APACHE SPARK KULLANILARAK BÜYÜK BOYUTLU GÖRÜNTÜLERİN ANALİZİ

Betül DOLAPCI

2020
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Tez Danışmanı
Dr. Öğr. Üyesi Caner ÖZCAN

APACHE SPARK KULLANILARAK BÜYÜK BOYUTLU GÖRÜNTÜLERİN ANALİZİ

Betül DOLAPCI

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

Tez Danışmanı
Dr. Öğr. Üyesi Caner ÖZCAN

KARABÜK
Temmuz 2020

Betül DOLAPCI tarafından hazırlanan "APACHE SPARK KULLANILARAK BÜYÜK BOYUTLU GÖRÜNTÜLERİN ANALİZİ" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Caner ÖZCAN .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından Oy Birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 06/07/2020

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. İsmail Rakıp KARAŞ (KBÜ)	.....
Üye : Dr. Öğr. Üyesi Caner ÖZCAN (KBÜ)	.....
Üye : Dr. Öğr. Üyesi Yasemin GÜLTEPE (KÜ)	.....

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Hasan SOLMAZ .....
Lisansüstü Eğitim Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Betül DOLAPCI

ÖZET

Yüksek Lisans Tezi

APACHE SPARK KULLANILARAK BÜYÜK BOYUTLU GÖRÜNTÜLERİN ANALİZİ

Betül DOLAPCI

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Caner ÖZCAN
Temmuz 2020, 72 sayfa

Günümüzde yaşanan dijital dönüşüm süreci ve internetin küreselleşmesinden kaynaklı kolay erişilir olması, yüksek hacimlerde ve her türde (görüntü, ses, video, metin vb.) veri üretilebilmesine olanak sağlamıştır. Üretilen verinin boyutu, düzensizliği ve çeşitliliği gibi sebeplerden dolayı, veri üzerinde analiz yapılması ve anlam çıkarımı gittikçe zorlaşmaktadır. Görüntü verilerinin daha küçük parçalara bölünmesi ve bu parçalardan elde edilen ayırt edici ve bağımsız özelliklere sahip bir vektörle temsil edilmesi analiz işlemini kolaylaştırmaktadır. Bu nedenle, öncelikle görüntü verilerini küçük piksel bloklarına bölen bir blok bölümü yöntemi uygulanır. Büyük verinin boyut azaltımına gidilerek daha küçük boyutlarda ifade edilmesi özellik vektörü ile gerçekleştirilir. Bu çalışmada, veri tipi olarak kullandığımız görüntünün analizi için hibrit bir öznitelik vektörü oluşturulmuştur. Görüntülerin renk ve doku özelliklerinden çıkarılan alt özelliklerin bir arada kullanılması ile oluşturulan hibrit vektör, makine

öğrenmesi yöntemleri ile görüntülerin sınıflandırılması amaçlı kullanılmıştır. Sınıflandırma işlemlerinde Apache Spark'ın MLlib kütüphanesi kullanılmıştır. Bu kütüphane içerisinde yer alan Naif Bayes, Karar Ağaçları ve Rastgele Orman yöntemleri kullanılarak Kaggle platformunda paylaşılan gemi görüntü verileri üzerinde deneysel çalışmalar gerçekleştirilmiştir. Bu tez çalışmasının amacı, gemi görüntüleri üzerinde öznitelik çıkarımı yöntemleri ile elde edilen hibrit vektör ile Apache Spark'ın MLlib kütüphanesi kullanılarak sınıflandırma yapmaktır. Deneysel çalışmaların sonuçları grafik ve çizelgeler ile sunularak detaylı bir şekilde analiz edilmiş ve tartışılmıştır.

Anahtar Sözcükler : Büyük veri, öznitelik çıkarımı, makine öğrenmesi, sınıflandırma, görüntü işleme ve Apache Spark.

Bilim Kodu : 92414

ABSTRACT

M. Sc. Thesis

ANALYSIS OF LARGE DIMENSIONAL IMAGES USING APACHE SPARK

Betül DOLAPCI

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Caner ÖZCAN
July 2020, 72 pages

Today's digital transformation process and easy access to the internet due to the globalization have enabled high-volume and all kinds of data (image, sound, video, text etc.) to be produced. Due to reasons such as the dimension, irregularity and diversity of the produced data, analysis and feature extraction on the data becomes more and more difficult. Dividing the image data into smaller blocks and representing them with a vector with distinctive and independent properties facilitates the analysis process. For this reason, a block division method is applied first, dividing the image data into small pixel blocks. Large data is reduced in dimension and expressed in smaller blocks is realized with the feature vector. In this study, a hybrid feature vector was created for the analysis of the image we use as the data type. The hybrid vector created by using the sub-features extracted from the color and texture features of the images was used for the classification of the images with machine learning methods. The MLlib library of Apache Spark was used for classification. Experimental studies were carried out on the ship images shared on the Kaggle platform using Naive Bayes,

Decision Trees and Random Forest methods in this library. The purpose of this thesis is to classify the ship images with the hybrid vector obtained by the feature extraction methods using the MLlib library of Apache Spark. The results of the experimental studies are analyzed and discussed in detail by presenting them with graphs and tables.

Key Word **:** Big data, feature extraction, machine learning, classification, image processing and Apache Spark.

Science Code : 92414

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren, bu satırlarla ifade edilemeyecek saygıyı hak eden sayın hocam, Dr. Öğr. Üyesi Caner ÖZCAN'a sonsuz teşekkürlerimi sunarım.

Tez yapımı ve yazımında her zaman birlikte olduğum, yanımdan eksik olmayan enerjisi hiç bitmeyen Faruk ÇAKMAKLI'ya ve çok değerli dostum olan, teori ve bilgileri ile zihnimi sürekli açık tutan sevgili Yakup TAÇYILDIZ'a teşekkür ederim.

Aynı zamanda bu tez çalışmamızı "FYL-2019-2044" proje numarası ile desteklemeye layık gören Karabük Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne teşekkürlerimi sunarım.

Sevgili aileme manevi hiçbir yardımı esirgemeden yanımda oldukları için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

μ : Ortalama

σ : Standart Sapma

KISALTMALAR

ALS : Alternating Least Squares (Alternatif En Küçük Kareler Yaklaşımı)
API : Application Programming Interface (Uygulama Programlama Arayüzü)
CART : Classification and Regression Trees (Sınıflandırma ve Regresyon Ağaçları)
CCM : Color Co-occurrence Matrix (Renk Eş-Oluşum Matrisi)
CD : Compact Disk (Yoğun Disk)
CIE : Commission Internationale de l'Eclairage (Uluslararası Aydınlatma Komisyonu)
DAG : Directed Acyclic Graph (Yönlü Düz Graflar)
DOG : Difference of Gaussians (Gaussian'ın Farkı)
DSÖ : Dünya Sağlık Örgütü
DT : Decision Tree (Karar Ağacı)
DVM : Destek Vektör Makineleri
FS : First Order Statistics (Birinci Dereceden İstatistikler)
GBT : Gradient Boosted Trees (Gradyan Artırma Ağaçları)
GLCM : Gray Level Co-Occurrence Matrix (Gri Seviye Eş-Oluşum Matrisi)
GNU : GNU's Not Unix (GNU Unix Değildir)
GUI : Graphic User Interface (Grafik Kullanıcı Arayüzü)
HDFS : Hadoop Distributed File System (Hadoop Dağıtılmış Dosya Sistemi)
HOG : Histogram of Gradients (Gradyanların Histogramı)
HSL : Hue Saturation Lightness (Ton Doygunluk Parlaklık)
HSV : Hue Saturation Value (Ton Doygunluk Değer)
IBM : International Business Machines (Uluslararası İş Makineleri)
JVM : Java Virtual Machine (Java Sanal Makinesi)
KNN : K-Nearest Neigbour (K-En Yakın Komşuluk)
LDA : Latent Dirichlet Allocation (Gizli Dirichlet Ayrımı)
LOG : Laplace of Gauss (Gauss'un Laplası)

LSD : Line Segment Detector (Çizgi Segmenti Bulucu)

ML : Machine Learning (Makine Öğrenmesi)

MLLIB : Machine Learning Library (Makine Öğrenmesi Kütüphanesi)

NB : Naive Bayes (Naif Bayes)

RDD : Resilient Distributed Datasets (Esnek Dağıtılmış Veri Kümeleri)

RF : Random Forest (Rastgele Orman)

RGB : Red Green Blue (Kırmızı Yeşil Mavi)

SAR : Sentetik Açıklıklı Radar

SARS-COV-2 : Severe Acute Respiratory Syndrome Coronavirus 2 (Şiddetli Akut Solunum Yolu Sendromu Koronavirüsü 2)

SD : Standard Deviation (Standard Sapma)

SIFT : Scale Invariant Feature Transform (Ölçek Değişmez Özellik Dönüşümü)

SURF : Speeded Up Robust Features (Hızlandırılmış Sağlam Özellikler)

SQL : Structured Query Language (Yapılandırılmış Sorgu Dili)

BÖLÜM 1

GİRİŞ

Veri, yaşanılan zaman itibari ile çağımıza yön veren en değerli kavramların başında gelmektedir. İnternetin ortaya çıkmasıyla günümüzde insanların internet üzerinden birbirleri ile etkileşimi ve nesnelerin birbiriyle iletişimi sayesinde, büyük veri denilen kavram ortaya çıkmıştır. Birçok türde ve boyutta yüksek hacimli verilerin meydana getirdiği büyük veriyi analiz etmek ve verimli sonuçlar alabilmek günümüzün en önemli problemlerinden biridir. Veri türünün büyük boyutlu görüntü olması, veriyi analiz etmeyi daha çok zorlaştırmaktadır. Piksellerin ifade ettiği değerlerin sayısallaştırılması, kirli görüntü verilerinin gürültü giderimi sonrası görüntü işleme teknikleri ile analiz edilmeye uygun hale getirilmesi işlemleri çok kapsamlıdır. Bu nedenle, görüntüyü, kendisini ifade edebilen daha az boyutta bir veri ile temsil etmek faydalı bir ön adımdır.

Öznitelik vektörü, görüntüden elde edilen ayırt edici ve bağımsız değişkenlerin oluşturduğu, her görüntüye özgü, sayısal veri kümeleridir. Her bir vektör, sahip olduğu ayırt edici sayısal verilerle ait olduğu görüntüyü temsil etmektedir. Vektörler, görüntülerden çok farklı yollarla ve farklı içeriklerden çıkarılabilmektedir. Renk, şekil ve doku içeriklerinden elde edilen özellikler öznitelik vektöründe birleştirilir.

Bu çalışmanın amacı, büyük boyutlu görüntüler üzerinde öznitelik çıkarımı işlemleri gerçekleştirerek, hibrit bir öznitelik vektörü ortaya çıkarılması, büyük boyutlu görüntü üzerinde doğru sonuçların elde edilmesini sağlayacak makine öğrenmesi yöntemlerinin tespiti ve sınıflandırma sonuç analizlerinin Apache Spark teknolojisi yardımıyla daha hızlı elde edilmesidir. Analiz sürecindeki ilk ve en önemli işlem, görüntü verilerinin ön işleme adımından geçirilerek, bazı görüntü işleme teknikleri ile analiz edilip, renk ve doku özelliklerinden elde edilen öznitelik vektörü oluşturulma adımıdır. Bu öznitelikler istatistiki hesaplamalar sonrası olabilecek en uygun biçimde

art arda getirilerek hibrit bir öznitelik vektörü geliştirilmiştir. Geliştirilen vektörler, makine öğrenmesi yöntemleri ile Apache Spark üzerinde görüntülerin sınıflandırılması amaçlı kullanılmıştır. Makine öğrenmesi yöntemleri olarak Naif Bayes, Karar Ağaçları ve Rastgele Orman yöntemleri tercih edilmiştir.

Öznitelik vektörleri ile alınan deneysel bulgular ve sonuçlar, son kısımda çizelgelerle belirtilmektedir. Analiz süreci boyunca geçen işlem süreleri ise ayrıca grafiklerle sunulmaktadır.

BÖLÜM 2

BÜYÜK VERİ

İnsanlık tarihi boyunca her zaman var olan ve aslında yeni bir terim olmayan veri, çağın teknolojik gelişimi ve dijitalleşme süreci ile çok farklı bir boyut kazanmıştır. Tarih boyunca insanlar kendilerine gereken bilgileri, ihtiyaç duydukları belgeleri ve verileri farklı yöntemlerle kaydetmiştir. Verileri analiz etmek istediklerinde, istatistiki açıdan sonuçlar almak veya farklı açılardan sonuçlar elde etmek için hesaplamalar yapmışlar ve veriyi değerlendirmişlerdir.

Günümüzde, internetin yaygınlaşması, insanlar açısından kolay erişilir olması ve nesnelerin birbirine internet ile bağlanması sonucu, büyük veri denilen veri yığınları ile karşı kaşıya kalınmaktadır. Gün geçtikçe katlanarak artan büyük veri, depolaması ve analiz edilmesi karmaşık hale gelen ve zorlaşan bir süreç olarak karşımıza çıkmaktadır.

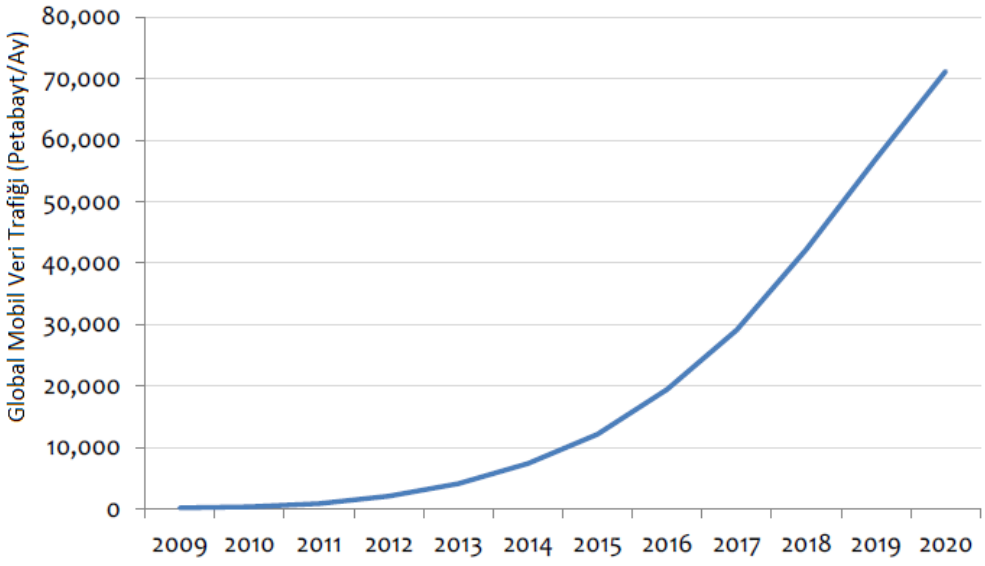
2.1. BÜYÜK VERİNİN ÖZELLİKLERİ

Büyük verinin 4 temel özelliği bulunmaktadır. 4V olarak geçen bu özellikler, hacim (volume), hız (velocity), çeşitlilik (variety), düzensizlik ya da karmaşıklık (veracity) gibi İngilizce terminolojisinde karşılık bulmuş kelimeler ile ifade edilmektedir. Bu özellikler her yerde evrensel olmayıp çoğunlukla 5V olarak da tanımlanabilmektedir. Buradaki beşinci V, değer (value) anlamına gelip veriyi değerli olması açısından ele almamızı sağlamaktadır. Fakat büyük veri, tanımı itibariyle değerli bir argüman olduğundan çoğu literatür tanımlamayı 4V özelliği ile sınırlı bırakmaktadır.

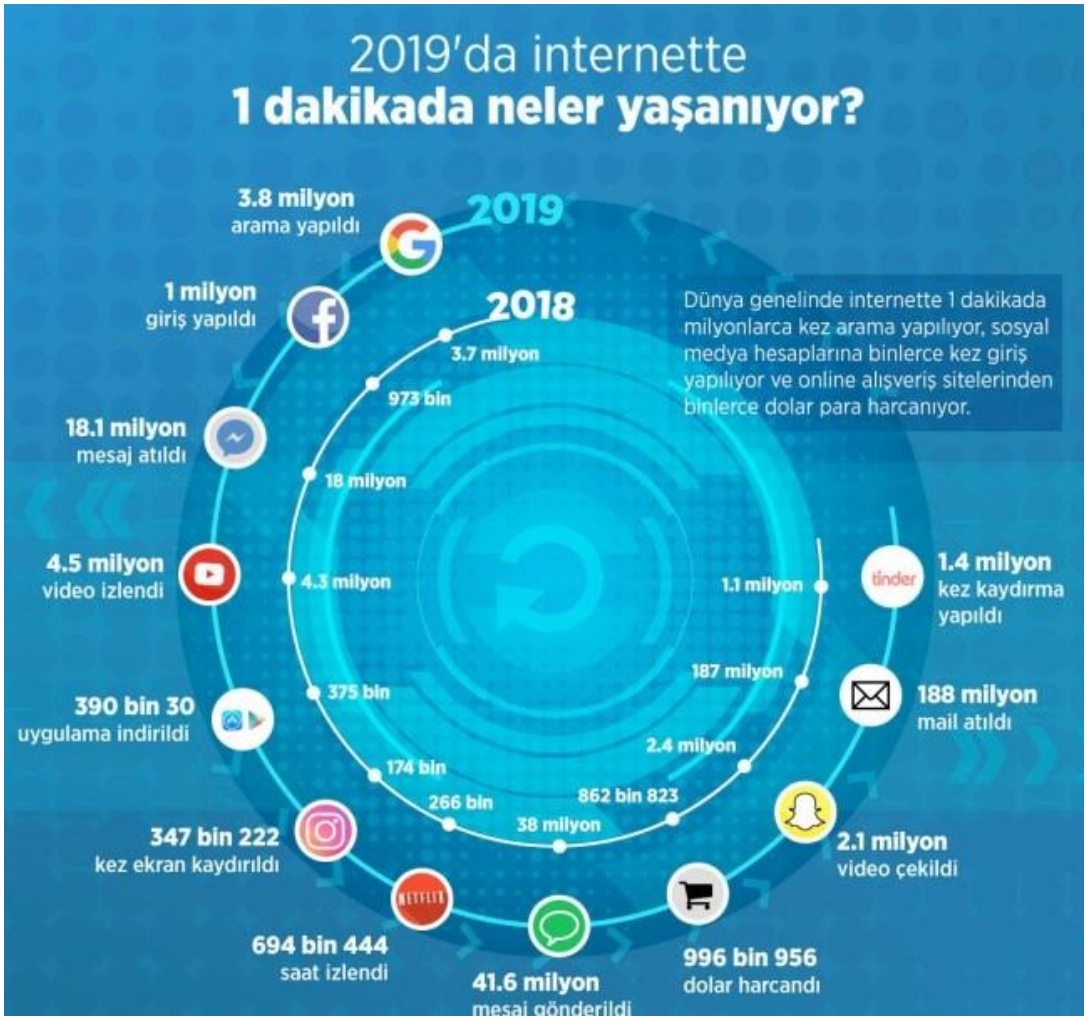
Hacim denilen özellik, verinin yüksek boyutlu olduğunu belirtir. Sosyal medyada, akıllı ev sistemlerinde kullanılan sensörlerde, uzay keşif araçlarının bilgisayar ve kamera sistemleri gibi birçok entegre kartlar dahilinde her saniye terabaytlarca veri

üretilmektedir. Nitekim Alman film yönetmeni Werner Herzog, 2016 yapımı "Lo and Behold" filminde verinin yüksek hacimli olmasından şöyle bahsetmektedir: Filme göre, 90'lı yılların başlarında yaygın olarak tercih edilen yoğun disklere (Compact Disk - CD), bugün 1 günde üretilen veri kaydedilmek istense, verilerin kaydedildiği bu CD'ler üst üste konarak buradan Mars'a kadar ulaşabileceği gerçeğinin doğru olmasıdır [1]. Bu gerçek, petabaytlarca veriyi temsil etmektedir.

Nesnelerin interneti ve insan etkileşimi her saniye petabaytlarca verinin üretilmesine neden olmakta ve veriler giderek daha büyük hacimlere ulaşmaktadır. Bu süreç, verilerin depolanması için gerekli alanın artmasına neden olmaktadır. Artan alan demek, büyük veri analizinde depolama birimlerine bağlı olarak dezavantaj demektir. Bu veriler analiz edilmek istediğinde verilerin depolandığı birimler birden fazla ve birbirlerinden bağımsız oldukları için, bu depolama birimlerinin birbirleri ile iletişimi özellikle analiz sırasında önem kazanmaktadır. Alanın artması, depolanan verilerin incelenmesi gerektiğinde, günler hatta aylar sürecek bir zaman dilimini kapsamaktadır. Şekil 2.1'de belirtildiği üzere her yıl mobil veri trafiği, önceki yıllara göre katlanarak artış göstermektedir. Şekil 2.2'de, 2019 yılında bir dakika içinde üretilen veri hacmi ve türleri belirtilmiştir. Bu problemi aşmak için, verilerin dağıtık sistemlerde paralel işlenmesi çözümü önerilmektedir. Bu çözüm kümeleme mimarisinin anlatıldığı Bölüm 3'te Apache Spark başlığı altında ayrıntılı olarak ele alınmaktadır.

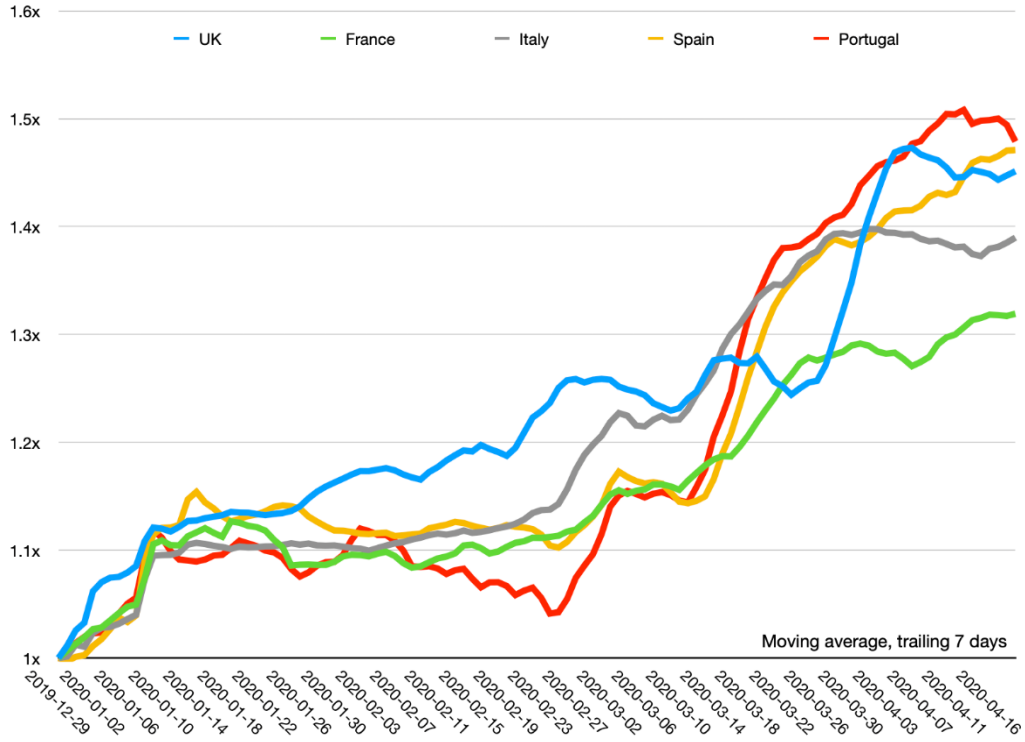


Şekil 2.1. Mobil veri trafiğindeki büyümenin yıllara göre değişimi [2].



Şekil 2.2. 2019 yılında bir dakika içinde üretilen veri hacmi [3].

Hız özelliği, verinin nesnelerin interneti ile beraber çok daha kısa sürelerde üretilmesi ve yayılması demektir. Sosyal medya hesaplarının çeşitliliği mobil teknolojilerin gelişimi ile birlikte, insanlar paylaşmak istedikleri bilgileri internet aracılığıyla bağlantılı olduğu herkese gönderebilmektedir. Dünya üzerinde internet bulunmayan birkaç izole alan dışında tüm alanlar sürekli veri üretmektedir. Gelişen teknoloji ile yüzlerce insana aynı anda veri göndermeye yarayan platformlar sayesinde, paylaşılmak istenen veri milyonlarca hatta milyarlarca insana ulaşabilmektedir. Twitter takipçisi milyon üzerinde bulunan hesaplar buna örnek verilebilmektedir. Akıllı ev ve akıllı şehir sistemlerinde kullanılan sensörlerde dakikalar hatta saniyeler içinde hızla üretilen bu verilerin analizinin de orantılı bir hızla sonuçlanması gereklidir. Şekil 2.3'te koronavirüs pandemisi sonrası internet trafiğindeki hız artışı gösterilmektedir.



Şekil 2.3. Koronavirüs sonrası internet kullanım yoğunluğu [4].

Çeşitlilik özelliği, verinin belirli bir yapısının olmadığını ve değişkenliğini temsil eder. Üretilen veriler, ham veri de olabilmekte, yapılandırılmış veri de olabilmektedir. Görüntüler, ses dosyaları, text dosyaları örnek olarak verilebilir. Büyük veride, veriyi anlamlandırma sürecinde yüksek başarım elde etmek için çeşitlilik özelliği büyük önem taşımaktadır. Verinin ne tip veri olduğu, yapısı, düzenliliği, ham veri olup olmadığı gibi bilgiler bu açıdan çok önemlidir. Bu bilgiler başta belirlenerek, daha sonra karmaşıklığa yol açacak ve sonuçların alınması için gerekli zamanı uzatacak problemlerden uzak kalmamızı sağlamaktadır. Veri çeşidine göre kullanılması gereken makine öğrenmesi algoritmalarının belirlenmesi, analizin yapılacağı platformun seçilmesi ve kullanılacak programlama dilinin uygunluğu tamamen bu özelliğe bağlıdır.

Düzensizlik ya da karmaşıklık olarak bilinen özellik ise verinin kirli olması demektir. Bu özelliği örneklemek için bir hastanenin hasta bilgilerinin kayıt altında tutulduğu veritabanında bulunan eksik bilgiler düşünülebilir. Veritabanında hastaların ad, soyad, doğum tarihi, hastalık tanısı gibi birçok bilgi yer almaktadır. Günler ve yıllar geçtikçe yüzlerce hatta milyonlarca verinin biriktiği bu veritabanından daha iyi hizmet

verilmesi ve sağlık adına verilerin analiz edilmesi istendiğinde, eksik bilgiler nedeniyle anlamlı sonuçlar çıkarılması mümkün olmayacaktır. Hedeflenen yorumlar ve kazanımların elde edilmesi açısından verilerin veritabanına kaydedilme anı doğru biçimde kayıt altına alınması büyük önem arz etmektedir.

Bu duruma günümüzden örnek vermek gerekirse, Dünya 2020 yılı itibari ile büyük bir salgınla mücadele etmektedir. Şiddetli akut solunum yolu sendromu koronavirüsü 2 (SARS-CoV-2) olarak da bilinen yeni tip koronavirüs, Dünya Sağlık Örgütü (DSÖ) tarafından pandemi olarak ilan edilmiştir [5]. Fakat büyük veri sayesinde bu pandemi aslında DSÖ'nden önce Kanada'lı bir veri analiz şirketi olan Blue Dot tarafından tahmin edilmiştir. Onlarca farklı veri analiz edilerek Çin'in gösterildiği nokta atışı ile lokasyon verisini sunarak salgının başladığı yer bilinip, aynı zamanda salgının hangi ülkelerde yayılacağı da tahmin edilmiştir [6]. Büyük veri, özellikle bu gibi örnekler açısından hayati önem arz etmektedir. Birçok olumsuz durumun önceden tahmini sayesinde alınacak tedbirlerle olumlu bir yaşam ortamı yaratılabilir, iklim değişikliklerinin önüne geçilebilir, hastalık ve salgınlar durdurulabilir. Trafik kazaları arabalardaki sensörlerden elde edilen verilerden anlamlı bilginin üretilmesi ile engellenebilir.

Makine öğrenmesi ve yapay zeka teknolojilerinin de yardımı ile büyük veri sayesinde dijital dönüşümün dönüm noktalarından biri yaşanmaktadır. Yeni bir çağın başlangıcı olarak yorumlanabilecek büyük verinin kullanımı ile beraber bu verilerin depolanması, güvenlik ve mahremiyeti, düzensiz ve kirli verilerin düzenli hale getirilme işlemi gibi problemler ortaya çıkmaktadır. Süreç, veri madenciliği, bilgisayar bilimi, makine öğrenmesi, veri tabanı yönetimi, matematiksel algoritmalar ve istatistik gibi disiplinlerarası bir ekip çalışmasını gerektirmektedir. Özellikle yeni teknolojilerin daha yaygın hale gelmesiyle birbirinden farklı kanallar üzerinden hızlı bir şekilde üretilen ve yüksek hacme ulaşan veriler sunucularda depolanmaktadır. Bu durumda üretilen veri hacmi ile orantılı olarak depolama birimlerinin arttırılması gereklidir. Fiziksel belleklerin yetmediği bu problemin üstesinden gelmek için bulut bilişim dediğimiz teknoloji geliştirilmiştir. Verinin yüksek boyutlu olması problemi, depolama birimlerinin yetersizliği gibi çözülmesi gereken problemler dışında verinin güvenliği de oldukça önem arz etmektedir. Veri bilimcileri, iş analistleri ve bu alanda

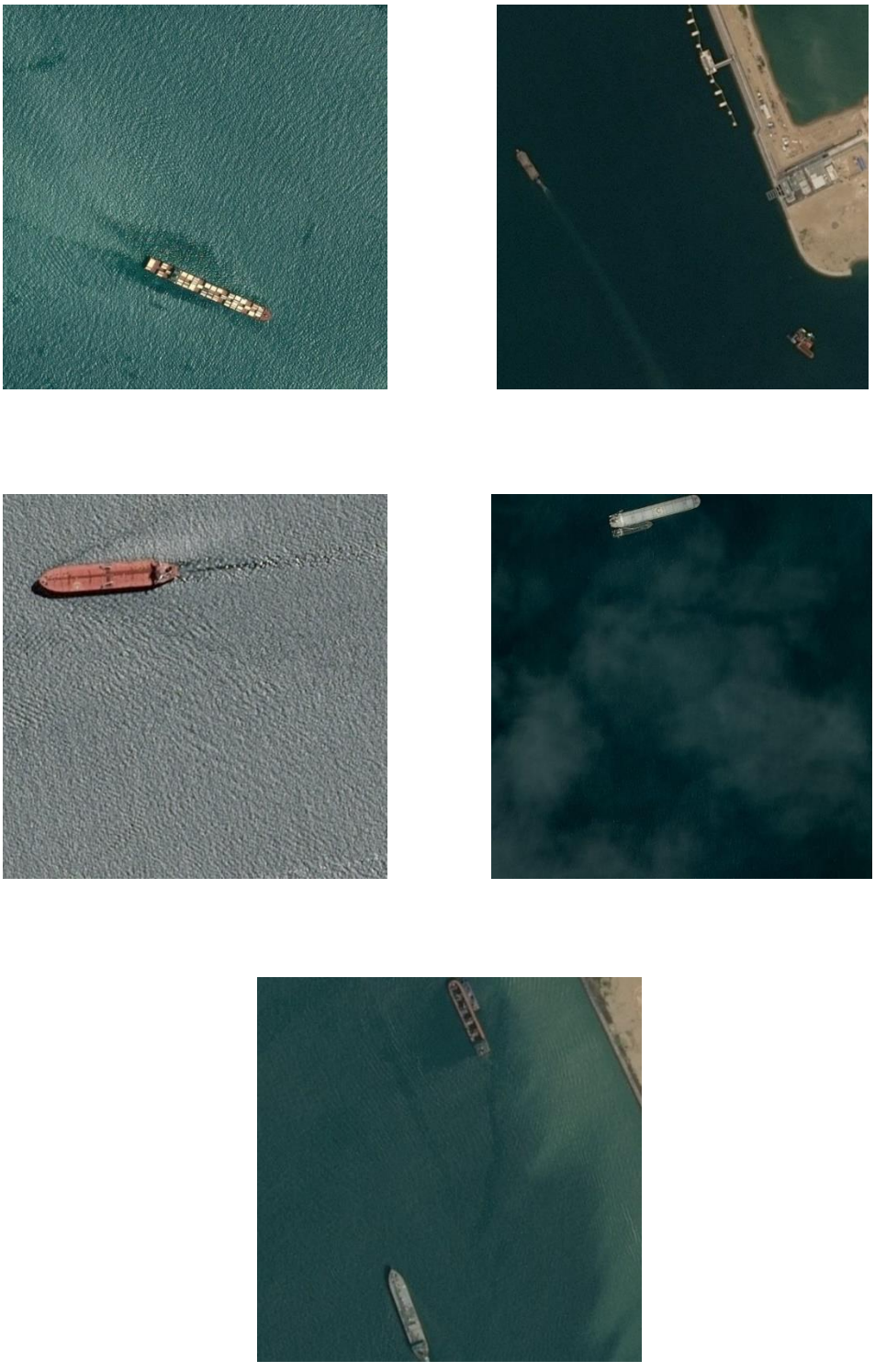
çalışmalar yapan diğer disiplinlerdeki araştırmacılar, sadece devasa değil aynı zamanda değişen ve çok hızlı biriken büyük veri ortamı için şifreli ve mahremiyet korumalı veri tabanı yönetim sistemleri ve ürünleri üzerine çalışmaktadır [7].

Büyük veriyi oluşturan veri türlerinden biri de büyük boyutlu görüntülerdir. Bu görüntülerin her biri tek başına birçok anlam çıkarılabilecek bilgiye sahiptir. Görüntü verisi piksellerden meydana gelmekte ve bu durum veri tipi olarak görüntünün daha karmaşık olmasına, analizinin daha farklı yöntemler gerektirmesine neden olmaktadır. Görüntü verisinin karmaşık yapısı piksellerin geniş aralıklarda seyreden frekans dağılımından kaynaklanmaktadır. Piksellerin sahip olduğu renk değerleri, doku perspektifleri, desen dağılımı, görüntünün gürültülü veya net olması birçok anlamı ifade edebilmekte ve bu durum görüntü verilerinin analizini zorlaştırmaktadır. Bu problemin üstesinden gelmek için büyük boyutlu görüntünün daha küçük boyutta sayısal değerlerle ifade edilmesi analiz açısından bir seçenektir. Görüntüyü ifade eden piksel değerlerinin sayısallaştırılması ve piksellerin matris olarak ifade edilmesi bu duruma örnek gösterilebilir.

2.2. BÜYÜK BOYUTLU GÖRÜNTÜ VERİSETİ

Airbus firması, kısa mesafeden çekilen gemi görüntülerini bir araya getirerek bir veriseti oluşturmuştur. Gemilerin gökyüzünden çekilen bu görüntüleri gemi güvenliğinden sorumlu firmalar için oldukça önemlidir. Gemi ile yapılan nakliye trafiğinin artması, çevreye zarar veren gemi kazaları, yasadışı kargo hareketleri ve illegal malzemelerin gemi ile ticareti gibi deniz ihlali ihtimallerinin takibi bu firma tarafından yapılmaktadır. Özel hizmet verilerini eğitimli veri analistleriyle paylaşarak güvenilir taşımacılık ve diğer gemi hareketleri için deniz takibinin yapılmasını sağlamaktadır [8]. Bu takipte denizdeki gemilerin tespiti ve gökyüzünden çekilen kısa mesafeli görüntülerde gemilerin sınıflandırılması işlemi önemlidir. Bu çalışmada, Airbus firmasının Kaggle platformunda sunduğu kısa mesafeli görüntü verileri [9] üzerinde tespit ve sınıflandırma amaçlı analiz çalışılmıştır. Görüntü veriseti içerisinden seçilen görüntülerle yeni bir alt veriseti kümesi oluşturulmuştur. Bu görüntüler, deniz üzerinden çekilmiş gemilerin olduğu veya olmadığı deniz görüntülerinden meydana gelmektedir. Verisetinde kara alanlarının ve daha farklı

mekansal dağılımı bulunan görüntüler de mevcuttur fakat bu çalışmada sadece, deniz üzerinden çekilmiş kısa mesafeli görüntüler ile çalışılmıştır. Şekil 2.4'te görüntü verisetinden beş örnek görüntü paylaşılmıştır.



Şekil 2.4. Görüntü verisetinden örnekler [9].

BÖLÜM 3

APACHE SPARK TANIMI VE MİMARİSİ

Apache Spark, yüksek hacimli verilerin oluşturduğu büyük veri kümeleri üzerinde paralel işlem yapılmasını sağlayan Scala ile geliştirilmiş açık kaynak kodlu bir kütüphanedir. Kaliforniya Üniversitesi'nden Matei Zaharia tarafından geliştirilmiştir [10]. Apache Spark, çok büyük veri kümelerinde analiz görevlerini hızlı bir şekilde gerçekleştirebilen bir veri işleme çerçevesidir ve ayrıca veri işleme görevlerini tek başına yapabilmekte veya diğer dağıtılmış bilgi işlem araçlarıyla birlikte birden çok bilgisayara dağıtabilmektedir. Bu iki özellik, büyük veri depoları ile uğraşmak için devasa bilgi işlem gücünün birleştirilmesini gerektiren büyük veri ve makine öğrenimi dünyaları için anahtardır.

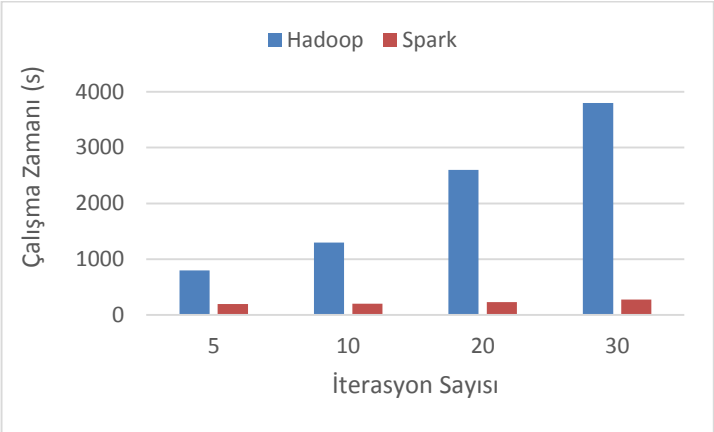
2009 yılında Berkeley'de geliştirilen Apache Spark, dünyadaki en büyük veri dağıtımı işlem çerçevelerinden biri haline gelmiştir. Spark Java, Scala, Python ve R programlama dilleri için yerel bağlamalar sağlamakta ve SQL, veri akışı, makine öğrenimi ve grafik işlemeyi desteklemektedir. Apache Spark bankalar, telekomünikasyon şirketleri, oyun şirketleri, hükümetler ve Apple, Facebook, IBM, Microsoft gibi tüm büyük teknoloji devleri tarafından kullanılan bir teknolojidir.

3.1. APACHE SPARK KÜMELEME SİSTEMİ

MapReduce ve benzer teknolojiler, ham veri kümelerinde analiz uygulamaları gerçekleştirmede oldukça başarılı olmuştur. MapReduce bir çevrimdışı veri akışı modeli üzerine inşa edilmiştir. Matei Zaharia'nın MapReduce kullanarak geliştirdiği sistem, birden çok paralel işlemde çalışan bir veri kümesini yeniden kullanabilme sürecidir. Bu, etkileşimli veri analiz araçlarının yanı sıra birçok yinelemeli makine öğrenme algoritmasını içermektedir. MapReduce'un ölçeklenebilirliğini ve hataya dayanıklılığını korurken, bu uygulamaları destekleyen Spark adında yeni bir çerçeve

Zaharia tarafından önerilmiştir. Bu hedeflere ulaşmak için Spark, esnek dağıtılmış veri kümeleri (Resilient Distributed Datasets-RDD) adı verilen bir soyutlama sunmaktadır. RDD, bir bölüm kaybolursa yeniden oluşturulabilen bir dizi makine arasında bölünmüş salt okunur bir nesne koleksiyonudur [10]. Apache Spark'ın birden fazla bağımsız alanda depolanan verileri paralel olarak işleme teknolojisinde RDD mimarisinin payı büyüktür. Böylece dağıtık sistemde veriler senkronize işlenirken analiz için gerekli zaman daha da azalarak işlem zamanı kısalmaktadır.

Spark, MapReduce yöntemine alternatif olarak geliştirilmiştir. Yinelemeli makine öğrenme işlerinde Hadoop'tan 10 kat daha iyi performans gösterebilmekte ve 39 GB'lik bir veri kümesini ikinci saniyenin altında yanıt süresiyle etkileşimli olarak sorgulamak için kullanılabilmektedir. Şekil 3.1'de Spark ve Hadoop arasındaki performans farkı gösterilmektedir. MapReduce'a benzer ölçeklenebilirlik ve hataya dayanıklılık özellikleri sağlarken, çalışma kümeleriyle uygulamaları destekleyen Spark, yeni bir küme bilgi işlem çerçevesi sunmaktadır [10].



Şekil 3.1. Apache Spark ve Hadoop performans farkı [10].

Spark'ın sunduğu bu yeni küme bilgi işlem modeli yaygın kullanılır hale gelmiştir. Burada yerellikten bağımsız zamanlama, hata toleransı ve yük dengelemesi sağlayan sistemler tarafından güvenilmez makinelerin kümelerinde veri paralel hesaplamaları yürütülmüştür. MapReduce bu modele öncülük ederken, Dryad ve Map-Reduce-Merge gibi sistemler desteklenen veri akışı türlerini genelleştirmiştir. Bu sistemler, ölçeklenebilirlik ve hata toleranslarına, kullanıcının girdi verilerini bir dizi

operatörden geçirmek için çevrimsel olmayan veri akış grafikleri oluşturduğu bir programlama modeli sağlayarak ulaşmaktadırlar. Bu, temel sistemin zamanlamayı yönetmesine ve kullanıcı müdahalesi olmadan hatalara tepki vermesine olanak tanımaktadır [10].

Bu veri akışı programlama modeli büyük bir uygulama sınıfı için yararlı olsa da döngüsel olmayan veri akışı olarak verimli bir şekilde ifade edilemeyen uygulamalar vardır. Bu uygulamalar birden çok paralel işlemde çalışan bir veri kümesini yeniden kullanma mantığı ile tasarlanmış süreçlerden oluşan uygulamalardır. Bu, Hadoop kullanıcılarının MapReduce'un eksik olduğunu bildirdikleri iki kullanım durumunu içermektedir:

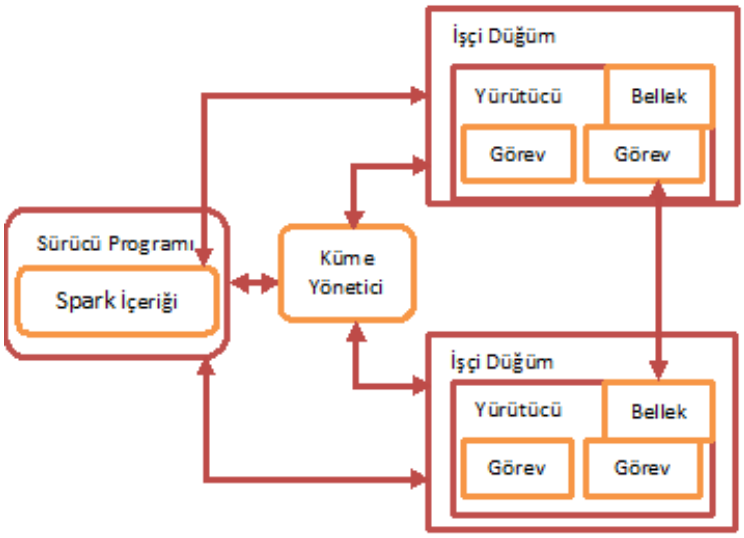
1. Bir çok yaygın makine öğrenme algoritması, bir parametreyi optimize etmek için aynı veri kümesine tekrar tekrar bir işlev uygular (örneğin, degrade iniş yoluyla). Her yineleme bir MapReduce / Dryad işi olarak ifade edilebilirken, her işin verileri diskten yeniden yüklemesi gerekir ve bu da önemli bir performans artışına neden olur.
2. Hadoop genellikle Pig ve Hive gibi SQL arabirimleri aracılığıyla büyük veri kümelerinde geçici keşif sorguları çalıştırmak için kullanılır. İdeal olarak kullanıcı bir dizi veri setini birden çok makinede belleğe yükleyebilir ve tekrar tekrar sorgulayabilir. Ancak Hadoop ile her bir sorgu ayrı bir MapReduce işi olarak çalıştığı ve diskten veri okuduğu için önemli gecikmelere (onlarca saniye) maruz kalır.

Zaharia, bu iki problemin üstesinden gelmek için MapReduce'a benzer ölçeklenebilirlik ve hataya dayanıklılık özellikleri sağlarken, çalışma kümeleriyle uygulamaları destekleyen Spark adında yeni bir küme bilgi işlem çerçevesini sunmaktadır [10]. Sunulan bu çerçevenin mimarisi sonraki başlıkta incelenmektedir.

3.2. APACHE SPARK MİMARİSİ

Spark'ı kullanmak için geliştiriciler uygulamalarının yüksek seviye kontrol akışını uygulayan ve paralel olarak çeşitli işlemleri başlatan bir sürücü programı yazarlar.

Spark, paralel programlama için iki ana soyutlama sağlar: esnek dağıtılmış veri kümeleri ve bu veri kümelerinde paralel işlemler (veri kümesine uygulanacak bir işlev iletilerek çağrılır). Buna ek olarak Spark, daha sonra açıklayacağımız kümede çalışan işlevlerde kullanılabilen iki kısıtlı paylaşılan değişken türünü destekler. Şekil 3.2'de Apache Spark mimarisi gösterilmektedir.

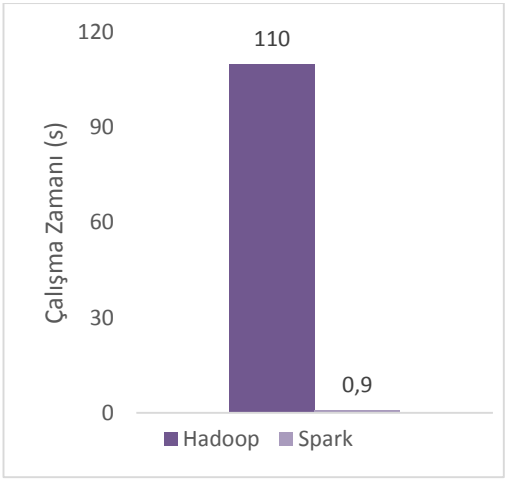


Şekil 3.2. Apache Spark mimarisi.

Apache Spark büyük veri işleme projelerinde oldukça sık kullanılmaya başlanmış bir platformdur. Bunun nedeni petabaytlarca verilerin dağıtık sistemlerde paralel olarak kısa süreler içinde işlenmesi ve gerekli sonuçların alınmasını sağlamasıdır. Spark'ın Hadoop'tan hızlı olmasının nedeni kümeleme sistemi kaynaklı mimari yapısıdır. Büyük veri analizi açısından önem arz etmesi, Yönlendirilmiş Döngüsüz Grafik (Directed Acyclic Graph - DAG) motoruna sahip olmasından ve bellek-içi (in-memory) veri işleme, analiz özelliklerini sağlamasındandır. Hadoop'tan daha performanslı olduğu bir diğer alan yapay öğrenme algoritmalarının dağıtık implementasyonu konusundaki performansıdır. Bu nedenle, Apache Mahout projesi Hadoop ile değil Spark üzerinde çalışacak şekilde geliştirilmeye devam etme kararı alınmıştır [11].

Spark bu gibi avantajlara sahip olsa da, Hadoop yerine her bakımdan tercih edilebilecek bir teknoloji değildir. Gerek kümeleme mimarisi gerek hız performansı

ile Hadoop'tan önde olmasına rağmen yapısı içinde büyük veriyi saklayacak depolama alanı açısından Hadoop sisteminden eksiktir. Petabaytlarca verinin depolanması da, bu çalışmanın büyük veri kısmında bahsedildiği üzere çok önemli bir konudur. Spark az hacimde veriyi bellek içi olarak analiz etmek ve çok yüksek hızda sonuç almak amacıyla mimarileştirilmiştir. Bu, yüksek hacimli veriler için önemli bir farktır. Spark, Hadoop Dağıtılmış Dosya Sistemi (Hadoop Distributed File System-HDFS) gibi herhangi bir depolama birimi sunmamakla beraber, HDFS üzerinden okuma yazma yapabilmektedir. Büyük boyutlu verilerin işlenmesi ile beraber depolanmasının da önem arz ettiği işler için Hadoop daha avantajlı olabilir. Bunun yanında hızlı sonuç almak için Spark tercih edilebilir. Bu tercih süreci yapılacak analizin boyutu, amacı, mimari avantaj ve dezavantajları düşünülerek gözden geçirilmelidir. Şekil 3.3'te Lojistik regresyon algoritmasının Hadoop ve Spark üzerinde çalıştırılması sonucu elde edilen performans gösterilmektedir.



Şekil 3.3. Lojistik regresyon algoritmasının Hadoop ve Spark performansı [10].

Temel düzeyde, bir Apache Spark uygulaması iki ana bileşenden oluşur: Bunlar; kullanıcının kodunu çalışan düğümlere dağıtılabilen birden fazla göreve dönüştüren bir sürücü ve bu düğümlerde çalışan ve kendilerine atanan görevleri yürüten yürütücüler olarak bilinmektedir. İkisi arasında aracılık yapmak için bir tür küme yöneticisi gereklidir.

Küme (cluster) yapısı dışında Spark, kümedeki her makinede Apache Spark çerçevesini ve Java Sanal Makinesi (Java Virtual Machine – JVM)'ni gerektiren

bağımsız tek başına bir küme modunda çalışabilir. Bununla birlikte, işçi makineler ve bir ana makineyi kapsayan daha sağlam bir kaynak veya küme yönetim sistemi modu temel çalışma modudur. Bu normalde Hadoop YARN (Cloudera ve Hortonworks dağıtımlarının Spark işlerini nasıl yürüttüğü) üzerinde çalışmak anlamına gelir. Ancak Apache Spark ayrıca Apache Mesos, Kubernetes ve Docker Swarm'da da çalışabilir. Apache Spark, kullanıcının veri işleme komutlarını Yönlendirilmiş Bir Döngüsel Grafik veya DAG içinde oluşturmaktadır. DAG, Apache Spark'ın zamanlama katmanıdır; hangi düğümlerde ve hangi sırada hangi görevlerin yürütüleceğini belirler [10].

3.2.1. Esnek Dağıtılmış Veri Kümeleri (RDDs)

Apache Spark'ın merkezinde bilgisayar kümesine bölünebilen değişmez bir nesne koleksiyonunu temsil eden bir programlama soyutlaması olan Esnek Dağıtılmış Veri Kümesi kavramı yer alır. RDD'ler üzerindeki işlemler de küme boyunca bölünebilir ve paralel bir toplu işlemde yürütülebilir, bu da hızlı ve ölçeklenebilir paralel işlemeye yol açar. Esnek dağıtılmış veri kümesi, bir bölüm kaybolursa yeniden oluşturulabilen bir dizi makine arasında bölünmüş salt okunur bir nesne koleksiyonudur. RDD öğelerinin fiziksel depolamada bulunması gerekmez; bunun yerine bir RDD tanıtıcısı, güvenilir depolama alanındaki verilerden başlayarak RDD'yi hesaplamak için yeterli bilgi içermektedir. Bu, düğümler başarısız olursa RDD'lerin her zaman yeniden oluşturulabileceği anlamına gelmektedir. RDD'ler basit metin dosyalarından, SQL veritabanlarından, NoSQL mağazalarından (Cassandra ve MongoDB gibi), Amazon S3 kovalarından ve çok daha fazlasından oluşturulabilir. Spark Core API'nin çoğu, geleneksel harita oluşturma ve işlevselliği azaltmanın yanı sıra veri kümelerini birleştirme, filtreleme, örnekleme ve toplama için yerleşik destek sağlayan bu RDD konsepti üzerine inşa edilmiştir.

Spark, bir Spark uygulamasını görevlere ayıran ve bu işi yapan birçok yürütücü işlem arasında dağıtan bir sürücü çekirdek işlemini birleştirerek dağıtılmış bir şekilde çalışır. Bu yürütücüler, uygulamanın gereksinimleri için gerektiği gibi ölçeklendirilebilir [12].

Spark'da her RDD bir Scala nesnesi ile temsil edilir. Spark, programcıların RDD'leri dört şekilde oluşturmasına izin vermektedir:

1. Hadoop Dağıtılmış Dosya Sistemi gibi paylaşılan bir dosya sistemindeki bir dosyadan.
2. Bir Scala koleksiyonunu (örneğin, Bir dizi) sürücü programında "paralelleştirerek". Bu işlem, koleksiyonun birden fazla düğüme gönderilecek birkaç dilime bölünmesi anlamına gelmektedir.
3. Mevcut bir RDD'yi dönüştürerek.
4. Mevcut bir RDD'nin kalıcılığını değiştirerek. Varsayılan olarak, RDD'ler tembel ve kısa ömürlüdür. Yani, bir veri kümesinin bölümleri paralel bir işlemde kullanıldıklarında (örneğin bir dosya bloğunu bir harita işlevinden geçirerek) isteğe bağlı olarak gerçekleşir ve kullanımdan sonra bellekten atılmaktadırlar [10].

3.2.2. Paralel İşlemler

RDD'lerde birkaç paralel işlem gerçekleştirilebilir:

1. reduce: Sürücü programında sonuç üretmek için ilişkilendirilebilir işlev kullanarak veri kümesi öğelerini birleştirir.
2. collect: Veri kümesinin tüm öğelerini sürücü programına gönderir. Örneğin, bir diziyi paralel olarak güncellemenin kolay bir yolu diziyi paralelleştirmek, eşlemek ve toplamaktır.
3. foreach: Her öğeyi kullanıcı tarafından sağlanan bir işlevden geçirir. Bu yalnızca işlevin yan etkileri için yapılır (bu, verileri başka bir sisteme kopyalamak veya paylaşılan bir değişkeni aşağıda açıklandığı gibi güncellemek olabilir) [10].

3.2.3. Paylaşılan Değişkenler

Programcılar ve geliştiriciler fonksiyonları (kapatmaları) kullanarak harita, filtre ve azaltma gibi işlemleri başlatırlar. Fonksiyonel programlamada tipik olarak bu kapaklar oluşturuldukları kapsamdaki değişkenleri ifade edebilir. Normalde, Spark bir çalışan düğümünde bir kapatma çalıştırdığında, bu değişkenler çalışana kopyalanır. Bununla

birlikte, Spark ayrıca programcıların iki basit ancak ortak kullanım şeklini desteklemek için iki kısıtlı paylaşılan değişken türü oluşturmasına izin vermektedir:

1. Yayın değişkenleri: Birden fazla paralel işlemde büyük bir salt okunur veri parçası (örneğin, bir arama tablosu) kullanılıyorsa, her kapanışta paketlemek yerine işçilere yalnızca bir kez dağıtılması tercih edilir. Spark, programcının değeri saran ve her çalışana yalnızca bir kez kopyalanmasını sağlayan bir "yayın değişkeni" nesnesi oluşturmasına izin verir.
2. Akümülatörler: Bunlar, çalışanların yalnızca ilişkilendirilebilir bir işlem kullanarak "ekleyebilecekleri" ve yalnızca sürücünün okuyabildiği değişkenlerdir. MapReduce'daki gibi sayaçları uygulamak ve paralel toplamlar için daha zorunlu bir sözdizimi sağlamak için kullanılabilirler. Akümülatörler "ekleme" işlemi ve "sıfır" değeri olan herhangi bir tip için tanımlanabilir. "Salt eklenebilir" semantikleri nedeniyle, hataya dayanıklı hale getirmek kolaydır [10].

Apache Spark veya Hadoop kullanılma farklarına göz atmak gerekirse, Apache Spark ve Apache Hadoop'un biraz yanlış isim olduğunu belirtmek gerekmektedir. Spark'ı bugünlerde çoğu Hadoop dağıtımında bulabilirsiniz. Ancak iki büyük avantajı nedeniyle Spark büyük verileri işlerken Hadoop'u öne çıkaran eski MapReduce paradigmasını geçerek tercih edilen çerçeve haline gelmiştir.

İlk avantajı hızdır. Spark'ın bellek içi veri motoru, belirli durumlarda özellikle durumların diske geri yazılmasını gerektiren çok aşamalı işlerle karşılaştırıldığında belirli durumlarda MapReduce'dan yüz kat daha hızlı görevler gerçekleştirebileceği anlamına gelir. Özünde, MapReduce veri haritalama ve azaltma işlemlerinden oluşan iki aşamalı bir yürütme grafiği oluştururken, Apache Spark'ın DAG'sinin daha verimli dağıtılabilen birden çok aşaması vardır. Verilerin bellek içinde tamamen içerilemediği Apache Spark işleri bile MapReduce muadillerinden yaklaşık 10 kat daha hızlıdır [13].

İkinci avantajı geliştirici dostu Spark Uygulama Programlama Arayüzü (Application Programming Interface – API)'dür. Spark'ın süreci hızlandırması kadar önemli olan Spark API'nın kolaylığının daha da önemli olduğu söylenebilmektedir.

3.2.4. Spark Çekirdeği

MapReduce ve diğer Apache Hadoop bileşenleri ile karşılaştırıldığında, Apache Spark API, geliştiriciler için çok kolay ve basit yöntem çağrılarının arkasında dağıtılmış bir işleme motorunun karmaşıklığının çoğunu saklamaktadır. Bunun kanonik örneği, bir belgedeki kelimeleri saymak için neredeyse 50 satır MapReduce kodunun Apache Spark'ın birkaç satırına (burada Scala'da gösterilmiştir) nasıl azaltılabileceğidir:

```
val textFile = sparkSession.sparkContext.textFile("hdfs:///tmp/words")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs:///tmp/words_agg")
```

Apyt Spark, Python ve R gibi veri analizi için popüler dillere ve daha kurumsal dostu Java ve Scala'ya bağlantı sağlayarak uygulama geliştiricilerinden veri bilimcilerine kadar herkesin ölçeklenebilirliğini ve hızını erişilebilir bir şekilde kullanmasına izin vermektedir [13].

3.3. SPARK MLLIB

Spark'ın dağıtılmış makine öğrenmesi kütüphanesi olan MLlib, sınıflandırma, regresyon, işbirlikçi öğrenme, kümeleme ve boyutsallık azaltma gibi ortak öğrenme ortamları için standart öğrenme algoritmalarının hızlı ve ölçeklenebilir uygulamalarından oluşmaktadır. MLlib, Spark'ın ölçeklenebilir makine öğrenimi kütüphanesidir [14]. Spark, hızlı bellek içi hesaplamayı ve verilerin yinelemeli sorgulanmasını desteklediği için makine öğrenimine iyi bir şekilde katkıda bulunmaktadır. Bunun yanı sıra, derin sinir ağlarını modelleme ve eğitme olanaklarını da içermektedir. Spark MLlib; Java, Scala ve Python'da, uygulama programlama arayüzü kullanımını sağlamakta, bu da görüntüden öznitelik çıkarımı ve sınıflandırma için OpenIMAJ kullanan mevcut bir Java uygulamasıyla entegrasyonu kolaylaştırmaktadır [15].

MLlib, Spark'ın makine öğrenimi (ML) kütüphanesidir. Amacı, pratik makine öğrenimini ölçeklenebilir ve kolay hale getirmektir. Yüksek düzeyde aşağıdaki gibi araçlar sağlar [16]:

1. ML Algoritmaları: Sınıflandırma, regresyon, kümeleme ve işbirlikçi filtreleme gibi yaygın öğrenme algoritmaları.
2. Özellik: Özellik çıkarma, dönüşüm, boyut küçültme ve seçim.
3. Boru Hatları: ML Boru Hatlarını oluşturma, değerlendirme ve ayarlama araçları.
4. Kalıcılık: Algoritmaları, modelleri ve boru hatlarını kaydetme ve yükleme.
5. Yardımcı programlar: Doğrusal cebir, istatistik, veri işleme, vb.

MLlib, verimli dağıtılmış öğrenme ve tahmini desteklemek için birçok optimizasyon içermektedir. Örneğin, ALS algoritması JVM çöp toplama yükünü azaltmak ve daha yüksek seviyeli doğrusal cebir işlemlerini kullanmak için engellemeyi dikkatli bir şekilde kullanmaktadır. Karar ağaçları, iletişim maliyetlerini azaltmak için verilere bağlı özellik ayrıklaştırması gibi PLANET projesinden (Panda ve diğerleri, 2009) birçok fikir kullanır ve ağaç toplulukları hem ağaçlarda hem de ağaçlarda öğrenmeyi paralel hale getirir. Genelleştirilmiş doğrusal modeller, işçi hesaplamaları için hızlı C++ tabanlı doğrusal cebir kitaplıkları kullanılarak gradyan hesaplamasını paralelleştiren optimizasyon algoritmaları aracılığıyla öğrenilir. Birçok algoritma etkili iletişim ilkellerinden yararlanır; özellikle ağaç yapılı toplama sürücünün darboğaz olmasını önler ve Spark yayını büyük modelleri hızla işçilere dağıtır [17]. Şekil 3.4'te Spark MLlib'in versiyonlar arasındaki hızlandırılması verilmiştir.

Şekil 3.4. Spark MLlib'in versiyonlar arasındaki hızlandırması [17].

Makine öğrenimi boru hatları genellikle bir dizi veri ön işleme, özellik çıkarma, model yerleştirme ve doğrulama aşamalarını içermektedir. Makine öğrenimi kütüphaneleri genelde, boru hattı yapımı için gereken çeşitli işlevler kümesi için yerel destek sağlamaz. Özellikle büyük ölçekli veri kümeleriyle uğraşırken, uçtan uca bir boru hattını bir araya getirme süreci, ağ ek yükü açısından hem emek yoğun hem de pahalıdır. Spark mimarisinden faydalanarak, bu gibi problemlerin üstesinden gelmek için, MLlib bir paket içerir. Spark.ml olarak adlandırılan bu paket, kullanıcıların standart bir öğrenmeyi değiştirmesine olanak tanıyan API'ler de dahil olmak üzere tek tip bir üst düzey API sağlayarak çok aşamalı öğrenme boru hatlarının geliştirilmesini ve ayarlanmasını basitleştirir [17].

Apache Spark'ın MLlib kütüphanesi altında sınıflandırma ve regresyon amaçlı çeşitli algoritmalar kullanılabilmektedir. Destek Vektör Makineleri, Naif Bayes, Karar Ağaçları, Rastgele Ağaçlar, Gradyanla Güçlendirilmiş Ağaçlar, öneri için kullanılan Alternatif En Küçük Kareler Yaklaşımı (Alternating Least Squares - ALS), Gauss Karışımları, Gizli Dirichlet Tahsisi (Latent Dirichlet Allocation - LDA) yöntemleri örnek olarak gösterilebilmektedir. Yine Spark MLlib altında yapay sinir ağları ve derin öğrenme yöntemlerinin kullanıldığı alternatif yaklaşımlar mevcuttur. Bu çalışmada, en yaygın kullanılan Naif Bayes, Karar Ağaçları ve Rastgele Orman yöntemleri kullanılmıştır.

BÖLÜM 4

ÖZNİTELİK ÇIKARIMI

Öznitelik çıkarımı, görüntünün anlamlandırılıp daha kısa sayısal ifadelerle temsil edilebilmesi için gerekli en önemli görüntü işleme tekniğidir. Görüntü verisinin nasıl elde edildiği analiz işlemlerine yön veren önemli bir adımdır. Görüntünün sentetik açıklıklı radar (SAR) veya optik uzaktan algılama görüntüsü olması farklı analiz işlemlerini gerektirir. Uzaktan algılama görüntüleri uydu sensörlerinden elde edilen yoğun hacimli verilerdir. İçerdiği zengin hiperspektral bilgi, görüntünün boyutunu arttıran en önemli özelliğidir. Uzaktan algılama görüntüleri yüksek hiperspektral çözünürlüklere sahiptirler [18]. Kısa mesafeden elde edilen renkli görüntü verilerinin daha az karmaşık yapıda olması, görüntünün analiz ve sınıflandırma işleminde avantaj sağlamaktadır [19]. Uydu görüntülerinden elde edilen büyük boyutlu verilerde nesne tanıma analizi için yerel öznitelik tabanlı algoritmalar kullanılmaktadır. Yerel öznitelik tabanlı algoritmalar, görüntünün üç temel içeriği olan şekil, renk ve doku özelliklerinden şekil özelliğine daha çok yer vermektedir. Şekil tabanlı öznitelik çıkarımlarında GrabCut [20] isimli otomatik bölütleme algoritmasından faydalanılmıştır [21]. Bölütleme algoritmaları, bir görüntüyü kendisi oluşturan parçalara veya nesnelere ayırmaktadır. Bölütleme, sayısal görüntü işleme teknikleri içerisinde uygulaması ve analiz süreci zor olan yöntemlerdendir.

Sınıflandırma, nesne tanıma, segmentasyon gibi analiz işlemlerinin birçoğunda kullanılan öznitelik çıkarımı ham ve büyük boyutlu veriden en önemli ayırt edici özellikleri alma işlemi olarak tanımlanmaktadır. Öznitelik çıkarımı, bir görüntünün şeklini, rengini ve dokusunu benzersiz şekilde tanımlayan parametre kümelerinden oluşmaktadır. Bu işlem sonucunda her görüntünün kendisini ifade eden bir öznitelik vektörü ortaya çıkarılmış olur [22]. Öznitelik vektörü ait olduğu nesneye dair birçok önemli bilgi içermektedir. Görüntü işleme açısından değerlendirildiğinde vektörler gradyan büyüklüğü, renk, gri tonlama yoğunluğu, kenarlar, alanlar, doku ve şekle dair

birçok bilgi yerine kullanılabilecek sayısal dizilerdir. Literatürde piksel tabanlı görüntü sınıflandırması için Gri Seviyesi Eşlenik Matrisi (Gray Level Co-occurance Matrix, GLCM) [23], Gabor Süzgeçleri [24,25] ve Yönlü Gradyan Histogramı (Histogram of Oriented Gradient, HOG) [26] yöntemlerinden elde edilen öznitelik vektörleri geliştirilmiştir [27]. Bir başka çalışmada, bazı öznitelik çıkarma teknikleri karşılaştırılarak görüntü sınıflandırma işlemlerinde tekniklerin başarısı değerlendirilmiştir [28]. İçeriğe dayalı görüntü analizleri için de öznitelik vektörleri büyük önem taşımaktadır. Görüntü içeriği renk, şekil ve doku olmak üzere üç temel görsel özelliği ifade etmektedir. Görsel özellikler içinde renk, görüntü için en önemli detayları vermesi nedeniyle öznitelik vektörünün önemli bir parçasını oluşturmaktadır [29].

Doku özellikleri çıkarılmasında GLCM ve Birinci Dereceden İstatistikler (First Order Statistics, FS) yaygın tercih edilen yöntemlerdir [30]. Literatürde görüntü doku analizi ve özellik algılama üzerinde çalışmaları ile bilinen GLCM yöntemi [23] en önemli doku özellik çıkarma tekniklerinden biridir. RASAT uydu görüntülerinin kullanıldığı bir başka çalışmada, çizgi ve doğru parçaları gibi parçaları çıkarmaya yarayan Çizgi Parçası Bulucu (Line Segment Detector, LSD) metodu kullanılmaktadır [31]. Gemi görüntüleri üzerinde öznitelik vektörü çıkarılarak analiz edilmiş çalışmalar literatürde mevcuttur. Analizler incelendiğinde, amaca yönelik birkaç noktaya odaklanılarak o noktalar üzerinden bir özellikler parametresi elde edilmeye çalışıldığı görülmektedir [32,33]. Farklı içeriklerden çıkarılan ve her biri kendi içerisinde sınıflandırma başarısı yüksek vektörlerin art arda getirilerek oluşturulması ile elde edilen bir öznitelikler vektörü, analiz edilecek görüntü için çok daha sağlam sonuçlar üretilmesini sağlamaktadır. Bu vektör, görüntüye dair ayırt edici tüm parametreleri içerdiğinden daha doğru sınıflandırma yapılmakta ve makine öğrenmesi daha verimli sonuçlar üretmektedir.

4.1. ÖZNİTELİK ÇIKARIMI YÖNTEMLERİ

Literatürde öznitelik çıkarımında kullanılmak üzere geliştirilmiş birden çok yöntem ve algoritma bulunmaktadır. Bunlardan biri Yönelimli Gradyanların Histogramı (Histogram of Gradient - HOG) algoritmasıdır. HOG algoritmasının kullanıldığı en

yaygın alanlardan biri nesne tanıma ve algılamadır. Nitekim algoritmanın ilk çıkış amacı, yaya tanıma sistemlerinde kullanılabilecek tanımlayıcılar üretmektir [34]. Nesne ve örüntü tanıma problemlerinde yüksek başarım elde etmesi öznitelik çıkarımı süresince tercih edilme sebebi olmuştur. HOG, nesne algılama amacıyla bilgisayarla görme ve görüntü işlemede kullanılan bir özellik tanımlayıcıdır. Bu yöntem, kenar yönlendirme histogramlarına ölçek değişmez özellik dönüştürme tanımlayıcılarına ve şekil bağlamlarına benzer. Ancak eşit aralıklı hücrelerin yoğun bir ızgarasında hesaplanması ve gelişmiş doğruluk için örtüşen yerel kontrast normalleştirmesi kullanması bakımından farklılık göstermektedir. Wayland Research Inc.'den Robert K. McConnell, ilk olarak HOG'un arkasındaki kavramları 1986'da patent başvurusunda HOG terimini kullanmadan açıklamıştır. 1994 yılında kavramlar Mitsubishi Electric Research Laboratories tarafından kullanılmıştır. Ancak, 2005 yılında Fransız Bilgisayar Bilimi ve Otomasyonu Araştırma Ulusal Enstitüsü (INRIA) araştırmacıları Navneet Dalal ve Bill Triggs'in, bilgisayar vizyonu ve örüntü tanıma konferansında (CVPR) HOG tanımlayıcıları ile ilgili ek çalışmalarını sunmalarından sonra giderek yaygınlaşmıştır. Bu çalışmada statik görüntülerde yaya tespitine odaklanmışlar, ancak o zamandan beri testlerini videolarda insan tespitini ve statik görüntülerdeki çeşitli yaygın hayvan ve araçlara dahil edecek şekilde genişletmişlerdir [35].

Bir diğer öznitelik çıkarımı yöntemi Renk Histogramları yaklaşımıdır. Renkli görüntülerde, farklı renk uzayları kanalları üzerinden öznitelik çıkarımı için yaygın olarak kullanılan yöntemlerdendir. Görüntüyü ele aldığı perspektif renk kanallarının ayrı ayrı her bir bileşenidir. Bir görüntüdeki renk kutularının frekans dağılım bilgisi, renk histogramını vermektedir [36]. Histogram, her türdeki piksel sayısını sayar ve her görüntü pikseli sadece bir kez okunarak ve histogramın uygun bölmesi arttırılarak hızla oluşturulabilmektedir. Renk histogramı çeviriye, görüntüleme ekseni etrafında dönmeye, küçük eksen dışı dönmeye, ölçek değişikliklerine ve kısmi oklüzyona nispeten değişmezlik göstermektedir. Literatürde, iki tür renk histogramı bulunmaktadır. Bunlar, küresel renk çubuk grafik ve yerel renk çubuk grafik türündeki histogramlardır. Renk histogramı, bir görüntüdeki her istatistiksel renk frekansını analiz eden global bir renk tanımlayıcı olarak önerilmektedir. Çeviri, döndürme ve görüş açısı değişikliği gibi problemleri çözmek için kullanılmaktadır. Yerel renk

histogramı görüntünün tek tek bölümlerine odaklanır. Yerel renk histogramı, küresel renk histogramlarında kaybolan pikselin uzamsal dağılımını dikkate alır. Renk histogramının hesaplanması kolaydır ve görüntüdeki küçük varyasyonlara duyarsızdır, bu nedenle görüntü veritabanının endekslenmesi ve alınması için çok önemlidir. Bu avantajların yanı sıra, iki büyük dezavantajla karşı karşıyadır [37]. İlk olarak, genel uzamsal bilgiler dikkate alınmaz. İkincisi, benzer renk dağılımına sahip iki farklı görüntü benzer histogramlara yol açtığından histogramın sağlam ve benzersiz olmaması, ışık açısından farklı pozlama ile aynı açıdan elde edilen görüntüler farklı histogramlar oluşturmaktadır [37].

Diğer bir öznitelik çıkarımı algoritması, Ölçek Değişmez Özellik Dönüşümü (Scale Invariant Feature Transform - SIFT), görüntülerdeki yerel özellikleri algılamak ve tanımlamak için bilgisayar görüşünde bir özellik algılama algoritmasıdır. Kanada'da British Columbia Üniversitesi tarafından patentlenmiş ve 1999 yılında David Lowe tarafından yayınlanmıştır [38]. SIFT, bir görüntünün, aydınlatma, döndürme ve ölçeklendirmeye karşı değişmeyen bölgesel özelliklerini belirleyip tanımlayan bir algoritmadır. Algoritmanın ilk olarak amacı, görüntü üzerindeki anahtar noktalarını (keypoints) bulmaktır. Bunun için, verilen görüntüye farklı ölçeklerde Gaussian filtresi uygulanır. Gaussian filtresi ile bulanık hale gelen bu görüntüler arasındaki farklar alınır. Farklı ölçeklerde alınan Gaussian farkının ekstremum noktaları anahtar noktalarını vermektedir [39]. Algoritmada dört temel adım bulunmaktadır. İlki, ölçek alanı tepe seçimidir. Nesneler belirli bir ölçekte anlamlıdır. Duvarda asılı bir uzay posteri tek başına anlamlı görünebilir. Fakat tüm samanyoluna birden bakılırsa bu poster görünürde yok olur. Görüntüye farklı ölçeklerde Gaussian filtresi uygulanmasının nedeni budur. İkinci adım, anahtar noktalarını doğru şekilde bulmaktır. Üçüncü adım anahtar noktalara yönlendirme atamaktır ve son adım anahtar noktaları yüksek boyutlu bir vektör olarak tanımlamaktır [39].

SIFT algoritması ölçekden bağımsız olma konusunda başarılı sonuçlar sergilerken bir yandan hesaplama maliyetinin fazla olması yavaş çalışmasına neden olmaktadır. Bu nedenle SIFT'in dönmeden etkilenmeme ve ölçekten bağımsız olma gibi özelliklerine sahip ve aynı zamanda hızlı çalışan bir algoritmaya olan ihtiyaç, Hızlandırılmış Sağlam Özellikler (Speeded Up Robust Features, SURF) algoritmasının

geliştirilmesini sağlamıştır [40]. SURF algoritması SIFT'in hızlandırılmış versiyonudur. SIFT'de Gauss'un Laplace'ı (LoG), Gauss'ların farkı (DoG) ile yaklaşım yapılırken SURF'de kutu filtre (box filter) yaklaşımı kullanılmaktadır.

Piksellerin uzamsal ilişkisini dikkate alan dokuyu incelemek için istatistiksel bir yöntem, gri düzey uzamsal bağımlılık matrisi olarak da bilinen gri düzey ortak oluşum matrisidir. GLCM fonksiyonları, bir görüntüde belirli değerlere sahip ve belirtilen bir uzamsal ilişkide piksel çiftlerinin ne sıklıkta meydana geldiğini hesaplayarak, bir GLCM oluşturarak ve daha sonra bu matristen istatistiksel ölçümler çıkararak bir görüntünün dokusunu karakterize eder [23]. Gri Seviye Eş-Oluşum Matrisi, gri düzey uzamsal bağımlılık olarak da bilinen piksellerin uzamsal ilişkisini dikkate alan doku özelliklerini incelemek için istatistiksel bir yöntemdir. Burada, yoğunluk değeri i olan bir pikselin j değeri olan bir piksele spesifik bir uzamsal ilişkide ne sıklıkta oluştuğunu hesaplayarak bir GLCM matrisi oluşturulur. GLCM, görüntüde iki pikselin belirli bir vektörle ayrıldığı frekanslardan oluşur. Matristeki dağılımın mesafeye ve açısal ya da pikseller arasındaki yatay, dikey, diyagonal, anti-diyagonal ilişki gibi yönlere bağlı olacağı GLCM özellikleri, öznitelik vektörü oluşturma için kullanılmaktadır. Bir görüntüdeki dokunun birçok istatistiksel özelliği, görüntüdeki gri düzeylerin piksel ilişkisinin ikinci sırasını temsil eden eş-oluşum matrisine dayanır. Birlikte ortaya çıkma matrislerinin çeşitli istatistiksel ve bilgi teorik özellikleri dokusal özellikler olarak işlev görebilir ve bu özelliklerle ilgili kısıtlamanın hesaplanması zahmetlidir ve görüntü sınıflandırması ve alımı için çok verimli değildir. Haralick [23], her biri Gri Seviye Birlikte Oluşma Matrisinden çıkarılan 28 çeşit dokusal özellik önermiştir [23]. Bu çalışmada; kontrast, korelasyon, enerji ve homojenlik olan dört dokusal öznitelik kullanılmıştır.

Gri Seviye Eş Oluşum Matrisi, doku analizi yöntemi, doku analizi için en sık kullanılan yöntemdir. Bununla birlikte, renk bilgisi ve gri-GLCM dokusal özellikleri arasındaki ilişki hakkında nispeten az şey bilinmektedir. Ek olarak, gri-GLCM dokusal özellikleri küresel olarak uyarlanabilir, ancak yerel olarak optimize edilmez. Shearer [41], geleneksel gri düzeyli doku analizinin eksikliklerinin üstesinden gelmek için renk dokusu analizinin kullanılmasını önermiştir. Tarımsal ürün işlemede, RGB, HSL, HSV ve L*a*b* gibi yaygın olarak kullanılan renk uzayları görüntü işlemede sıklıkla

kullanılmaktadır [42]. HSL, HSV ve L*a*b* renk uzayları, insan renk algısını çoğaltmak için geliştirilmiştir. Renk Birlikte Oluşumu Matrisi (CCM) yöntemi kullanılarak renk dokusu analizi, görünür spektrumda renk özelliklerinin kullanılmasının geleneksel gri düzey gösterime göre ek görüntü özellikleri sağladığı hipotezine dayanır. Birçok çalışma biyosensing teknikleri için doku analizi kullanmanın yararını kanıtlamıştır [43]. Literatürde bir çalışmada, biyo-ürün algılama teknolojisi için CCM'ye dayalı doku analiz yazılımı geliştirilmiştir [43]. Bu çalışmada, GLCM özellikleri ve renk özellikleri ile öznitelik çıkarımı yapılmıştır.

4.2. HİBRİT ÖZNİTELİK VEKTÖRÜ

Önişleme safhasından geçirilen görüntüler daha temiz veriler olduğundan görüntüleri anlamlandırma işlemi daha doğru sonuçlar almamızı sağlayacaktır. Görüntü verileri daha düzenli ve analize uygun hale getirilmiştir.

4.2.1. Bloklara Ayırma

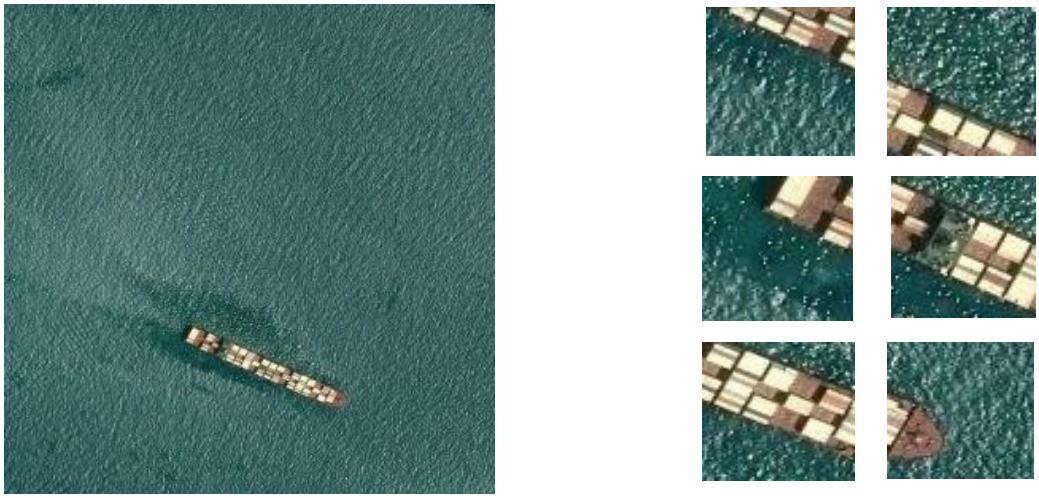
Blok temelli yaklaşım, piksel temelli yaklaşımın aksine daha anlamlı ve bütünsel analiz yapılmasını sağlar. Görüntünün renk ve doku içeriğine bağlı olarak daha homojen bilgi edinilmesini sağlayan bu yöntem ile vektörün hızlı oluşturulması da sağlanır. Bu çalışmada özellikler, blok bölümünün uygulanmasıyla elde edilmiştir. Bu blok bölümü, görüntüleri küçük boyutlu ve sabit boyutlu gemi olmayan ve gemi olan bloklara ayırır. Bir piksel yaklaşımına kıyasla, bu blok bölümü, sınıflandırma işleminin karmaşıklığını önemli ölçüde azaltır, çünkü sınıflandırılacak eleman sayısı önemli ölçüde azalır. Eleman sayısının azalması, büyük verinin boyut azaltımı açısından önemlidir. Bu çalışmada üç blok boyutu için analizler yapılmış ve karşılaştırmalar Sonuçlar kısmında çizelgeler halinde sunulmuştur. Süreç ile ilgili grafikler de çizelgeler ile beraber sunulmuştur. Çalışılan blok boyutları; 16x16 piksel, 32x32 piksel ve 64x64 pikseldir.

Analizi yapılacak verinin, artan hacmini hesaba katmakla beraber farklı blok bölümleri ile analizin tekrarlanması işlemi, verimli ve başarılı sonuçlar elde edebilmek içindir. Aynı görüntüler, farklı blok boyutları ile işlendiğinde, en iyi sonucu hangi blok boyutu

için vermekte, hangi blok bölümünün, analizin derinlemesine yapılmasına katkı sağlamakta gibi soruların cevabı alınmaktadır. Çalışmada genelde, en ufak blok bölümü ile daha kaliteli sonuçlar elde edilmektedir.

Görüntünün bloklara ayrılması, veri hacmini önemli ölçüde arttırmaktadır. 50 adet, 768x768 piksel boyutlarındaki görüntünün 16x16 piksellik bloklara bölünmesi demek, 115 200 alt görüntü ile analiz gerçekleştirmek demektir. Görüntülerin farklı blok boyutlarına ayrılması (32x32, 64x64 veya 128x128 gibi) ve görüntülerin sayısının artması ile (50 görüntünün 150, 300, 500 olarak artması gibi) hali hazırdaki verilerin hacmi katlanarak artmaktadır ve burada büyük verinin hacim özelliği ön plana çıkmaktadır.

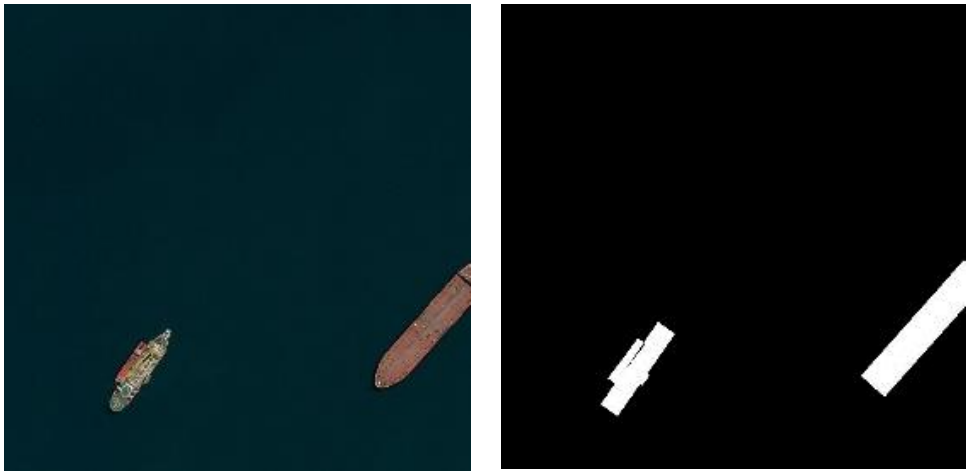
Görüntülerin bloklara ayrılma ve öznitelik çıkarımı işlemleri MATLAB ortamında gerçekleştirilmiştir. Eğitim verisi olarak kullanılacak görüntüler, öncelikle 16x16, sonra 32x32 daha sonra da 64x64 piksellik bloklara bölünerek, bir klasörde kayıt altına alınmıştır. Şelik 4.1'de bir görüntünün, 64x64 bloklara ayrılmış bazı kısımları gösterilmektedir.



Şekil 4.1. 64x64 piksel bloklara bölünmüş bir görüntü örneği.

Blok bölümünden sonra her blok için, etiketlendirme amaçlı ikili maske (binary mask) uygulaması gerçekleştirilmiştir. Bu yaklaşımın amacı, görüntüleri önceden gemi var veya gemi yok olarak etiketlemektir. Sonraki safhalarda gerçekleştirilecek

sınıflandırmanın doğruluğu, etiketli veri ile karşılaştırılarak yapılacaktır. Bu yüzden verileri doğru etiketlendirmek çok önemlidir. Gemi blokları, gemi içinde bulunan piksel bölgelerini temsil eder. Gemi dışı bloklar, su ve gökyüzü bölgeleri gibi gemi sınırlarının dışında bulunan piksel bölgelerinden oluşur. İkili maske (binary mask) oluşturulmasının nedeni, eldeki görüntü verilerinin etiketlendirilme işlemleri içindir. Görüntü verilerinde siyah bölgeler 0, beyaz bölgeler 1 ile etiketlendirilmiştir (Şekil 4.2.).



(a) (b)

Şekil 4.2: (a) Orijinal görüntü. (b) Görüntüye ikili maske uygulanmış hali.

4.2.2. Renk Öznitelikleri

Renkler, piksellerin görsel algısını, ışığa göre ton dağılımını tanımlar ve kromatik yoğunlukları hakkında bilgi verir. Bu yaklaşımda, RGB (Red-Green-Blue), HSV (Hue-Saturation-Value) ve CIE Lab olmak üzere üç farklı renk alanı değerlendirilmektedir.

RGB, insan görsel sistemine benzer çalışması nedeniyle bilgisayar grafiklerinde yaygın olarak kullanılan renk modeline dayanan bir renkli görüntü alanıdır. Bu modelde, ana renkler RGB bileşenlerinin her birinin değeri ile temsil edilen kırmızı, yeşil ve mavi renkliliklerle tanımlanır [19].

HSV, nesne tanıma ve görüntü segmentasyonu gibi uygulamalarla bilgisayar görme ve görüntü analizinde kullanılan bir renk alanıdır. HSV'nin ana avantajlarından biri, insan beyni tarafından gerçekleştirilene benzer şekilde yoğunluk ve renk bilgisi arasındaki ayrımdır. Ton (H), rengin gölgesini ve bu rengin renk spektrumunda bulunduğu konumu açıklar. Doygunluk (S), beyaz bir referansa göre renk tonunun saflığını temsil eder. Değer (V), hafifliğin bir ölçüsüdür [19].

CIE Lab renk alanı, farklı dalga boylarının insan algısına dayanmaktadır ve ortalama bir insan gözlemci tarafından algılanan herhangi bir rengi tanımlayabilir. CIE Lab, cihaza bağlı renkler olan RGB ve HSV ile karşılaştırıldığında, cihazdan bağımsız bir renktir. CIE Laboratuarı'nda, üç parametre bir küre ile temsil edilir. Dikey eksen L* hafifliği temsil eder. Yatay eksen a* kırmızı ve yeşil bileşenler arasındaki farkı ölçer ve yatay eksen b* mavi ve sarı bileşenler arasındaki farkı ölçer [19].

Renk uzaylarını öznitelik olarak kullanabilmek, her bir renk bileşeni için her bloktan ortalama ve standart sapma çıkarımı yapılmasına bağlıdır. Eşitlik 4.1'de ortalama Eşitlik 4.2'de standart sapma formülleri verilmiştir:

$$\mu = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} I(x,y)}{M\ x\ N} \tag{4.1}$$

$$\sigma = \sqrt{\frac{\sum_{x=1}^{M}\sum_{y=1}^{N} \left(I(x,y) - \mu\right)^2}{M\ x\ N}} \tag{4.2}$$

burada I(x,y), (x,y)'de bulunan pikselin renk bileşenidir, M, her bloğun piksel cinsinden genişliğidir ve N, her bloğun piksel cinsinden yüksekliğidir. Alt bloklara ayrılmış görüntüler, önceki kısımda bir klasöre kaydedilmişti. Bu kısımda, renk içeriğine ait öznitelik çıkarımı yapılması işlemleri gerçekleştirilecektir. Bu iş, MATLAB ortamında yazılan kod ile gerçekleştirilmiştir. Burada incelenen 3 renk uzayı için ayrı ayrı öznitelikler çıkarılmıştır.

Öncelikle, RGB renk uzayının, kırmızı, yeşil ve mavi renk bileşenleri aşağıdaki kod satırları ile bulunmuştur:

```
redChannel = image(:, :, 1);
greenChannel = image(:, :, 2);
blueChannel = image(:, :, 3);
```

Daha sonra bu renk kanallarının her birine ait standard sapma bulunmuştur:

```
sdR = std2(redChannel);
sdG = std2(greenChannel);
sdB = std2(blueChannel);
```

En son bu renk kanallarının her birine ait ortalama değeri hesaplanmıştır:

```
avgRGB = mean(reshape(image, [], 3), 1);
```

Bu işlemler sonucu, RGB renk uzayı için, 6x1 öznitelik vektörü elde edilmiştir. Yukarıdaki işlemler HSV ve CIE Lab renk uzayları içinde gerçekleştirilmiştir. Bu iki renk uzayından elde edilen 6x1 boyutundaki vektörler ilki ile birleştirilerek, 18x1 boyutunda son vektör elde edilmiştir.

4.2.3. Doku Öznitelikleri

Doku, bir bölgedeki piksellerin yapısını ve uzamsal niteliklerini temsil eden bir özelliktir. Doku, gri bir seviyede piksellerin yoğunluk özellikleri ve aralarındaki uzamsal ilişki ile karakterize edilebilir. Renk özelliklerinin aksine, doku özellikleri tek tek piksel özellikleri yerine bölge tabanlı özellikleri açıklar. Doku özelliklerini çıkarmak için görüntüler önce parlaklık bileşenini korurken renk tonu ve doygunluk bilgisini ortadan kaldırarak gri tonlamaya dönüştürülür. Bu dönüşümden sonra, her bloktan iki tür doku özelliği çıkarılır: birinci dereceden istatistikler ve Gri Seviye Eş-Oluşum Matrisleri. GLCM tabanlı doku analizinde, bu algoritmanın sağladığı bazı istatistik verilerden yola çıkılmıştır. İstatistiki veriler aşağıdaki çizelgede açıklanmaktadır. FS tabanlı istatistiki veriler ise hemen diğer çizelgedeki gibidir [19]:

Çizelge 4.1. GLCM istatistiki verileri (öznitelikler).

Öznitelik	Tanımı
Kontrast	Yoğunluk ve gri seviye varyasyonları
Korelasyon	Gri seviye değerleri lineer bağımlılığı
Enerji	Piksellerin homojenlik ölçütü
Homojenlik	Farklı bölgelerdeki benzerlik ölçütü

Bu çalışmada kullanılan FS öznitelikleri Çizelge 4.2'de belirtildiği gibidir. Bunun için öncelikle renkli görüntü blokları griye dönüştürülmüştür. Sonrasında ise, görüntünün her bir pikseli double değerle ifade edilecek bir matris yapısına dönüştürülmüştür.

Çizelge 4.2. FS istatistiki verileri (öznitelikler).

Öznitelik	Tanımı
Ortalama	Piksel değerleri ortalaması
Standart Sapma	Varyans bilgisinin karekökü
Varyans	Ortalamadan sapmanın karesi
Çarpıklık	Dağılımının asimetrisinin ölçütü
Entropi	Gri seviye uzaysal düzensizliği
Enerji	Piksellerin homojenlik ölçütü

Oluşturulan sayısal matris değerleri üzerinden Çizelge 4.2'deki 6 öznitelik her bloktan çıkarılmıştır. Burada 6x1 boyutunda öznitelik elde edilmiştir. Aynı görüntüden GLCM özniteliklerinin Çizelge 4.1'de belirtilen öznitelikler 4x1 vektör boyutu olarak çıkarılmıştır. Sonuç olarak birleştirilen bu iki vektör ile 10x1 boyutunda vektör elde edilmiştir.

Naif Bayes, Karar Ağaçları ve Rastgele Orman gibi sınıflandırma algoritmalarında, istenen özelliklerin elde edilmesi için uygun niteliklerin seçilmesi önemli bir karardır. Bu seçim, görüntünün bölünen bloklarına ve kullanılabilir (piksel tabanlı bozukluğu olmayan) görüntülerin türüne bağlıdır. Kısa mesafeden elde edilen görüntülerde gemi tespiti durumunda, deniz ortamı özellik olarak kullanılabilecek faydalı görsel

özellikler sağlar. Bu çalışma, görüntülerin her bir bloğundan renk ve doku özelliklerinin çıkarılmasını önermektedir. Çıkarılan özellikler birbiri ardına eklenir ve öznitelik vektörü oluşturulur. Daha sonra bu özellikler eğitim ve sınıflandırma aşamalarında kullanılır.

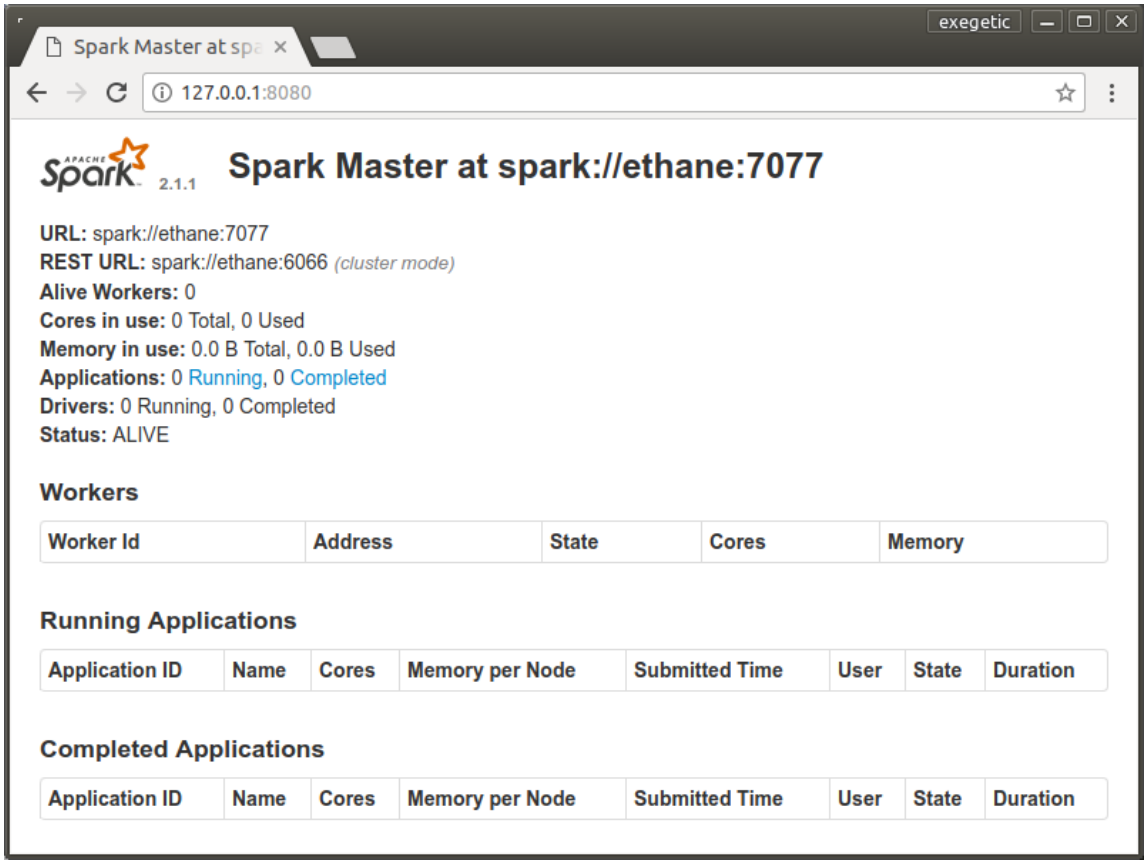
BÖLÜM 5

DENEYSEL ÇALIŞMALAR

Bu bölümde Apache Spark kümeleme mimarisinin tasarım aşamaları, makine öğrenmesi yöntemleriyle bu mimari üzerinde görüntü verilerinin sınıflandırma süreçleri ve süreçler sonucu elde edilen bilgiler anlatılmaktadır. Görüntü verileri Spark'ın MLlib kütüphanesi altındaki makine öğrenmesi yöntemleri ile analiz edilmiş ve sınıflandırılmıştır. Sınıflandırma esnasında oluşan eğitim modelleri kullanılarak test verilerinin sınıflandırılması üzerinde de bazı analizler gerçekleştirilmiştir. Spark kümeleme mimarisi test verilerinin sınıflandırılması esnasında kullanılmıştır. Elde edilen sonuçlar çizelge ve grafiklerle değerlendirilmiştir.

5.1. APACHE SPARK KÜMELEME MİMARİSİ TASARIMI

Apache Spark'ın kümeleme mimarisi kullanılarak oluşturulan eğitim modelleri üzerinden test verileri üzerinde de sınıflandırma işlemleri yapılmış ve bu sınıflandırma işleminin dağıtık sistemler üzerinde yapılması ile Spark'ın hız performansı test edilmiştir. Büyük verilerin işlenme süreci, Spark üzerinde master (ana bilgisayar) – worker (işçi bilgisayarlar veya süreçler) mimarisi kullanılarak yapıldığında, worker sayısına bağlı olarak uzamakta veya kısalmaktadır. Bu çalışma için elimizdeki görüntü verilerinin test veriseti için ayrılan kısmı üzerinde çalışılmıştır. Toplam 18 432 000 görüntü test verisi üzerinde öncelikle 1 master ve 4 worker, sonra 1 master ve 3 worker, daha sonra 1 master ve 2 worker ve en son 1 master ve 1 worker oluşturarak sınıflandırma süreleri analiz edilmiştir. Yapılacak sınıflandırma ve analiz işinin, birden fazla worker tanımlamasıyla 2-3 kat hızlandığı tespit edilmiştir. GNU/Linux ortamında, terminalden Spark üzerinde tek master oluşturma işleminin sonucunu Web GUI (Graphic User Interface) üzerinden Şekil 5.1'deki gibi görebiliriz:



Şekil 5.1: Web GUI üzerinden Spark ile tanımlı master

Şekil 5.1.'deki gibi master oluşumu için terminale gidilerek, aşağıdaki komut girilmelidir. Buradaki /usr/local/spark-2.4.4 kısmı, her bilgisayara göre, Spark sürümüne göre değişiklik gösterebilmektedir.

$ sudo /usr/local/spark-2.4.4/sbin/start-master.sh

Bir kümeleme mimarisi oluşturmak için worker tanımlaması yapmak gereklidir. Bunun için Spark klasörünün içinde bulunan conf klasörü açılır ve içindeki spark-env.template dosyası kopyalanarak template kısmı silinir ve sh dosyası haline getirilir. Daha sonra içine,

export SPARK_WORKER_INSTANCES=1

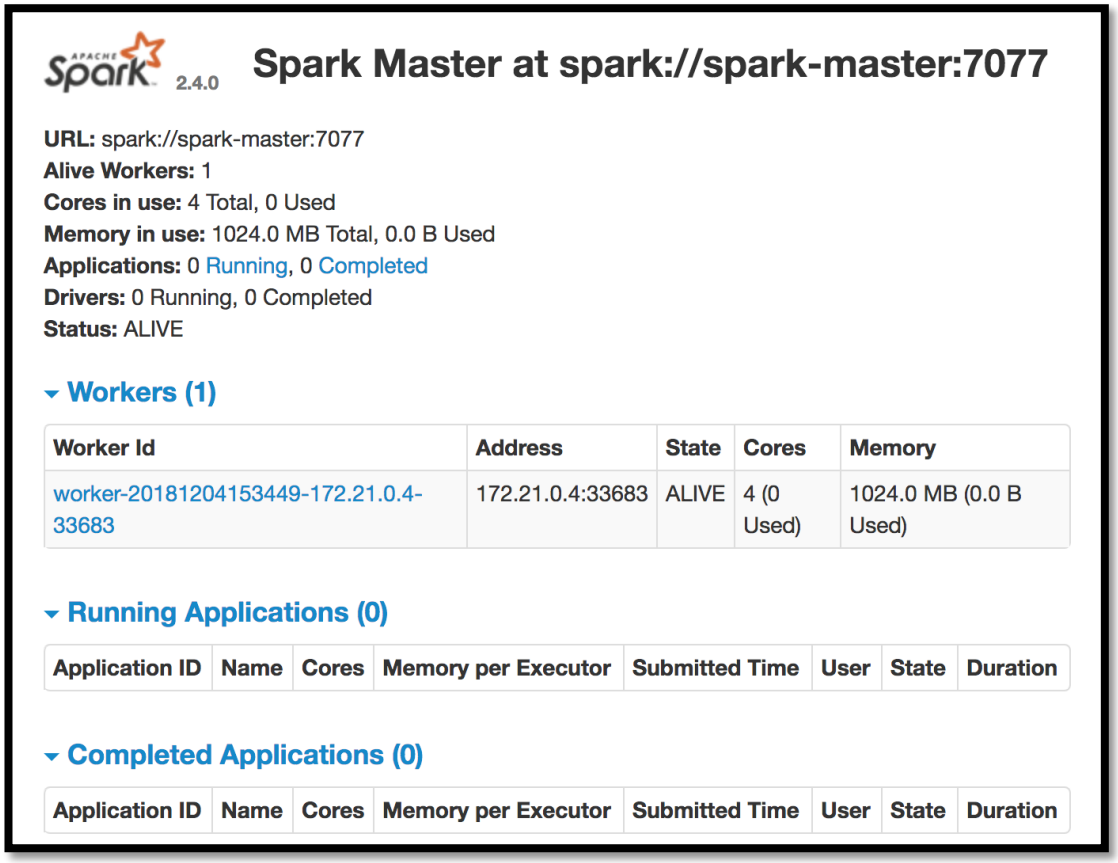
export SPARK_WORKER_CORES=1

export SPARK_WORKER_MEMORY=8g

satırları kaydedilerek tanımlama gerçekleştirilmiş olur. Burada 8 gigabayt bellek miktarına sahip 1 worker tanımlanmıştır.

$ sudo '/usr/local/spark-2.4.4/sbin/start-all.sh komutu çalıştırılarak master ve worker lar aktif hale getirilir. Şekil 5.2.'de 1 master ve 1 worker tanımlaması

gösterilmektedir. Ardıntan uygulama, terminal üzerinden spark-submit komutu ile çalıştırılabilir. Durdurmak için ise **stop-all.sh** dosyası çalıstırılmalıdır. Master oluşturduktan sonra bir worker tanımlarsak, yine Web GUI üzerinde Şekil 5.2'deki gibi görüntülenecektir:



Şekil 5.2: Web GUI üzerinden Spark ile Tanımlı Worker

Terminal üzerinden Spark kümeleme mimarisinin oluşum süreci bu şekildedir fakat uygulamamızın içine bu mimariyi tanıtacak bir kod bloğu yazmamız gereklidir. Uygulamamızda Spark nesnesinin oluşturulduğu kod satırı, aşağıdaki gibidir:

```
SparkConf conf = new
SparkConf().setMaster("spark://192.168.1.4:7077").setAppName("Sar_Analysis").set("spark.driver.memory", "4g")
    .set("spark.executor.memory", "1g")
    .set("spark.executor.cores","1")
    .set("spark.executor.instances","1");
```

Burada setMaster içine, Spark Web GUI'den de göreceğimiz master ip adresi:port numarası girilmelidir. driver.memory ile sürücü bellek miktarı, executor.memory ile yürütücü bellek miktarı, executor.cores ile yürütücü çekirdek miktarı ve executor.instances ile yürütücü örnek sayısı belirlenmektedir. Worker adedi executor örnek sayısı ile belirlenmektedir. Hem buradan hem de yukarıda bahsedilen spark-env.sh içindeki export SPARK_WORKER_INSTANCES=1 satırı değiştirilerek worker tanımlaması yapılmış olur. İstenilen konfigürasyonda master worker mimarisi oluşturulduktan sonra, terminale gidilerek,

bin/spark-submit --class Sar_Image_Analysis.Main_Sar_Analysis –master spark://192.168.1.4:7077 /home/betul/Masaüstü/SparkJob.jar

komutu çalıştırılır. Böylece uygulamamızı terminal üzerinden Spark kümeleme mimarisine göndererek workerlar üzerinde çalıştırabilmekteyiz.

5.2. MAKİNE ÖĞRENMESİ SINIFLANDIRMA YÖNTEMLERİ

Genel olarak, denetimli bir öğrenme süreci, eğitim ve sınıflandırma olmak üzere iki aşamadan oluşur. Görüntü sınıflandırma durumunda, görüntüler önce eğitim ve sınıflandırma için kullanılan bir eğitim setine ve bir test setine ayrılır. Eğitim seti, makine öğrenmesi sınıflandırıcısını eğitmek için kullanılan görüntülerden oluşur. Bu yaklaşımda, eğitim için seçilen özellikler bu eğitim görüntülerinin bloklarından çıkarılır ve hibrit bir öznitelik vektöründe birleştirilir. Eğitimden önce bağımsız değişkenlerin aralığını standartlaştırmak için bu öznitelik vektörü daha önce normalizasyon işlemlerinden geçirilmiştir.

Literatürde makine öğrenmesi ile sınıflandırma işlemlerinde kullanılan çeşitli algoritmalar mevcuttur. Gradyanla Güçlendirilmiş Ağaçlar Yöntemi (Gradient-Boosted Tree - GBT), bunlar içerisinde yaygın tercih edilen yöntemlerden biridir. GBT algoritması [44] birkaç regresyon ağacını birleştirme prensibine dayanan bir tahmin modelidir. Özellikle, regresyon ağaçları, ağaç sığsa yüksek sapma ve düşük sapma hataları veya ağaç derinse düşük sapma ve yüksek sapma hataları olmasıyla karakterize edilen modellerdir. Bu sorunun üstesinden gelebilmek amacıyla yüksek hataları azaltmak için birkaç regresyon ağacını birleştiren iki algoritma ailesi vardır. İlk aile rastgele ormanlardır ve torbalama ilkesine dayanmaktadır [45], yani buradaki

amaç, düşük önyargıyı korurken varyansı azaltmak için düşük önyargı ve yüksek sapma hatası olan modelleri birleştirmektir. Özellikle orijinal eğitim veri seti ilk önce farklı torbalanmış numuneler oluşturmak için değiştirilerek örneklenir. Daha sonra numunelerin her biri için derin bir ağaç eğitilir, yani bahsedilen yüksek varyanslı ve düşük önyargı hatasına sahip bir modeldir. Torbalı örneklerin hepsi birbirinden farklı olduğundan her ağacın tahmini farklıdır. Son olarak, nihai tahmin tüm karar ağaçlarının çoğunluk oylama kuralı kullanılarak oluşturulur. İkinci aile eğimi kuvvetlendiren ağaçlardır ve kuvvetlendirme prensibine dayanır [45], yani düşük bir varyansı korurken eğilimi azaltmak için modelleri yüksek sapma ve düşük varyans hatası ile birleştirmek amaçlıdır. Ayrıntılı olarak derin ağaçlar ve farklı eğitim veri kümeleri kullanmak yerine, güçlendirici ağaçlar girdi-çıktı ilişkisinin özellikleri konusunda eğitilmiş sığ ağaçlar kullanır. Özellikle ardışık sığ ağaçlarda n. ağaç, n'den önce oluşan ağaçlardaki tahmin hatalarını azaltmak amacıyla seri olarak eğitilir. Hatalı ve sağlıklı arasında ayrım yapmak için ilk ağaç tahmin edilir: ağaç, voltaj değerine dayalı bir karar sınırı çizer. Daha sonra, ikinci ağacın birinci ağacın yanlış sınıflandırılmış örneklerini düzelttiği tahmin edilmektedir; bu ağaç geçerli bir değer kullanarak ikinci bir sınır çizer. Son olarak, üçüncü ağacın ilk ikisinin hatalarını düzelttiği tahmin edilmektedir. Sonunda, modelin tahmini üç ağacın seri kombinasyonuna dayanmaktadır.

En yakın komşu (kNN) algoritması, parametrik olmayan klasik bir sınıflandırma yöntemidir. Spesifik olarak bir sorgu ve bir eğitim seti verildiğinde, kNN algoritmasının amacı eğitim setinden sorgunun en yakın komşularını bulmak ve çoğunluk oylama kuralı aracılığıyla sorguya bir sınıf etiketi atamaktır. Basitliği, etkinliği ve sezgiselliği nedeniyle kNN algoritması örüntü tanıma, özellik seçimi [46], aykırı algılama ve benzeri birçok alanda yaygın olarak kullanılmaktadır. kNN algoritmasının diğer yöntemlere kıyasla birçok faydası vardır. Parametrik olmayan bir sınıflandırma yöntemi olarak kNN algoritmasının bir eğitim sürecine ihtiyacı yoktur. Özellikle eğitim örneklerinin istatistiksel özellikleri hakkında önceden bilgi gerektirmez ve sorguyu eğitim seti tarafından sağlanan bilgilere dayanarak doğrudan sınıflandırabilir. Ayrıca, toplam eğitim örneği sayısı ve kNN algoritmasının sınıflandırma doğruluğunun en uygun Bayes sınıflandırıcısının iki katına yaklaşabileceği kanıtlanmıştır. Sınıflandırma amaçlı en yaygın kullanılan diğer

yöntem Destek Vektör Makineleridir (DVM). Bu modeller istatistiki öğrenme teorisinden [47] fikirlerin algoritmik uygulamalarıdır ve bu da verilerden tutarlı tahmin ediciler oluşturma sorunu ile ilgilidir: bir modelin bilinmeyen bir veri seti üzerindeki performansı yalnızca modelin özellikleri göz önüne alındığında nasıl tahmin edilebilir, ve bir eğitim setindeki performans algoritmik olarak destek vektör makineleri kısıtlı kuadratik optimizasyon problemini çözerek veri setleri arasında optimum ayırma sınırları oluşturur [48,49]. Farklı çekirdek fonksiyonları kullanılarak modele değişen derecelerde doğrusal olmayanlık ve esneklik dahil edilebilir. İleri istatistiksel fikirlerden türetilebildiği ve genelleme hatası üzerindeki sınırlar onlar için hesaplanabildiğinden DVM son yıllarda önemli araştırma ilgisi almıştır. Tıbbi literatürde diğer makine öğrenimi algoritmalarına göre veya bu makineyi aşan performanslar bildirilmiştir. DVM'nin dezavantajı, sınıflandırma sonucunun tamamen ikiye ayrılması ve sınıf üyeliği olasılığının verilmemesidir. Makine öğrenmesi algoritmalarında DVM sınıflandırma ve regresyon analizi için kullanılan verileri analiz eden ilişkili öğrenme algoritmalarına sahip denetimli öğrenme modelidir. Her biri iki kategoriden birine veya diğerine ait olarak işaretlenmiş bir dizi eğitim örneği verildiğinde, bir DVM eğitim algoritması bir kategoriye veya diğerine yeni örnekler atayan bir model oluşturmakta ve bu da olasılık dışı olmayan bir ikili doğrusal sınıflandırıcı haline gelmektedir. DVM, iki sınıflı doğrusal veri sınıflandırma problemlerinde yaygın olarak tercih edilen yöntemdir. Bu çalışmada analizler iki ayrı sınıf üzerinde gerçekleştirildiğinden bu yöntem tercih edilmiştir. Yöntemin amacı sınıfların birbirinden ayrıldığı hiper düzlemin bulunması işlemidir. İki sınıflı doğrusal veriler için farklı sınıflarda yer alan birbirine en yakın bulunan iki örneğin arasındaki mesafenin maksimum olarak belirlenmesi ile düzlem tespit edilmektedir [50].

i sayıda örnek içeren veri kümesinde yer alan örnekler $\{x_i, y_i\}$ ile belirtilip, $k = 1,2,\ldots\ldots i$ olarak düşünüldüğünde hiper düzleme ait denklemler aşağıdaki gibi olmalıdır [30]:

$$y_i = \begin{cases} -1, & wx_i + \beta \leq -1 \\ +1, & wx_i + \beta \geq +1 \end{cases} \tag{5.1}$$

Eşitlik 5.1'de y sınıf etiketini, w ağırlık vektörlerini ve β eğilim değerini ifade etmektedir. $wx + \beta \leq -1$ ve $wx + \beta \geq +1$ eşitsizlikleri destek vektörleri olarak adlandırılmaktadır.

Naif Bayes (NB) sınıflandırma algoritması, istatistiksel çıkarım için kullanılan Bayes Teoremi'ne dayanmaktadır [37]. Teorem, A olayının B olayını gözlemleme olasılığının, A verilen B olasılığı ile A ve B olasılıklarının birbirinden bağımsız olarak belirlenebileceğini belirtir. Naif Bayes sınıflandırıcısı, Bayes teoremine dayanan olasılıklı bir sınıflandırıcıdır. Bir özelliğin varlığının bir sınıfa verilen diğer özelliklerin varlığından şartlı olarak bağımsız olduğunu varsayar. Bir eğitim seti verildiğinde, her bir sınıfın önceki olasılığı ve her bir sınıfın her bir özelliğinin verilen koşullu olasılığı eğitim sürecinde tahmin edilebilir. Karar ağaçları ve rastgele orman gibi yöntemlerle karşılaştırıldığında sınıflandırma üzerindeki başarısı veri boyutuna bağlı olarak yüksek veya düşük olabilmektedir. Veri ne kadar çok olursa o oranda detay sağlayacağı için sınıflandırmada daha başarılı olacaktır.

Sınıflandırma ve Regresyon Ağacı (Classification and Regression Tree, CART) algoritması Breiman ve arkadaşları tarafından oluşturulmuş bir sınıflandırma yöntemidir [51]. CART bir ikili ağaçtır ve örnek bir kümeden bir karar ağacı oluşturmak için bir ikili segmentasyon yöntemi kullanır. CART sınıflandırmasında ağacın her bir düğümünde Gini dizini, örnek kümesini alt kümelere en etkili şekilde ayıran veri niteliğini seçmek için kullanılır. En düşük Gini dizinine sahip özellik en iyi özellik olarak seçilir.

Rastgele Ağaçlar (Random Forest, RF) algoritması [52-53] karar ağaçlarının komitesini (topluluğunu) kullanır. RF algoritmasındaki yüksek kalite sınıflandırması çok sayıda basit sınıflandırıcıyı (karar ağaçları) birleştirerek elde edilir. Nihai sınıflandırma sonucu birçok ağacın tepkileri toplanması temelinde elde edilir. RF algoritması bir karar ağacıyla karşılaştırıldığında aşırı takılma sorununu azaltır ve sınıflandırma kalitesini iyileştirir. RF algoritması, torbalama, bootstrap toplama ve rastgele altuzay yöntemlerini birleştirir. RF algoritmasında, torbalamada olduğu gibi sınıflandırıcıların eğitimi aynı veri kümesinde aynı ağaçları oluşturma sorununu çözen eğitim setinin farklı alt kümelerinde bağımsız olarak gerçekleşir. Sonuç olarak,

herhangi bir nesnenin sınıfı, bir ağacın bir oyu olduğu varsayılarak ağaçların çoğunun oyladığı sınıfa eşit olacaktır. RF algoritması uygulaması durumunda parametrelerin değerlerinin tanımlanması önemlidir. RF algoritmasının ana parametreleri şunlardır: ormandaki ağaç sayısı, en iyi ayrımı ararken dikkate alınması gereken özellik sayısı, ağacın maksimum derinliği, yarma kriteri.

Genel olarak, denetimli bir öğrenme süreci eğitim ve sınıflandırma olmak üzere iki aşamadan oluşur. Analizi gerçekleştirilecek görüntüler eğitim ve test verisi olmak üzere önceden belirlenen bir oranda ikiye ayrılır. Eğitim seti, makine öğrenmesi sınıflandırıcısını eğitmek için kullanılan görüntülerden oluşur. Bu yaklaşımda, eğitim için seçilen özellikler bu eğitim görüntülerinin bloklarından çıkarılır ve hibrit bir öznitelik vektöründe birleştirilir. Eğitime geçmeden oluşturulan öznitelik vektörü normalizasyon işlemlerinden geçirilir. Eğitim sırasında her bir eğitim bloğunun doğru sınıflandırması, blokların önceden doğru bir şekilde etiketlendiği ikili maskeler (Şekil 4. (b)) aracılığıyla da sağlanır.

Makine öğrenmesi algoritmaları ile görüntü verileri eğitildikten sonra sınıflandırma işlemleri değerlendirme için kullanılan görüntüler tarafından oluşturulan test setinde gerçekleştirilir. Eğitim sırasında oluşturulan sınıflandırıcılar, bu test görüntülerindeki blokların doğru sınıflandırmasını tahmin eder ve bunları gemi veya gemi olmayan bloklar olarak sınıflandırır. Gemi blokları ve gemi olmayan bloklar sırasıyla beyaz pikseller ve siyah piksellerle temsil edilmektedir. Test görüntüleri üzerinde yapılan sınıflandırma analizleri sonuçları, dört farklı makine öğrenmesi algoritması tarafından incelenerek değerlendirmeler yapılmıştır. Bu algoritmalar tarafından yapılan sınıflandırılmalarda yüksek başarım, analiz için seçilen blok boyutuna bağlıdır. En ufak boyutlara sahip bloklar, görüntülerden çok daha fazla detayın hesaba katılmasını sağlamaktadır. Küçük blok boyutları ile karşılaştırıldığında, daha büyük blok boyutları kullanılarak yapılan sınıflandırmalar daha az başarım göstermektedir. Ayrıca, görüntülerin parlaklık, hava koşullarına bağlı aydınlık olup olmaması gibi faktörlerin yanında, deniz gemilerinin çoğunlukla düşman kuvvetleri tarafından izlenmelerini önlemek için deniz renklerine benzer tonlar kullanılarak kamufle edilmesi, sınıflandırma aşamasında dezavantaj olarak düşünülmektedir. Bu dezavantaj, renkten elde edilen özelliklerin yanında doku özelliklerinin kullanılması ile çözülmektedir. Bu

çalışmada, yukarıda bahsedilen en yaygın üç yöntem olan Naif Bayes, Karar Ağaçları ve Rastgele Orman yöntemleri kullanılmıştır.

5.2.1. Naif Bayes

Naif Bayes algoritması, denetimli bir makine öğrenme algoritmasıdır. Özellikler arasında bağımsızlık varsayımı ile birden fazla sınıflandırma için basit bir olasılık modelidir. NB, her özelliğin bir sınıfa atanan olasılıklara bağımsız olarak katkıda bulunduğunu varsayar. NB sınıflandırıcısı aşağıdaki formüle göre analiz işlemlerini gerçekleştirmektedir.

$$P(c|F) = (P(F|c)P(c))/(P(F)) \tag{5.2}$$

Eşitlik 5.2'de $P(c)$ ve $P(F)$, c ve F olaylarının önsel olasılıklarıdır, $P(c|F)$ F olayının gerçekleştiği durumda c olayının gerçekleşme olasılığını, $P(F|c)$ c olayının gerçekleştiği durumda F olayının gerçekleşme olasılığını gösterir. Naif Bayes algoritması ile veri kümesini eğitmek kısmen daha zordur fakat eğittikten sonra oldukça hızlı çalışan bir sınıflandırma algoritmasıdır. Bir durumun olma ihtimalinin en yüksek olma koşuluna göre hareket eder. Dezavantajlı olduğu durum ise verinin sürekli değiştiği durumdur. Çünkü her yeni veri eğitim sürecini uzatacaktır [54].

5.2.2. Karar Ağaçları

Makine öğrenmesi algoritmaları içinde en etkin sonuçların üretilmesini sağlayan ikinci yöntem, karar ağacı algoritmalarıdır. Sınıflandırma ve regresyon amaçlı kullanılabilmektedir. Bir karar ağacı; düğüm, dal ve yapraktan oluşmaktadır. En üst kısım kök, kökten diğer düğümlere giden yol dal, bu dallar üzerinden en son varılan sonuç ise yaprağı ifade etmektedir [55]. Bu algoritma ile eğitim gerçekleştirilecek verilere bir dizi soru sorulmakta, elde edilen cevaplar doğrultusunda sonuca ulaşılmaktadır. Karar ağacı oluşurken, ağaçtaki dallanmanın hangi kriter veya öznitelik değerine göre yapılması gerektiği bilgi kazancı ve bilgi kazanç oranı yaklaşımları ile hesaplanmaktadır [56]. Karar ağaçları bir dizi karar kuralı uygulayarak yukarıdan aşağı yinelenen bir şekilde bölme ve fethetme şeklinde ilerleyen açgözlü bir

algoritma çeşididir [57]. Bu algoritmada, bir ağaç yapısı oluşturulur ve sınıf etiketleri ağacın yapraklarında ifade edilir. Son ağaç, yaprak düğümüne ulaşan tüm örnekler için aynı etiketi öngörür. Her bölüm, bir ağaç düğümünde bilgi kazanımını en üst düzeye çıkarmak için olası bölünmeler kümesinden en iyi ayrımı seçerek belirlenir. Her ağaç düğümünde seçilen bölünme, bölünmüş bir v'nin T veri kümesine uygulandığında, bilgi kazancının en üst düzeye çıkarılması için gerekli olan argümanlar IG(T,v) hesaplanarak elde edilir. Burada, iki farklı ölçü (Gini safsızlığı ve entropi safsızlığı) veri kümesini sınıflandırma için önerilmektedir [57]. Gini safsızlığı:

$$\sum_{a=1}^{C} f_{\mathrm{a}}(1 - f_{\mathrm{a}}) \tag{5.3}$$

şeklinde hesaplanmaktadır. Eşitlik 5.3'te C, benzersiz etiketlerin sayısıdır ve f_{a}, bir düğümdeki a etiketinin frekansıdır. Entropi için tanımlanan safsızlık ölçüsü ise Eşitlik 5.4'teki gibidir:

$$\sum_{a=1}^{C} -f_{\mathrm{a}} \log(f_{\mathrm{a}}) \tag{5.4}$$

Bilgi kazancı, ana düğüm safsızlığının iki alt düğüm safsızlığının ağırlıklı toplamından çıkarılmasına dayanır. Bilgi kazancı Eşitlik 5.5'teki gibi tanımlanır:

$$\mathrm{IG(T,v)}=Gini(T) - \frac{N_{sol}}{N} Gini(T_{sol}) - \frac{N_{sa\check{g}}}{N} Gini\left(T_{sa\check{g}}\right) \tag{5.5}$$

Burada N boyutuna sahip veri kümesi T, sırasıyla bölümleri ve N_{sol} ve $N_{sa\check{g}}$ boyutlarında T_{sol} ve $T_{sa\check{g}}$ terimlerini bölerek elde edilir.

5.2.3. Rastgele Orman

Rastgele orman yöntemi, en başarılı makine öğrenimi modellerinden biridir. Rastgele Orman (Random Forest, RF) sınıflandırma gibi denetimli öğrenme görevlerini çözmek için karar ağaçlarının bir araya gelerek oluşturduğu, gürültüye karşı iyi bir toleransa

sahip ve aşırı uyuma eğilimli olmayan bir topluluk öğrenme algoritmasıdır. Naif Bayes ve karar ağaçları yaklaşımına kıyasla, çok daha yüksek başarımlı sınıflandırma sonuçları vermektedir. Daha doğru ve istikrarlı bir tahmin elde etmek için daha güçlü modeller üreterek birden fazla karar ağacını bir araya getirmektedir. Algoritma, eğitim aşamasında rastgele bir veri örneği kullanarak farklı veri alt kümelerine dayanan birden fazla karar ağacından oluşan bir model oluşturur. Bu rastgelelik RF modelinin avantajlı bir özelliğini oluşturur, bu da modelin tek bir karar ağacından daha sağlam olmasını ve eğitim verilerinin aşırı uyumlu ve birbirine benzer olması probleminin üstesinden gelmesini sağlar [57].

Aşırı uyum, veri üzerinde eğitim gerçekleştirildiği sırada, modelin veriyi aşırı öğrenmesi, ezberlemesi olarak tanımlanmaktadır. Karar ağaçları yaklaşımının bir dezavantajı olan aşırı uyum probleminin üstesinden gelmek için Rastgele Ağaçlar yaklaşımı, veri setinden ve öznitelik vektörlerinden rastgele alt ağaçlar oluşturmakta ve bunları eğitmektedir. Her biri farklı birer karar ağaçlarından oluşan bu yapıda, en çok oy alan tahminler aracılığı ile sınıflandırma işlemi gerçekleştirilmiş olur. Örnek küme T ‘ yi bölmek için b öznitelikleri kullanılarak elde edilen bilgi kazancı (BK) ve Gini endeksi aşağıda verilen Eşitlik 5.6 ve Eşitlik 5.7’de düğüm bölme formülü ile gösterilmektedir [58]:

$$BK(T,b) = Ent(T) - \sum_{n=1}^{V} -\frac{\left|T^{(n)}\right|}{|T|} Ent(T^n) \tag{5.6}$$

$$Gini(T,b) = \sum_{n=1}^{V} -\frac{\left|T^{(n)}\right|}{|T|} Gini(T^n) \tag{5.7}$$

Bu çalışmada NB, DT ve RF makine öğrenmesi yöntemleri kullanılarak büyük boyutlu görüntüler üzerinde sınıflandırma yapılmıştır. 768x768 boyutlarında olan gemi görüntüleri 16x16 blok boyutlarına ayrıldığında, bir görüntüden 2 304 blok görüntü elde edilirken 283 adet görüntü için toplam 652 032 blok görüntü elde edilmektedir. Aynı görüntü verileri, 32x32 blok boyutlarına ayrıldığında 163 008 blok görüntü, 64x64 blok boyutlarına ayrıldığında ise, 40 752 blok görüntü elde edilmiştir. Toplamda, 283 görüntüden elde edilen 855 792 blok görüntü ile analiz işlemleri

gerçekleştirilmiştir. Bu çalışma sonuçları Çizelge 5.1 ve Çizelge 5.3'te sunulmaktadır. Öznitelik vektörü ve etiket vektörü kullanılarak sınıflandırma işlemlerinin başarımı Eclipse Oxygen sürümü ortamında Apache Spark'ın MLlib kütüphanesi kullanılarak hesaplandı. Sonuçlar, GNU/Linux işletim sistemi dağıtımlarından Ubuntu 16.04 ortamında alınmıştır.

Çizelge 5.1. Hibrit öznitelik vektörü ile 3 farklı sınıflandırma algoritmasının 3 farklı blok boyutu için doğruluk sonuçları (%).

	16x16	32x32	64x64
NB	80,12	70,12	82,33
DT	99,58	90,54	86,09
RF	**99,62**	**94,88**	**90,56**

Yukarıdaki çizelgede görüldüğü üzere, en iyi sınıflandırma başarısı RF algoritması ile elde edilmiştir. Blok boyutları küçüldükçe DT ve RF algoritmaları çok daha başarılı sınıflandırma ölçütü vermektedir. Blok boyutları açısından değerlendirildiğinde en yüksek başarım 16x16 blok boyutlarına aittir. Bunun nedeni, görüntüye dair daha detaylı ve daha fazla bilgi barındırıyor olmasıdır.

Çizelge 5.2. Öznitelik çıkarımında kullanılan özelliklerin kısaltması.

Kısaltmalar	Özellikler
RGB	RGB bileşenlerinin ortalaması
HSV	HSV bileşenlerinin ortalaması
LAB	LAB bileşenlerinin ortalaması
SD	Renk bileşenleri standart sapması
FS	Birinci Dereceden İstatistikler
GLCM	Gri Seviye Eş-Oluşum Matrisi

Bu çalışmada kullanılan sınıflandırma başarım ölçütlerini farklı bir bakışla değerlendirmek için oluşturulan hibrit vektörü birkaç öznitelik bazında parçalara ayırarak sınıflandırma başarısı test edilmiştir. Bunun için önce her bir renk uzayında oluşturulan vektörler ile ayrı ayrı sınıflandırma sonuçları alınmıştır Daha sonra her bir

renk uzayı vektörüne SD ve FS öznitelikleri eklenerek sonuçlar alınmıştır. En son, GLCM öznitelikleri de eklenerek hangi vektörle daha iyi sınıflandırma sonucu alındığı ölçülmüştür. Bu doğrultuda alınan sonuçlar Çizelge 5.3'teki gibidir:

Çizelge 5.3. Parçalı öznitelik vektörü ile 3 farklı sınıflandırma algoritması doğruluk sonuçları (%).

Kısaltmalar	NB	DT	RF
RGB	54,53	96,64	97,54
HSV	98,76	98,79	98,82
LAB	54,56	94,05	94,37
RGB+SD+FS	78,68	98,68	98,70
HSV+SD+FS	63,76	98,68	98,73
LAB+SD+FS	67,39	98,59	98,84
RGB+SD+FS+GLCM	69,81	97,38	98,21
HSV+SD+FS+GLCM	79,12	97,86	**98,92**
LAB+SD+FS+GLCM	68,29	98,03	98,68

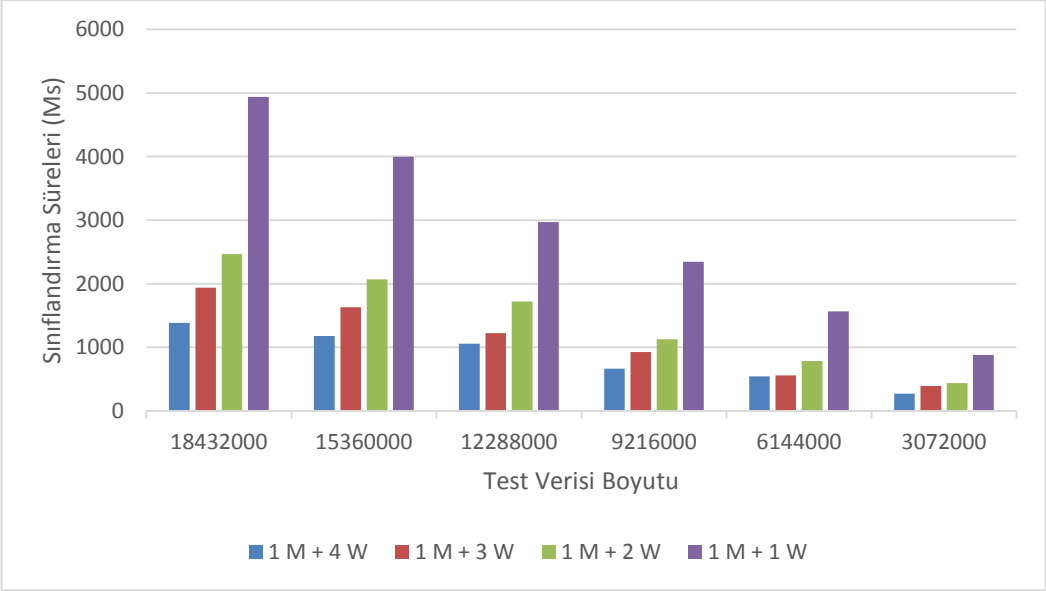
Çizelge 5.3.'e göre, 28x1 boyutlu hibrit öznitelik vektörü, dokuz farklı öznitelik bazında parçalanarak hangi özniteliklerin baskın derecede sınıflandırmaya katkısı olduğu belirtilmektedir. Çizelge 5.3'e göre görüntüler üzerinde üç farklı sınıflandırma algoritmasında en yüksek başarımı elde eden vektör HSV+SD+FS+GLCM özniteliği ile oluşturulmuş vektördür. Ayrıca Naif Bayes algoritması, RGB+SD+FS, HSV+SD+FS, LAB+SD+FS vektörleri ile eğitildiğinde sınıflandırma başarımının düştüğü görülse de RF algoritmasının dokuz vektörün hepsi ile eğitildiğinde, sınıflandırma başarısının %94'ün altına düşmediği görülmektedir.

Görüntü üzerinde geminin var veya yok olmasına göre yapılan sınıflandırma sonucunda test verileri üzerinde de sınıflandırma yapılmıştır. Master worker mimarisi kullanılarak yapılan bu sınıflandırmada farklı görüntü veri boyutlarının farklı master-worker mimarisi ile analizi değerlendirilmiştir. Spark hız performansının veri boyutuna göre ve worker adedine göre değişimi gözlemlenmiştir. Elde edilen bulgular çizelgeler eşliğinde sunulmuştur. Üç farklı yöntem kullanılarak geliştirilen süre sonuçları değerlendirilmiştir. Soldan sağa, worker sayısı azaldıkça; test işlem süresinin arttığı gözlemlenmiştir. Bu da paralel mimarinin büyük veri işlemede süre açısından önemli olduğunu göstermektedir.

Çizelge 5.4. Kümeleme mimarisi ile NB yöntemi sınıflandırma süreleri (ms).

Veri Boyutu	1 Ms+4 Wr	1 Ms+3 Wr	1 Ms+2 Wr	1 Ms+1 Wr
18 432 000	1384,67	1939,33	2467,67	4939,67
15 360 000	1179,67	1628,67	2066,33	3999
12 288 000	1056,67	1223,67	1720,67	2972
9 216 000	663,67	926	1127,33	2345,33
6 144 000	541	558	783,33	1563,33
3 072 000	271,33	391,67	438,67	882,33

Çizelge 5.4'te NB algoritması ile 18 432 000 veri üzerinde, 1 master ve 4 worker ile 1 master ve 1 worker arasında 3,5 kat hız artışı olduğu görülmektedir. Veri boyutu azaldıkça aradaki zaman farkı veri ile orantılı düşüş gösterse de birden fazla worker tanımlanmış süreçlerin çok hızlı olduğu görülmektedir. Veri boyutu her bir alt satıra geçerken sınıflandırma süreleri değerlendirmesi için belli oranda azaltıldığında yine hız artışı gözlemlenmektedir. Büyük veri söz konusu olduğunda, kümeleme mimarisinin tercih edilmesi böylece çok daha anlaşılır olmaktadır.

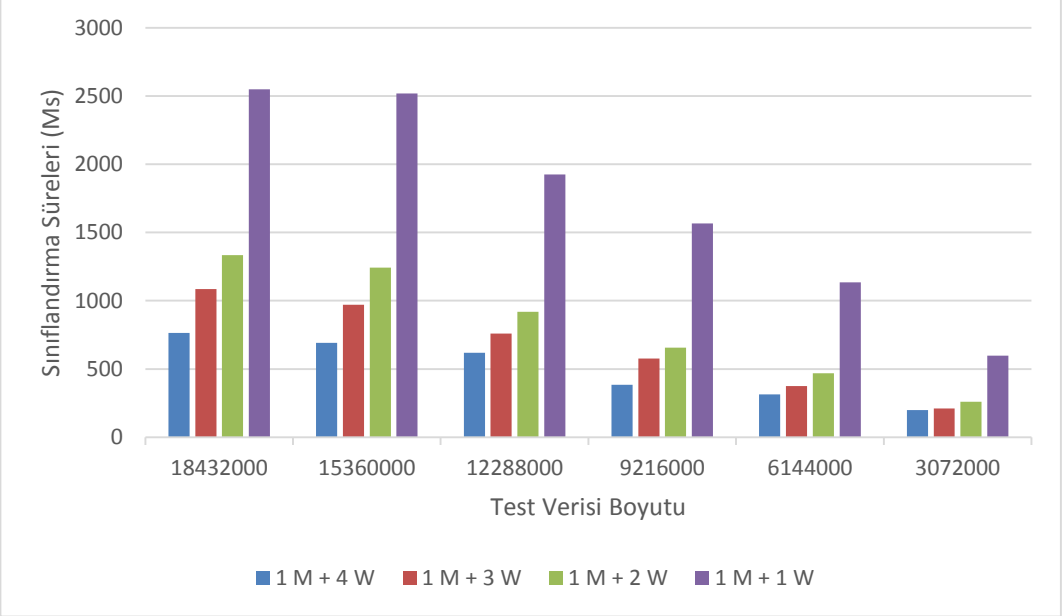


Şekil 5.3: Kümeleme mimarisi ile NB yöntemi sınıflandırma süreleri

Çizelge 5.5. Kümeleme mimarisi ile DT yöntemi sınıflandırma süreleri (ms).

Veri Boyutu	1 Ms+4 Wr	1 Ms+3 Wr	1 Ms+2 Wr	1 Ms+1 Wr
18 432 000	763,67	1085	1333,33	2549,67
15 360 000	692,67	970	1243	2519,33
12 288 000	617,67	760,67	919,33	1925,33
9 216 000	383,67	576,67	656,67	1566
6 144 000	315	376	468,67	1134,33
3 072 000	199,67	211,67	260,33	596,67

Çizelge 5.5'te DT algoritması ile 18 432 000 veri üzerinde, 1 master ve 4 worker ile 1 master ve 1 worker arasında 3,3 kat hız artışı olduğu görülmektedir. Hız artışı RF algoritmasına göre düşüş gösterse de iki yöntem arasında çok büyük fark bulunmamaktadır.



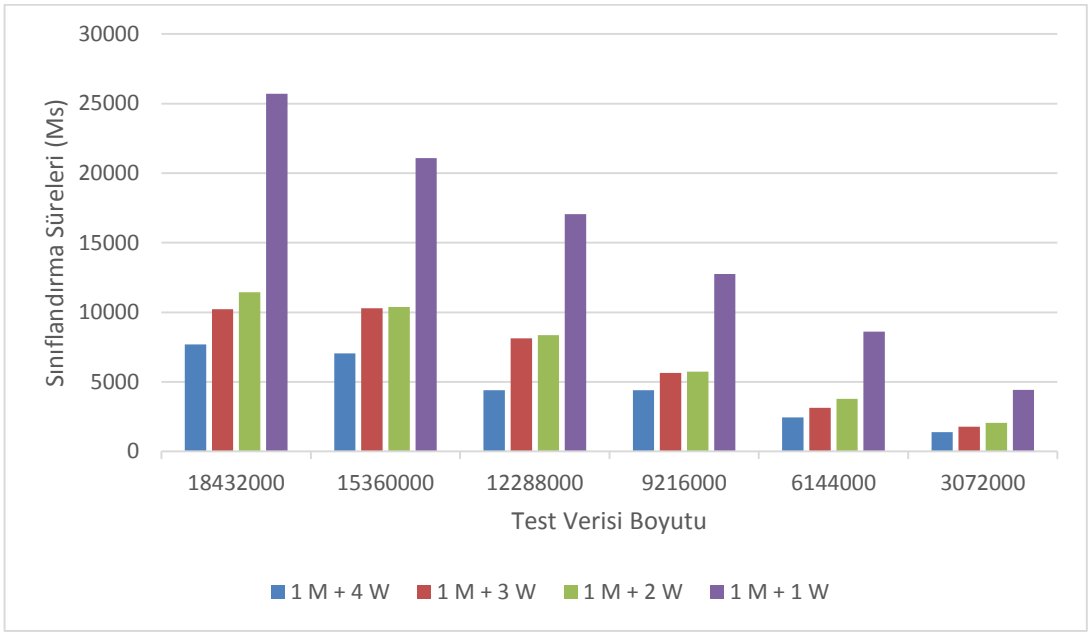
Şekil 5.4: Kümeleme mimarisi ile DT yöntemi sınıflandırma süreleri

DT sınıflandırma algoritması ile 18 432 000 veri üzerinde yapılan analiz işlemleri sonucu Şekil 5.4'te grafik olarak sunulduğunda, 1 master ve 1 worker ile yapılan analizin oldukça uzun sürdüğü net olarak görülmektedir.

Çizelge 5.6. Kümeleme mimarisi ile RF yöntemi sınıflandırma süreleri (ms).

Veri Boyutu	1 Ms+4 Wr	1 Ms+3 Wr	1 Ms+2 Wr	1 Ms+1 Wr
18 432 000	7680	10228	11442,33	25710,67
15 360 000	7054	10286	10389,67	21075,33
12 288 000	4400,33	8128,33	8358,67	17059
9 216 000	4410,33	5647	5740,33	12754
6 144 000	2454,33	3126,67	3790	8610,33
3 072 000	1398,33	1775	2063,33	4419,67

Çizelge 5.6' da RF algoritması ile 18 432 000 veri üzerinde, 1 master ve 4 worker ile 1 master ve 1 worker arasında 3,3 kat hız artışı olduğu görülmektedir. 15 360 000 veri boyutu için, 1 master 4 worker ile 1 master 1 worker arasında 2,9 hız artışı görülmektedir. Kümeleme mimarisinin en verimli sonuçlar verdiği tespit ve sınıflandırma yöntemi RF yöntemidir.



Şekil 5.5: Kümeleme mimarisi ile RF yöntemi sınıflandırma süreleri

Şekil 5.5'te görüldüğü üzere kümeleme mimarisi yapısı ile birden fazla worker tanımlayarak bunların paralel çalıştırılmasını tasarlamakla, hem bilgisayar sistemlerindeki tüm kaynakların etkin olarak kullanımını sağlanmakta hem de

sınıflandırma ve tespit işlemlerinin daha kısa sürede bitirilmesini gerçekleştirmekteyiz.

Apache Spark kümeleme mimarisi kullanılarak büyük boyutlu görüntü üzerinde gerçekleştirilen sınıflandırma ve analiz işlemleri çok daha kısa sürelerde etkili sonuçlar alınmasını sağlamıştır. Sınıflandırma sürelerindeki bu azalma ile amaca yönelik hizmetlerde kullanıcılara daha hızlı yanıt olabilmek ve sorunlarına hemen çözüm sunabilmek kümeleme mimarisi sayesindedir.

BÖLÜM 6

SONUÇ VE ÖNERİLER

Bu çalışmada görüntü içeriklerinden renk ve doku öznitelik çıkarımları yapılarak hibrit bir vektör elde edilmiştir. Hibrit öznitelik vektörünün uzunluğu 28x1 boyutlarındadır. Çizelgeler ve grafikler ile farklı blok boyutları üzerinden elde edilen sonuçlar ve sınıflandırma başarısı değerlendirilmektedir. Öznitelik vektörü ve etiket vektörü kullanılarak sınıflandırma işlemlerinin başarımı Eclipse Oxygen sürümü ortamında Apache Spark'ın MLlib kütüphanesi kullanılarak hesaplandı. Sonuçlar, GNU/Linux işletim sistemi dağıtımlarından Ubuntu 16.04 ortamında alınmıştır. Vektör farklı boyutlarda, sadece renk uzayları değerlendirilmiş olarak, sadece doku özellikleri değerlendirilmiş olarak yeni boyutlarda oluşturulup, analiz ve sınıflandırma işlemleri yapılmıştır. Farklı öznitelik boyutları kullanılarak yapılan sınıflandırma sonuçları da değerlendirilmiştir.

Bu çalışmanın amacı, makine öğrenmesi yöntemleri kullanılarak, kısa mesafeli görüntüler üzerinde gemi ve gemi dışı bloklar arasında sınıflandırma yapmaktır. En yüksek başarımı %99,62 oranı ile Rastgele Orman yöntemi vermiştir. Değerlendirilen üç renk alanı arasındaki karşılaştırmalı çalışmada, HSV+SD+FS+GLCM öznitelik vektörü ile en yüksek başarım oranı elde edilmiştir.

Gelecek çalışmalar için, daha karmaşık özelliklerin öznitelik olarak değerlendirilmesi, çok daha verimli sonuçlar elde edilmesini sağlayabilir. Farklı makine öğrenmesi yöntemleri denenerek verimli çalışmalar elde edilebilir. Yapay sinir ağları ve derin öğrenme yöntemleri ile öznitelik çıkarımı yapılabilir ve Apache Spark kümeleme mimarisi Docker teknolojisi kullanılarak tasarlanabilir.

KAYNAKLAR

1. Özlük, Ö., "Big Data 101", ***Webrazzi Online: Big Data***, 27 Ekim (2016).

2. İnternet: "Growth of Mobile Data Traffic", Source: General Mobile, Cisco, **https://anritsu.typepad.com/.a/6a0163040dc8ca970d01b8d1bd3939970c-popup.**

3. İnternet: Türkiye Radyo Televizyon Kurumu Haber, "2019'da İnternette 1 Dakikada Neler Yaşanıyor?", **https://www.trthaber.com/haber/dunya/2019da-internette-1-dakikada-neler-yasaniyor-409609.html**, (2019).

4. İnternet: Graham-Cumming J., "Internet performance during the COVID-19 emergency", **https://blog.cloudflare.com/recent-trends-in-internet-traffic/**, (2020).

5. İnternet: Dünya Sağlık Örgütü, "Coronavirus disease (COVID-19) pandemic", **https://www.who.int/emergencies/diseases/novel-coronavirus-2019**, (2019).

6. Sırman, Alp, Mayıs, ***Popular Science Türkiye***, 54-56, (2020).

7. Atalay, M. ve Çelik, E., "Büyük Veri Analizinde Yapay Zeka ve Makine Öğrenmesi Uygulamaları", ***Mehmet Akif Ersoy Üniversitesi Sosyal Bilimler Enstitüsü Dergisi***, 9 (22): 155-172 (2017).

8. İnternet: Airbus Ship Detection, "Airbus Ship Detection Challenge", **https://www.kaggle.com/c/airbus-ship-detection**, (2018).

9. İnternet: Airbus Ship Detection, "Airbus Ship Detection Challenge", **https://www.kaggle.com/c/airbus-ship-detection/data**, (2018).

10. Zaharia, M., Chowdhury, M., Franklin, M., J., Shenker, S. ve Medjahed I., "Spark: Cluster Computing with Working Sets", Temmuz, (2010).

11. İnternet: Apache Spark Foundation "Apache Spark Lightning-fast Unified Analytics Engine", **https://spark.apache.org/**, (2020).

12. Chao, Z., Shi, S., Gao, H., Luo, J. ve Wang, H., "A gray-box performance model for Apache Spark", ***Future Generation Computer Systems***, 89, 58-67, (2018).

13. İnternet: Pointer, I., "What is Apache Spark? The big data platform that crushed Hadoop", **https://www.infoworld.com/article/3236869/what-is-apache-spark-the-big-data-platform-that-crushed-hadoop.html,** Mart (2020).

14. Belouch, M., El Hadaj, S. ve Idhammad, M., "Performance evaluation of intrusion detection based on machinelearning using Apache Spark", ***Procedia Computer Science***, 127 (1-6), (2018).

15. Mustafa, S., Elghandour, I. ve A. I. Mohamed, "A Machine Learning Approach for Predicting Execution Time of Spark Jobs", ***Alexandria Engineering Journal***, 57, (3767-3778), (2018).

16. İnternet: Apache Spark Foundation "Machine Learning Library (MLlib) Guide", **https://spark.apache.org/docs/latest/ml-guide.html**, (2020).

17. Meng, X., Talwalker, A., "MLlib: Machine Learning in Apache Spark Xiangrui".

18. Cavallaro G, Riedel M, Richerzhagen M, Benediktsson JA, Plaza A. "On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods". ***IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing***, 8, 4634-4646, 2015.

19. Morillas JRA, Garsia IC, Zölzer U. "Ship Detection Based on SVM Using Color and Texture Features". ***2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)***, Cluj-Napoca, Romania, 3-5 September 2015.

20. Peng, B., Zhang, L. ve Zhang, D., "A survey of graph theoretical approaches to image segmentation", ***Pattern Recognition,*** 46(3), 1020-1038, (2013).

21. Ergül M, Alatan AA. "Geospatial Object Recognition From High Resolution Satellite Imagery". ***2013 21st Signal Processing and Communications Applications Conference (SIU)***, Haspolat, Turkey, 24-26 April (2013).

22. Medhi S, Ahmed C, Gayan R. "A Study on Feature Extraction Techniques in Image Processing". ***International Journal of Computer Sciences and Engineering***, 4, (2016).

23. Haralick RM, Shanmugam K, Dinstein I. "Textural Features for Image Classification". ***IEEE Transactions on Sysytems, Man, and Cybernetics***, SMC-3(6), 610-621, 1973.

24. Jain AK, Farrokhnia F. "Unsupervised texture segmentation using Gabor filters". ***Pattern Recognit***, 24(12), 1167–1186, 1991.

25. Majunath BS, Ma WY. "Texture features for browsing and retrieval of image data". ***IEEE Trans. Pattern Anal. Mach. Intell.***, 18(8), 837–842, 1996.

26. Dalal N, Triggs B. "Histograms of oriented gradients for human detection". ***2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)***, San Diego, CA, USA, USA, 20-25 June 2005.

27. Tombak A, Aptoula E, Kayabol K. “Pixel-Based Classification of SAR Images Using Features”. ***IEEE Geoscience and Remote Sensing Letters***, 16, 564-567, 2019.

28. Medjahed SA. “A Comparative Study of Feature Extraction Methods in Images Classification”. ***I.J. Image, Graphics and Signal Processing***, 3, 16–23, 2015.

29. Srivastava D, Wadhvani R, Gyanchandani M. “A Review: Color Feature Extraction Methods for Content Based Image Retrieval”. ***IJCEM International Journal of Computational Engineering & Management***, 18, 2015.

30. Hlaing KNN. “First Order Statistics and GLCM Based Feature Extraction for Recognition of Myanmar Paper Currency”. ***2016 Second Asian Conference on Defence Technology (ACDT)***, 21-23 Jan. 2016.

31. Temizkan E, Bilge HŞ. “Airport Detection by Combining Geometric and Texture Features on RASAT Satellite Images”. ***2017 25th Signal Processing and Communications Applications Conference (SIU)***, Antalya, Turkey, 15-18 May 2017.

32. Li Y, Zhang H, Guo Q, Li X. “Machine Learning Methods for Ship Detection in Satellite Images”.

33. Nie T, He B, Bi G, Zhang Y, Wang W. “A Method of Ship Detection under Complex Background”. ***International Journal of Geo-Information***, 2017.

34. Acharjya DP, Ahmed K. “A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools”. ***International Journal of Advanced Computer Science and Applications***, 7, 2016.

35. Oğul İÜ, Özcan C, Hakdağlı Ö. “Fast Text Classification with Naive Bayes Method on Apache Spark”. ***2017 25th Signal Processing and Communications Applications Conference (SIU)***, Antalya, Turkey.

36. Oğul İÜ, Özcan C, Hakdağlı Ö. “Text Classification with Spark Support Vector Machine”. ***1. Ulusal Bulut Bilişim ve Büyük Veri Sempozyumu***, Antalya, 2017.

37. Özcan C, Ersoy O, Oğul İÜ. “Classification of SAR Image Patches with Apache Spark Using GLCM Texture Features”. ***International Conference on Advanced Technologies, 3rd World Conference on Big Data***, İzmir, 28 - 30 Nisan 2018.

38. Wang N, Chen F, Yu B, Qin Y. “Segmentation of large-scale remotely sensed images on a Spark platform: A strategy for handling massive image tiles with the MapReduce model”. ***ISPRS Journal of Photogrammetry and Remote Sensing***, 162, 137-147, 2020.

39. Han J, Kamber M, Pei J. Data Mining: Concepts and Techniques. Waltham, MA, USA: Elsevier, Second Edition, 2006.

40. Kaya Ç, Yıldız O. "Makine Öğrenmesi Teknikleriyle Saldırı Tespiti: Karşılaştırmalı Analiz". ***Marmara Fen Bilimleri Dergisi.***

41. Kavzaoğlu T, Çölkesen İ. "Karar Ağaçları ile Uydu Görüntülerinin Sınıflandırılması: Kocaeli Örneği", ***Electronic Journal of Map Technologies***, 2, 2010.

42. Özcan C, Ersoy O, Oğul İÜ. "Fast texture classification of denoised SAR image patches using GLCM on Spark", ***Turkish Journal of Electrical Engineering & Computer Sciences***, 28, 2020.

43. Man W, Ji Y, Zhang Z. "Image Classification Based on Improved Random Forest Algorithm", ***2018 the 3rd IEEE International Conference on Cloud Computing and Big Data Analysis***, 20-22 April 2018, Chengdu, China.

44. Cortes C, Vapnik Vladimir. "Support-Vector Networks", ***Machine Learning***, 20, 273-297 (1995).

45. Gonzalez RC, Woods RE, Eddins SL**.** ***Digital Image Processing using Matlab***. New Jersey, Prentice Hall, 2003.

46. Kaya Ç, Yıldız O. "Makine Öğrenmesi Teknikleriyle Saldırı Tespiti: Karşılaştırmalı Analiz". ***Marmara Fen Bilimleri Dergisi.***

47. Kavzaoğlu T, Çölkesen İ. "Karar Ağaçları ile Uydu Görüntülerinin Sınıflandırılması: Kocaeli Örneği", ***Electronic Journal of Map Technologies***, 2, 2010.

48. Özcan C, Ersoy O, Oğul İÜ. "Fast texture classification of denoised SAR image patches using GLCM on Spark", ***Turkish Journal of Electrical Engineering & Computer Sciences***, 28, 2020.

49. Man W, Ji Y, Zhang Z. "Image Classification Based on Improved Random Forest Algorithm", ***2018 the 3rd IEEE International Conference on Cloud Computing and Big Data Analysis***, 20-22 April 2018, Chengdu, China.

50. Cortes C, Vapnik Vladimir. "Support-Vector Networks", ***Machine Learning***, 20, 273-297 (1995).

51. Gonzalez RC, Woods RE, Eddins SL**.** ***Digital Image Processing using Matlab***. New Jersey, Prentice Hall, 2003.

52. Kaya Ç, Yıldız O. "Makine Öğrenmesi Teknikleriyle Saldırı Tespiti: Karşılaştırmalı Analiz". ***Marmara Fen Bilimleri Dergisi.***

53. Kavzaoğlu T, Çölkesen İ. "Karar Ağaçları ile Uydu Görüntülerinin Sınıflandırılması: Kocaeli Örneği", ***Electronic Journal of Map Technologies***, 2, 2010.

54. Özcan C, Ersoy O, Oğul İÜ. "Fast texture classification of denoised SAR image patches using GLCM on Spark", ***Turkish Journal of Electrical Engineering & Computer Sciences***, 28, 2020.

55. Man W, Ji Y, Zhang Z. "Image Classification Based on Improved Random Forest Algorithm", ***2018 the 3rd IEEE International Conference on Cloud Computing and Big Data Analysis***, 20-22 April 2018, Chengdu, China.

56. Cortes C, Vapnik Vladimir. "Support-Vector Networks", ***Machine Learning***, 20, 273-297 (1995).

57. Gonzalez RC, Woods RE, Eddins SL**.** ***Digital Image Processing using Matlab***. New Jersey, Prentice Hall, 2003.

58. Gonzalez RC, Woods RE, Eddins SL**.** ***Digital Image Processing using Matlab***. New Jersey, Prentice Hall, 2003.

ÖZGEÇMİŞ

Betül DOLAPCI 1991 yılında Konya'da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı. Meram Atatürk Mesleki ve Teknik Anadolu Lisesi Bilişim Teknolojileri Bölümü'nden mezun oldu. 2009 yılında Selçuk Üniversitesi Teknik Bilimler Meslek Yüksekokulu Bilgisayar Programcılığı Bölümü'nde öğrenime başlayıp 2011 yılında mezun oldu. 2013 yılında Karabük Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği (İngilizce) Bölümü'nde öğrenime başlayıp 2017 yılında mezun oldu. 2017 yılında Karabük Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda başlamış olduğu yüksek lisans programını, Karabük Üniversitesi Lisansüstü Eğitim Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı altında sürdürmektedir. 2019 yılında Kastamonu Üniversitesi Bilgisayar Mühendisliği Bölümü'nde araştırma görevlisi olarak göreve başladı ve halen aynı yerde çalışmaya devam etmektedir.

ADRES BİLGİLERİ

Adres : Kastamonu Üniversitesi
Mühendislik ve Mimarlık Fakültesi
Bilgisayar Mühendisliği Bölümü
Kuzeykent / KASTAMONU
Tel : (0366) 280 2979
E-posta : bdolapci@kastamonu.edu.tr