 61$" aralığının bit vektör değeri "00001111"'dir. Bu iki aralık birleştirerek tek bir aralık elde edilir ve bit vektör değeri birleştirilmiş aralığa atanır. Yeni birleştirilmiş aralığın ilk ve son değeri "$-\infty - 61$" olarak değiştirilir ve bit vektör değeri "00001111" olarak yazılır. Tüm aralık bit vektör değerleri kontrol edilerek, birbirine eşit olan ardışık bit vektör değerleri için benzer işlemler gerçekleştirilir. Sonuç olarak, her özellik kümesi için daha küçük kümeler elde edilmektedir.

Özellik setteki azalma, gelen aramanın daha az aralığa girmesine ve daha az karşılaştırmayla sonuçlanmasını sağlayacaktır. Birleştirme işlemi, arama hızını artırırken arama gecikmesinde önemli bir düşüş sağlayacaktır.

Bu çalışmada kullanılan veri setinde, en büyük özellik set uzunluğu 62'dir (MAX-Set). Birleştirme işleminden sonra, MAX-Set uzunluğu 12 olacaktır. Benzer şekilde, *Çizelge 6.1*'de gösterildiği gibi diğer özellik setlerin uzunlukları da azalmaktadır. Birleştirme işlemi, kullanılan donanım kaynaklarının miktarını azaltacak ve bu durum arama hızında belirgin bir artış ve arama gecikmesinde önemli bir azalma sağlayacaktır. Optimize edilmiş *BC-SC* ($BC\text{-}SC^{opt}$) veri yapısının aralık sayısı analizi Çizelge 6.6'da gösterilmektedir.

Çizelge 6.1 BC-SC Aralık Sayısı Analizi

	Özellik Setler (FS)					
	Prtcl-Set	SP-Set	DP-Set	Avg-Set	Max-Set	Min-Set
Aralık Sayısı	1	59	59	46	62	40
Aralık Sayısı (Opt.)	1	40	40	22	12	20

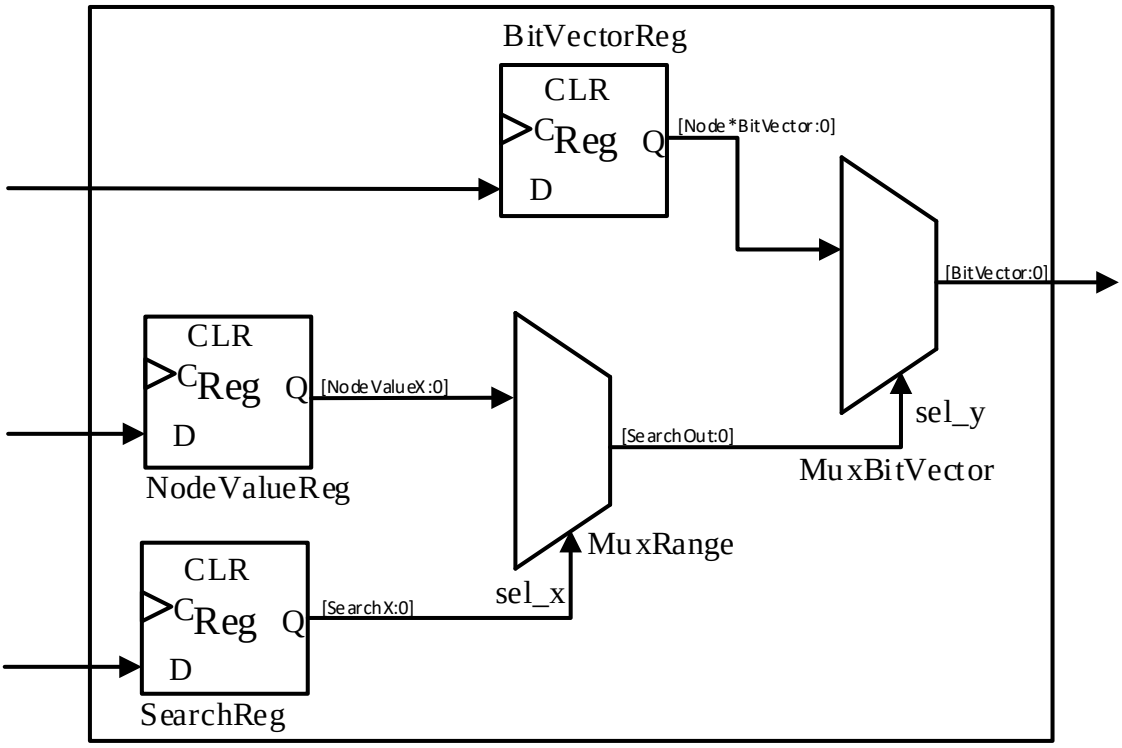
Bir diğer iyileştirme, uygulama sınıfı sayısı azaltılarak yapılabilir. Bu azaltma işlemi şu şekilde gerçekleştirilir: *Şekil 6.1a* incelenirse uygulama sınıfı sıralaması (soldan itibaren) Skype TCP, Sopcast, Sopcast, Sopcast, PPLive, Sopcast, MSN ve PPLive şeklinde olacaktır. Sıralamada görüleceği üzere 3 adet Sopcast uygulama sınıfı art arda gelmiştir. Bu durumda bu sınflar birleştirilip tek uygulama sınıfı olarak yazılabilir. Birleştirme işleminden sonra yeni uygulama sınıfı sıralaması Skype TCP, Sopcast, Sopcast, Sopcast, PPLive, Sopcast, MSN ve PPLive şeklinde olacaktır. Görüldüğü gibi, birleştirme işleminden sonra, *Şekil 6.1* için uygulama sınıfı uzunluğu 8 bit yerine 6 bit olarak azalacaktır. Dolayısıyla bit vektörler bu yeni duruma göre oluşturulabilir.

Simple CART ağacında yaprak düğümlerdeki uygulama sınıfı sayısı 112'dir. Dolayısıyla bit vektör değeri 112 bit olacaktır. Yapılan uygulama sınıfı optimizasyonu sonrasında uygulama sınıfı listesi değeri ve bit vektör uzunluğu 88 olacaktır. Uygulama sınıfındaki azalma bit vektör uzunluğunu azaltarak donanım kaynaklarının etkin kullanılmasını sağlayacaktır. Yapılan optimizasyonun uygulama sonuçları Çizelge 6.6'da gösterilmektedir.

6.3. BC-SC Mimarisi Ve FPGA Uygulaması

6.3.1. *BC-SC* Modüler PE Yapısı

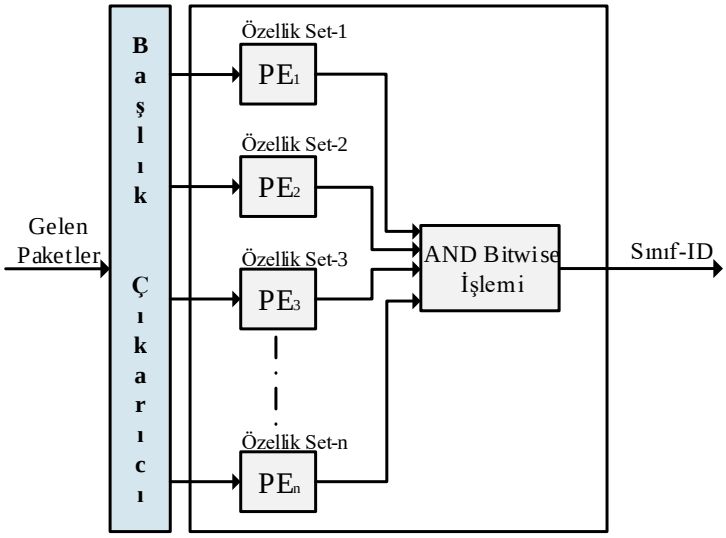
İşleme Elemanı (Processing Element - *PE*), *Şekil 6.2'*te gösterildiği gibi iki MUX'den ve registerlardan meydana gelmektedir. Registerlar, özellik set aralık değerlerini saklamak için kullanılırken; MUX'lar, aranan değerin menzilde olup olmadığını (MuxRange) ve eğer öyleyse iç aralıklarda yer alan *bit vektör* değerini (MuxBitVector) belirlemek için kullanılır. Her *PE*, gelen değere karşılık olarak bir *bit vektör* değeri üretmektedir.



Şekil 6.2 Bir BC-SC PE Dizaynı

6.3.2. *BC-SC* Mimarisi

BC-SC için önerilen mimari, *Şekil 6.3*'te gösterilmektedir. Her bir *PE* mimarisi bir özellik set saklamaktadır. Mimarideki *PE*, özellik setleri sayısına bağlıdır. Önerilen veri setinde 6 özellik yer aldığı için mimarideki PE sayısı 6 olacaktır. Her PE kendi özellik setinde yer alan aralık sayısından bir eksik aralık değeri içermektedir ($PE\ aralık\ sayısı\ =\ özellik\ set\ aralık\ sayısı\ -1$). Örneğin *Çizelge 6.3*'de Max-Set için aralık sayısı 62 olduğu için PE'de 61 aralık değeri tutulmaktadır. Diğer özellik setlerinin aralık değerleri *Çizelge 6.3*'de gösterilmektedir. FPGA üzerinde tüm PE'ler paralel olarak arandığı için sınıflandırma tek adımda gerçekleştirilmektedir. PE'den elde edilen değer AND-Bitwise işlemi ile işlenir ve bit vektör değeri elde edilir. Elde edilen *bit vektör* değerinde "1" değerine karşılık gelen, ilgili *Uygulama Sınıf Listesi*nden elde edilen değer olan Sınıf-ID (C_{ID}) değeri, arama sonucunu verir. Önerilen mimaride BRAM kullanılmadığından, arama hızı oldukça yüksektir ve sınıflandırma gecikmesi çok çok düşüktür.



Şekil 6.3 BC-SC Mimarisi

6.4. Performans Değerlendirmesi

6.4.1. Deney Düzeneği

Bu bölümde de Tstat'tan (Tstat, 2016) alınan 8 uygulama sınıfı içeren veri seti kullanılarak analizler yapılmıştır.

6.4.2. Performans Karşılaştırması

Çizelge 6.2, bir makine öğrenme aracı olan WEKA (Hall, vd., 2009) programı kullanılarak mevcut sınıflandırma tabanlı ML algoritmalarının performansını *Doğruluk* (Kolon 4), *Kappa* (Sütun 6) ve *F-Ölçümü* değerleri (Kolon 7) açısından karşılaştırmaktadır. Ayrıca, karar ağacı tabanlı algoritmalar, *ağaç derinliklerine* (Sütun 5), *yaprak düğümlerinin sayısına* (Sütun 8) ve *ağaç düğümlerinin toplam sayısına* (Sütun 9) göre de karşılaştırılmaktadır.

Önerilen *BC-SC* veri yapısı, ağaç veri yapısı olmadığından *BC-SC* için *Çizelge 6.2*'de yer alan yaprak düğüm sayısı aslında önerilen mimarideki uygulama sınıf listesini ve toplam düğüm sayısı ise toplam aralık sayısını göstermektedir.

BC-SC'nin diğer yapılardan farkı ağaç dönüşümü yapılırken asıl doğruluk oranını (bitmap dönüşümü yapılmadan önce algoritma analizi yapılırken elde edilen doğruluk oranı) korumasıdır. *Çizelge 6.2*'de en yüksek doruluk oranına sahip *C4.5* ile *BC-SC* karşılaştırıldığında doğruluk oranında %0,3'lük fark olmasına rağmen BC-SC sınıflandırma süresi ve arama gecikmesi oldukça düşüktür. Bunun nedeni BC-SC'nin ağaç yapılarından farklı olarak paralel aralık arama işlemini gerçekleştirmesidir.

Çizelge 6.2 Makine Öğrenmesi Algoritmalarının Performans Karşılaştırması

No	Algoritma	Tür	Toplam Doğruluk (%)	Derinlik	Kappa	F-Ölçümü	Yaprak Düğüm Sayısı	Toplam Düğüm Sayısı
1	BC-SC	Karar Ağacı	96.8125	NA	0.9636	0.9680	112	267
2	BC-SC^{Opt}	Karar Ağacı	96.8125	NA	0.9636	0.9680	112	267
3	SCF (n=1)	Karar Ağacı	94.0625	15	0.9320	NA	52	97
4	SCF^{Opt} (n=2)	Karar Ağacı	96.6719	15	0.9620	NA	51	98
5	SCF^{Opt} (n=4)	Karar Ağacı	95.0781	18	0.9440	NA	71	140
6	Simple CART (E-SC)	Karar Ağacı	96.8125	16	0.9636	0.9680	112	223
7	Simple CART (A-SC)	Karar Ağacı	93.9219	18	0.9305	0.9390	223	445
8	C4.5	Karar Ağacı	97.1875	44	0.9679	0.9720	103	205
9	BFTree	Karar Ağacı	96.7656	17	0.9630	0.9680	127	253
10	k-NN	Mesafe Tabanlı	96.0781	NA	0.9552	0.9610	NA	NA
11	BayesNET	İstatiksel	95.1094	NA	0.9441	0.9510	NA	NA
12	Naive Bayes	İstatiksel	94.6875	NA	0.9393	0.9470	NA	NA
13	RIPPER	Kural Tabanlı	94.9375	NA	0.9421	0.9490	NA	NA
14	YSA	Fonksiyon Tabanlı	75.0469	NA	0.7148	0.7570	NA	NA

6.4.3. Bellek Kullanımı

BC-SC, Simple CART karar ağacını bit vektörlere dönüştürür. Önerilen veri yapısında her özellik kümesinin aralık değerleri ardışık olarak düzenlenmektedir. En büyük değerler içeren DP-Set ve SP-Set için sınır değerleri 0 ve 65536'dır. Bu nedenle, DP ve SP aralık alt ve üst değeri olarak "16" bit tutulmaktadır. Avg-Set, Max-Set ve Min-Set aralıklarında "11" bit ve Prtcl-Set aralıklarında "1" bit tutulmaktadır. Ayrıca, her aralığın iç aralığında (aralık ve bir sonraki aralık arasında *alt aralık* yer almaktadır), *bit vektör* olarak adlandırılan "112" bit tutulmaktadır. Her özellik setteki aralık sayısı, her özellik değerinin düğümlerindeki benzersiz aralık değerleridir ($aralık\ sayısı = aralık - 1$) ve DP ve SP için bu sayı "59" bittir. Diğer değerler *Çizelge 6.3*'da gösterilmektedir.

Çizelge 6.3 BC-SC Veri Yapasının Bellek Gereksinimi

	Özellik Setler (FS)					
	Prtcl-Set	SP-Set	DP-Set	Avg-Set	Max-Set	Min-Set
Aralık Sayısı	1	59	59	46	62	40
Bellek (Bayt)	14	930	930	693.25	939.25	601

Çizelge 6.3, aynı zamanda önerilen *BC-SC* veri yapısının ayrıntılı hafıza sonuçlarını göstermektedir. *Çizelge 6.4* bu tez çalışmasında önerilen yapıların performanslarını karşılaştırmaktadır. Bu tezde önerilen veri yapılarından *BC-SC* veri yapısı en kötü bellek kullanımına sahip olmasına rağmen, optimize edilmiş *BC-SC* veri yapısı 2.02 Kbayt ile diğer veri yapılarına göre en iyi (en az) bellek kullanımına sahiptir.

Çizelge 6.4 Simple CART Yapılarının Bellek Gereksinimi

Mimari	**Derinlik**	**Aşama -1 Düğüm Sayısı**	**Aşama -2 Düğüm Sayısı**	**Toplam Düğüm Sayısı**	**Bellek (Bayt)**
BC-SC	NA	NA	NA	267	4107.5
BC-SC ^{Opt}	NA	NA	NA	135	2068.4
SCF (n=1)	15	267	97	364	2275.0
SCF^{Opt} (n=2)	15	267	98	365	2284.0
SCF^{Opt} (n=4)	18	267	140	407	2767.0
E-SC	22	267	223	490	3632.0
SC	16	NA	NA	223	55666.1

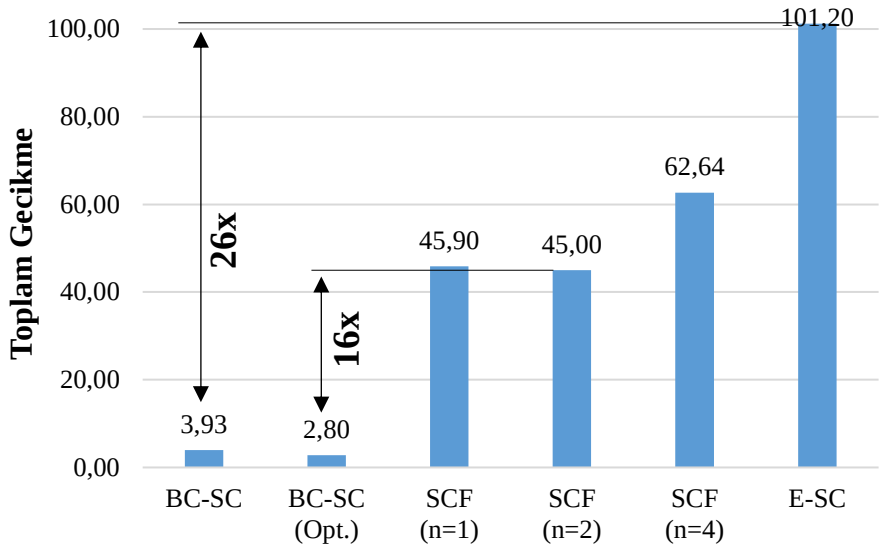
6.4.4. Gecikme

Çizelge 6.5'da bu tezde önerilen mimarilerin gecikme sonuçları gösterilmektedir. *E-SC* veri yapısı 22 $(6 + 16)$ saat döngüsüne sahiptir ve sınıflandırma süresi 4.60 ns'dir ve $döngü\ x\ zaman$: 22 x 4.60 $=$ 101.2 ns'dir. Önerilen *BC-SC* veri yapısı bir adımda sınıflandırıldığından ve sınıflandırma zamanı 3.93 ns olduğundan, $döngü\ x\ zaman$: 1 x 3.93 $=$ 3.93 ns'dir. *BC-SC* mimarisi *E-SC* mimarisine göre nerede ise $27x$ daha az gecikme ve optimize edilmiş *BC-SC* mimarisi ise optimize edilmiş *SCF* $(n = 2)$ mimarisine göre nerede ise $16x$ daha az gecikme elde etmektedir.

Çizelge 6.5 Önerilen Mimarilerin Gecikme Sonuçları

Mimari	Derinlik	Saat	Toplam Gecikme
BC-SC	NA	3.93	3.93
BC-SC ^{Opt}	NA	2.80	2.80
SCF (n=1)	15	3.06	58.95
SCF^{Opt} (n=2)	15	3.00	45.00
SCF^{Opt} (n=4)	18	3.48	62.64
E-SC	22	4.60	101.20

Önerilen mimarilerin gecikme karşılaştırması *Şekil 6.4*'da gösterilmektedir. *Şekil 6.4*'ya göre en iyi sonucu optimize edilmiş *BC-SC* $(2.80\,ns)$ mimarisinin verdiği görülmektedir.



Şekil 6.4 Önerilen Mimarilerin Gecikme Karşılaştırması

6.4.5. İş Oranı

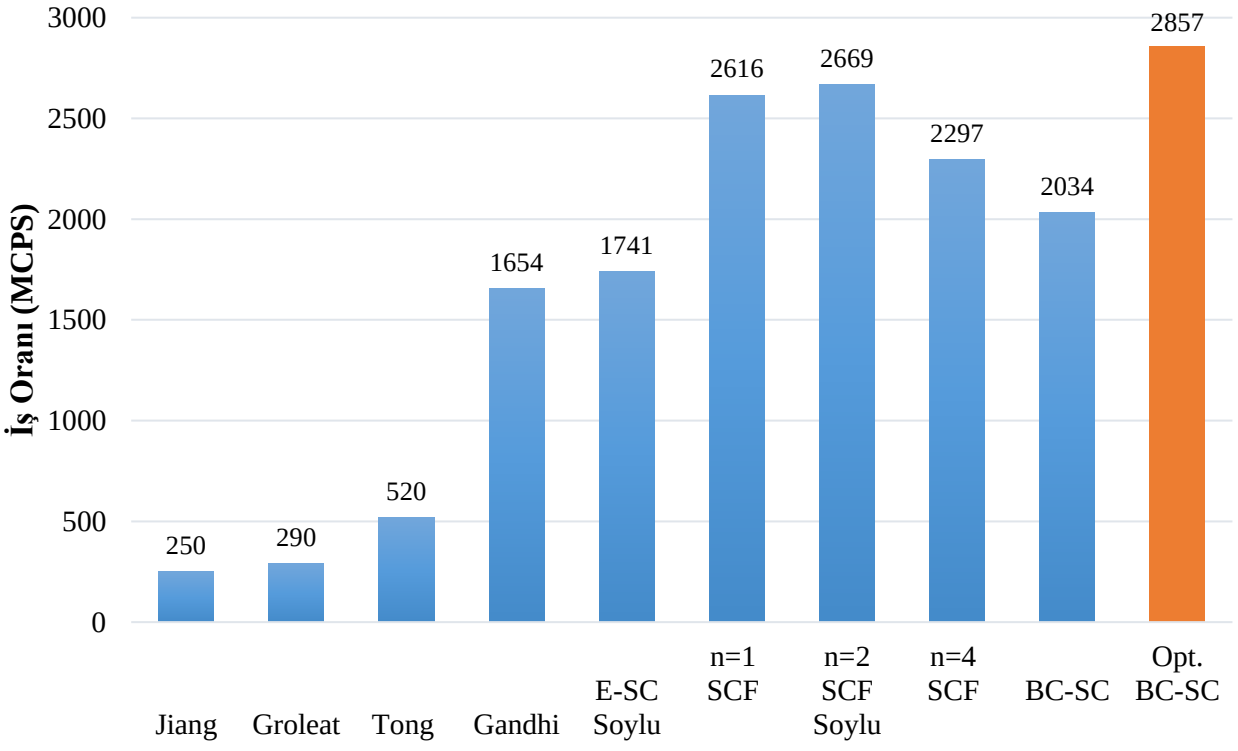
Önerilen *BC-SC* mimari tasarımı Xilinx Vivado 2018.1 (en son sürüm) kullanılarak Verilog donanım tanımlama dili (HDL) ile Xilinx Virtex-Ultra XCVU440FLGA2892 aygıtı üzerinde hedef olarak -3 hız aralığında uygulandı. Önerilen mimarinin FPGA sonuçları *Çizelge 6.6*'de gösterilmektedir. Karşılaştırma sonucuna göre, optimize edilmiş *BC-SC* mimarisinin diğer tüm mimarilerden daha iyi performans gösterdiğini, saatte bir $357.1\,MHz$ hızına ulaştığını ve $914\,Gbps$ veya 2857 milyon sınıflamasını $(MCPS)$ desteklediğini göstermektedir. Buna karşın *BC-SC* mimarisi

Çizelge 6.6'de gösterildiği gibi en çok IO kullanan tasarımdır (83.48 %). Önerilen mimaride 8 uygulama sınıfı yer almakta olup gelecek çalışmalarda daha çok uygulama sınıfı yer alması durumunda mevcut FPGA aygıtları destekleyemeyebilir. Dolayısıyla önerilen mimarinin bu yönü en büyük dezavantajı olarak ortaya çıkmaktadır.

Çizelge 6.6 Uygulama Sonuçları

Mimari	**İş Oranı** (Gbps)	**İş Oranı** (MCPS)	**Süre** (ns)	**Frekans** (MHz)	**LUT** Sayısı	**Slice** Sayısı	**BRAM** (36-Kb)	**BRAM** (18-Kb)	**Bağlı** IOB
BC-SC	650	2034	3.93	253.2	3827(0.20%)	808(0.02%)	0(0.00%)	0(0.00%)	504(38.50%)
BC-SC Opt	914	2857	2.80	357.1	2089(0.12%)	664(0.02%)	0(0.00%)	0(0.00%)	374(83.48%)
SCF (n=1)	837	2616	3.06	322.5	10106(3.30%)	2894(3.81%)	0(0.00%)	0(0.00%)	139(23.17%)
SCF^{Opt} (n=2)	854	2669	3.00	333.3	6560(2.20%)	1812(2.39%)	0(0.00%)	0(0.00%)	130(21.67%)
SCF^{Opt} (n=4)	735	2297	3.48	285.7	5615(1.80%)	1650(2.17%)	4(0.39%)	4(0.19%)	130(21.67%)
E-SC	557	1741	4.60	217.0	3933(1.30%)	1351(1.8%)	130(12.6%)	6(0.30%)	201(33.5%)

Şekil 6.5'de, liretatürdeki popüler çalışmaların ve bu tezde önerilen 3 çalışmanın performans sonuçları gösterilmektedir. Önerilen her üç çalışmanın da literatürdeki çözümlerden daha iyi sonuçlar elde ettiği görülmektedir.



Şekil 6.5 FPGA Tabanlı Tasarımların İş Oranı (MCPS) Karşılaştırması (BC-SC)

6.5. Sonuç

Bu bölümde, yüksek performanslı trafik sınıflandırması için tek adımlı bir *BC-SC* veri yapısı önerilmiş ve FPGA'lar tarafından sağlanan bol paralellik avantajları kullanılmıştır. Klasik karar ağaçlarında, sınıf sayısındaki artışla birlikte sınıfları birbirinden ayırmak için çok sayıda düğüm gereklidir. Klasik karar ağaçlarının aksine önerilen *BC-SC* veri yapısında yeni eklenen uygulama sınıfı sadece bit vektör genişliğini arttırır. *BC-SC* mimarisinde arama gecikme performansı klasik karar ağaçlarına veya ağaç tipi yapıların aksine tek adımda sınıflandırma ile üstündür. Önerilen tasarım, alt aralık değerleri aynı olan aralık değerleri ve uygulama sınıfı değerleri aynı olan uygulama sınıflarını birleştirilerek donanım kaynakları etkin bir şekilde kullanılabilir. Deneysel sonuçlar, *BC-SC* mimarisinin donanım üzerinde kolaylıkla eşleştirilebileceğini ve FPGA tabanlı optimize edilmiş *BC-SC* mimarisinin, tüm mevcut yaklaşımları iş oranı, arama gecikmesi ve bellek gereksinimi açısından geride bıraktığını kanıtlamaktadır.

BÖLÜM 7

SONUÇLAR

7.1. İnternet Trafik Sınıflandırma

İnternet trafik sınıflandırma ISS'ler, kamu kurumları veya özel şirketler için oldukça önemli bir kullanım alanı oluşturmaya devam etmektedir. Her kurumun amacı farklı olduğu için bir internet trafik sınıflandırıcıdan beklentisi de farklıdır. Bu önem ve beklenti trafik sınıflandırıcı tasarımında da istenen sonucu en iyi şekilde verebilecek ve bazı kriterleri sağlayabilen mimari tasarımlara ihtiyacı artırmaktadır. Trafik sınıflandırma yapılırken en önemli kriterlerin başında doğruluk gelmektedir. Bir sınıflandırıcı ne kadar doğru sınıflandırıyorsa o kadar iyi bir sınıflandırıcı olarak tanımlanır. Fakat doğruluk tek başına yeterli bir kriter değildir. Doğruluk yanında sınıflandırma hızı da birçok kullanım amacı için önemlidir. Gerçek zamanlı trafik sınıflandırma yapabilmek için saniyede $100 + Gbps$ üzerinde hızlara ulaşmak gerekliliğini göz önünde bulundurmak gerekmektedir. Aynı zamanda belirtilen hususlar kadar önemli bir nokta ise arama gecikmesi kavramıdır. Bir sınıflandırıcının arama gecikmesinin mümkün olduğu kadar az olması istenmektedir.

Bu tezde gerçek zamanlı trafik sınıflandırma mimari tasarım için 3 farklı mimari tasarım önerilmiştir: (i) *E-SC* (Genişletilmiş Simple CART – Extented Simple CART) mimari tasarımı, (ii) *SCF* (Simple CART Ormanları – Simple CART Forest) mimari tasarımı, (iii) *BC-SC* (Bit Vektör Kodlu Simple CART – Bit Vector Coded Simple CART) mimari tasarımı.

Önerilen her mimarinin birbirlerinden üstünlükleri ve birbirlerine göre zayıf yönleri vardır. Ayrıca önerilen mimarilerin kullanım alanlarına göre avantajları da olabilir. Örneğin, bir kurum için internet trafik sınıflandırma mimarisinden beklentisi sınıflandırma hızının çok iyi olması değil sınıflandırma doğruluğunun yüksek olması ise tercih edeceği sınıflandırıcı buna uygun bir sınıflandırıcı olmalıdır.

Çizelge 7.1 Önerilen Mimarilerin Uygulama Sonuçları

Mimari No	**Mimari** Adı	**Doğruluk** %	**İş Oranı** (Gbps)	**İş Oranı** (MCPS)	**Saat** (ns)	**Frekans** (MHz)	**Bellek** Bayt	**Gecikme** ns
i	E-SC	96.8125	557	1741	4.60	217.0	3632.0	101.20
ii	SCF (n=1)	94.0625	837	2616	3.06	322.5	2275.0	58.95
	SCF^{Opt} (n=2)	96.6719	854	2669	3.00	333.3	2284.0	45.00
	SCF^{Opt} (n=4)	95.0781	735	2297	3.48	285.7	2767.0	62.64
iii	BC-SC	96.8125	650	2034	3.93	253.2	4107.5	3.93
	BC-SC ^{Opt}	96.8125	914	2857	2.80	357.1	2068.4	2.80

Çizelge 7.1'de bu tezde önerilen mimarilerin uygulama sonuçları yer almaktadır. *Çizelge 7.1*'e göre önerilen mimarilerden sınıflandırma doğruluğu en yüksek trafik sınıflandırıcısı *E-SC* mimarisi (96.8125) ve *BC-SC* (96.8125) mimarisidir. Fakat sınıflandırma hızı bakımından ise *BC-SC* $(2.80\,ns)$ mimarisi en iyisidir. Eğer tercih edilen sınıflandırma kriteri yüksek iş oranı ise önerilen trafik sınıflandırıcılar arasında yüksek iş oranına sahip trafik sınıflandırıcıların optimize edilmiş *BC-SC* mimarisi ve *SCF* $(n = 1, 2, 4)$ mimarileri olduğu görülmektedir. Bu mimariler arasında ise $2857\ MCPS$ veya $914\ Gbps$ ile en yüksek iş oranına sahip sınıflandırıcı optimize edilmiş *BC-SC* mimarisidir. Eğer tercih edilecek kriter bellek kullanımı ise yine önerilen trafik sınıflandırıcılar arasında düşük bellek kullanımına sahip trafik sınıflandırıcıların optimize edilmiş *BC-SC* mimarisi ve *SCF* $(n = 1, 2, 4)$ mimarileri olduğu görülmektedir. Bu mimariler arasında ise 2068.4 bayt bellek kullanımı ile en düşük bellek kullanımına sahip sınıflandırıcı optimize edilmiş *BC-SC* mimarisidir. Tercih edilecek kriter arama gecikmesi ise *BC-SC* mimarileri düşük arama gecikmesine sahiptir ve bu mimariler arasında optimize edilmiş *BC-SC*^{opt} mimarisi (toplam $2.80\,ns$) en düşük arama gecikmesine sahip mimari olduğu görülmektedir.

7.2. Gelecek Çalışma

Önerilen veri yapıları üzerinde iyileştirmeler ile elde edilen performansları (doğruluk, iş oranı, sınıflandırma süresi, bellek kullanımı ve gecikme gibi) arttırılabilir.

Mevcut ML algoritmaları internet trafik sınıflandırma işlemine özel olarak yazılan algoritmalar olmadığı için bu alana uyarlamak zorunluluğu doğmaktadır. Örneğin FPGA mimarisine en uygun yapılar karar ağacı yapılarıdır fakat ağaç derinliği arttıkça FPGA tabanlı mimarilerin sınıflandırma performansı düşmektedir. Karar ağaçları yapısında ağaç derinliğini belirli noktalarda sınırlamanın sonuçlar üzerinde çok etkili olacağını gözlemledik (*Bölüm 0*). Dolayısıyla bu sorunu çözmek için ağaç derinliklerini otomatik olarak ayarlayabilen veya sınırlayabilen ve aynı zamanda sınıflandırma doğruluğundan da ödün vermeyen bir ML karar ağacı algoritması tasarlamayı planlamaktayız.

Gerek literatür taramasında gerekse önermiş olduğumuz çalışmalarda karşılaşılan sorunların başında ölçeklenebilirlik sorunu gelmektedir. Uygulama sınıflarının artması günümüzde mevcut çalışmalarda önerilen mimarilerin bu kadar fazla sayıda uygulama sınıfı ile etkin sonuç vermeyebilir. Uygulama sınıfının çok fazla artması doğruluk oranı üzerinde olumsuz etkiye neden olacaktır. Dolayısıyla kullanılacak algoritma da çok sayıda sınıfı birbirinden ayırmaya uygun bir algoritma olmalıdır. Ayrıca bu artış ağaç derinliğini olumsuz etkileyecektir. Dolayısıyla uygulama sınıfındaki artış doğruluk, iş oranı, sınıflandırma süresi, bellek kullanımı ve gecikme üzerinde olumsuz etkiye neden olacaktır. Ölçeklenebilirlik sorununa çözüm üreten mimariler tasarlamayı planlamaktayız.

Makine öğrenmesi tabanlı çalışmalarda ağaç derinliği, uygulama sınıfı sayısının artması ile doğrudan etkilenmektedir. Ağaç derinliğinin artması ise mimarilerde olumsuz sonuçlara (doğruluk, iş oranı, sınıflandırma süresi, bellek kullanımı ve gecikme üzerinde olumsuz etkilemekte) neden olmaktadır. Ağaç derinliğinin artmasını önlemek için ise önerilen algoritmanın çıktısı üzerinde bazı düzenlemeler yapmak gerekmektedir. Bu düzenlemelerin başında aşırı dallanan düğümleri budayarak kısaltmak yer almaktadır. Bu işlemi belirli kurallar dahilinde yaparak yeni öneriler sunmayı planlamaktayız.

Bu tez kapsamında önerilen mimarilerin tamamı FPGA üzerinde uygulanmıştır. Mevcut mimarileri günümüzde artık önemli bir gelişme sağlayan Grafik İşlemciler (GPU), Çok Çekirdekli İşlemciler (Multi Core) veya ASIC üzerinde de uygulamayı planlamaktayız.

Bu tezde önerilen mimariler sadece internet trafik sınıflandırma için kullanılmıştır. Önerilen mimariler oldukça genel mimariler olup başka uygulama alanlarında da kullanılabilir olduğunu düşünmekle birlikte bu alanları araştırmayı planlamaktayız.

Bu tezde kullanılan uygulama sınıfları arasında sıkıştırılmış uygulamalar veya şifrelenmiş uygulamalar yer almamaktadır. Önerilen mimarileri sıkıştırılmış uygulamalar ve şifrelenmiş uygulamaları da sınıflandırabilecek şekilde revize etmeyi planlamaktayız.

KAYNAKLAR

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6(1), 37–66.

Alshammari, R. and Zincir-Heywood, A. (2009). Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype. *Computational Intelligence for Security and Defense Applications, CISDA, Ottawa, Ontario, Canada July 08-10*, 289-296.

Angevine, D. and Zincir-Heywood, N. (2008). A Preliminary Investigation of Skype Traffic Classification Using a Minimalist Feature Set. *Availability, Reliability and Security, ARES 08, Third International Conference, Barcelona, Spain.*

Baker, F., Foster, B., and Sharp, C. (2004). *Cisco Architecture for Lawful Intercept in IP Networks.* The Internet Society, Report, https://tools.ietf.org/html/rfc3924.

Bonfiglio, D., Mellia, M., Meo, M., Rossi, D. and Tofanelli, P. (2007). Revealing Skype Traffic: When Randomness Plays with You. *SIGCOMM '07 Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 37*(4), 37-48.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A Training Algorithm For Optimal Margin Classifiers. *COLT'92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, New York, NY, USA*, 144-152.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.I. (1984). *Classification And Regression Trees*. Belmont, California: Wadsworth.

Chen, X. W. and Wasikowski, M. (2008). FAST: A Roc-Based Feature Selection Metric For Small Samples And İmbalanced Data Classification Problems. *KDD'08 Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Las Vegas, Nevada, USA, August 24-27, 2008

Cohen, W. W. (1995). Fast Effective Rule Induction. *ICML'95 Proceedings of the Twelfth International Conference on International Conference on Machine Learning, Tahoe City,* California, USA, 115-123.

Crotti, M., Dusi, M.,Gringoli, F. and Salgarelli, L. (2007). Traffic Classification Through Simple Statistical Fingerprinting. *ACM SIGCOMM Computer Communication Review, 37*(1), 5-16.

Dunham, M. (2003). *Data Mining Introductory and Advanced Topics.* New Delhi, India: Prentice Hall of India.

Erman, J., Mahanti, A., Arlitt, M., Cohen, I. and Williamson, C. (2007). Semi-Supervised Network Traffic Classification. *SIGMETRICS'07, Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems,* San Diego, California, USA, *35*(1), 369-370.

Este, A., Gringoli, F. and Salgarelli, L. (2009). Support Vector Machines for TCP Traffic Classification. *Computer Networks: The International Journal of Computer and Telecommunications Networking, 53*(14), 2476-2490.

Fayyad, U. M. and Irani, K. B. (1991). Multi-Interval Discretization of Continuous Valued Attributes for Classification Learning. *IJCAI - Proceedings of the Twelfth International Conference on Artificial Intelligence,* August 24-30, Sydney, Australia.

Fisher, R. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annual Eugenics, 7*(2), 179–188.

Frank, J. (1994). Machine Learning and Intrusion Detection: Current and Future Directions. *National 17th Computer Security Conference, 14*(1), 31-31.

Gandhi, V. R., Qu, Y. R., and Prasanna, V. K. (2014). High-Throughput Hash-based Online Traffic Classification Engines on FPGA. *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig),* Cancun, Mexico.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA, USA: Addison-Wesley Longman Addison-Wesley Longman Publishing Co., Inc.

Groleat, T., Arzel, M. and Vaton, S. (2012). Hardware Acceleration of SVM-based Traffic Classification on FPGA. *8th International Wireless Communications and Mobile Computing Conference (IWCMC),* Limassol, Cyprus.

Haffner, P., Sen, S., Spatscheck, O. and Wang, D. (2005). ACAS: Automated Construction of Application Signatures. *MineNet'05 Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data*, Philadelphia, Pennsylvania, USA.

Haijian, S. (2007). *Best-first Decision Tree Learning.* (Thesis of Master), The University of Waikato, Hamilton, New Zealand.

Hall, M.A. and Holmes, G. (2003). Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, *15*(6), 1437– 1447.

Hall, M.A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. (2009). The Weka Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10–18.

Harthi, A. F. A. (2015). *Designing an Accurate and Efficient Classification Approach for Network Traffic Monitoring*, (Doctor of Philosophy), RMIT University, Melbourne, Victoria, Australia.

Heckerman, D., Mamdani, A. and Wellman, M. P. (1995). Real-World Applications of Bayesian Networks. *Communications of the ACM*, *38*(3), 24-26.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *National Academy of Sciences of the USA,* 79(8), 2554–2558.

Jiang, W. and Gokhale, M. (2010). Real-Time Classification of Multimedia Traffic Using FPGA. *Field Programmable Logic and Applications (FPL), International Conference,* Milano, Italy.

John, G. H. and Langley, P. (1995). Estimating Continuous Distributions In Bayesian Classifiers. *UAI'95 Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montréal, Qué, Canada.

Karagiannis, T., Broido, A., Brownlee, N., Claffy, K. and Faloutsos, M. (2004a). Is P2P Dying or Just Hiding?. *Globecom,* Dallas, TX, USA.

Karagiannis, T., Broido, A., Faloutsos, M., and Claffy, K. (2004b). Transport Layer Identification of P2P Traffic. *IMC'04 Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement,* Taormina, Sicily, Italy.

Karagiannis, T., Papagiannaki, T., ve Faloutsos, M. (2005). BLINC: Multilevel Traffic Classification in the Dark. *SIGCOMM'05 Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications,* Philadelphia, Pennsylvania, USA.

Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M. and Lee, K. (2008). Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices, *CoNEXT'08, ACM CoNEXT Conference*, Madrid, Spain.

Kohavi, R. and John, G. (1997). Wrappers For Feature Subset Selection. *Artificial Intelligent, 97*(1-2), 273–324.

Kohavi, R. (1995). A Study Of Cross-Validation And Bootstrap For Accuracy Estimation And Model Selection. *IJCAI'95 Proceedings of the 14th international Joint Conference on Artificial Intelligence, 2*, 1137-1143.

Korczynski, M. (2012). *Classifying Application Flows and Intrusion Detection in Internet Traffic.* (Thesis of Ph.D), Grenoble Institute of Technology, Grenoble, France.

Landis, J. and Koch, G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics, 33*(1), 159–174.

Lim, Y. S., Kim, H. C., Jeong, J., Kim, C. K., Kwon, T., and Choi, Y. (2010). Internet Traffic Classification Demystified: On the Sources of the Discriminative Power. *Co-NEXT'10 Proceedings of the 6th International Conference,* Philadelphia, Pennsylvania, USA.

Lin, Z., Lo, C. and Chow, P. (2012). K-means Implementation on FPGA for High-Dimensional Data using Triangle Inequality. *Field Programmable Logic and Applications (FPL), 22nd International Conference*, Oslo, Norway.

Luo, Y., Xiang, K. and Li, S. (2008). Acceleration of Decision Tree Searching for IP Traffic Classification. *ANCS'08 Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems,* San Jose, California, USA.

Maxfield, C. (2004). *The Design Warrior's Guide to FPGAs*. Burlington, MA 01803, USA: Newnes Elsevier.

Mitchell, T. M. (1997). *Machine Learning*. New York, USA: McGraw-Hill, Science, Engineering, Math.

Monemi, A., Zarei, R., Marsono, M. N. and Khalil-Hani, M. (2013). Parameterizable Decision Tree Classifier on NetFPGA. *Advances in Intelligent Systems and Computing*, *182*, 119–128.

Moore, A. W. and Papagiannaki, K. (2005). Toward the Accurate Identification of Network Applications. *Passive and Active Network Measurement*, Springer, Berlin, Heidelberg, Germany.

Moore, A. W. and Zuev, D. (2005). Internet Traffic Classification using Bayesian. *SIGMETRICS'05, Proceedings of the 2005 International Conference on Measurement and Modeling of Computer Systems*, Alberta, Canada.

Nguyen, T. T. and Armitage, G. J. (2006a). Training on Multiple Sub-Flows to Optimise The Use of Machine Learning Classifiers in Real-World IP Networks. *Local Computer Networks, Proceedings, 31st IEEE Conference*, Tampa, FL, USA.

Nguyen, T. T. and Armitage, G. J. (2006b). Synthetic Sub-Flow Pairs for Timely and Stable IP Traffic Identification. *Australian Telecommunication Networks & Applications Conference (ATNAC)*, Australia.

Nguyen, T. T. and Armitage, G. J. (2008). A Survey of Techniques for Internet Traffic Classification using Machine Learning. *IEEE Communications Surveys & Tutorials*, *10*(4), 56–76.

Papadonikolakis, M. and Bouganis, C. S. (2012). Novel Cascade FPGA Accelerator for Support Vector Machines Classification. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(7), 1040–1052.

Park, J., Tyan, H. R and Kuo, C. J. (2006). GA-Based Internet Traffic Classification Technique for QoS Provisioning. *Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP'06 International Conference*, Pasadena, CA, USA.

Qu, Y. R. and Prasanna, V. K. (2014). Compact Hash Tables for High-Performance Traffic Classification on Multi-core Processors. *SBAC-PAD*, Jussieu, France

Qu, Y. R. and Prasanna, V. K. (2015). Enabling High Throughput and Virtualization for Traffic Classification on FPGA. *Field-Programmable Custom Computing Machines (FCCM)*, Vancouver, BC, Canada.

Quinlan, J. R. (1987). Simplifying Decision Trees. *International Journal of Man Machine Studies, 3*(27), 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Reich, Y. and Fenves, S. J. (1991). *The Formation and Use of Abstract Concepts in Design. Concept Formation: Knowledge and Experience in Unsupervised Learning, Francisco*. CA, USA: Morgan Kaufmann.

Roughan, M., Sen, S., Spatscheck, O. and Duffield, N. (2004). Class-of-Service Mapping for Qos: A Statistical Signature-Based Approach to IP Traffic. *IMC '04 Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, Taormina*, Sicily, Italy.

Sen, S., Spatscheck, O. and Wang, D. (2004). Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. *13th international conference on World Wide Web*, New York, NY, USA.

Shi, Z. (1992). *Principles of Machine Learning. Beijing*. China: International Academic Publishers.

Silver, B. (1990). Netman: A Learning Network Traffic Controller. *IEA/AIE'90 Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 2,* 923-931.

Soylu, T., Erdem, O., Carus A. and Güner, E. S. (2017). Simple CART Based Real-Time Traffic Classification Engine on FPGAs. *27th International Conference on ReConFigurable Computing and FPGAs (ReConFig'17),* Cancun, Mexico.

Soylu, T., Erdem, O., Carus A. and Güner, E. S. (2018). Real-Time Traffic Classification using Simple CART Forest on FPGA. *IEEE 19th International Conference on High Performance Switching and Routing (HPRS'2018),* Bucharest, Romania

Stewart, L., Armitage, G., Branch, P. and Zander, S. (2005). An Architecture For Automated Network Control of Qos Over Consumer Broadband Links. *TENCON 2005 - 2005 IEEE Region 10 Conference,* Melbourne, Australia.

Tong, D., Sun, L., Matam K. and Prasanna, V. (2013). High Throughput and Programmable Online Trafficclassifier on FPGA. *FPGA '13 Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays,* 255–264.

Tong, D., Qu, Y. R. and Prasanna, V. K. (2014). High-Throughput Traffic Classification on Multi-core Processors. *2014 IEEE 15th International Conference, High Performance Switching and Routing (HPSR),* Vancouver, BC, Canada.

TSTAT. (2016). Available: http://tstat.tlc.polito.it/traces.shtml.

Wang, Y. (2013). *Automatic Network Traffic Classification*. (Thesis of Ph.D.), Geelong VIC 3220 - Australia: Deakin University, Australia.

William, W. (1995). Fast Effective Rule Induction. *ICML'95 Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, 115-123.

Williams, N., Zander, S. and Armitage, G. (2006). A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *ACM SIGCOMM Computer Communication Review, 36(*5), 5-16.

Winston, P. H. (1984). *Artificial Intelligence (2nd ed.).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (Second Edition).* Francisco, CA, USA: Morgan Kaufmann Publishers of Elsevier,

Xilinx. (2018). Available: https://www.xilinx.com

Yohannes, Y. and Webb, P. (1999). *Classification and Regression Trees (CART): A User Manual for Identifying Indicators of Vulnerability to Famine and Chronic Food Insecurity*. Washington BC, USA: International Food Policy Research Institute.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Tuncay SOYLU

Doğum Yeri ve Tarihi : Terme/Samsun – 1981

e-mail : soylu.tuncay@gmail.com

web : www.tuncaysoylu.com

Eğitim Bilgileri

Lise	:	Elektrik, Çarşamba Teknik ve Endüstri Meslek Lisesi, Samsun	1998
Lisans	:	Elektrik Eğitimi, TEF, Marmara Üniversitesi, İstanbul	2004
Lisans	:	Elektrik-Elektronik Mühendisliği, Mühendislik Fakültesi, Trakya Üniversitesi, Edirne	2018
Y. Lisans	:	Bilgisayar Mühendisliği, Trakya Üniversitesi, FBE, Edirne	2012
Doktora	:	Hesaplamalı Bilimler, Trakya Üniversitesi, FBE, Edirne	2018

İş Deneyimleri

2006 – 2009	Gülpano Elektrik San. Tic. Ltd. Şti.	Üst Düzey Yönetici
2009 –	İpsala Meslek Yüksekokulu, Trakya Üniversitesi	Öğretim Görevlisi

Yabancı Diller

İngilizce ve Almanca

YAYINLAR

1. T. Soylu, O. Erdem, A. Carus And E. S. Güner**, "Simple CART Based Real-Time Traffic Classification Engine on FPGAs"**, *27th International Conference on ReConFigurable Computing and FPGAs (ReConFig'2018)*, Cancun - Meksika, 2017.
2. T. Soylu, O. Erdem, A. Carus and E. S. Güner, **"Real-Time Traffic Classification using Simple CART Forest on FPGAs"** *IEEE 19th International Conference on High Performance Switching and Routing (HPRS'2018)*, Bükreş - Romanya, 2018.
3. T. Soylu, O. Erdem and A. Carus **"Bit Vector-Coded Simple CART Structure for Low Latency Traffic Classification on FPGAs"**, *28th International Conference on ReConFigurable Computing and FPGAs (ReConFig'2018)*, Cancun - Meksika, 2018, (Submitted).