BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
DOKTORA PROGRAMI

VİDEOLARDA İNSAN EYLEMLERİNİ TANIMADA POLİGON TABANLI ÖZNİTELİK ÇIKARIMI

HAZIRLAYAN

OĞUL GÖÇMEN

DOKTORA TEZİ

ANKARA – 2023

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
DOKTORA PROGRAMI

VİDEOLARDA İNSAN EYLEMLERİNİ TANIMADA POLİGON TABANLI ÖZNİTELİK ÇIKARIMI

HAZIRLAYAN

OĞUL GÖÇMEN

DOKTORA TEZİ

TEZ DANIŞMANI

PROF. DR. MURAT EMİN AKATA

ANKARA – 2023

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik Elektronik Mühendisliği Anabilim Dalı Elektrik Elektronik Mühendisliği Doktora Programı çerçevesinde Oğul GÖÇMEN tarafından yapılan bu çalışma, aşağıdaki jüri tarafından Doktora Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 6 / Ocak / 2023

Tez Adı: Videolarda İnsan Eylemlerini Tanımada Poligon Tabanlı Öznitelik Çıkarımı

Tez Jüri Üyeleri (Unvanı, Adı - Soyadı, Kurumu)	**İmza**
Prof. Dr. Murat Emin AKATA, Başkent Üniversitesi	……
Prof. Dr. Mehmet Reşit TOLUN, Çankaya Üniversitesi	……
Prof. Dr. Hasan Şakir BİLGE, Gazi Üniversitesi	……
Doç. Dr. Ahmet Güngör PAKFİLİZ, Başkent Üniversitesi	……
Doç. Dr. Mustafa SERT, Başkent Üniversitesi	……

ONAY

Prof. Dr. FARUK ELALDI
Fen Bilimleri Enstitüsü Müdürü
Tarih : / Ocak / 2023

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLER ENSTİTÜSÜ

DOKTORA TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 6 / Ocak / 2023

Öğrencinin Adı, Soyadı : Oğul GÖÇMEN

Öğrencinin Numarası : 20411058

Anabilim Dalı : Elektrik Elektronik Mühendisliği

Programı : Elektrik Elektronik Mühendisliği

Danışmanın Unvanı/Adı, Soyadı : Prof. Dr. Murat Emin AKATA

Tez Başlığı : Videolarda İnsan Eylemlerini Tanımada Poligon Tabanlı Öznitelik Çıkarımı

Yukarıda başlığı belirtilen Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 107 sayfalık kısmına ilişkin, 28 / Aralık / 2022 tarihinde şahsım tarafından, Turnitin adlı intihal tespit programından, aşağıda belirtilen filtrelemeler uygulanarak, alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 7'dir.

Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası

ONAY

Tarih:……./ Ocak / 2023

Öğrenci Danışmanı: Prof. Dr. Murat Emin AKATA

……

Canım aileme

TEŞEKKÜR

Tez danışmanım Prof. Dr. Sayın Murat Emin AKATA'ya çalışmanın sonuca ulaştırılmasında ve karşılaşılan güçlüklerin aşılmasında, yardımcı ve motive edici yaklaşımı için; Prof. Dr. Sayın Nizami GASİLOV'a, makale yazım sürecinde düzeltmeler aşamasında harcadığı yoğun emek için; Doç. Dr. Sayın Mustafa Sert ve Doç. Dr. Sayın Ahmet Güngör PAKFİLİZ'e, Tez İzleme Komiteleri süresince yapıcı ve yol gösterici eleştirileri ile katkıları için teşekkür ederim.

Sevgili annem Filiz GÖÇMEN ve babam Korkmaz GÖÇMEN'e, bana olan inançlarını hiç yitirmedikleri için; Sevgili eşim Pelin Ö. GÖÇMEN'e ve dünya tatlısı evlatlarım Bilgin GÖÇMEN ve A. Bilge GÖÇMEN'e her zaman yanımda oldukları ve desteklerini hiç esirgemedikleri için teşekkür ederim. İyi ki varsınız.

ÖZET

Oğul GÖÇMEN

VİDEOLARDA İNSAN EYLEMLERİNİ TANIMADA POLİGON TABANLI ÖZNİTELİK ÇIKARIMI

Başkent Üniversitesi Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı
2023

Videolardaki insan eylemlerinin tanınması ve sınıflandırılması için, silüet tabanlı yeni bir öznitelik çıkarma yöntemi önerilmiştir. Bu amaçla, "Hareket Geçmişi Görüntüsü (HGG)" ve "Poli-Siluet (PoS)" adı verilen yeni görüntü şekilleri, yeni bir "Poligonlaştırma Algoritması (PoG)" ve yeni bir "Poligon Kodlama Algoritması (PoC)" geliştirilmiştir. Önerilen yöntem, "Silüet Videolarından (SiVi)" , HGG'lerin elde edilmesi, daha az köşeye indirgenerek poligon formuna dönüştürülmesi ve kodlanmasına dayanmaktadır.

Geleneksel siluet oluşturma yöntemleri siluetlerin gövde bütünlüğünü tek başlarına karşılayamadıklarından, bu sorunu aşmak için, görüntü ve videolarda yer alan insan ve nesneler üzerinde örnek bölütlemesi yapabilen Yolact++'ın kodları değiştirilerek, modelin ürettiği insan maskeleri ile SiVi'ler oluşturulmuştur. Hareketi daha iyi tanımlayan HGG'ler, ardışık olmayan SiVi karelerindeki silüetleri ağırlık merkezlerine göre toplanması ile elde edilir. Poligonlaştırma işlemi, HGG görüntülerine uygulanır.

PoG, eğri şekilleri ve görüntülerin poligonlaştırılmasına dayansa da, çalışması tam olarak bunlara benzemez. PoG, HGG'nin kontur koordinatlarını hızlı bir şekilde istenilen boyuta indirger. Elde edilen poligonlaştırılmış HGG görüntülerine PoS adı verilir. PoC algoritması, PoS görüntüsünü parametre olarak alır ve içerisindeki şeklin sol üst köşe koordinatından başlayarak, saat yönünün tersinde dolanırken, üzerinden geçilen her bir kenar ile bir vektör oluşturur. Bu esnada vektörün, çalışmada belirlenen sekiz adet açısal alandan hangisine iz düşümü olduğunu hesaplayarak, ilgili vektörün kodunu belirler. Bir PoS'un tüm kenar vektörlerinin kodları, dolaşılma sırasıyla dizi içine kaydedildiğinde, o vektörler ile oluşan PoS için poligon kodu elde edilir.

Uzun poligon kodları ile kolay işlem yapılabilmek ve farklı uzunlukta vektörler oluşturulabilmek için, genetik algoritmada kullanılan k-mer gruplandırma tekniği benzeri bir yaklaşım çalışmada geliştirilmiştir. Düşünülen işlem poligon kodunun normalizasyonunu sağlamakla kalmamış, aynı zamanda poligon kodunun adaptif bir çekirdeğe göre uyarlanmasını doğurarak çalışmada önceden düşünülmeyen, farklı bir boyuta bu çalışmayı taşındığı görülmüştür. Gruplama k boyu olarak farklı uzunluklar seçilmiş ve poligon kod grupları, k boyuna göre oluşabilecek en büyük kodlara bölünerek normalize edilmiştir.

Önerilen öznitelik çıkarma yöntemi, herhangi bir aksiyon videosunun karelerinden elde edilmiş PoS üzerinde eşit uzunlukta öznitelik vektörlerinin üretilmesini garanti eder, bu durum, boyut sorununun üstesinden gelmek için ekstra boyut indirgeme algoritmalarının kullanımına ihtiyacı ortadan kaldırır.

Farklı k-mer uzunlukları, çalışmada kullanılan veri setleri üzerinden elde edilen öznitelik vektörlerinde denenerek "işlem hızı" ile "sınıflandırma doğruluğu" karşılaştırmaları yapılmıştır. Yöntem, HMDB51 ve UCF101 veri setleri üzerinde denenmiş en iyi performans veren k-mer boyu ve sınıflandırma doğruluk sonuçları, diğer çalışmalarla karşılaştırılmış ve daha başarılı sonuçların elde edildiği görülmüştür. Ayrıca, çalışmaya özgü hazırlanan Yoga veri seti üzerinde HGG görüntülerinin farklı formları ile ek olarak derin ağ uygulamaları da gerçekleştirilmiş ve kayda değer cesaretlendirici sonuçlar alınmıştır.

ANAHTAR KELİMELER: Poligonlaştırma Algoritması, Poligon Kodlama, Videolarda İnsan Eylemlerinin Tanınması, Poligon Tabanlı Öznitelik Çıkarımı

ABSTRACT

Oğul GÖÇMEN

POLYGON BASED FEATURE EXTRACTION FOR HUMAN ACTION RECOGNITION IN VIDEOS

Başkent University Institute of Science

Department of Electrical and Electronics Engineering

2023

A new silhouette-based feature extraction method for the recognition and classification of human actions in videos is proposed. For this purpose, new image forms called “Motion History Image (HGG)” and “Poly-Silhouette (PoS)”, a new “Polygonization Algorithm (PoG)” and a new “Polygon Encoding Algorithm (PoC)” have been developed. The proposed method is based on obtaining HGGs from “Silhouette Videos (SiVi)”, converting them to polygon form with fewer corners and encoding them.

Since traditional silhouette creation methods cannot meet the body integrity, to overcome this problem, the codes of Yolact++ were changed and human masks produced by the model is saved as SiVis. HGGs that better describe motion have been obtained by summing the silhouettes in non-consecutive SiVi frames according to bounding box centroids. The polygonization process has been applied to the HGG images.

PoG quickly reduces the contour coordinates of the HGG to the desired size. The resulting polygonized HGG images are called PoS. The PoC algorithm takes the PoS image as a parameter and creates a vector with each edge, starting from the upper left corner coordinate, while tracing in counterclockwise direction. It calculates the related vector code by projecting the vector on to the area that are presented in the study. When the codes of all edge vectors of a PoS are recorded, the polygon code for the PoS is formed.

An approach similar to the k-mer grouping technique in genetic algorithms has been developed in order to be able to process with long polygon codes and to create feature vectors of different lengths. This idea not only provided for the normalization of the code,

but also for the adaptation of the code according to an adaptive kernel. Different lengths were chosen as k and groups were normalized by dividing them into the largest codes according to the k length.

The proposed feature extraction method guarantees the generation of feature vectors of equal length on PoS, eliminating the need for the use of extra dimension reduction algorithms to overcome the size problem.

Different k-mer lengths were tested on HMDB51 and UCF101 datasets and comparisons of "processing speed" and "classification accuracy" were made. The results of best k-mer and classification accuracy were compared with studies and it was found that more successful results were obtained. In addition, extra deep network applications were performed out on the yoga dataset created specifically for the study, and remarkable encouraging results were obtained.

KEYWORDS: Polygonization Algorithm, Polygon Coding, Recognition of Human Actions in Videos, Polygon Based Feature Extraction

ÖNSÖZ

Bu tez çalışmasında, günümüzün popüler ve karmaşık konularından biri olan insan eylemlerinin, videolarda otomatik tanınması ve sınıflandırılması ile ilgili bir çalışma gerçekleştirilmiştir.

Çalışma içerisinde bu amaç uğruna kullanılabilecek yenilik içeren birden fazla yaklaşım sergilenmiş, önerilen özgün algoritmalar ve özgün öznitelik çıkartım yöntemi, diğer akademik çalışmalarda kullanılan ve literatürde iyi bilinen büyük video veri setleri üzerinde denenip, başarılı sonuçlar alınmıştır.

Tez çalışması içerisinde aynı zamanda çok katmanlı yapılar, derin ağ mimarileri ile ilgili çalışmalar da yapılmış ve tez için özel olarak hazırlanan YouTube üzerinden toparlanan yoga hareketlerini barındıran bir veri seti üzerinde denenmiştir. Bu denemelerde de başarılı sonuçlar elde edilmiştir. Yoga veri seti akademik kullanım için GitHub üzerinden paylaşılmıştır.

Ocak, 2023

Oğul GÖÇMEN

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

BoW Bag Of Words (Görsel Kelime Çantası)
CCTV Closed-Circuit Television **(**Kapalı Devre Televizyon)
DVM Destek Vektör Makineleri
HGG Hareket Geçmişi Görüntüsü
HOF Histogram Of Flow – (Akış Histogramı)
HOG Histogram Of Oriented Gradients – (Yönlü Gradyanlar Histogramı)
k-NN k-Nearest Neighbour – (k-En Yakın Komşuluk Algoritması)
PCA Principle Component Analysis – (Temel Bileşen Analizi)
PoC Poligon Kodlama Algoritması
PoG Poligonlaştırma Algoritması
PoS Poli-Siluet (Poligonlaştırılmış Siluet)
SIFT Scale Invariant Feature Transform (Ölçekten Bağımsız Öznitelik Dönüşümü)
SURF Speed-Up Robust Features (Hızlandırılmış Gürbüz Öznitelikler)
SiVi Silüet Videosu

1. GİRİŞ

Rus fizikçi Leon Theremin tarafından 1927 yılında geliştirilen CCTV kameralar, o dönemde askeri amaçlı gözetleme sistemi olarak kullanılmaya başlanmış; Siemens firması tarafından geliştirilerek fırlatılan füzelerin gözetlenmesi amacıyla, 1942 yılında Almanya tarafından kullanılmış ve ilk ticari modeli ile 1949 yılında günlük kullanıma girmiştir[1]. Güvenlik kameraları, yıllar ilerledikçe insanların gözetlenmesi amacı ile de kullanılmaya başlanmıştır. Görüntülerdeki insan hareketlerinin tanınması, analizi amaçlı kullanım şekline de ilk 80'li yıllarda başlandığı görülmektedir.

Ülkelerin, artan nüfusları ile beraber şehirler genişlemekte; irili ufaklı yerleşim merkezlerinin, inşa edilen binaların, cadde ve sokakların sayısı her geçen gün giderek artmaktadır. Kalabalıklaşan toplulukları denetlemek, onların güvenliğini sağlayabilmek daha da zor hale gelmektedir. Bu yüzden günümüzde hem açık hava hem de kapalı ortamlarda kullanılan güvenlik kameralarının sayısı artmaktadır. Bir olay olması durumunda kaydedilen görüntülerin içeriklerinin denetlenmesi de kaydedilen medyanın dosya boyutunun büyümesinden dolayı zaman almaktadır. Artık bu gibi bir durum olduğunda geçmişe dönük arşivleri incelemek ile zaman harcamak yerine, akıllı sistemler kullanarak bu incelemelerin yapılması yaygınlık kazanmıştır [2].

İlerleyen teknoloji ve insanlığın artan bilgi birikimi, bu sistemleri daha da akıllı hale getirecek gelişmiş kontrol yazılımlarının üretimini sağlamaktadır. Bu sistemlerin insanlar yerine karar verebilecek önemli görevlerde yer alması, insan faktörlü hataların azalmasına neden olmuştur. Güvenlik senaryolarında insan hareketlerinin analizini yapmak demek, normal insan aktivitelerine göre, o ortam içinde insanların yaptığı anormal aktiviteleri tespit etmek demektir. Örneğin, bir alışveriş merkezinde normal denebilecek insan aktivitesi yürüme hareketi iken, ortamda koşan bir insanın olması, gözetleme sistemine göre normal olmayan bir aktivite olarak nitelendirilmelidir. İnsan hareketlerinin bir sistem ile analiz edilmeye çalışılması, problemdeki parametrelerin çokluğu ve hareketlerin serbestlik derecesinin fazla olmasından dolayı oldukça zor bir problemdir.

Hareket tanıma sistemlerinin günümüzde sadece güvenlik amacıyla kullanılmadığı; spor, sağlık ve eğlence gibi insanların farklı amaçlarına hizmet etmek amacıyla da kullanıldığı görülmektedir. Mesela, sağlık sektöründe fizik tedavi uygulamalarında

egzersizleri yapan hastanın, vücut pozlarının doğruluğunun denetimi, günümüzde kameralı sistemler ile takip edilebilmekte ve hareketin ne denli doğru yapıldığına dair raporlar üretebilmektedir [3], [4]. Yalnız yaşayan, bakıma muhtaç kişilerin bulundukları ortam içindeki hareketlerinin kameralarla anlık gözetlenmesi, acil durum gerektiren vakalarda, örneğin düşme, bayılma gibi durumlarda, müdahalenin anında yapılabilmesi için önemlidir [5].

Spor alanında yine benzer uygulamalar bulunmaktadır. Hatta spor karşılaşmalarında oyuncuların performans istatistikleri yine görüntü analizi gerçekleştirilerek yapılabilmektedir [6].

Gelişen eğlence sistemlerinde çeşitli tipteki kameralarla takip edilen insanlar, bedenleri, pozları, uzuv hareketleri ile fiziksel enerji harcayarak sanal dünya içerisindeki karakterlerini yönetmekte, bu da beraberinde evden çıkmamaya başlayan topluma egzersiz yaptırmasını sağlarken, oyuncuların sanal karakterlerine hareket kazandırmaktadır [7], [8], [9].

Kameralardan alınan canlı görüntülerin yanı sıra, kaydedilmiş video görüntüleri üzerinde de analiz çalışmalarının yapıldığı bilinmektedir [10], [11]. Bu çalışmaların çoğunda, vücut eklem yapısı ve eklemlerin serbestlik derecesinin yüksek olmasından dolayı yapılacak veya tanınacak insan hareketlerine sınırlamalar getirildiği görülmektedir [12], [13].

Videolardaki insan hareketlerinin tanınmasında üstesinden gelinmesi gereken birçok konu mevcut olup, bunlar hareketin formunun dışındaki unsurlar da olabilir. Örnek vermek gerekirse, hareketin tipinden bağımsız olarak, vücut pozunu elde edebilmek ve analizi gerçekleştirmek için, video arka zemini ile ön zeminin (insanın) birbirinden ayrıştırılması, yani arka planın çıkartılması gerekebilir [14], [15].

Videoda arka planın çıkartılması işlemine arka plan kestirimi denmektedir. Bu işlemde, videonun arka planında hareketli olup olmaması veya ortamda birden fazla insanın hareket edip etmemesi, hareketi görüntülemiş olan kameranın sabit veya hareketli olması gibi aşılması gereken zorluklar bulunmaktadır [16]. Yapılan hareketin hızı [17], hareketin kendini örten kısımlarının olup olmadığı [18], insan anatomisinin birbirinden farklılığı, hareketlerin aynı şekilde aynı kişi tarafından bile tekrar aynı şekilde yapılamayışı, elbise seçimindeki çeşitlilik, kameranın görüntüyü çekerken duruş açısı,

ortam ışıklandırması vb. gibi unsurlar bu problemin çözümüne ulaşılması için, aşılması veya kısıtlanması gereken engellerden bazılarıdır.

1.1. Motivasyon ve Amaç

Günümüzde dijital içeriklerin zenginleşmesi, işlenen ve depolanan bilginin boyutunun ve miktarının artması, bilginin kolay ve hızlı yayılabilir olması, bu verilerin denetiminin ve kontrolünün bir insan tarafından yapılabilir olmasını zorlaştırmaktadır. Bu amaçla, aranan bilginin veri yığını içerisinden otomatik tespit edilmesi ve bu işin insan gücü olmadan yapılması, günümüzde tercih edilir çözüm yolu olarak kabul görmektedir. Depolanmış çoklu ortam verisinin tipine göre kullanılabilecek birçok yöntem günümüze kadar gelişerek gelmiştir. Gelişen bilgisayar donanımlarıyla, bu verilerin analizinin yapılarak anlam çıkartılması ve büyük veri analizlerinin insan faktörü olmadan gerçekleştirilebilmesi mümkündür. Fakat belirtilen işlemleri canlı kamera kayıtları üzerinde gerçekleştirebilmek ya da kaydedilen videoları sonradan analiz etmek, yapılacak işlemler için sınır koyulmadığı sürece kolay olmayan bir hale dönüşmektedir. Bir de videolarda aranan kaynağın insan olması, insan bedeninin ve yapabileceği hareketlerin serbestlik derecesinin fazlalığı, bu analizleri daha da zor hale sokmaktadır.

Bu tez çalışmasının motivasyonu, yukarıda belirtilen ihtiyaçlara zaman içerisinde bulunan çözümler arasında olan siluet tabanlı yaklaşımların eksikliklerini gidermek, klasik yöntemler ile elde edilen siluetleri daha iyi tanımlamaktır. Bunun yanı sıra çözüm yönteminde günümüzün gelişmiş derin mimarilerini kullanarak, yeni bir yaklaşım ile bu yapılarla entegre olabilecek bir sistem inşa edebilmek hedeflenmiştir.

Bu tez çalışmasının amacı, insan hareketlerini içeren videolarda, hareket tanıma ve sınıflandırma esasına dayalı bir işlemi gerçekleştirmektir. Bu bağlamda video içeriklerinden elde edilen silüetler, poligon tabanlı yeni bir öznitelik çıkartma yöntemi uygulanarak gerçekleştirilmiştir.

1.2. Hipotez

Literatür araştırmaları sonucu insan eylemlerinin olduğu videolarda farklı hareket tiplerini tanımak ve sınıflandırabilmek için birçok yöntem kullanıldığı bulgulanmıştır.

Siluet tabanlı yöntemler de bunlardan birisidir. Fakat incelenen çalışmalarda, bu yöntemin her zaman tek başına kullanılmadığı görülmüştür. Örneğin optik akış gibi başka yöntemler ile sonuçların desteklendiği ve bu yüzden de mevcut sistemlerin performanslarının belli bir kısmının, diğer yöntemlerle desteklenmek amacıyla harcandığı düşünülebilir.

Bu araştırmaya, "İnsan eylemlerinin olduğu videolarda, farklı hareket tiplerini tanıyabilmek ve sınıflandırabilmek için, poligonlaştırılmış siluetlerden çıkartılan poligon tabanlı öznitelik ile hareketler birbirinden ayrılabilir ve sınıflandırılabilir." hipotezi ile başlanmıştır.

Yapılan araştırmada, kişilerin vücut pozlarının, farklı hareket tiplerinin ayrımında daha belirleyici olduğu görülmüştür. Hareketin tamamı izlenmese de, o harekete ait bir kare görüntüden bile içeriğin, yani hareket tipinin anlaşılabiliyor olması, bu çalışmanın ilgi alanını pozlar üzerine kaydırmıştır. Bu pozlar kullanılarak insan hareketlerini daha iyi tanımlayabilecek bir sistem oluşturulmaya çalışılmıştır.

Bu tez çalışmasında, insan siluetinin videolardan yeni bir yaklaşım ile çıkartılmasını; hareketi daha ayrık temsil edebilecek yeni bir görüntü tipinin oluşturulmasını; silueti çevreleyen kenar koordinatlarının, önerilen algoritma ile poligonlaştırılmasını ve bu poligonu oluşturan kenarları yönlü bir doğru parçası (vektör) gibi ele alıp yeni bir yaklaşım ile bu poligonların kodlanmasını; kodlanan bu bilginin farklı uzunluklarda gruplanarak hız ve performans dengesi yakalanabilecek bir sonuç çıkarılmasını sağlayan bir dizi yöntem önerilmiş ve önerilenler çalışmada ele alınan üç veri seti üzerinde gerçekleştirilmiştir.

1.3. Problemin Tanımı

İnsan vücudunun ve yapabileceği hareketlerinin serbestlik derecesinin çokluğundan dolayı, çalışmaya hareket konusunda belirli kısıtlar getirilerek başlanmış ve bir kaç yoga hareketinin sabit kamera ve sabit arka planlı sahnelerde tanınması ile sınırlandırılmıştır. Bu amaç doğrultusunda yoga hareketlerinden oluşmuş bir video veri seti arayışına başlanılmıştır. Fakat bulunan veri setlerinin içerikleri ya belirlenen kısıtlara uymamış ya da yeterince zengin olmadığı görülmüştür. Bu yüzden tez çalışması içerisinde özgün bir "Yoga Hareket Veri Seti" oluşturulmuştur. Veri seti, YouTube üzerinden kaydedilen, farklı insanların yaptığı yoga hareketlerini içermektedir.

Araştırma ve geliştirme sürecinde bu kısıtlar ile ilerlense de çözüm için önerilen yöntemin diğer akademik çalışmalarla kıyaslanabilir olması için literatürde kullanılan ve hareket tipinde, sayısında, arka plan koşulunda, kamera çekiminde, renklerde, vb. hiç kısıtları olmayan veri setleri üzerinde de yürütülmesi gerektiği anlaşılmıştır.

Bu veri setleri çok büyük boyutlu olduğundan, evrişimli ağ uygulamaları çevrimiçi sistemler üzerinden yürütülmüştür. Bu bağlamda, HMDB51 ve UCF101 video veri setleri üzerlerinde yöntemimiz Colab üzerinden denenmiş, elde edilen sonuçlar k-NN ve DVM gibi sınıflandırıcılarla sınıflandırılmıştır. Literatürdeki aynı veri setlerini ve aynı sınıflandırıcıları kullanan diğer çalışmalarla sonuçlar karşılaştırılmış ve elde edilen iyileştirilmiş katkılara göre de makaleler hazırlanmıştır.

Yöntemin başarısı hem kendi oluşturduğumuz veri seti üzerinde, hem de akademik çalışmalarda sıkça rastlanan veri setleri üzerinde denenmiş ve yöntemin genel anlamda her tip video ve hareket üzerinde çalıştığı ispatlanmıştır.

1.4.Tez Çalışmasının Bilime Katkıları

Tezde uygulanan yöntemin bilime katkıları maddeler halinde aşağıda belirtilmiştir. Bunlar;

- Çalışmada siluet çıkartma yöntemi olarak klasik yöntemlerin kullanılması yerine yeni bir yaklaşım düşünülmüştür. Bir görüntü bölütleme derin ağ modeli olan Yolact++'ın ürettiği maskeler, modelin kodları yeniden şekillendirilerek, sadece insan nesnesini algılamasını, arka plandan ayrıştırmasını ve insan maskelerini bir siluet görüntüsü olarak kaydetmesini sağlayacak şekilde değiştirilmiştir. Bu sayede klasik yöntemlerle oluşturulan siluet görüntülerindeki eksiklikler giderilerek, siluet görüntülerinin gövde bütünlüğü sağlanmıştır. Yeniden şekillendirilmiş Yolact++ kodu, GitHub üzerinden genel kullanıma açılarak, siluet görüntüsü ile ilgili çalışmalar yapmak isteyen kişilerin kullanımına sunulmuştur.

- Siluet görüntüsünün poligonlaştırılması aşamasında, yeni bir algoritma önerilmiştir. Bu öneri, ihtiyaca göre uyarlanabilir bir poligonlaştırma algoritmasıdır. Bu algoritma ile herhangi bir ikilik görüntü (siluet), istenilen kenar ve köşe sayısı ile yeniden ifade edilebilmektedir. Yöntem, eğri veya görüntü poligonizasyonuna dayansa da onlar gibi çalışmamaktadır. Hedef poligon sayısı algoritmada seçilebilmekte ve bu işlem gerçek zamanlı kenar ve köşe boyutu indirgeme hızında yapılabilmektedir. Yöntemin

oluşturduğu poligon formu sadece tez çalışmasında değil, sanat ile ilgili alanlarda da örneğin animasyon oluşturma gibi bir amaçla kullanılabilir. Haritacılık alanında yeryüzü şekillerinin poligonla ifadesi çok sık kullanılan gösterim şekilleri arasındadır. Bu algoritma yeryüzü haritalarının hedef poligon sayısında ifade edilmesinde kullanılabilir.

- Çalışmada önerilen poligonlaştırma aşaması, önerilen yeni öznitelik çıkartma aşaması için bir ön işlemdir. Poligonlaştırma ve kodlama ikilisi bir araya geldiğinde olaylar, şekil içerisinde şekil arama gibi bir işlemi, metin içerisinde metin arama gibi bir işleme dönüşmekte ve bu da bu çalışmanın çıktısını farklı amaçlar için farklı doğruluk ve hıza uyarlanabilir kılmaktadır. Yöntem, bir derin ağ mimarisi omurgasını kullandığından, tezde önerilen öznitelik çıkartma adımının, destek aldığı mimariye entegre edilmesi durumunda bu işlem modelin saniyede 33fps'lik performansından sonuna kadar yararlanacaktır. Bu entegrasyon, gerekli donanım imkanlarının olmayışından dolayı yapılamamıştır.

- Çalışma içerisinde bir başka yenilik içeren yaklaşım da, poli-siluet adı verilen, hareketi daha iyi tanımlayan görüntü şekillerinin üretilerek, özniteliklerin bu görüntüler üzerinden çıkartılmasıdır. Polisiluet'ler, ardışık olmayan video kareleri içerisinden saniyedeki video kare sayısı süresinin yarısındaki karelerde üretilmiş maskeler (siluetler) ile elde edilir. Bu maskeler, Şekil 1.1.'de gösterildiği gibi ağırlık merkezlerine göre toplanarak ve poligonlaştırılarak polisiluet adı verilen görüntü tanımları oluşturulur. Polisiluet'ler anahtar pozlar olup, birbirine benzeyen eylem pozlarını, birbirinden ayırt etmekte büyük fayda sağlar. Bu yaklaşım videolarda eylem sınıflandırması konulu çalışmalarda siluet tabanlı yöntemi kullanan çalışmalar tarafından tercih edilebilir.

(a) (b) (c) (d)

Şekil 1.1. Ağırlık merkezine göre toplama işlemi (a)'dan (c)'ye kadar olan maskeler (siluetler), ağırlık merkezine göre toplanır ve (d) elde edilir,

- Çalışmada yeni önerilen poligon kodlama tekniği, poligonun kenarlarını oluşturan doğru parçalarının, yönlü doğru parçaları şeklinde, yani vektörler ile yeniden ifade edilmesiyle başlar. Poligon şeklinin sol üst köşesinden, saat yönünün tersi yönünde kenarlar üzerinde gidilirken, poligona ait köşe koordinat değerlerinin ardışık ikilileri kullanılarak kenar vektörleri oluşturulur. Bu vektörlerin, sekiz alana ayrılmış Kartezyen koordinat sistemi üzerindeki kırk beş derecelik dilimlerden oluşan alanlara iz düşümleri hesaplanarak vektör kodları bulunur. Şeklin sol üst köşesinden başlanıp, saat yönünün tersi yönde şeklin kenarında gezinilirken, üzerinden geçilen her kenar vektörünün kodu, başlangıç noktasına ulaşılana kadar sırayla bir dizi içerisinde saklanır. Kenar vektörü sayısı kadar basamağı olan bu sayı dizisi, o poligonun "poligon kodunu" oluşturur. Her farklı poligon şekli için oluşturulacak kod farklıdır. Öznitelik vektörü bu kod kullanılarak oluşturulur. Önerilen yöntemin, yeni bir öznitelik çıkartma yöntemi olup, insan hareketlerini içeren videoların başarılı bir şekilde sınıflandırılması için yeterli miktarda bilgi içerdiği çalışmada gösterilmiştir. Kodlama sisteminin bir getirisi de oluşturulan öznitelik vektörlerinin sınıflandırma algoritmaları tarafından direkt kullanılabilir olmasıdır. Normal şartlar altında elde edilen öznitelik vektörleri, boyut indirgeme algoritmalarından geçirilerek sistemleri çıkmaza sokan boyut lanetinden kurtarılarak kullanılır. Fakat önerdiğimiz çözüm yönteminde bu tür bir ihtiyaç doğmamakta, boyut indirgeme algoritmalarına ihtiyaç kalmamaktadır. Bu da ek bir algoritma çalıştırılmasına gerek duyulmadığından süreci hızlandırmaktadır.

- Yöntemimizin bir getirisi de, sınıflandırıcıların girdi olarak aldıkları bilginin her farklı örnek için eş uzunlukta olması gerekliliğidir. Önerdiğimiz poligonlaştırma ve poligon kodlama algoritmaları, farklı siluet görüntüleri için eşit ve istenilen boyda kod üretilmesini garanti eder. Bu da beraberinde çıktıların her farklı tip sınıflandırıcı ile rahatlıkla kullanılmasını sağlar.

- Bir başka yenilik, oluşturulan kod boyunun gruplama yapılarak boyutu ve hassasiyetinin uyarlanabilir olarak değiştirilebilmesidir. Öznitelik çıkartılırken poligon kodun içerisindeki 3, 4 ve 5 birimlik tüm olası kod sekanslarının poligon kodun içerisinde bulunma sıklığı tespit edilir ve öznitelik vektörü gruplanmış bu değerler ile yaratılır. Bu yaklaşım, öznitelik vektörü oluşturulduktan sonra da sınıflandırma performansı üzerindeki hız denemelerinin gerçekleştirilebilir olmasını

etkin kılmıştır. Çalışmada önerilen poligon kodlama tekniği içerisinde genetik algoritmalarda kullanılan bir yaklaşıma benzer bir uygulama yapılmış ve kod dizileri 3'lü, 4'lü ve 5'li olacak şekilde daha küçük parçalarda gruplanarak öznitelik vektörleri oluşturulmuştur. Bu gruplamanın poligon kodları üzerinde kullanılması, sınıflandırma için yüksek öneme sahip normalize edilmiş veri oluşturma işleminin gerçekleştirilebilmesini sağlamak amacı ile ilk olarak düşünülmüş olup, devamında farkı amaç ile (hız vs. başarım performansı) kullanılabileceği keşfedilmiştir. Poligon kodu aynı gen dizileri gibi uzun N basamaklı bir sayı dizisidir. Bu dizinin elemanlarının genetik algoritmada kullanılan gruplama yöntemi olan k-mer gruplaması şeklinde daha küçük parçalara ayrılması ile normalizasyon işlemi uzun kod dizisi üzerinde gerçekleştirilebilmiştir. Bu yaklaşımın, aynı zamanda kod dizisinin içerisinde kod parçalarının bulunma sıklık frekansını hesaplayabilmek için arama işlemlerini kolaylaştırdığı görülmüştür. Normalizasyon işlemi, beraberinde sınıflandırma başarımını yükseltmektedir. Yaklaşım aynı zamanda farklı işlem hızı - performans ikilisi seçeneklerinin yapı üzerinde denenebilir ve karşılaştırılabilir olmasını sağlamaktadır. Büyük gruplamalar kod dizisini küçülttüğünden işlem hızını arttırmaktadır. Ama diğer yandan başarımı da belli bir yüzdede azaltmaktadır. Bu ikili uygun veya ihtiyaca göre ayarlandığında, işlem hızları %87 orana kadar hızlandırılabilmekte buna karşın başarım yüzdeleri sadece %1.96 civarında düşmektedir. (Bknz. UCF101 k-NN sınıflandırıcısı sonuçları.) Yöntem, Yolact++ gibi, gerçek zamanlı çalışabilen derin ağ modellerine adapte edilebilecek şekilde parametrik oluşturulmuştur.

1.5. Literatür Taraması

Gerçek videolardan insan siluetini ve vücut kinematiğini çıkarmak zordur. Çünkü videolardaki insan hareketlerine ait siluetler ve vücut uzuvlarının kinematiği hareket sürecinde sürekli değişim gösterir [35], [36]. Bu yüzden hareket tanıma yaklaşımlarında görünüş tabanlı ve hareket tabanlı metotlar genellikle birbirini desteklemek amacı ile beraber kullanılır. Görünüş tabanlı metotlarda hareket, iki boyutta, vücut pozlarının serisi olarak incelenir. Tek bir postür (poz veya duruş), arka zemini çıkartılmış ve çeşitli ön işlemlerden geçirilmiş görüntülerdeki insan siluetlerinden veya siluetin dış hattındaki sınırlarından oluşan iki boyutlu, ikilik görüntü olarak tanımlanır [19, 21, 22, 33, 34].

Hareketler için çıkartılacak anahtar pozlar sistemin eğitim kısmı için önem taşımaktadır. Anahtar poz, bir hareketi diğer hareketten farklı kılan, o hareketin ayırt edici pozudur [32]. Sistem ne kadar tanımlayıcı (ayrıklık bilgisi) bilgi ile eğitilirse, çıkarsamadaki (gruplandırma/tanıma) becerisi o kadar doğru olacaktır. Hareketler için çıkartılacak anahtar pozlar ile eğitilmiş olan sistemler, yeni gelen poz bilgisini çeşitli yöntemlerle sınıflandırarak bilgi ile eşleştirip hareketin ne hareketi olduğunu tanımaya çalışılır. Literatürde kullanılan tanıma/sınıflandırma yöntemlerinden bazıları; destek vektör makineleri, çok sınıflı destek vektör makineleri, öz düzenleyici haritalar (som) [20], maksimum benzerlik, gizli markov modelleridir [28, 30].

Poz tabanlı metotların hepsi kameranın görüş açısına, kişinin dış görünüşüne, hareketli nesnenin görüntüdeki boyutsal değişimi gibi unsurlara karşı duyarlıdır. İki boyutlu yaklaşımların içerisinde optik akış yöntemi (hareket tabanlı) yani hareketin yönlü değişim hızını (hareketin gradiyentini) inceleyerek çözüm arayan yöntemler popülerdir. Optik akış yöntemi her ne kadar hareketi yapan kişinin pozundan etkilenmese de akış hesabındaki hatalardan, genel görüntüdeki istenmeyen ortam gürültüsünden çok etkilenmektedir [28]. Bunun yanı sıra büyük hareketlerle başa çıkamaması ve görüntüde hareket içeren bölgelerin birden fazla nesne içerme ihtimali olması, bu yöntemin tek başına kullanılma dezavantajları arasında sayılabilir. Optik akış yöntemi literatürde yer alan çalışmalarda daha çok yapılan hareketin ne olduğunun tespiti için değil, sahnede hareketin nerede yapıldığının yani "nesnelerin hareketinin algılanması" için çoğunlukla kullanılmaktadır. Yukarıda anlatılan eksikliklerinden dolayı optik akış yöntemi hareket tanıma sistemlerinde doğrudan kullanılmayıp, görüş tabanlı yöntemlerin eksik kaldığı noktalarda çözüme ulaşılmak için diğer yöntemleri destekleyecek şekilde kullanılmaktadır [25, 32]. Tanınacak hareketlerin iki boyutlu şekillerinin birbirine benzemesi, fakat yapılış hızlarının farklı olması örneğin bir görüntüde yapılan hareketin, koşma veya yürüme hareketi olduğunun kestirimi esnasında, hareketin hızının da çözüme getirisinin olabileceği göz önünde bulundurularak optik akış yönteminden faydalanılabilir [28, 31, 37].

İnsan aktivitelerinin dinamik karakteristiği ve yapısının beraber kullanımı hareket tanıma için önemlidir. Her ne kadar literatürde yapılan birçok çalışmanın "tek insan, tek hareket ve basit arka zemin" [26, 27] üçlüsünde yoğunlaştığı görülse de bu üçlü gerçek dünyanın sadece çok az bir bölümünü özetlemektedir. Fakat bu tür çalışmalar için hazırlanan genel amaçlı karşılaştırma veri setlerinin yukarıda belirtilen formda hazırlanmış

olması ve karmaşık hareketler içeren setlerin işlenmelerinin çok zaman alması göz ardı edilemez bir gerçektir. Çok açıdan görüntüleme yöntemlerinin [29] uygulanabilmesi için teknik alt yapı yeterliliğinin sağlanması gerekir. Yapılan hareketin her açıdan görüntülenebilmesi, hareketin yapılması esnasında harekete ait görüntülenemeyen kısımlarının olmamasını sağlar. Vücut uzuvlarının hareket esnasında birbirini örterek engellemesi veya görüntülenen açıya göre uzvun görülememesi gibi bir problem yaşanmaz.

Literatürde hareketin birden fazla kamera ile görüntülenmesi, stereo kameralarla görüntülenmesi [23, 25, 30], hareketi yapan kişiye hareket yakalama sensörlerinin takılması [24], hareketi yapan kişi üzerine eklem referans noktalara renkli işaretleyiciler konulması sayesinde üç boyutlu yöntemler ve çok açılı görüntüleme yöntemleri uygulanabilir.

Genelde hareket yakalama veri setleri çalışmalara özel olup, açık paylaşımlı kaynakları bulunmamaktadır. Aynı zamanda kurulu bir donanım sistemi olduğundan dolayı literatürde yapılan ve ortak veri tabanları kullanan sistemlerin, eğitim amaçlı kullandıkları videoları kullanmadıkları için, yapılan çalışmaların kendi içlerinde tutarlılığı olup, ortak veri tabanları kullanan çalışmalarla test aşamalarının kıyaslanmaması gereklidir. Hatadan mümkün olduğunca arındırılmış olan bu steril sistemlerin hareket yakalama sayısal verileri ile eğitilmiş olması, aynı zamanda başarı yüzdelerini de arttırmaktadır. Pahalı donanımlar gerektiren bu yapıların yaygın kullanımı pek mümkün değildir.

1.6.Tez Çalışmasında Kullanılan Donanım ve Üretilen Yazılım

Görüntüye bağlı bu tür çalışmalarda son zamanlarda sıklıkla kullanılan derin ağ mimarilerinin başarımları, klasik yöntem ve sınıflandırma algoritmalarına göre çoğunlukla yüksektir. Günümüz donanımlarının grafik kartları ve işlemcilerinin gelişmesi, bu işlemlerin artık bilgisayarın işlemcisi yerine grafik kartlarının işlemcileri kullanılarak yapılmaya başlanmış, hatta bu donanımlara birden fazla grafik kartı bağlayıp iş yükü dağıtılarak daha da büyük veri setleri işlenebilir kılınmıştır.

Bu tez kapsamında elimizde olan donanım 2GHz'lik i7 işlemci ve 12GB RAM'i olan, grafik kartı bulunmayan bir Mac Mini Server olup, içerisinde Intel HD Graphics

3000 tümleşik grafik kartı olması ve bağımsız GPU barındırmadığından, çalışmanın derin ağ mimarisi ile ilgili kısmı büyük çoğunlukla Google'ın Colab sistemi üzerinden yürütülmüştür. Fakat bu sisteminde ücretli sürümün bile limitlerinin kısıtlı olması ve kullanıcı yoğunluğuna göre tahsis edilen işlemci ve grafik kartı RAM boyutu farklı zamanlarda hep farklı olduğundan, çalışmanın derin ağ mimarisi kısmı büyük zorluklar içinde tamamlanabilmiştir.

Çalışmanın ilk zamanlarında C++, Matlab ve Python dilleri ile karma bir yazılım inşa edilmişken, çalışmanın sonlarına doğru tüm kodlamalar Python diline kaydırılmış ve yaklaşık 30.000 satırlık özgün kod yazılmıştır.

2. TERMİNOLOJİ, VERİ SETLERİ VE METODOLOJİLER

Bu bölüm, çalışmada uygulanan makine öğrenmesi ve görüntü işleme tekniklerinde sıkça kullanılan terminolojinin verilmesi ile başlayıp, çalışmadaki veri setlerini tanıtarak devam eder. Bu veri setleri hem çalışmaya özgün olarak hazırlanan, hem de literatürde kullanılan veri setleridir. Farklı ihtiyaçlar için seçilen çözüm metotları bu bölüm içerisinde anlatılmaktadır. Bu metotlar, literatürde bilindik yöntemler olabileceği gibi, tez çalışması için düşünülen özgün metodolojileri de kapsamaktadır.

2.1. Terminoloji

Aktivite ve hareket tanıma alanında sıkça kullanılan makine öğrenme yöntemleri, istatistiksel ve matematiksel metotlar kullanarak, mevcut verilerden çıkarımlar yapar. Bu çıkarımlarla bilinmeyene dair tahminlerde bulunmaya çalışır. Belirli bir veri kümesi kullanılarak eğitilmiş modellerden faydalanan bu metotlar, sınıfının ne olduğu bilinmeyen bir doğrulama kümesini, hata oranı en düşük olacak şekilde sınıflandırabilmek amacıyla eğitilirler.

Bu süreçte kullanılan modelin ve verinin, çözümlenmesi istenen problemin karmaşıklık düzeyine göre uygun seçilmesi önemlidir. Çünkü büyük modellerin az veri ile eğitilmesi ya da küçük modellerin benzer örneklerle eğitiminin zamanında kesilmemesi, bu sistemlerin öğrenememesi ya da ezberlemesi gibi riskleri ortaya çıkartabilir.

Günlük kullanımda geçen bazı terimler makine öğrenme konusu içerisinde de sıkça kullanılmaktadır. Bu terimlerin konunun takibi sırasında anlam kargaşası çıkarmaması için tanımlarının biliniyor olması gereklidir. Buna göre;

Ezberleme (overfitting), sistemin sadece eğitim için kullanılmış verileri tanıyabilme yani farklı tipte veri geldiğinde çözüm bulamaması olarak yorumlanır. Bu tür tanıma ve sınıflandırma işlemlerinde model doğru seçildikten sonra, sistemin genel başarısı, eğitim verisi olarak kullanılan bilgi miktarı ve kullanılan veri tipi ile doğru orantılıdır. Sistem ne kadar çok doğru veri ile eğitilirse, o denli başarılı çıkarsama yapabilir.

Öğrenmek veya değerlendirmek için kullanılan her bir veriye '**gözlem**' (obsevation) denir. Bir gözlemi temsil eden verilere '**özellikler / öznitelik**' (features) ve gözlemlere atfedilen kategorilere de '**etiket**' (label) denir.

Makine öğrenmede kullanılan ve bu veriyi kullanarak çıkarımlar yapması beklenen algoritmanın öğrenmesi ya da eğitilmesi için kullanılan veri dizine '**eğitim verisi**' (training data) denir. Bu veri seti modelin görmediği verileri doğru sınıflandırabilmesi için model eğitiminde kullanılan veri setleridir. Algoritmanın şekillendirdiği modelinin gerçeğe ne kadar yakın sonuçlar verdiğini test edebilmek için kullanılan veriye '**doğrulama verisi**' (validation data) denir. Doğrulama veri seti, aynı zamanda, kullanılan algoritma için hangi modelin daha iyi olduğunu belirlemek, belirlenen model için de en doğru parametreleri ayarlamak amacıyla bir test platformu sağlar. Eğitilecek tüm modeller doğrulama veri setine gerek duymayabilir. Bir de modeli eğitirken kullanılan veri setinin alt kümelerinde olmayan, modelin daha önce hiç görmediği, verilere ihtiyaç duyulur. Bu verilere de **'test verisi'** (test data) denir. Eğer test verisinde sonuçlar eğitim aşamasına göre beklendiği gibi çıkmıyorsa bir ezberleme durumu ile karşılaşıldığı anlaşılır. Ezberleme, modelimizin eğitim için kullanılan veri üzerinde gereğinden fazla çalıştırılması sonucu ortaya çıkar. Bu yüzden eğitimin belli bir yüzdeye ulaştıktan sonra kesilmesi daha doğru bir yaklaşım olur. Eğitim ve test verisinin sınırlı sayıda olduğu durumlarda, önceden etiketlenmiş ve eğitim esnasında kullanılan veri, etiketlendirilmesi çıkarılarak test işlemlerinde de kullanımı mümkündür. Bu yöntem çok fazla önerilmese de veri miktarının az olduğu durumlarda kurtarıcı olmaktadır.

Doğrulama ve eğitim aşamalarında kullanılan verilerin içeriğinin öznitelik vektörünün yapısına benzemesi gereklidir. '**Öznitelik vektörü**' (feature vector) aynı nesnenin birden fazla ve farklı özelliğinin bir araya gelmesi ile sıralı bir şekilde oluşturulmuş veri yığınıdır. Öznitelik, görüntülerden çıkartılan, insanlar tarafından elle hesaplanması ve bakılarak algılanması zor olan, görüntüyü tanımlayıcı numerik bilgilerdir. Bilgisayarın nesneleri otomatik tanıması için nesnelerin tanımlarının olması yani özniteliklerinin olması gereklidir. Dolayısıyla tanıma işleminden önce nesnelerin özniteliklerinin çıkarılması 'elde edilmesi' gereklidir. Bu işleme **özellik** veya **öznitelik çıkarımı** (feature extraction) denir. Görüntülerden çıkartılan öznitelikler genellikle görüntüden daha düşük boyutludur. Bu sayede görüntünün işlenme hızı arttırılabilir.

Uygulamalara bağlı olarak görüntülerden iki farklı tipte öznitelik çıkartılır. Bunlar **genel öznitelikler** (global features) veya **yerel öznitelikler** (local features) ya da **tanımlayıcılar** (descriptors) olarak adlandırılır. Genel öznitelikler ve tanımlayıcıları, genellikle **görüntü geri getiriminde** (image retrieval), görüntüde nesne tespiti, **görüntü bölütleme** (image segmentation), **görüntü sınıflandırma** (image classification) gibi işlemlerde kullanılır. Yerel öznitelikler ve tanımlayıcıları ise genelde **nesne algılama ve sınıflandırma** (object detection and classification), **izleme ve hareket tahmini** (object tracking and motion estimation), **görüntü düzeltme** (image registration) gibi konularda tercih edilir. Aynı zamanda yerel öznitelikler, nesnenin boyutundaki değişim, dönme ve örtüşme gibi tespit edilmesi zor olan unsurların algoritmalar tarafından üstesinden gelinmesini daha iyi sağlar.

Tespit ve tanıma arasında çok büyük fark olup, biri nesnenin görüntüdeki varlığını bulurken bir diğeri kimliğini belirler. Genel öznitelikler görüntüye bir bütün olarak yaklaşıp genelleme yaparken çoğunlukla tek bir öznitelik vektörü kullanır. Yerel tanımlayıcılar görüntüyü yamalara ayırarak ve **anahtar noktaları** (key points) kullanarak birden fazla öznitelik vektöründen elde edilmiş bilgi ile daha **güçlü** (robust) sonuçlara ulaşır. Bir nesnenin genel öznitelikleri kontur ve şekil tanımlamalarından oluşurken, yerel öznitelikleri **doku** (texture) bilgisi kullanılarak elde edilir.

Şekil matrisleri, bağımsız momentler olan Hu ve Zernike momentleri, HOG (Yönlü Gradyanların Histogramı) [38], genel tanımlayıcılara verilebilecek örneklerden bazılarıdır. Yerel tanımlayıcılardan bazıları ORB [39], SURF[40], LBP (Local Binary Points) [41], SIFT [42] şeklindedir. Genel ve yerel tanımlayıcıların birleştirilerek kullanılması durumunda başarı oranını arttığı görülmektedir [43].

Makine öğrenmede kullanılan iki temel yöntem vardır. Birisi **Gözetimli Öğrenme** Supervised-Learning, diğeri de **Gözetimsiz Öğrenme** Unsupervised-Learning olarak adlandırılmaktadır.

Gözetimli öğrenmede; işe başlarken elde sınıflandırma yapmak istenilen iş için kullanılacak ve ne olduğu bilinen, hangi sınıfa dâhil olduğu bilinen veri kümeleri vardır. Yeni gelen verinin hangi sınıfa ait olacağı bulunurken, elde bulunan verilerle benzerlik ölçütüne bakılarak sınıflandırma yapılır. Yani '**ön bilgi**' (Apriory Knowledge) işe

başlamadan önce mevcuttur ve bu bilgi kullanılarak sınıflandırma ve tanıma işlemleri gerçekleştirilir.

Gözetimsiz öğrenmede; işe başlarken elde bulunan veriler karmaşık bir şekildedir. Bir ön bilgi yoktur. Sınıf veya grup yoktur. Veri karma karışık bir yığın şeklindedir. Bilginin ne olduğu, o yığın içerisinde kaç farklı türde veri olduğu başlangıçta belli değildir. Dolayısıyla yeni gelen bir verinin hangi sınıfa dâhil olduğunu bulmadan önce eldeki veri birbirleri ile karşılaştırılarak benzerlik ölçütüne göre önce sınıflandırılıp daha sonra yeni verinin bulunan sınıflarla olan benzerliğine bakılarak sınıfı belirlenir.

Aynı sınıftan olan iki verinin birbiri ile karşılaştırılması sonucu '**benzerliğinin**' (similarity) yüksek çıkması beklenir. Verilerin '**benzeşmezlik**' oranı (disimilarity value) yüksekse birbirinden farklı sınıflara ait verilerdir denebilir.

2.2. Veri Setleri

Bu doktora tezi kapsamında çalışmaya özgü yoga hareketlerinden oluşan bir veri seti hazırlanmış ve bunun yanı sıra, video sınıflandırma, hareket tanıma, içerik analizi gibi literatürde birçok amaçla kullanılan ve iyi bilinen büyük ölçekli veri setleri olan HMDB51 ve UCF101 isimli veri setleri de kullanılmıştır.

2.2.1. HMDB51 Veri Seti

HMDB51 veri seti içerisinde çoğu film sahnesine ait olan YouTube, Google ve Prelinger koleksiyonuna ait çeşitli kaynaklardan toparlanmış açık kaynaklı bir video veri setidir. Bu veri seti her biri 100'den fazla video içeren 51 eylem kategorisine bölünmüş olup yaklaşık 6849 adet kısa videodan oluşmaktadır. Veri setinin toplam büyüklüğü 2 GB kadardır [44]. Veri seti içerisinden alınmış örnek videolar Şekil 2.1.'de görülmektedir.

Eylem kategorileri beş türde gruplandırılabilir:

1- Genel yüz hareketleri gülümsemek, çiğnemek, konuşmak.
2- Nesne manipülasyonu ile yüz hareketleri: sigara içmek, yemek yemek, sıvı içmek.

3- Genel vücut hareketleri: takla atma, el çırpma, tırmanma, merdiven çıkma, dalış, yere düşme, ters vuruş, amuda kalkma, zıplama, yukarı çekme, yukarı itme, koşma, oturma, ayağa kalkma, dönme, yürüme, dalgalanma.

4- Nesne etkileşimli vücut hareketleri: saç fırçalamak, yakalamak, kılıç çekmek, golf oynamak, bir şeye vurmak, topa vurmak, almak, dökmek, bir şey itmek, bisiklete binmek, ata binmek, top atmak, yay atmak, silah atmak, beyzbol sopası sallamak, kılıç egzersizi, fırlatmak.

5- İnsan etkileşimi için vücut hareketleri: sarılma, birini tekmeleme, öpme, yumruk atma, tokalaşma, kılıç dövüşü.

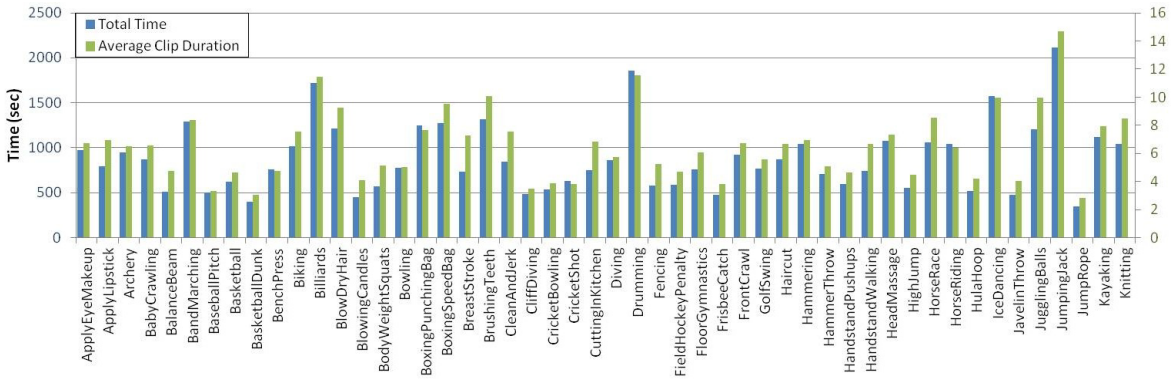
Şekil 2.1. HMDB51 video veri seti içerisindeki her sınıftan seçilmiş örnek video kareleri.

2.2.2. UCF101 Veri Seti

UCF101 veri seti, ortalama 25 fps olan 320x240 boyutlu, 101 eylem kategorisine ait, içinde 13.320 adet YouTube videosu olan ve toplamda 27 saate yakın, gerçekçi eylem videolarından oluşan bir veri setidir. Bu veri seti, 50 eylem kategorisine sahip UCF50 veri setinin bir uzantısıdır. Bu veri kümeleri, eylemler açısından en geniş çeşitliliği sunar. Kamera hareketi, nesne görünümü ve duruşu, nesne ölçeği, bakış açısı, karmaşık arka plan, aydınlatma koşulları vb. gibi zorlayıcı etkileri de barındıran bir veri setidir. Mevcut eylem tanıma veri setlerinin çoğu gerçekçi olmadığından ve aktörler tarafından sahnelendiğinden, UCF101, yeni gerçekçi eylem kategorilerini öğrenerek ve keşfederek eylem tanıma konusunda daha fazla araştırmayı teşvik etmeyi amaçlamaktadır. 101 aksiyon kategorisindeki videolar, her grup 4-7 aksiyon videosundan oluşabilen 25 gruba ayrılmıştır. Aynı gruptaki videolar, benzer arka plan, benzer bakış açısı gibi bazı ortak özellikleri

paylaşabilir [45]. Eylem kategorileri beş türe ayrılabilir: 1)İnsan-Nesne Etkileşimi 2) Yalnızca Vücut-Hareketi 3) İnsan-İnsan Etkileşimi 4) Müzik Aletleri Çalmak 5) Spor. Veri seti içerisinden alınmış örnek videolar Şekil 2.2'de görülmektedir. Şekil 2.3'de bu veri seti içerisindeki videolara ait istatistiki bilgiler incelenebilir.

Şekil 2.2. UCF101 video veri seti içerisindeki her sınıftan seçilmiş örnek video kareleri.



Şekil 2.3. UCF101 Veri setine ait istatistiki bilgiler.

2.2.3. Çalışmaya Özgü Hazırlanmış Yoga Veri Seti

Hazırlanan Yoga veri seti, HMDB51 ve UCF101 veri setlerine göre nispeten daha küçük olduğundan, bu videolar üzerinde hem derin ağ mimarisine dayanan, hem de klasik sınıflandırıcı ile gerçekleştirilebilecek bir takım uygulamalar yapılabilmiştir. Ek-1'de görüldüğü gibi 190 civarında asana olsa da bir yogi eğitmeni bunların 70 civarını kullandığı, popüler olarak da yoga yapanların 30-40 tanesini uyguladığı yapılan araştırmalar sonunda öğrenilmiştir.

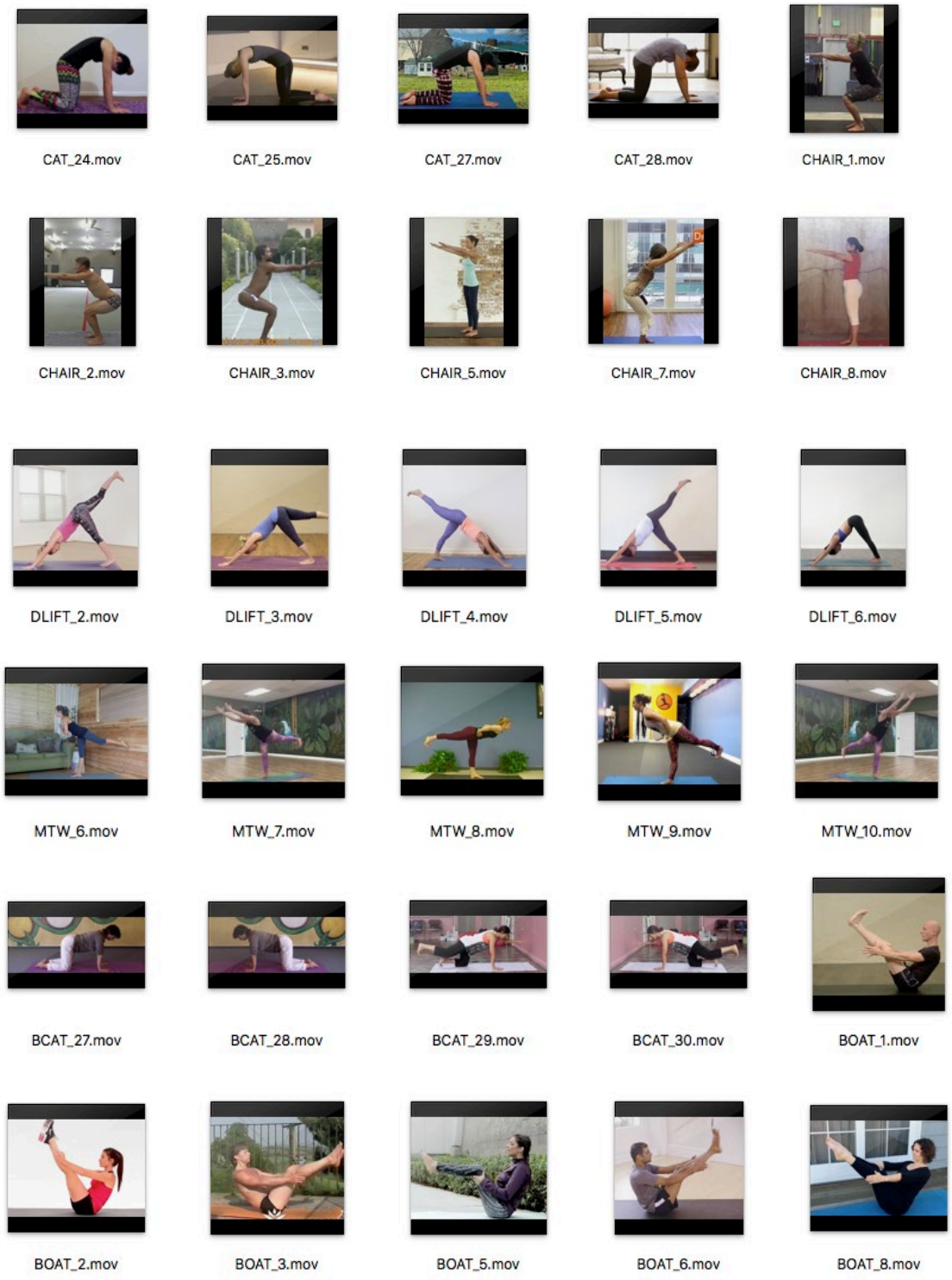
Seçilen asanaların video içerisinden bulunması işlenmesi uygun formatta kayıt edilmesi çok emek isteyen ve zaman alan işlemlerdir. Veri tabanı farklı insanların yaptığı aynı asanalardan oluşturulmuştur.

Veri tabanı içerisindeki örnek videolar Şekil 2.4'de görülmektedir. Bazı asanaların yapılışı diğerlerine göre daha uzun sürmektedir. Videoların ortalama uzunlukları üç ile sekiz saniye arasında değişim göstermektedir. Çalışılan işletim sisteminin Mac OS X olması ve gerçekleştirilen bir çok işlemin (video yakalama, kayıt etme, işleme, dışa aktarma) işletim sisteminin imkanları dahilindeki programlarla yapılması yaratılan videoların MOV formatında hazırlanması konusunda rahatlık sağlanmaktadır. Hazırlanan videoların çoğu QuickTimePlayer'ın içerisinde yer alan ekran yakalama ve kayıt etme işlemi ile gerçekleştirilmiştir.

Hazırladığımız veri seti, sıklıkla yapılan 40 asanadan 8 tanesine odaklanmıştır. Bu asanaların isimleri ve her asanadan hazırlanan video adedi aşağıdaki listede görülebilir.

1- Adho Mukha Savasana (38 adet video),
2- Boat Pose (32 adet video)
3- Balancing Cat (61 adet video
4- Cat Cow Pose (41 adet video)
5- Chair Pose (42 adet video)
6- Downward Dog Leg Lift (22 adet video)
7- Mountain To Warrior (32 adet video)
8- Trikosana (42 adet video)

8 adet asana'nın yer aldığı, yaklaşık bir saate yakın video 600 x 600 çözünürlük ve 60 fps'de MOV formatında YouTube üzerinden kaydedilmiştir. Veri setimizde toplamda 310 adet video mevcuttur.



Şekil 2.4. Yoga veri setinde yer alan 8 Asana'ya ait örnek video kareleri.

2.2.3.1. Yoga Veri Seti Hazırlama Adımları

YouTube üzerinde milyonlarca video yer almaktadır. Bu video içeriklerini YouTube'da hesap açan herkes yükleyebilmektedir. Video içeriklerinin denetimi ile ilgili bir yaklaşım söz konusu değildir. İçerik bazı durumlarda video başlığı ile eşleşmeyebilir. Bu yüzden veri seti hazırlanırken yürütülen ilk adımlarda, ön eleme işlemleri gerçekleştirilmek zorundadır. İçeriklerin ilintili olduğu anlaşılan videolar devamında aşağıda belirtilen işlemlerden geçirildikten sonra kayıt edilir.

1. YouTube üzerinden Asana isimleri ile arama
2. Bulunan sonuçları hızlıca izleme ve ön elemeden geçirme
3. Uygun olan videoları kayıt etme
4. Video düzenleme yazılımları ile işleme
5. Asana isimleri altında gruplama

YouTube videolarının bir standarda bağlı olmadığı ve farklı kişilerce çekildiği için her bulunan sonuç önce hızlıca izlenmiş ve uygun olmayanlar elenmiştir. Devamında alt maddelerde açıklanan video işleme adımları yürütülmüştür.

2.2.3.2. Otomatik Reklamlardan Arındırma

YouTube kâr güden bir kuruluş olduğundan dolayı ücretsiz yükleme hakkı tanıdığı kullanıcı videolarının çeşitli zaman dilimlerine otomatik reklam eklentisi yapmaktadır. Genellikle video girişlerinde karşımıza çıkan bu reklamlar, izleme anında da karşımıza çıkabilmektedir. Örnek bir reklam sahnesi Şekil 2.5'de görülmektedir.

Şekil 2.5. Video içerisinde anlık çıkan YouTube reklamları.

Videoların yakalanması esnasında otomatik reklamlar manuel olarak çıkartılmıştır.

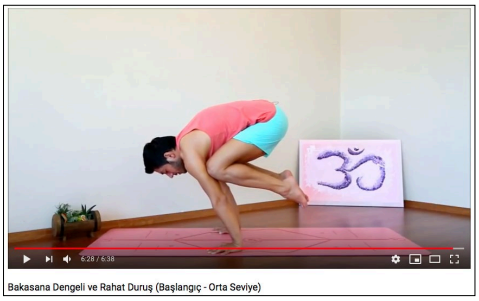
2.2.3.3. Tanıtım Sahnelerinin Elenmesi

YouTube videolarının girişlerinde veya çıkışlarında (**Intro/Outro**) karşımıza çıkan videoyu hazırlayan kişi veya organizasyon tarafından eklenen, akılda kalıcılık veya reklam amacı ile konulmuş tanıtım logoları veya yazılarından oluşan bölümlerdir. Çoğunlukla video giriş ve çıkışlarında bazılarına da video oynarken anlık olarak karşılaşılmıştır. Bu ve benzer sahneler video işleme uygulaması ile montaj esnasında kesilip atılmıştır. Örnek bir tanıtım sahnesi Şekil 2.6'de görülmektedir.

Şekil 2.6. Video girişinde yer alan tanıtım sahneleri.

2.2.3.4. Uzun Videoların İşlenmesi

Genelde Yoga hareketi videoları, duruşun ve pozun eğitimi amaçlı hazırlanır. Bunun için asana'yı gerçekleştiren yogi çoğunlukla 1 adet asanayı yaparken nasıl yapılacağını da sözle anlatır. Bu nedenle videolar, gerçek hareket serisine ait olmayan hareketleri de barındırmaktadır. Örneğin Şekil 2.7'de görülen "turna duruşu" olarak isimlendirilen "bakasana" pozu bir videoda 6 dakikada anlatılmaktadır. Benzer uzun anlatımlar çoğu videoda mevcuttur.

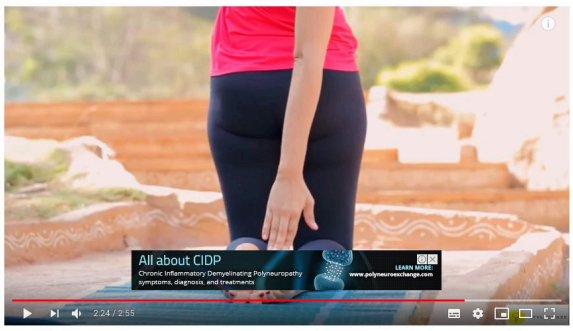


Şekil 2.7 Altı dakikada anlatılan "Bakasana" yoga asanası

Genellikle, duruşun gerçekleştirildiği özet anlatımların olduğu, pozun gerçek görselini içeren kısımlar video işleme yazılımı kullanılarak kırpılmış ve yalın içerikler elde edilmiştir. Bu bilgiler ile video dosyaları şekillendirilmiş ve içerikleri istenilen uzunluğa ayarlanabilmiştir.

2.2.3.5. Vücut Bütünlüğünün Kesildiği Anlar

Öz çekim yapan kişilerin kameraya olan uzaklıklarını tam olarak ayarlayamayışları hareketin tamamının görülememesine neden olmaktadır. Aynı zamanda hareket esnasında kamera için ayarladıkları çerçeveden farkında olmadan çıkmaları da aynı sıkıntıyı doğurmaktadır. Şekil 2.8'deki gibi kesilmelerin olduğu sahneler video ön işleme esnasında kurgu aşamasında kırpılmıştır.

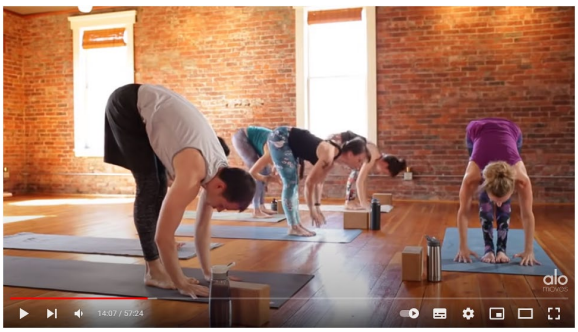


5 Quick Yoga Poses For THYROID Problems & Disorders

Şekil 2.8 Poz kesilmeleri.

2.2.3.6. Videoda Birden Fazla Kişinin Olması

Video içeriklerinde yoga hareketi yapan insanların duruş pozları üzerinden çözümleme gerçekleştireceğimiz için, yoga asanalarına ait görüntüleri oluştururken, veri tabanımızdaki videoların içerisinde birden fazla kişi olmasına dikkat edilmiştir. Şekil 2.9,'daki gibi aynı asananın bir grup halinde yapıldığı videolar yoga video veri tabanına dâhil edilmemiştir.



Şekil 2.9. Birden fazla aynı asanayı yapan kişilerin olduğu video örneği.

2.2.3.7. Pan - Tilt - Zoom Kamera Hareketleri

Birçok videoda hareket yapılırken, eğer çekim için ikinci bir kişi kameraman olarak kullanıldıysa, kameramanın yapılan harekete çekim anında aşırı zum girmesi veya sahnenin genelini göstermek adına zum çıkması söz konusu olabilir. Ayrıca, bazı videolarda çok büyük pan/tilt hareketlerine de rastlanmıştır. Bu hareketler video izlenmeden işin başında belirlenemediği için, videoların ayrıştırılması amacı ile tüm videolar hızlandırılmış bir şekilde izlenmiş uygun olmayanlar başta elenmiş veya video işleme anında sabit kamera ile çekilmiş görüntüler birbiri peşi sıra montajlanmıştır.

2.3. Metodolojiler

Bu bölümde, tez çalışmasında kullanılan ve çalışma için özel olarak hazırlanan video veri setinin hazırlık aşamalarını anlatılarak başlanmaktadır. Devamında çalışma içerisinde kullanılan çözüm metodolojileri tanıtılmıştır. Bunlar arasında iyi bilinen klasik sınıflandırıcıları ve çalışmada kullanılan derin ağ modelleri mevcuttur. Bu kavramlar anlatılırken, terminolojik bilgilendirme devam etmektedir.

2.3.1. Yoga Veri Seti İçerik Zenginleştirme Metodu

Bu bölümde 2.2.3 başlığında bahsi geçen adımlardan sonra elde edilen videoların, derin ağ mimarilerinde kullanımı amacıyla veri setinin nasıl zenginleştirildiği hakkında genel bilgilendirme içermektedir. Bu videoların ham hali, akademik kullanımın hizmetine sunulmak amacıyla GitHub linki https://github.com/ogulgocmen üzerinden genel paylaşıma açılmıştır.

2.3.1.1.Video Ön İşleme Metodu

Tez kapsamındaki yoga veri setinde yer alan videolar, YouTube içeriğinden oluşturulmuştur. Bu işlem ekran yakalama yöntemi ile yapılmıştır. Görüntü kalitesini yüksek tutmak adına 600x600 boyutlarında videolar kaydedilmiştir. Bu videoları işleme aşamasından önce sisteme yük getirmemesi adına 320x240 ebatlarına indirgenmiştir. Görüntüdeki parlaklık, renk, çekim açısından kaynaklanan heterojen durumlar, olduğu gibi değiştirilmeden bırakılmıştır. Bu videoların hepsi veri ihlali barındırmayan kamuya açık içeriklerdir.

2.3.1.2.Veri Çoğaltma Metotları

Oluşturulan yoga videolarının adedi ile derin öğrenme mimarilerinin inşa edilebilirliği veya var olan hazır mimarilerin öğrenmenin aktarımı (transfer learning) ile yeniden eğitilebilir olması, veri sayısının azlığında gerçekleştirilemez. Derin öğrenme uygulamaları için görsel içerik toplama veya bu içerikleri hazırlama işlemleri oldukça maliyetlidir. Sistemin başarılı bir çıktı üretebilmesi için çok miktarda veri olması gereklidir. Genelde görüntü ön işleme yöntemlerinde kullanılan yöntemler veri çoğaltma amacıyla kullanılır. Var olan veriden, veri çoğaltma işlemine (**data augmentation**) denir [46]. Bu yöntem aynı zamanda derin ağ mimarilerinde zaman zaman karşılaşılan ve ezberleme (**over fitting**) olarak dile getirilen mimarinin tüm bilgiyi öğrenmesi ve yeni bilgiler ile eşleştirebilme yapamaması problemi için de kullanılan adımlardandır. Ayrıca, düzgün dağıtılmamış veri setleri içerisinde, sınıf sayılarının eş sayıya getirilmesi ve genel sistem performansının arttırılması gibi amaçlar için de kullanılır. Veri çoğaltma yöntemlerinde eldeki veriye morfolojik işlemler uygulayarak mevcut görüntünün, kaykılmış, ötelenmiş, dönmüş, ebadı büyütülüp küçültülmüş vb. gibi farklı formları elde edilir, var olan bilgi sanal olarak çoğaltılabilir.

Çalışmada veri çoğaltma işlemi, tez çalışması kapsamında hazırlanan yoga video veri tabanı içeriğinin, Tablo 2.1'de gösterilen * işaretli derin ağ mimarilerinde kullanabilmek adına gerçekleştirilmiştir. Mimariler üzerinde öğrenmenin aktarımı işlemlerinin gerçekleştirilebilmesi için, aynı zamanda Tablo 2.1'de görülen mimarilerin giriş katmanında verilen "Giriş Görüntü Boyutu" sütununda görülen genişlik ve yükseklik ölçülerine indirgenmesi gereklidir. Bu mimariler giriş bilgisi olarak görüntü dosyası

formatını kullanır. Bu yüzden, video veri tabanı bu mimarilere giriş olarak verilmeden önce, videonun her karesinin ayrı ayrı görüntü dosyası olarak kayıt edilmesi ve uygun ebada yeniden boyutlandırılması gereklidir.

Tablo 2.1. Derin Ağlar – Katman ve Parametre Sayıları – Giriş Boyutları

Kullanılan	Derin Ağ Modeli	Derinlik	Parametre(Milyon)	Giriş Görüntü Boyutu
	Alexnet	8	61	227 x 227
*	Vgg16	16	138	224 x 224
*	Vgg19	19	144	224 x 224
	Squeezenet	18	1,24	227 x 227
	Googlenet	22	7	224 x 224
	Inceptionv3	48	23,9	299 x 299
	Densenet201	201	20	224 x 224
	Mobilenetv2	53	3,5	224 x 224
	Resnet18	18	11,7	224 x 224
	Resnet50	50	25,6	224 x 224
	Resnet101	101	44,6	224 x 224
	Xception	71	22,9	299 x 299
	Inceptionresnetv2	164	55,9	299 x 299
	Shufflenet	50	1,4	224 x 224
	Nasnetmobile	*	5,3	224 x 224
	Nasnetlarge	*	88,9	331 x 331

Literatürde, görüntü çoğaltma yöntemleri ile ilgili "geleneksel" ve "derin öğrenme tabanlı" yöntemler mevcuttur. Bu yöntemlerin hangisinin diğerine göre iyi olduğuna ilişkin kesin bir şey başta söylenemez. Çünkü yöntemler, yöntemin uygulandığı veri seti ve bu veri seti ile yapılacak olan çalışmanın amacına göre aynı olmayan başarımlar gösterebilmektedir. Görüntü sınıflandırmak için geleneksel yöntemler ile derin öğrenme mimarili veri çoğaltma yöntemleri [47] numaralı çalışmada karşılaştırılmıştır. Görüntüleri döndürmek, çevirmek ve kırpmak için geleneksel teknikler ile derin mimarilerden olan GAN kullanılmış ve geleneksel veri çoğaltma yöntemlerinin daha etkili olduğu sonuçta gösterilmiştir.

Diğer çalışmalardan birinde, veri çoğaltma metotları olan rastgele bir yerden kesme ve yatayda çevirme, temel bileşen analizi (PCA - Principal Component Analysis,) yöntemleri denenmiştir. Sonuçta, veri çoğaltma işlemlerinin, modelin sınıflandırma

performansına olumlu yönde etkilediği bildirilmiştir [48]. Geleneksel yöntemlerle ilgili yöntemler literatürde birçok çalışmada geçmektedir [49], [50], [51], [52].

a. Görüntü Döndürme İşleminin Uygulanması

Görüntü döndürme, elde bulunan görüntünün istenilen bir açı kadar sağa veya sola doğru döndürmesi işlemidir. Döndürme açı değeri sıfırdan büyük ise görüntüleri sağa, sıfırdan küçük ise görüntüleri sola doğru döndürülmesi işlemi gerçekleşir. Döndürme işleminin uygulandığı görüntüde en boy oranında değişim olabilir bu nedenle morfolojik işleme tabi tutulan görüntüler tekrar orijinal boyutuna çevrilmelidir.

b. Yatay veya Düşey Eksene Göre Simetri Alma

Görüntüleri x veya y eksenlerine göre simetrilerinin alınması işlemidir. Görüntüler zenginleştirilirken, dönme işlemlerinin her ikisi veya bir tanesi peş peşe birden fazla kez orijinal görüntüye uyarlanabilir. Çalışmada, her ikisi de görüntüler üzerine uygulanmıştır.

c. Görüntü Kaykılma İşlemi

Düzlem geometrisinde, kayma, her noktayı sabit bir yönde, o yöne paralel olan ve orijinden geçen çizgiden işaretli mesafesiyle orantılı bir miktarda kaydıran doğrusal bir haritadır. Bir örnek vermek gerekirse, koordinatları (x,y) olan herhangi bir noktayı (x+2y,y) noktasına götüren eşlemedir. Bu durumda, yer değiştirme, sabit çizginin x ekseni olduğunu ve işaretli mesafenin y koordinatı olduğunu yerde 2 kat yataydır. Referans çizgisinin zıt taraflarındaki noktaların zıt yönlerde yer değiştirdiğine dikkat edilmelidir.

d. Görüntü Öteleme İşlemi

Düzlem geometrisinde, öteleme, her noktayı sabit bir değer eklemektir. Eklenen değerin pozitif veya negatif olmasına göre veya eklenen eksenin x ya da y olmasına göre görüntü, aşağı, yukarı, sola, sağa veya çapraz doğrultularda, boyut değiştirmeden yer değiştirir. Öteleme, $x = x + Tx$ ve $y = y + Ty$ şeklinde bir formüle sahiptir. Bir örnek

vermek gerekirse, koordinatları (x,y) olan herhangi bir noktayı (x+2,y) noktasına götüren eşleme görüntüyü x ekseninde 2 birim ötelemiştir.

e. Görüntü Boyut Değiştirme İşlemi

Düzlem geometrisinde, ölçekleme, bir grup noktayı daha büyük veya daha küçük gruptaki noktalara evirmedir. Örneğin, küçültme işleminde grup içerisindeki değerlerden rastgele birinin değeri kullanılarak tüm diğerleri bu tek bir değere evirmek en kolay yöntemidir. Seçilen grup sayısı kadar görüntü boyutu ufalacaktır. Tersi işlem boyut büyütme amacı ile uygulanabilir. Boyut büyütmede değeri belli olmayan noktaların hesabı komşu değerlerin ortalaması alınarak yapılabilir.

2.3.2. Sınıflandırma Algoritmaları ve Çok Katmanlı Algılayıcılar

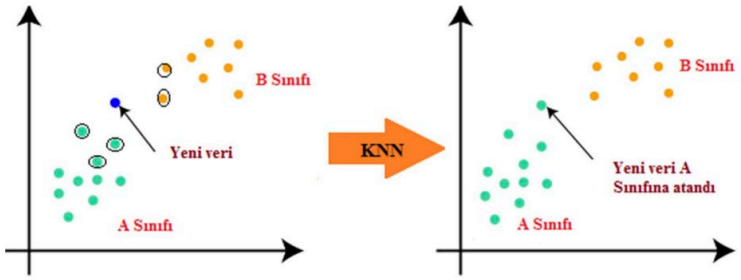
Bu başlık altında, k- En Yakın Komşuluk Sınıflandırıcısı, DVM – Destek Vektör Makineleri, Çok Katmanlı Algılayıcılar – MLP (Multi Layer Perceptron) konularına değinilmiştir. Sınıflandırıcılarda kullanılan ölçütlerin tanımları ve formülleri verilmiştir.

2.3.2.1. K - En Yakın Komşuluk Sınıflandırıcısı

KNN veya k-NN olarak da bilinen k-en yakın komşu algoritması, bireysel bir veri noktasının gruplandırılması hakkında sınıflandırmalar veya tahminler yapmak için yakınlığı kullanan, parametrik olmayan, denetimli bir öğrenme sınıflandırıcısıdır. P. E. Hart ve T. M. Cover ikilisi tarafından önerilmiştir. Algoritma basit, etkili ve örnek tabanlı bir örüntü sınıflandırma algoritmalarından biridir. Algoritma makine öğrenme alanında sıkça tercih edilen popüler bir sınıflandırıcıdır. Yapay zeka uygulamaları, veri madenciliği teknikleri, örüntü tanıma gibi branşlarda çok sık kullanıldığı görülmüştür. Bu algoritmada, öğrenme işlemi eğitim setinde bulunan verilere dayalı olarak yapılır.

Algoritmanın gerçekleştiriminin basitliği, analitik olarak takip edilebilir, yerel bilgilere adapte edilebilir, paralel olarak yürütmelere uygun, gürültü içeren eğitim verilerinde tutarlılığı dikkat çekici özelliklerindendir. Algoritmanın bu gibi avantajları

olduğu için sınıflandırmada özellikle tercih edilmektedir fakat uygulamada bu algoritma çok miktarda bellek ihtiyacı vardır. Kullanılmak istenildiği yerde öznitelik vektörünün boyutu arttıkça işlem yükü ve işlem maliyetinin önemli ölçüde artması, performansın k adet komşu sayısı, öznitelik sayış, uzaklık ölçütü gibi parametre bağlı olarak etkilenmesi bu algoritmanın dezavantajı olarak sayılabilir [53].



Şekil 2.10. KNN algoritmasının nasıl çalıştığını gösteren çizim.

Şekil 2.10'da da görüldüğü gibi, algoritmanın çözüm yönteminde; yeni karşılaşılan bir örneğin, eğitim setinde yer alan örneklerle arasındaki mesafe ölçülür, mesafenin küçük olması, yeni örneğin o gruba benzer olduğu anlamını doğurur. Bu benzerliğe göre, yani mesafenin en kısa olduğu gruba örnek atanır [54].

k-En yakın komşuluk sınıflandırıcısının çözüm sözde kodu;

- *k değerini ve uzaklık ölçütünü belirle.*
- *Veri setinde bulunan verilerle yeni veri arasındaki uzaklıkları hesapla*
- *Hesaplanan uzaklıkları küçükten büyüğe sırala ve ilk k adedinin etiketine bak ve en çok sayıdaki etiket ile yeni gelen veriyi işaretle.*

Algoritma, k komşu sayısı bilgisine göre sınıflandırma yapar. Sınıflandırma esnasında, yeni değer, k değeri 1 olduğunda, sadece en yakın komşunun bulunduğu sınıfa yeni gelen veri atanırken, k değeri örnek sayısına (N) yaklaştıkça, veri setinde yer alan tüm diğer veriler dikkate alınarak, oylamayla seçim gerçekleşmektedir. Komşular için ağırlıkların atanması ile sınıflandırılmakta olunan örneğe daha yakın olan komşular, çoğunluk oylamasına daha fazla katkı koyması amaçlanır. En sık kullanılan ağırlık değeri atama yöntemlerinde her bir komşunun ağırlığının, d komşular arası uzaklık olmak üzere, 1/d ya da $1/d^2$ şeklinde alınmasıdır. Sınıflarla olan uzaklık hesaplanırken kullanılabilecek birden

fazla ölçüt vardır. Bu ölçütler, Öklid Mesafesi, Minkowski Uzaklığı, Manhattan Uzaklığı ve Chebyschev Uzaklığı'dır [55].

Öklid Uzaklığı; kümeleme ve sınıflandırma algoritmalarında en çok tercih edilen uzaklık ölçütüdür. Herhangi 2 nokta, P ile Q arasındaki Öklid mesafesi, $\mathbf{P} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_n})$ ve $\mathbf{Q} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, \boldsymbol{y_n})$ iki nokta olmak üzere, eşitlik 2.1'e göre hesaplanır [56].

$$d = \left(\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \right) \qquad (2.1)$$

Öklid uzaklığı, K-NN algoritması, K-ortalama kümeleme algoritmasındaki gibi kümeleme ve sınıflandırma algoritmalarında uzaklığın ölçülmesi için kullanılan uzaklık ölçütüdür.

Manhattan uzaklığı, Herhangi 2 nokta, P ile Q arasındaki Manhattan mesafesi, $\mathbf{P} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_n})$ ve $\mathbf{Q} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, \boldsymbol{y_n})$ iki nokta olmak üzere, eşitlik 2.2'ye göre hesaplanır [57].

$$d = \left(\sum_{i=1}^{n} |x_i - y_i| \right) \qquad (2.2)$$

Chebyschev uzaklığı; Herhangi 2 nokta, P ile Q arasındaki Chebyschev mesafesi, $P = (x_1, x_2, x_3, x_n)$ ve $Q = (y_1, y_2, y_3, y_n)$ iki nokta olmak üzere, eşitlik 2.3'e göre hesaplanır [58].

$$\lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} = max_{i=1}^{n} (|x_i - y_i|) \qquad (2.3)$$

Minkowski uzaklığı; Herhangi 2 nokta, P ile Q arasındaki Minkowski mesafesi, $P = (x_1, x_2, x_3, x_n)$ ve $Q = (y_1, y_2, y_3, y_n)$ iki nokta olmak üzere, eşitlik 2.4'e göre hesaplanır [59].

$$\lim_{p \to -\infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} = min_{i=1}^{n} (|x_i - y_i|) \qquad (2.4)$$

2.3.2.2. DVM – Destek Vektör Makineleri

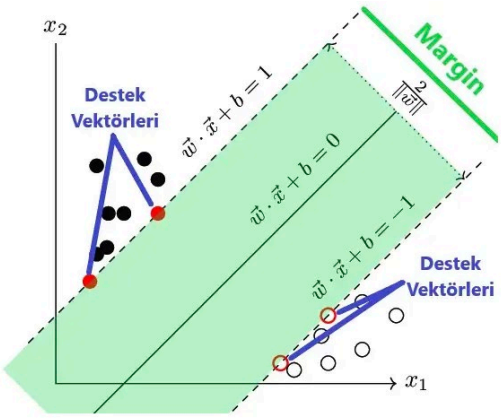
Destek vektör makineleri, bir sınıflandırma metodudur. İstatistiksel öğrenme teorisine dayanır. İlk çıktığında, iki sınıfa ayrılmış doğrusal verilerin ayrıştırılması amacıyla tasarlanmış bu yöntem, zamanla doğrusal olmayan ve ikiden daha fazla sınıflı verilerin sınıflandırılması ihtiyacını karşılamak üzere iyileştirilmiştir. Basit olarak iki adet sınıfı birbirinden ayırabilen bir hiper düzlemin tanımlanması esasına dayanır. Yüksek boyutlu veriler karşısında aldığı yüksek başarıdan ve farklı tipteki veri kaynaklarını esnek bir şekilde modelleyebilmesinden dolayı, DVM'ler geniş bir kullanım alanına sahiptir. Değişkenler arası örüntülerin ne olduğunun bilinemediği veri setlerindeki sınıflandırma problemlerinde kullanılır. Sınıflama, regresyon, aykırı değer tespit etmek için kullanılabilen "gözetimli öğrenme" yöntemidir [60].

DVM'ler parametrik olmayan sınıflandırıcılardır. Verinin dağılımı ile ilgili herhangi bir ön bilgisi yoktur. DVM'de eğitim veri setlerindeki girdi ve oluşturulan çıktılar birbirleri ile eşlenir. Bulunan bu eşleşmeler aracılığıyla, test verisinde ve yeni veri setlerindli girdi bilgisini sınıflandırabilecek karar fonksiyonları elde edilir. Girdi bilgisi; doğrusal olarak ayrılabildiğinde, bu bilgiyi ayırabilecek sonsuz sayıda doğru içerisinden marjini birbirine göre en yüksek olan doğruları seçmeyi hedefler [60] .

Veri, Denklem (2.5)'de verilen DV'ler ile doğrusal olarak ayrıştırılamadığından, orijinal veriyi yüksek boyuta dönüştürmek için, doğrusal olmayan haritalamalar kullanmaktadır.

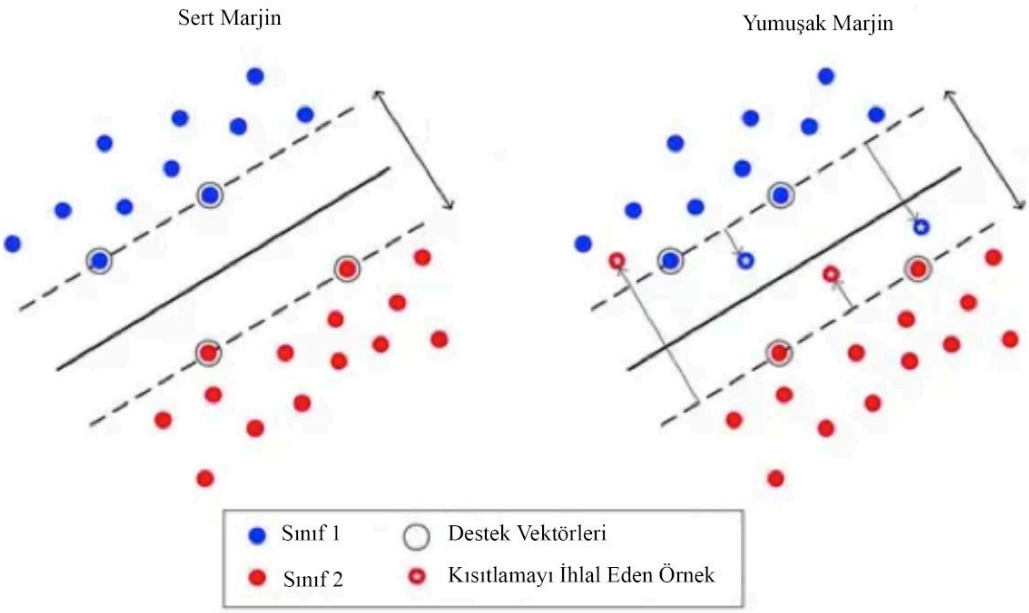
$$f(\vec{x}) = \begin{cases} 0, eğer\ \vec{w}^T.\vec{x} + b < 0 \\ 1, eğer\ \vec{w}^T.\vec{x} + b \geq 0 \end{cases} \qquad (2.5)$$

Denklem (2.5)'de, $\vec{w}$, ağırlık vektörü, $\vec{x}$, girdi vektörü, b'de sapmayı temsil etmektedir. Verinin taşındığı yeni boyutta Şekil 2.11'de görülen marjini en büyük (optimal) ayırıcı düzlemi araştırmaktadır. Marjin ne kadar fazla ise, 2 sınıf o kadar iyi ayrıştırılır. Şekil 2.11 üzerinde görülen kesikli çizgilerle ifade edilen bu çizgiler üzerinde görülen vektörler destek vektörler ve yumuşak ayrım (soft marjin) çizgisi bu vektörler üzerinden geçer. İki yumuşak ayrım çizgisinin ortasından geçen çizgi ise sert ayrım olarak isimlendirilir ve $f(\vec{x}) = \vec{w}^T.\vec{x} + b = 0$ fonksiyonuyla çizilir.



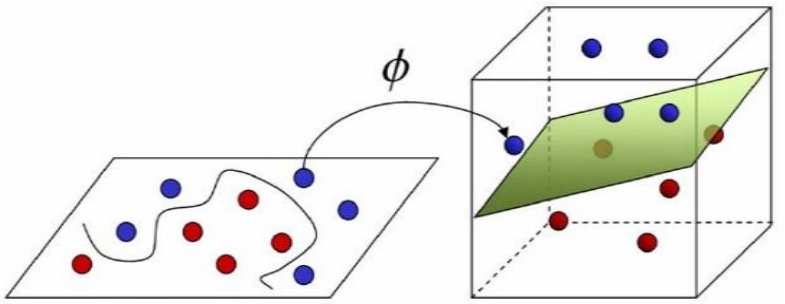
Şekil 2.11 – DVM, sınıflar arası kurulan marjin ve destek vektörleri.

Yeni değer için çıkan sonuç 0'dan küçük ise, yeni değer, Şekil 2.11'de görülen beyaz noktalara yakın olacaktır. Eğer, çıkan sonuç büyük ya da 0'a eşit çıkarsa, bu durumda yeni değer, siyah noktalara yakın kabul edilecektir. Marjin her zaman bu şekilde veriyi %100 performansta ayıramayabilir. Bazen örnekler Şekil 2.12'de görüldüğü gibi marjin bölgesine girebilir. Buna yumuşak marjin denir. Sert marjin, veri doğrusal olarak ayrılabiliyorsa çalışır. Aykırı değerlere karşı duyarlıdır. Bu yüzden, bazı durumlarda yumuşak marjinin tercih edilmesi gerekebilir.



Şekil 2.12 – Yumuşak marjin ve sert marjin kavramı için doğrusal destek vektör makinesi modeli temsili gösterimi.

Veriler marjin içine düştüğü veya karşı tarafına düşmesi durumunda yanlış ayrım yapıldığı anlamına gelir. Verilerin minimum hata yapılacak şekilde ayrılması gereklidir. Bu durumda doğrusal sınıflandırıcıdan, doğrusal olmayan sınıflandırıcılara geçişi sağlayacak çekirdek fonksiyonlarının kullanılması gereklidir. Çoğu durumda doğrusal olmayan sınıflandırıcı daha iyi sonuç verebilir. Buna nazaran, doğrusal sınıflandırıcıların da basit eğitim algoritmasına sahip olmaları gibi bir avantaj da söz konusudur. Bu durumda dikkate alınması gereken öncelikli şey, doğrusal sınıflandırıcı mekanizmasının doğrusal olmayan karar verme sınırlarının oluşturulabilmesi amacıyla genişletilip genişletilemeyeceğidir. Doğrusal sınıflandırıcı üzerinden, doğrusal olmayan sınıflandırıcıya geçiş yapılabilmesinin en kolay yolu, doğrusal olmayan çekirdek fonksiyonları kullanmaktır. Bu fonksiyonlar Denklem (2.6)'da görülen $\emptyset : X \rightarrow F$ ile veriyi X giriş uzayından, F öznitelik uzayına Şekil 2.13'te gösterildiği gibi eşleştirmektedir.



Şekil 2.13 - Doğrusal sınıflandırılamayan girdi uzayının, bir üst boyuttaki uzaya çekirdek fonksiyonu ile haritalandırılması.

F uzayında diskriminant fonksiyonu tanımlaması Denklem 2.6'daki gibidir.

$$f(\vec{x}) = \vec{w}^T . \emptyset(\vec{x}) + b \qquad (2.6)$$

Çekirdek metotları, veriyi büyük boyutlu öznitelik uzayına eşleştirerek boyutsal karmaşıklığın önüne geçer. Burada α_i değerleri, karar sınırının binary gösterimini sağlar. Ağırlık vektörünün, $f(x) = \sum_{i=1}^{n} \alpha_i \vec{x}_i$ eğitim örneklerinin doğrusal kombinasyonu olarak ifade edildiği varsayılırsa, bu durumda, diskriminant fonksiyonu Denklem 2.7'deki gibi ifade edilir.

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \ \vec{x_i}^T . \vec{x} + b \qquad (2.7)$$

F öznitelik uzayında ise bu ifade Denkler 2.8'deki forma dönüşür;

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \ \emptyset(\vec{x})^T . \emptyset(\vec{x}) + b \qquad (2.8)$$

Burada $k(\vec{x}, \vec{x}')$ çekirdek fonksiyonu şu şekilde tanımlanması gerekir;

$$k(\vec{x}, \vec{x}') = \ \emptyset(\vec{x})^T \emptyset(\vec{x}') \qquad (2.9)$$

Diskriminat fonksiyonunu çekirdek fonksiyonu cinsinden yazılacak olursa;

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \ k(\vec{x}, \vec{x}') + b \qquad (2.10)$$

Çekirdek metotları datayı, üç boyutlu uzaydaki noktalar kümesine dönüştürerek daha büyük boyutlu öznitelik uzayına eşler. Büyük boyutlu bu uzayda, farklı metotlar kullanılarak veriler arasındaki bağıntılar bulunabilir. Eşleştirme durumu genel olabileceği gibi bu dönüşümle bulunan bağıntılar çok daha genel olabilir. Öznitelik uzayında işlem gören kernel fonksiyonları, bu uzaydaki datanın koordinatlarını hesaplamayıp, sadece öznitelik uzayındaki dataların tüm ikililerinin görüntüleri arasındaki skaler çarpımı hesaplar. Bu işlem, koordinatların hesaplanması işlemine göre maliyet olarak daha ucuzdur [61].

Tez çalışmasında kullanılan çekirdek fonksiyonları, doğrusal, polinomiyal ve Gaussian radyal tabanlı fonksiyon (RBF) şeklinde olup daha fazlası Tablo 2.2'de görülebilir.

Tablo 2.2 Çekirdek fonksiyonlarının çeşitleri

Çekirdek Fonksiyonları	Formüller
Lineer	$k(x,y) = x^T y + c$
Polinom	$k(x,y) = (\alpha x^T y + c)^d$
RBF	$k(x,y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$
Logaritmik	$k(x,y) = -\log(\|x-y\|^d + 1)$
Laplasyen	$k(x,y) = \exp\left(-\frac{\|x-y\|}{\sigma}\right)$
Sigmoid	$k(x,y) = \tanh(\alpha x^T y + c)$
Eksponansiyel	$k(x,y) = exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$
Kuvvet	$k(x,y) = -\|x-y\|^d$

Doğrusal çekirdek fonksiyonu, en basit çekirdek fonksiyonudur. <x,y> iç çarpımı ve bir c sabiti ile belirtilir. İlgili çekirdek fonksiyonu 2.11 denklemi ile ifade edilir [62].

$$k(x,y) = x^T y + c \qquad (2.11)$$

Polinomiyal çekirdek fonksiyonu, durağan olmayan bir çekirdektir. Polinomiyal çekirdeği, tüm eğitim verilerinin normalleştirildiği problemler için çok uygundur. İlgili çekirdek fonksiyonu Denklem (2.12) ile ifade edilir [62].

$$k(x,y) = (\alpha x^T y + c)^d \qquad (2.12)$$

Ayarlanabilir parametrelerden, α eğimi, c sabit terimi ve d polinom derecesini ifade etmektedir.

Gaussian radyal taban fonksiyonu çekirdeği, Denklem (2.13) ile ifade edilir.

$$k(x,y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2}) \qquad (2.13)$$

Alternatif olarak, Denklem 2.14'de kullanılır.

$$k(x,y) = \exp(-\gamma \|x-y\|^2) \qquad (2.14)$$

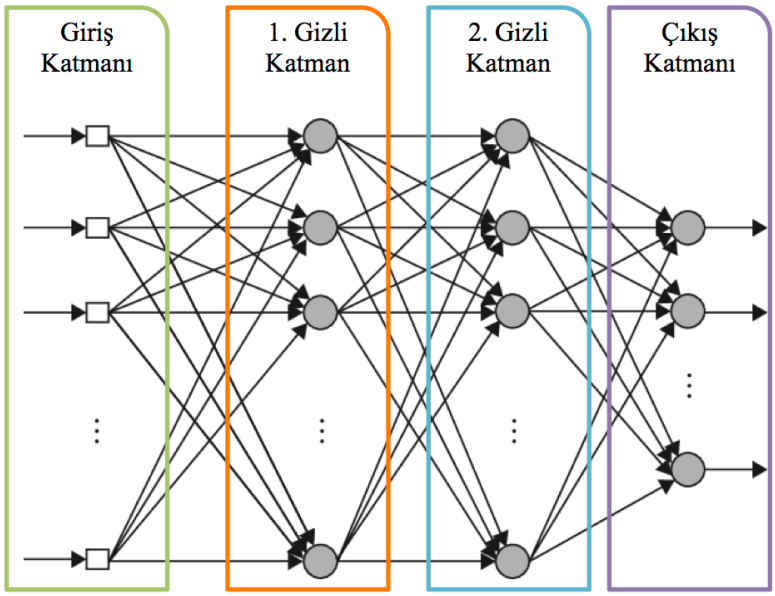
Değeri ayarlanabilir bir parametre olan sigma değeri (σ) çekirdeğin performansında önemli bir rol oynar. Sigma'nın değeri eldeki soruna göre dikkatle seçilmelidir. Aşırı tahmin söz konusu olduğunda, üstel ifade yaklaşık olarak doğrusallaşır. Bu durumda yüksek boyutlu projeksiyon doğrusal olmayan yapısını kaybeder. Öte yandan, eğer gerçek değerden daha küçük bir tahmin gerçekleşiyorsa, fonksiyon düzeltemeyecek ve karar sınırı da eğitim verisindeki gürültü değerlere karşı çok duyarlı olacaktır. Avantajları yüksek doğruluk, karmaşık karar sınırları modelleyebilme, çok sayıda bağımsız değişkenle çalışabilme hem doğrusal olarak ayrılan hem doğrusal olarak ayrılamayan verilere uygulanabilme, diğer birçok yönteme göre ezberleme sorununun az olması sayılabilir [62].

Bunun yanı sıra; en önemli avantajı, sınıflandırma problemini kareli optimizasyon problemine dönüştürüp çözmesidir. Böylece problemin çözümüne ilişkin öğrenme aşamasında işlem sayısı azalmakta ve diğer teknik/algoritmalara göre daha hızlı çözüme ulaşılmaktadır. Teknik bu özelliğinden dolayı, özellikle büyük hacimli veri setlerinde büyük avantaj sağlamaktadır. Ayrıca optimizasyon temelli olduğundan sınıflandırma performansı, hesaplama karmaşıklığı ve kullanışlılık açısından diğer tekniklere göre daha başarılıdır [63].

Dezavantajları olasılıksal tahminler üretememektir. Karmaşık ama küçük ve orta ölçekteki veri setleri için uygundur. Kullanım alanları, nesne tanıma, yüz tanıma, parmak izi tanıma, el yazısı tanıma, zaman serisi tahmin testleri v.b. sayılabilir.

2.3.2.3. Çok Katmanlı Algılayıcılar – MLP (Multi Layer Perceptron)

Kendi oluşturduğumuz video veri tabanının üzerinde, bir sonraki bölümde detaylı olarak bahsedilen öznitelik çıkartma yöntemimizi uyguladığımızda, metin tabanlı öznitelik vektörleri elde edilir. Bu öznitelik vektörleri, aşağıda genel bir tasarımı görülen Çok Katmanlı Algılayıcılar ile kullanılmış ve birtakım sonuçlar elde edilmiştir.



Şekil 2.14. Çok Katmanlı Algılayıcılar

Çok katmanlı ileri beslemeli ağ, mantıksal olarak düzenlenmiş iki ya da daha çok katmandan oluşur. Birer girdi katmanı ve çıktı katmanı vardır. Her biri en az bir nörona sahiptir. Girdi katmanındaki nöronlar, kendi girdileri olmadığından hipotetiktir ve hiçbir süreci işletmezler. Aktiflikleri (çıktı) ağın girdileri ile belirlenir. Girdi ve çıktı katmanları arasında gizli katmanları vardır. İleri beslemeli olmalarının sebebi bilginin tek yönde akmasından kaynaklanır. Her nörona girdiler, bir önceki katmanların nöronlarının çıktılarından gelir. Bu katmanın nöronlarının çıktıları da bir sonraki katmanın nöronlarına iletilir. Ağdaki her bir nöronun çıktısı, o nöronun girdisinin fonksiyonudur. Bu ağa girdi verildikten sonra, bütün çıktı katmanın aktiflik seviyeleri hesaplanabilir. Tekrarlamaya gerek olmayan ve tesadüfi değerler içermeyen bir süreçtir. Bazı modeller bağlantıların katmanları atlamalarına izin verir. Böylece nöronların çıktıları bir sonrakini atlayarak ardından gelen katmana aktarılabilir. Bu tür ağlara bu çalışma içerisinde yer verilmemiştir. Tek nöron şematik olarak Şekil 2.14'te gösterilmiş olup, n adet girdisi vardır ve bu girdi 0

ile n-1 arasında isimlendirilmiştir. Aynı zamanda varsayılmış bir girdisi vardır ve bias olarak adlandırılır, her zaman 1.0' a eşittir. Nöron n + 1 ağırlıkla karakterize edilmiştir. Her bir girdiyle çarpılmaktadır. Aktiflik fonksiyonu, girdinin ağırlıklandırılmış toplamına, çıktıyı üretmek için uygulanır. Girdilerin ağırlıklandırılmış toplamı, bias da dahil, sıklıkla ağ girdisi olarak adlandırılır. Böylece, eğer n girdi {x, i = 0, . . ., n 1} ise, nöronun çıktısı aşağıdaki gibi hesaplanır. Bu nöronun işlevsel karakteristikleri öncelikle ağırlıklar tarafından kontrol edilmektedir. Ayrıca, aktiflik fonksiyonu f(net) önemlidir. Temel gereksinimler karşılandığı sürece, nöronun işlevi, uygulamada aktiflik fonksiyonunun tam doğasından az etkilenmektedir. Diğer bir yandan ise, eğitim sürati aktiflik fonksiyonundan fazlasıyla etkilenmektedir. İleri beslemeli ağlar genellikle girdi ve çıktı katman arasında tek bir gizli katmana sahiptir. Böyle bir ağ üç katmanlı ağ olarak adlandırılır. Nadiren iki gizli katman gerekli olmaktadır.

$$f(net) = f(\sum_{i=0}^{n-1} x_i \; w_i + w_n) \qquad (2.14)$$

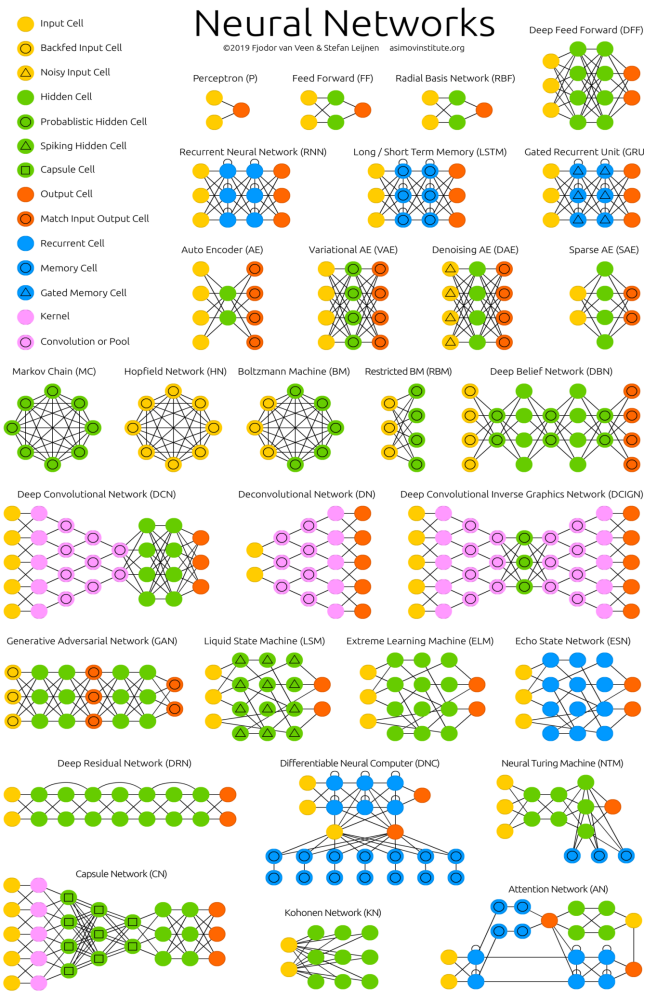
Dört katmanlı bir ağ şematik olarak Şekil 2.14'te gösterilmektedir. İleri beslemeli bir ağdaki yukarıdaki şekilde çember ile temsil edilmiş olan her nöron, eşitlik 2.14'te gösterilen biçimde işler. Girdileri bir önceki katmandan gelmektedir ve çıktıları bir sonraki katmana gitmektedir. Hemen her zaman tüm nöronlarda aynı aktiflik fonksiyonu kullanılır. Bir katmandan diğerine nöronları bağlayan ağırlıklardır. Daha önce de değinildiği gibi, girdi nöronlar hipotetik yapılardır. Aslında hiçbir süreci işletmezler. Ağın girdileri teorik olarak, girdi nöronların çıktılarıdır. Sadece, gizli (eğer varsa) ve çıktı nöronlar, girdilerini birleştirerek çıktı üretmek için süreç işletirler [64].

2.3.3. Çalışmada Kullanılan Derin Ağlar ve Diğer Metotlar

Kısaca derin ağ olarak da ifade edilen Evrişimli Sinir Ağları (CNN) sahip oldukları mimarilerle birbirinden ayrılır. Sık kullanılan mimari şekilleri ve bağlantılar Şekil 2.15'da görülmektedir.

Derin öğrenme yöntemleri, öznitelik çıkarma işlemi ve dönüştürme işlemi için birçok doğrusal olmayan katman kullanır. Her ardışıl katman, kendinden bir önceki

katmandaki çıktıyı, kendine girdi olarak alır. Algoritmalar denetimli örneğin "sınıflandırma problemi" veya denetimsiz örneğin "desen analizi" gibi olabilir. Derin öğrenme yöntemlerinde, verilerin birden çok öznitelik değerinin veya temsillerinin öğrenilmesine dayanan bir kurgu söz konusudur. Üst düzey öznitelikler, alt düzey özniteliklerden türetilerek hiyerarşik bir temsil oluşturur. Bu temsilde, soyutlamanın farklı seviyelerine karşılık gelen birden çok temsil seviyesini öğrenir. CNN mimarisinde genel olarak, evrişimsel katman, havuzlama katmanı, tam bağlı katman, toplu normalizasyon, ReLU ve seyreltme katmanlarının bir veya birden fazlası sıklıkla kullanılır. Evrişimsel katmanda, filtreler girdi görüntüsü üzerinde bir takım çarpma ve toplama işlemi gerçekleştirerek, filtre sayısı kadar öznitelik haritası oluşturur. Bu katmanda filtre boyu ve girdi üzerinde kaç pikselde bir işlem yapılacağının belirtildiği stride/adım parametresi, oluşturulacak öznitelik haritası üzerinde önemli bir rol oynar. Filtrelerdeki ağırlık (weights) değerleri rastgele atanarak eğitim aşamasında tekrarlamalı olarak güncellenir.



Şekil 2.15. Evrişimli Sinir Ağları - Mimari Modeller

Bu sayede görüntülerdeki öznitelikler eğitim aşamasından sonra ortaya çıkar. Kullanılmak istenen katman sayısına göre temel öznitelikler, ilkel örüntüler ve gelişmiş örüntüler elde edilebilir. Kullanılan her filtre görüntü üzerindeki en yüksek tepkileri veren farklı bölgeleri ortaya çıkarır. Böylece görüntüde aranan nesne yönden ve büyüklükten bağımsız olarak tespit edilebilir. Havuzlama katmanında, evrişimsel katman tarafından oluşturulan öznitelik haritalarının boyutları küçültülür. Bunun yapılmasının nedeni, öznitelik haritalarının boyutunun azalacağı için sistemin eğitim aşamasında harcadığı süreyi azaltmak ve n x m ebadındaki bölgelerde ortalama alma ya da maksimum değer elde etme gibi işlemler yapılacağından sistemin veriye olduğu gibi uymasının yani ezberlemenin önüne geçmektir. Tam bağlantılı katmanlar ise elde edilen öznitelik haritaları bir piksel vektörüne çevrildikten yani düzleştirildikten sonra girdi matrisini sınıflandırmaya hazır hale getirir. Tam bağlantılı katmandan sonra soft max ve sınıflandırma katmanları gelir. Softmax katmanı, girdi vektörünün mevcut sınıflara ait olma olasılığını hesaplaken sınıflandırma katmanı, girdi vektörünün sınıfını belirler. Tam bağlantılı katmandan önce seyreltme katmanı da kullanılabilir. Bu sayede her bir nöron belirlenen bir olasılık değerine göre ağırlıkları sıfırlanır ya da elenir. Bu da ezberleme probleminin önüne geçilmesi için ağlarda kullanılabilir. ReLU katmanı girdi matrisinin boyutunu değiştirmeden her bir elemanını bir eşik değerine tabi tutar. Evrişimsel katmanlardan sonra doğrusal olmayan bir aktivasyon fonksiyonu kullanılır. Toplu normalizasyon katmanı, doğrusal olmayan aktivasyon fonksiyonları (ReLU) ve evrişimsel katman arasında kullanılır. Eğitim sürecinin hızlandırılmasına katkı sağlar. Evrişimsel katman tarafından oluşturulan öznitelik haritaları burada normalize edilir. [78]

2.3.3.1. Öğrenmenin Aktarımı Metodu

Öğrenmenin aktarımı (transfer learning) günümüzde çoğunlukla makine öğrenimi konusu içerisinde geçse de gerçekte psikolojiye dayanmaktadır. Psikolojide, insanların bir konuda veya alanda öğrendiklerini, benzer özelliklerdeki başka bir konu veya alan da aktarma işlemidir. Bu konu, bilişsel psikolojinin temel aldığı araştırma alanlarından biridir. 1901 yılında Thorndike ve Woodworth ikilisi, Psikoloji literatürüne sokmuştur. "Öğrenme aktarımı bir konu öğrenilirken başka bir konudaki bilgi ve becerinin artması veya azalması şeklinde ortaya çıkar. Transfer, birbirine benzeyen konular arasında ise "yakın transfer", benzemeyen konular arasında ise "uzak transfer" olarak adlandırılır. Aktarımın

gerçekleşmesi, kişinin yetenekleriyle ilişkilidir. Kişinin benzerlik yorumlamasına dayalı olan sübjektif bir konudur. Benzerlik yargısının bazı alanları yaşam boyu gelişse de, temel mekanizmalar doğa tarafından zihnimize kodlanmıştır." şeklinde Uluslararası Eğitim Ansiklopedisinde 1992 yılında Perkins ve Salomon tarafından yazılmış bir açıklama bulunur. [65]

"Bir insanın matematik öğrenmesi, fizik çalışmasını kolaylaştırır, araba sürmeyi bilmesi, kamyon sürmeyi öğrenmesini kolaylaştırır, satranç bilmesi iş hayatında daha stratejik bir düşünür haline getirebilir. Transfer, eğitimde ve öğrenme teorisinde anahtar bir kavram olup çoğu örgün eğitim bu şekilde bir aktarımı amaçlamaktadır." [66]

Makine öğrenmesi alanında öğrenmenin aktarımı, daha önceden eğitilen bir modelin başka bir konuda kullanılabilmesi için kısmi eğitilmesi anlamına gelmektedir. Burada dikkat edilmesi gereken nokta "konuların ya da kullanım alanlarının birbirinden çok alakasız olmaması" gerekliliğidir. Yukarıda verdiğimiz örnekler ile şekillendirecek olursak "bisiklete binmeyi bilen bir kişi, kamyon kullanamaz". Eğitimle öğrenmiş model, bambaşka veri ile karşılaşırsa yeni veriyi anlamlandıramaz, hata miktarı büyür ve sonuç olarak veriyi düzgün kategorize edemez. Bu yaklaşım, makine öğrenmede kullanılan bir teknik olup yeni model daha önce öğrenilen bilginin üzerine inşa edildiğinden dolayı öğrenme süresinin daha kısa olması konusunda avantaj sağlar. Aynı zamanda büyük modellerin (derin ağların) eğitimi için güçlü donanımlar kullanılması gerektiğinden öğrenmenin aktarımı bu konuda da bize yardımcı olacaktır. Daha düşük performanslı donanımlar ile öğrenmenin aktarımı yöntemli eğitimler nispeten gerçekleştirilebilir.

Öğrenmenin aktarımı metodu uygulamaları derin ağ mimarilerinin eğitiminde önemli avantajlar sağlar. Birinci sırada, öğrenme hızındaki artış gelir. Eğitilmiş modellerin ağırlık katsayılarını barındırıyor olması bu hızın ortaya çıkmasındaki etkendir. Sınıftan öğrenen modellerin ihtiyacı olan büyük miktardaki veri yerine ince ayarlar (fine tuning) yapılarak daha az veri ile modeller eğitilip, başarılı sonuçlar alınabilir. Günümüzde evrişimli ağlar arasında iyi genelleme yapan ve büyük veri kümeleri ile eğitilip akademik çalışmalar için paylaşılan pek çok mimari bulunmaktadır. Modellerin isim ve özellikleri Tablo 2.1'de görülebilir. Bu modellerden türetilmiş birçok model de çeşitli amaçlarla kullanılmaktadır. Tablo 2.1'de görülen modellerin elde ettiği bilgiler, hedef modele doğrudan veya bir takım modifikasyonlar ile uyarlanabilir.

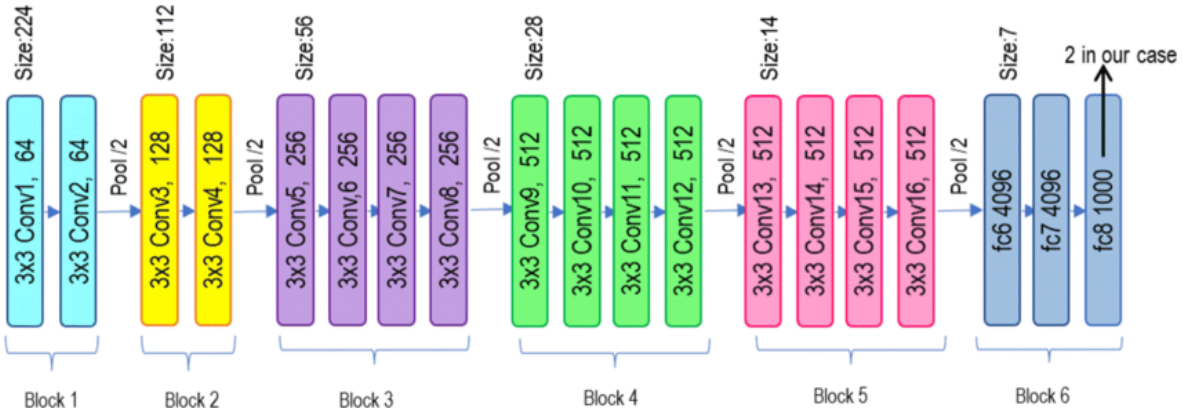
2.3.3.2. Vgg16 Derin Ağ Modeli

2014 yılında Oxford Üniversitesi Görsel Geometri Grubu tarafından oluşturulan ve on üç evrişim, üç tam bağlı ve diğer katmanların birleştirilmesiyle toplam 41 adet katman bulunmaktadır. Giriş katmanı görüntüsü 224x224x3 boyutundadır. Şekil 2.16'de VGG16 mimarisinin katman yapıları görülmektedir. Vgg-16, diğer modellere göre nispeten basit bir ağ modeli olup öncesindeki modellerden en önemli farkı evrişim katmalarının 2'li ya da 3'lü kullanılmasıdır. Tam bağlantı (FC) katmanında 7x7x512=4096 nöronlu bir öznitelik vektörüne dönüştürülür. İki FC katmanı çıkışında 1000 sınıflı softmax başarımı hesaplanır. Yaklaşık 138 milyon parametre hesabı yapılmaktadır. Diğer modellerde olduğu gibi girişten çıkışa doğru matrislerin yükseklik ve genişlik boyutları azalırken derinlik değeri (kanal sayısı) artmaktadır. Modelin her evrişim katmanı çıkışında farklı ağırlıklara sahip filtreler hesaplanır ve katman sayısı arttıkça filtrelerde oluşan öznitelikler görüntünün 'derinliklerini' simgelemektedir. [67]

Şekil 2.16 VGG16 mimarisinin katmanları.

2.3.3.3.Vgg19 Derin Ağ Modeli

2014 yılında Oxford Üniversitesi Görsel Geometri Grubu tarafından oluşturulan ve on altı evrişim, üç tam bağlı ve diğer katmanlarla birlikte toplam 47 adet katman bulunmaktadır. Giriş katmanı görüntüsü 224x224x3 boyutundadır. Şekil 2.17'de VGG19 mimarisinin katman yapıları görülmektedir.[67]



Şekil 2.17 VGG19 mimarisinin katmanları.

2.3.3.4. Mask-RCNN Derin Ağı

Mask R-CNN Facebook yapay zekâ ekibi tarafından önerilen ve gerçekleştirilen akademik çalışmalarda yaygın olarak kullanılan bir modeldir [68]. Bilinen nesne tespiti yanı sıra, tespit edilen nesneyi içeren çerçeve için maske oluşturarak, bölütlemeyi gerçekleştiren Mask RCNN'in basit ve etkili nesne bölütlemesi sunması bu yöntemin tez çalışması içerisinde seçilmesinde etkili olmuştur. Mask R-CNN yaklaşımına bakıldığında temelinde RESNET101 ve RESNEXT101 evrişimsel ağları ile nesne tespiti bulunduğu görülmektedir [68]. Bu yaklaşımda, görüntünün tamamına bölütleme işlemi yapmak yerine öncelikle nesne olması muhtemel alanlar yani ilgi alanları (ROI) tespit edilir. Sonrasında ise bu alanlar içerisinde nesne sınıflandırması ve bölütleme işlemi yapılarak maske çıkarılması işlemleri paralel olarak gerçekleştirilir [69]. Derin öğrenme yaklaşımları ile nesnelerin sınıflandırılması ve bölütlenmesinde ihtiyaç duyulan bir diğer kritik konu ise bu iş için kullanılacak olan veri setleridir. Tablo 2.3 ile literatürde faydalanılan farklı veri setleri karşılaştırmalı olarak değerlendirilmiştir. Tablo 2.4'te örnek bölütlemesinde yaygın olarak kullanılan veri setlerinin ismi ve açıklamaları yer almaktadır. Bu Veri setleri arasında yer alan MS-COCO veri seti nesne tespit ile bölütlemesi alanında derin öğrenme yaklaşımlarının geliştirilmesi ve sınanmasında sıklıkla kullanılan bir veri setidir [70]. Mask R-CNN ve Yolact++ ağ yapılarının hali hazırda bu veri seti ile eğitilmiş ağırlıkları bulunmaktadır. Bu ağırlıklar ile sayısal görüntüler üzerinde 80 farklı nesneyi bölütleri ile tespit etmek mümkündür.

Tablo 2.3 Yaygın Olarak Literatürde Kullanılan Instance Segmentation - Örnek Bölütleme Yöntemleri

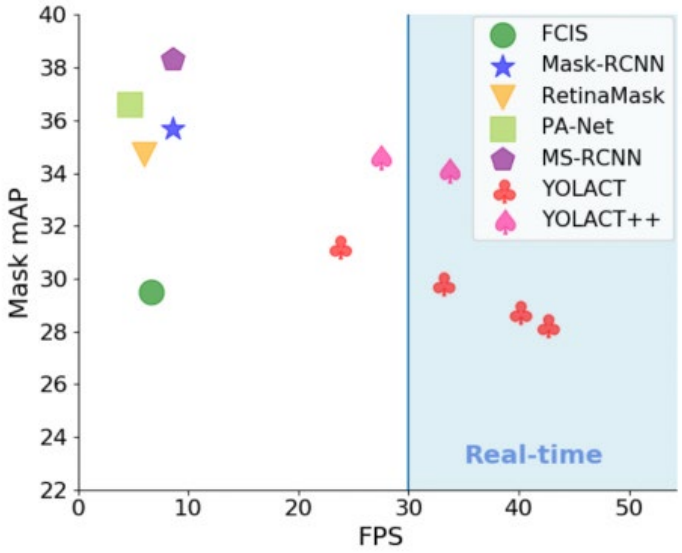
DeepMask (2015) [71]	Mask-RCNN yönteminin öncülü olan bu yöntemde, derin ağda öznitelik çıkartıldıktan sonra işlemler maskeleme ve sınıflandırma şeklinde devam eder. Yöntemi geliştiren ekip aynı zamanda Mask-RCNN'de yaratıcılarıdır.
PA-Net (2018) [72]	Uyarlamalı öznitelik ortaklaması yöntemi Pa-Net çalışmasında önerilmiştir. Mask-RCNN ile aynı alt yapıyı kullanmaktadır. Öznitelik çıkartma işleminden sonra elde edilen ilgi noktaları ve oluşturdukları bölgelerin doğruluğunu çalışmada arttırabilmişlerdir. Gerçekleştirilen testlerde Mask-RCNN'e göre daha başarılı olsa da ve performans olarak gerisinde kalmıştır.
RetinaMask (2019) [73]	RetinaMask, Retina-Net derin ağı üzerine kurulmuş bir yötemdir. Bu yöntem, Retina-Net'in tespit ettiği nesnelerin bölütlemesini yapacak şekilde bir ekleme ile oluşturulmuştur. Ortalama hassasiyet değeri olan mAP'değeri olarak Mask-RCNN'e çok yakın olsa da hız konusunda Mask-RCNN'in gerisinde kalmaktadır.
YOLACT (2019) [113]	YOLACT, diğer yöntemlere göre çok daha hızlı çalışabilen fakat mAP başarımı olarak geride kalan bir yöntemdir. Gerçek zamanlı örnek bölütlemesi yapabilir. Yöntem RetinaMask yöntemini temel almıştır.

Tablo 2.4.Görüntüde örnek bölütleme işleminde en yaygın kullanılan veri setleri

PASCAL VOC [74]	20 farklı sınıfta nesneler içerir. Her sınıf için fazla sayıda veri bulunmaktadır. 27.450 tane elle işaretli veri içeren 11.530 adet görsele sahtir. Bunlardan 6.929 adedi bölütleme için kullanılabilir. Veri setinde bulunan diğer verilerinde bölütleme işi için kullanılabilir olması konusunda çalışmalar devam ettiği bildirilmiştir.
Mapillary Vistas Dataset [75]	Günümüzde yaygınlaşmaya başlayan otonom arabalar için kullanılmak üzere geliştirilmiş ve köklü raç üreticileri tarafından da bu veri setinin geliştirilmesi devam ettiği bilinmektedir. Farklı caddelerden elde edilen 25.000 adet görsel içeren bu set içinde 37 farklı sınıf için bölütleme bulunabilir.
Cityscapes Dataset [76]	5000 adet görüntü içeren ve bunlardan 30 farklı sınıf için bölütleme barındıran yine otonom araçlar için kullanılabilecek cadde görüntülerinden oluşan bir veri setidir.
MS COCO [70]	Bölütleme işlemleri için tercih edilen en kapsamlı veri setidir. 80 farklı sınıfın olduğu 330.000 adet görüntü içerisinde 886.000 tane bölütleme bilgisi yer almaktadır.

2.3.3.5. Yolact++ Derin Ağı

Mask R-CNN modeli saniyede 8 ila 10 kare arasında maskeleme işlemi gerçekleştirebilirken, gerçek zamanlı bir örnek bölütlemesi sunan Yolact++ başarımı ispatlanan RetinaMask yöntemini temel almıştır. Saniyede 33,5 fps değerlere çıkabilen bu model hız açısından diğer yöntemleri geride bırakan bir performans sergilemektedir. [113]



Şekil 2.18 Yolact++'ın diğer metotlar ile hız ve başarım kıyaslama grafiği

Şekil 2.18'de görülebileceği gibi, gerçek zamanlı sistemlere uygulanabileceği belirtilen bu modelin tez çalışmasının başarımına omurga sağlamaktadır. Gerçekleştirilen tez çalışmasının ilk dönemlerinde uzun süre boyunca kullanılan Mask R-CNN, performansı çok düşük kalmış, bu nedenle büyük veri setlerinde kullanım için Yolact++ tercih edilmiştir. Her iki model de tez çalışmasında silüet maskeleri oluşturmak üzere yeniden şekillendirilmiştir. Öznitelik çıkartma yöntemi insanların siluetlerine dayalı olduğundan hazırlanan yoga videolarının içerisindeki insanlar öncelikli olarak tespit edilmiştir. Bu yöntem arka plan çıkartma, video kareleri arasında fark alma gibi veya ön plan tespitine ilişkin hareketli nesne tespiti gibi farklı algoritmalarla gerçekleştirilebilir. Fakat arka plan çıkarma işlemi genelde hareketli nesnenin olmadığı video karelerini içeren serilerde güzel sonuç verir. Karelerin farkının alınması ile de videolardaki insan siluetinin tamamı elde edilemeyip sadece hareketin olduğu nokta o video karesi içerisinde tespit edilebilir. Hazırlanan bu veri setinde kişiler yoga hareketlerini çok büyük oranda yer değiştirmeden yaptıkları için yukarıda bahsi geçen fark alma yöntemleri ile siluetlerin elde edilmesi mümkün olmamaktadır. Ön plan nesnesinin belirli bölgelerinin sabit kaldığı videolarda siluet tespit etme daha karmaşık bir işlem olup, modern yaklaşımlar kullanılarak gerçekleştirilmelidir. Bu yöntemler günümüzde derin öğrenme konu başlığı altında toplanan yöntemlerdir. Bu tez çalışmasında da videodaki insan siluetlerini elde etmek amacı ile Mask-RCNN ve devamında Yolact++ evrişimli yapay sinir ağlarından faydalanılmıştır. Her iki yöntemde görüntü içerisindeki birçok farklı nesne ve insanları

tespit edebilmektedir. Bulduğu nesneleri çevreleyen bir kutu ve her bir farklı nesne için ayrı bir maskeleme etiketi döndürmektedir. Yöntemin görüntü içerisinde bulabileceği nesnelerin listesi sınıf parametresinde aşağıdaki gibi tanımlanmıştır.

Class_names = [‘BG’, ‘person’, ‘bicycle’, ‘car’, ‘motorcycle’, ‘airplane’,’bus’, ‘train’, ‘truck’, ‘boat’, ‘traffic light’, ‘fire hydrant’, ‘stop sign’, ‘parking meter’, ‘bench’, ‘bird’, ‘cat’, ‘dog’, ‘horse’, ‘sheep’, ‘cow’, ‘elephant’, ‘bear’, ‘zebra’, ‘giraffe’, ‘backpack’, ‘umbrella’, ‘handbag’, ‘tie’, ‘suitcase’, ‘frisbee’, ‘skis’, ‘snowboard’, ‘sports ball’, ‘kite’, ‘baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']

Çalışmada her iki yöntemden faydalanırken, yöntemdeki ilgili fonksiyonun döndürdüğü, insan maskesini barındıran çevreleyen kutu koordinatları içerisindeki poz maskesi kopyalanarak, bu maskenin yeni bir görüntü olarak kaydedilmesi ve elde edilen bu RGB görüntünün ikilik görüntüye çevirdikten sonra konturunun çıkartması ve çıkan kontur üzerinden de öznitelik vektörünü oluşturmak için gerekli işlemlerin uygulaması şeklinde olmuştur.

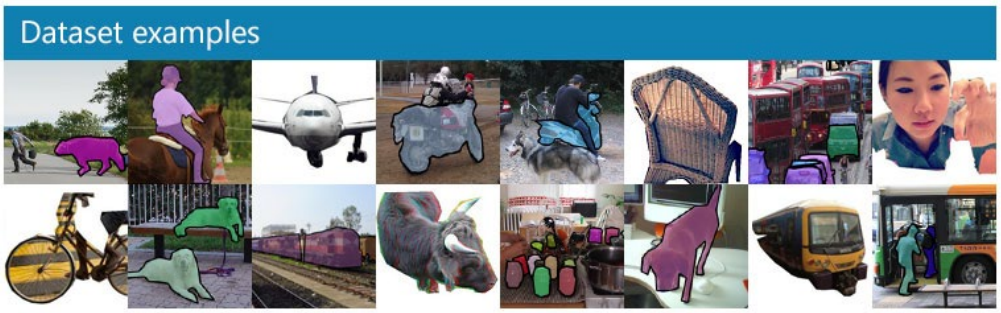
Derin öğrenme ile ilgili çalışmalar yaygın olarak Matlab ortamında veya Python dili ile geliştirilmiş paketlerde bulunabilir. Bu yöntemlerin ağır hesaplama maliyetleri olduğundan çoğunlukla büyük miktarda hafızalara sahip NVIDIA destekli grafik işlemciler kullanılarak hesaplamalar yapılmaktadır. Donanım olarak kullandığımız 12GB RAM’e sahip 2 GHz hızında i7 işlemcili Mac Mini Server’ın grafik işlemcisi 512MB’lık Intel HD Graphics 3000’dir. Dolayısı ile derin öğrenme paketlerinin GPU desteği bizim donanımımızda kullanılamamıştır. İşlemler ilk zamanlarda Matlab ve C++ dili kullanılarak yapılırken, çalışmanın sonuna doğru Python programlama diline kayılmıştır. Tüm çalışma için başlangıcından beri yazılan 30.000 satıra yakın Python kod’u vardır. Mask R-CNN’i kullanabilmek için mask_rcnn_coco.h5 modeli indirilip, model bu ağırlıklar ile çalıştırılmıştır.

2.3.3.6. COCO Modeli

Microsoft, FaceBook, Google, Mighty AI kuruluşlarının ve kar amacı gütmeyen CVDF (Ortak Görsel Veri Vakfı - Common Visual Data Foundation) destekleri ile yaratılmış, akademik çalışmalarda kullanılmak amacı ile hazırlanmış yüksek kaliteli, etiketli verilerin bulunduğu büyük boyutlu bir veri kümesidir. Şekil 2.19’da bir örneği görülen COCO,

büyük ölçekli bir nesne algılama, segmentasyon ve görüntü yazısı verisi barındırır. COCO'nun birçok özelliği vardır. Bunlar;

- Nesne segmentasyonu (object segmentation)
- Bağlam içinde tanıma (recognition in contex)
- Süper pikselli malzeme segmentasyonu
- 330K görüntü (> 200K etiketli)
- 1,5 milyon nesne örneği
- 80 nesne kategorisi
- 91 malzeme kategorisi
- Görüntü başına 5 görüntü yazısı
- Önemli noktaları olan 250.000 kişi



Şekil 2.19. COCO modelinin veri seti örnekleri

2.3.4. Sınıflandırma Metotlarının Performans Değerlendirme Ölçütleri

Kurulan modelin ya da algoritmanın, verilen görevi ne denli iyi yerine getirip getiremediğini öğrenebilmek için performansının değerlendirilmesi gerekmektedir. Performans değerlendirmesi makine öğrenmesinde önemli bir husus olup, bu amaçla birçok kıstas geliştirilmiştir. En sık kullanılan ölçütler Şekil 2.20'de bazılarının gösterildiği; doğru pozitif, yanlış pozitif, doğru negatif, yanlış negatif, doğruluk, kesinlik, duyarlılık, f - ölçütü, ROC eğrisi, AUC eğrisi, şeklindedir.

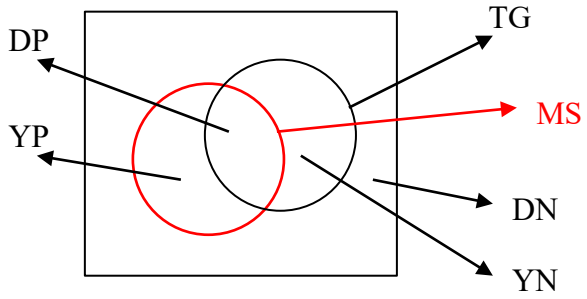
Denetimli öğrenme makine öğrenmesi yöntemleri kullanılarak gerçekleştirilecek tahmin/sınıflandırma çalışmalarında kullanılacak algoritmanın, var ise algoritmanın

parametreleri ve/veya hiper parametrelerinin seçimi, performans değerlendirilmesinde kullanılacak ölçütler ve veri setinin yapısı oldukça önemlidir.[79]

Seçilen sınıflandırma metodunun performans değerlendirme ölçütü (evaluation metric) iki farklı şekilde yapılabilir. Biri hassasiyet veya kesinlik (precision) bir diğeri de hatırlayabilme veya geri çağırma (recall) ölçütüdür. Bu ikilinin verdiği sayısal sonuç, kullanılan yöntemin seçilen özelliğin veya kurulan modelin ya da tüm sistemin, diğer yöntemler ile kıyaslarken ne kadar başarılı sonuçlar bulduğuna bir kanıttır sayılabilir. Sonuç başarısının ifade edildiği sayısal ikilidir. [80]

Örneğin bir koşma eylemi için metrik değerlerini yorumlarsak;

- **TG - Temel Gerçek:** Temel gerçek, çıkarımla sağlanan bilginin aksine, doğrudan gözlem ve ölçümle sağlanan, gerçek veya doğru olduğu bilinen bilgiler. Yani, koşma eylemi içeren videoların hepsi,
- **DP - Doğru Pozitif:** Doğru kabul edilenler. Ör: Koşma olarak etiketlendirilen ve gerçekten de koşma videolar,
- **YP - Yanlış Pozitif:** Yanlışlıkla kabul edilenler. Ör: Metodunun koşma olarak etiketlediği fakat, gerçekte koşma hareketi olmayan videolar,
- **DN - Doğru Negatif:** Doğru reddedilmişler. Ör: Koşma hareketi içermeyen reddedilmiş videolar (diğer sınıfa ait) Ör: Yürüme, vb.
- **YN - Yanlış Negatif:** Yanlışlıkla reddedilenler. Ör: Metodunun bulamadığı fakat koşma sınıfına ait olan videolar,
- **MS - Metodun Sonucu:** Sınıflandırıcının ve metodunun bulduğu ve koşmadır olarak etiketlediği videoların hepsi,



Şekil 2.20 Metrik değerlerin grafiksel ifadesi

Performans kıyaslaması için genellikle doğruluk (accuracy) değeri kullanılmaktadır ama bazen tek başına doğruluk değerinin kullanılması performans başarısının ölçümünde yetersiz olabilir. [81]

Tez çalışmasının tamamında tüm modellerin sınıflandırma başarımları, literatürde yaygın olarak kullanılan **Doğruluk (Accuracy)** olarak ifade edilen metrik değeri kullanılarak belirlenmiştir.

Doğruluk (Accuracy): Doğru sınıflandırılmış pozitif ve negatif örneklerin sayısının, toplam pozitif ve toplam negatif örnekler sayısına bölünmesi şeklinde hesaplanır ve Denklem 2.15'de gösterildiği şekilde ifade edilir..

$$Doğruluk\ (\ Accuracy) = \frac{DP + DN}{DP + DN + YP + YN} \qquad (2.15)$$

Geri Çağırma (Recall): Doğru sınıflandırılmış pozitif örnek sayısının, gerçek tüm pozitif miktarına bölünmesiyle hesaplanır ve Denklem 2.16'te gösterildiği şekilde ifade edilir.

$$Geri\ Çağırma\ (\ Recall\ ) = \frac{DP}{DP + YN} \qquad (2.16)$$

Kesinlik (Precision): Doğru sınıflandırılmış pozitif örnek sayısının, doğru pozitif tahminlerin, pozitif tahminlere oranıdır ve Denklem 2.17'te gösterildiği şekilde ifade edilir.

$$Kesinlik\ (Precision) = \frac{DP}{DP + YP} \qquad (2.17)$$

Özgünlük (Specificity): Gerçek negatif oranı olarak da adlandırılır. Sınıflandırıcının doğru olarak tanımlanan negatiflerin oranını yani olumsuz durumları ne kadar iyi tanımladığını ölçer ve Denklem 2.18'te gösterildiği şekilde ifade edilir.

$$Özgünlük\ (Specificity\ ) = \frac{DN}{DN + YP} \qquad (2.18)$$

Negatif Belirlilik Oranı: Sadece negatif olarak sınıflandırılan verilerin, gerçekteki negatif verilere oranını verir ve Denklem 2.19'te gösterildiği şekilde ifade edilir.

$$Negatif\ Belirlilik\ Oranı(\ Negative\ Predictive\ Value\ ) = \frac{DN}{DN + YN} \qquad (2.19)$$

F1 Skoru: Geri çağırma ve hassasiyetin, harmonik ortalamasıdır. Testlerin doğruluğunun bir ölçütüdür ve Denklem 2.20'te gösterildiği şekilde ifade edilir.

$$F1\ Skoru = \frac{2\ x\ Duyarlılık\ x\ Kesinlik}{Duyarlılık + Kesinlik} \qquad (2.20)$$

Ayrıca çalışmada sonuçlar **Karşıtlık Matrisi** (Confusion Matrix) kullanılarak verilmiştir. Karışıklık ya da karmaşıklık matrisi, sınıflandırıcının performansını gösterir. İkili (Binary) sınıflandırma için oluşturulan bir karşıtlık matrisi Şekil 2.43'de gösterilmiştir. Veri setinde yer alan farklı tüm sınıflar için hem satır ve hem de sütun ifadesi olmalıdır.

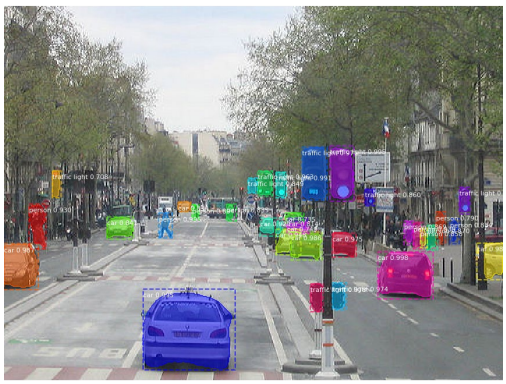
		Gerçek	
		Pozitif	Negatif
Tahmin	Pozitif	DP (Doğru Pozitif)	YP (Yanlış Pozitif)
	Negatif	YN (Yanlış Negatif)	DN (Doğru Negatif)

Şekil 2.21 – Karşıtlık Matrisi

3. ÇALIŞMAYA ÖZGÜ GÖRÜNTÜ TANIMI VE ÖZNİTELİK ÇIKARTMA YÖNTEMİ

Bu tez çalışmasında Mask-RCNN ve devamında Yolact++ siluet görüntülerini elde etmek amacı ile kodları yeniden şekillendirilerek kullanılmıştır. Mask-RCNN implemantasyonu, arka planda omurga yapısı olarak ResNet101'i kullanmaktadır. ResNet, Artık Ağ için kısa bir addır. Ağın adından da anlaşılacağı gibi, bu ağın sunduğu yeni terminoloji artık öğrenmedir. Derin evrimsel sinir ağları, görüntü sınıflandırma için bir dizi atılım sağlamıştır. Diğer birçok görsel tanıma görevi de çok derin modellerden büyük ölçüde yararlanmıştır. Bu yüzden yıllar geçtikçe daha derine inme, daha karmaşık görevleri çözme ve ayrıca sınıflandırma / tanıma doğruluğunu arttırma / geliştirme eğilimi vardır. Ancak, daha derine indikçe; sinir ağlarının eğitimi zorlaşır ve ayrıca doğruluk doygunluğa başlar ve sonra da düşer. Artık öğrenme, bu iki sorunu da çözmeye çalışır. Genelde, derin evrilmiş bir sinir ağında, birkaç katman istiflenir ve eldeki göreve göre eğitilir. Ağ, katmanlarının sonunda birkaç düşük / orta / yüksek seviye özelliği öğrenir. **Artık öğrenmede** (residual network), bazı öznitelikleri öğrenmeye çalışmak yerine, biraz kalıntı öğrenmeye çalışılır. Artık, basitçe o katmanın girişinden öğrenilen özelliğin çıkarılması olarak anlaşılabilir. ResNet bunu kısayol bağlantılarını kullanarak n'inci katmanının girişini bazı katmanların (n + x)'inci katmanına doğrudan bağlanması şeklinde yapar. Bu ağ türlerini eğitmenin basit derin evrimsel sinir ağlarını eğitmekten daha kolay olduğunu ve degradasyon doğruluğu probleminin çözüldüğünü kanıtlamıştır. ResNet'in temel konsepti budur **[77]**. Alt tarafta Şekil 3.1'de Mask-RCNN implemantasyonunun bir kaç görüntü üzerinde verdiği çıktılar görülmektedir. Eğitildiği nesnelerden olan "trafik ışıkları", "arabalar" ve "insan" yani yayaları etiketleyip her birini ayrı ayrı çevreleyen kutu içerisinde maskelemesi ve döndürdüğü sonuçlar görülebilir.

Şekil 3.1.a. Bir cadde görüntüsü



Şekil 3.1.b. NesnelerinMaskelenmiş hali

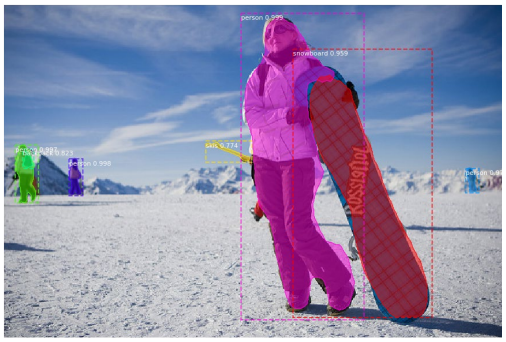
Yine Şekil 3.2'de model "bus" nesnesini bulmuş ve maskelemeyi gerçekleştirmiştir.

Şekil 3.2.a. Bir otobüs görüntüsü

Şekil 3.2.b. Maskelenmiş hali

"Snow Board" ve "Person" etiketlemeleri Şekil 3.3'de görülmektedir.

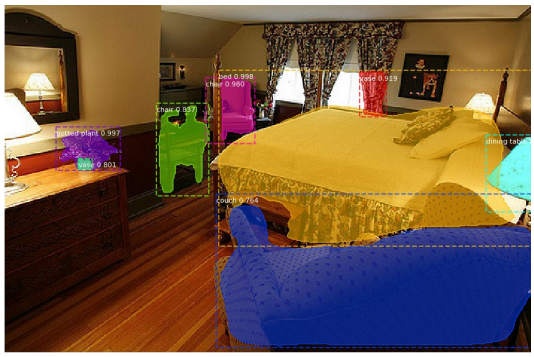
Şekil 3.3.a. Bir kayakçı görüntüsü



Şekil 3.3.b. Maskelenmiş hali

"Chair" , "Potted Plant", "Bed", "Couch", "Vase", "Dining Table" maskelemeleri ve çevreleyen kutu ile beraber tahmin edilme olasılığı Şekil 3.4'de görülebilir.

Şekil 3.4.a. Bir oda görüntüsü



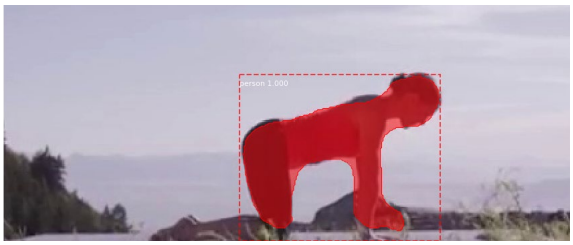
Şekil 3.4.b. Maskelenmiş hali

Çalışmada öncelikle Mask-RCNN'in Python dilinde yazılmış kodları yeniden şekillendirilerek, çalışmanın veri tabanındaki N adet videoyu seri bir şekilde açmasını ve

her bir videondan yarım saniyede bir video karesinde nesne algılama işlemini gerçekleştirmesini ve döndürdüğü sonuçları filtreleyip, bizim için önemli olan insan maskesinin elde edilmesi sağlanmıştır. İnsan figürünü çevreleyen kutu içerisindeki maskeler, videonun dosya ismi ve video kare numarasını birleştirerek, otomatik isimlendirip, veri tabanındaki her bir video sınıfı için ayrı bir klasör yaratarak, görüntü işleme yöntemleri ile ikilik forma çevrilen maskeler kaydedilmiştir. Bazı videolar için CNN, tez kapsamında etiketlenen görüntülerle eğitilemediği için aşağıda görüldüğü gibi maskeleme işlemleri esnasında eksiklikler ile karşılaşılmıştır.

Eksik etiketlemelere örnek olarak, Yoga sınıflarımızdan biri olan **"**Cat -Cow Pose" olarak bilinen Şekil 3.5'deki sırt ve omurga esnetme hareketini yapan kişinin diz kapağından ayak ucuna kadar olan bacak bölgesi eksik maskelenmiştir.

Şekil 3.5.a. Cat/Cow pozu



Şekil 3.5.b. Maskelenmiş hali

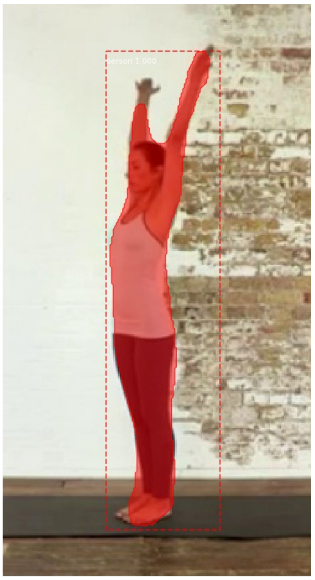
"Boat Pose" olarak bilinen Şekil 3.6'de görülen karın, bacak ve sırt bölgelerinin çalıştırıldığı videolardan birinde hareketi yapan kişinin bacakların kalça ve diz kapağı arasındaki bölgeyi maskeleyememiştir.

Şekil 3.6.a. Boat pozu

Şekil 3.6.b. Maskelenmiş hali

Kabul edilebilir hatalara bir örnek Şekil 3.7'da sağ elde kırpılma söz konusu fakat maske yüzde doksanın üzerinde pozu ifade ettiği için kullanılmıştır.

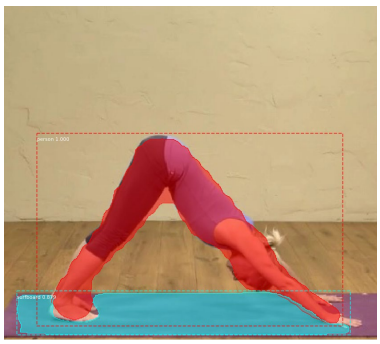
Şekil 3.7.a. Poz



Şekil 3.7.b. Maskelenmiş hali

Hatalı etiketlemelere örnekler; insan siluetini tam olarak tespit etse bile, Şekil 3.8.'deki gibi yoga matını "surf tahtası" olarak etiketlemesi.

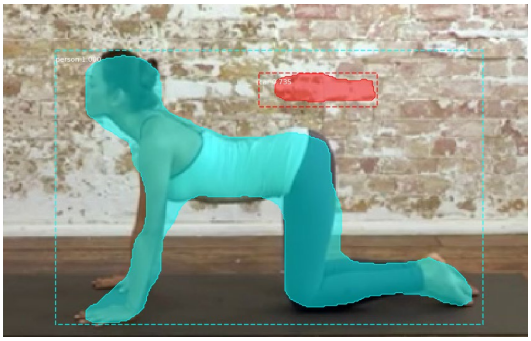
Şekil 3.8.a. Dog Leg Lift Pozu



Şekil 3.8.b. Hatalı maskelenmiş hali

İnsan siluetini tam olarak tespit etse bile, Şekil 3.9'daki duvar deseninin bir kısmını araba şekline benzetmesi ve etiketlemesi.

Şekil 3.9.a. Cat Pozu



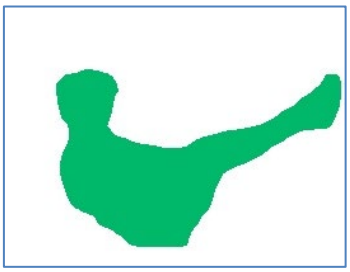
Şekil 3.9.b. Hatalı maskelenmiş hali

Kodları yeniden şekillendirilen MaskCNN uygulamasında, oluşturulan tüm video veri seti, maskeleme sonuçlarının tutarlılığını görmek amacı ile baştan verilmiştir. Bu model COCO veri seti ile eğitildiği ve donanımsal engellerden dolayı kendi yoga poz görüntüleri ile eğitilemediği için videoların bazılarında hiç maskeleme uygulayamamıştır. Hatalı sonuç üreten videolar kullanılmamış ve tamamen veri tabanı içinden ayıklanmıştır. Bu elemelerden sonra hazırlanan 310 adet videodan 291 adedi geriye kalmıştır.

Tek karelik görüntü için Python, Matlab ve C++ programlarının çıktıları aşağıda sırası ile Şekil 3.10 ve Şekil 3.11'de görülmektedir.

Şekil 3.10.a. Boat Pozu

Şekil 3.10.b. Tespit ve Maskeleme

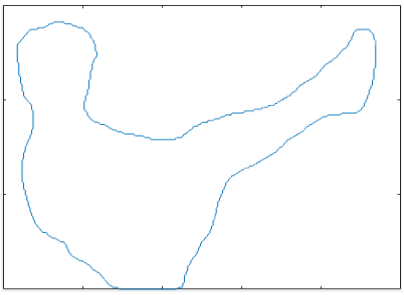


Şekil 3.10.c. Maskeyi Görüntüden Ayırma

Şekil 3.11.a. İkilik Forma Dönüştürme

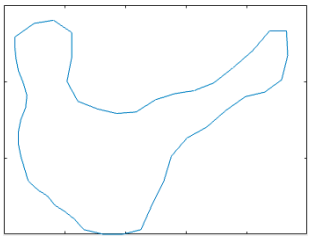
Şekil 3.11.b. Boş Alanları Kırpma

Şekil 3.12'de görülen orjinal ikilik görüntüde 639 adet koordinat noktası vardır. Yani bu görüntü aslında 639 poligondan oluşmaktadır.



Şekil 3.12 Boat pozuna ait Şekil 3.11'daki ikilik görüntünün orjinal kontur görüntüsü

Poligonlaştırma işlemleri sonucu Şekil 3.12'deki 639 koordinat, 50 adet koordinata düşürülmüştür ve Şekil 3.13'de gösterilmektedir.



Şekil 3.13 Hedef poligon sayısına indirgenmiş orjinal poz görüntüsü

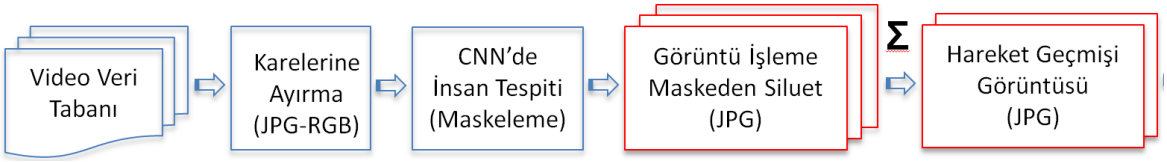
Bu işlemler tez çalışmasının ilk zamanlarında Mask-RCNN ve birden farklı dille gerçekleştirilirken, çalışmanın devamında tüm adımlar Python'a kaydırılmış ve bir bütünlük sağlanmıştır. Aynı zamanda tasarım bu altyapıyla çevrimiçi sistemler üzerinde de kullanılabilir kılınmıştır. Çalışmanın devamında benzer adımlar Yolact++ ile yürütülmüş ve modelin gerçek zamanlı cevap üretebilme performansı sayesinde çalışma hız kazanmıştır. İzlenen adımlar Mask-RCNN'de anlatılanlarla aynıdır. Çalışmanın bu adımından sonra 3.1'de anlatılan HGG görüntülerinin oluşturulması gelmektedir.

3.1 Hareket Geçmişi Görüntüsü (HGG) Oluşturma Metodu

Bu tez çalışmasında öğrenmenin aktarımı yöntemi, derin ağ mimarileri olan VGG16 ve VGG19 üzerinde kullanılmıştır. Yöntemin uygulaması, videoların içerisindeki siluetlerden elde edilen ve HGG olarak isimlendirilen, hareketi daha ayrık özetleyen,

"Hareket Geçmişi Görüntüsü" kullanılarak yapılmıştır. Şekil 3.14'te HGG'lerin oluşturulma sürecinin genel akış diyagramı görülmektedir. HGG'lerin çalışma içerisinde iki farklı tipi vardır. Bunlardan biri, öznitelik vektörü oluşturmak için kullanılacak olan renksiz formu, diğeri de VGG16 ve VGG19'un, öğrenmenin aktarımı yöntemine göre eğitimlerinde kullanılanılacak olan yapay renklendirilmiş modelidir.

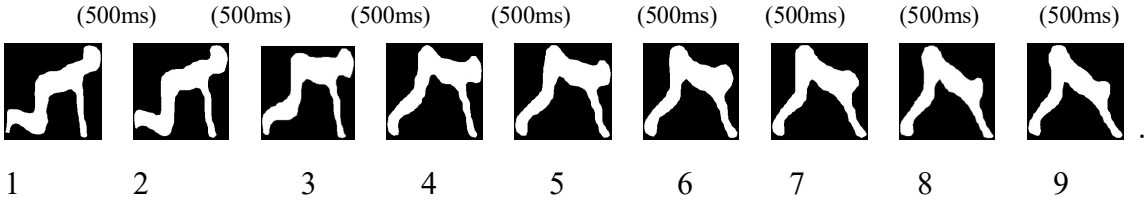
HGG'ler, video karelerinden elde edilen hareketin pozuna ait siluet görüntülerinin, arasında bir zaman aralığı bırakarak ve ardışıl olmayanları ağırlık merkezlerine göre toplanarak elde edilir. Bu HGG'lerin renksiz olanları Algoritma-2'ye gönderilip poligonlaştırılarak, öznitelik vektörü oluşturmak amacıyla kullanılır. Aynı HGG'lerin renkli olarak kaydedilenleri ise, derin ağ mimarilerinin öğrenmenin transferi metoduyla eğitimi için kullanılmıştır.



Şekil 3.14 HGG - "Hareket Geçmişi Görüntüsü" oluşum akış diyagramı.

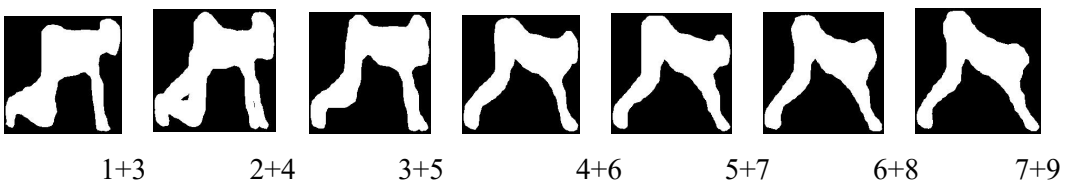
Tip-1 HGG oluşturma adımları;

1- Video içerisinden 500ms aralıklarla siluetler elde edilir.



Şekil 3.15 Video karelerinden kesikli zamanda elde edilen siluetler.

2- Ardışıl olmayan ları ağırlık merkezine göre toplanır ve renksiz HGG'ler Algoritma-2'ye gönderilir.



Şekil 3.16 Ardışıl olmayan siluetlerin ağırlık merkezine göre toplanması.

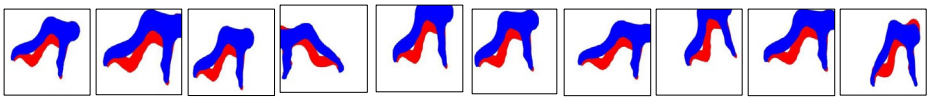
Tip-2 HGG oluşturma adımları;

1- Derin öğrenme modellerini eğitme amacı ile kullanmak için yukarıda anlatılan adım-1'de elde edilen silüetler kırmızı ve mavi renk ile yapay olarak renklendirilir, sonra ardışıl olmayanlarının toplanması ile Şekil 3.17'deki gibi bir formları elde edilir.

1+3 2+4 3+5 4+6 5+7 6+8 7+9

Şekil 3.17 Ardışıl olmayan siluetlerin yapay renk ile renklendirilip toplanması

2- Kaykılma, döndürme, öteleme, yansıma, büyütme küçültme benzeri işlemler uygulanarak Adım-1'de elde edilen her bir görüntü 10 katı miktarında fazlalaştırılır.

(2+4) x 10 adet 

Şekil 3.18 Veri çoğaltma adımı uygulanmış, Şekil 3.17'da yer alan ikinci görüntü.

3- Adım-2'de elde edilen tüm görüntüler, Tablo 2.1'de görülen CNN modellerinin girişine göre boyutlandırılarak artık kullanılabilir.

3.2. Siluet Görüntüsü Oluşturma ve Poli-Silüet Algoritması (PoS) (Algoritma-1)

Tez çalışmasında önerilen çözüm yöntemi, insan aksiyon videolarının silüet tabanlı sınıflandırması için yeni bir öznitelik çıkarma algoritmasından oluşmaktadır. Çalışmada bu amaçla birden fazla yenilik içeren algoritma sunulmuştur. Bu çözüm serisi, içerisinde silüet üretimi, silüet görüntülerinin poligonlaştırılması ve bu poligon şeklinin kodlanmasını kapsamaktadır. Poligonlaştırma algoritmamız eğri ve görüntü poligonlaştırmasına dayansa da bunlara benzemez [82 - 92].

Siluet elde etme ve tanımlama adımları yukarıda bahsedilen derin ağ mimarilerinin kodları yeniden şekillendirilmiş formları kullanılarak gerçekleştirilmiştir. Önerilen görüntü poligonlaştırma ve buna bağlı olarak çalışan poligon kodlama yöntemi bir bütün olarak

çalışmaktadır. Poligonlaştırma adına poli-silüet dediğimiz (PoS) görüntülerinin kontur koordinatlarında hızlı ve doğru çalışır. Hız, doğruluk ve siluet oluşturma için Yolact++ omurgasını etkin bir şekilde kullanır. Poligonlaştırma adımı ihtiyaca göre uyarlanabilir ve her ikilik görüntü için tamamen anında çalışır. Bu tez çalışmanın yeniliklerinden biri, silueti elde etmek için video görüntüleri arasındaki fark ve arka plan çıkarma teknikleri gibi geleneksel yöntemlere [93- 95] başvurmak yerine, derin öğrenme mimarilerinden biri olan Yolact++'ın kodlarını değiştirerek ara katmanlarında oluşan görüntünün siluet görüntüsü olarak kullanılmasıdır. Derin ağ mimarisinin gücünden yararlanılarak silueti bir bütün olarak elde edilmesi ve siluetin, geleneksel yollardan daha fazla doğrulukla oluşturabilmesi noktasında önemlidir.

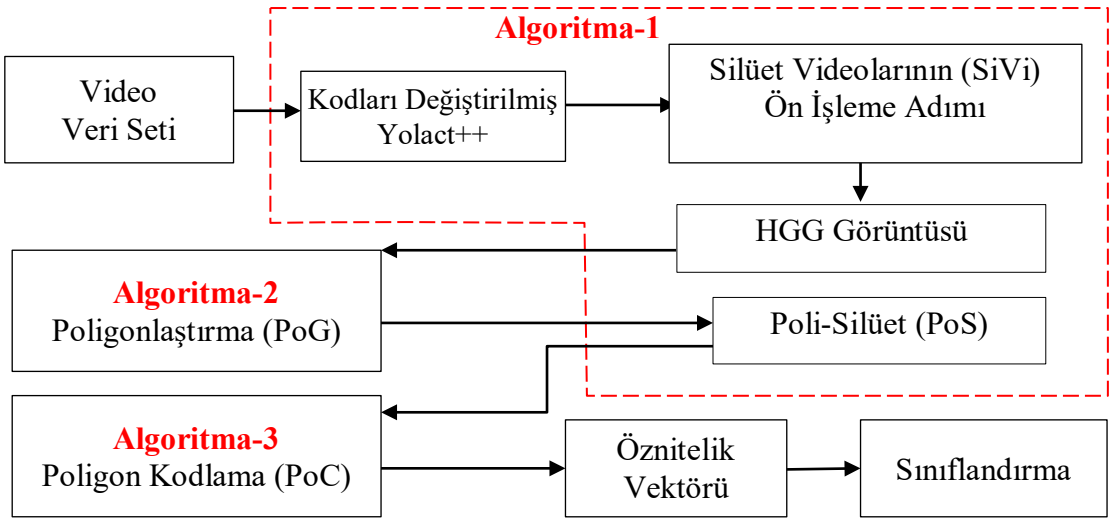
Çalışmadaki yeniliklerden bir diğeri, ardışık çerçevelerden elde edilen siluet görüntülerini toplanarak eylemi daha iyi tanımlayan görüntü ifadelerinin oluşturulması ve bu görüntülerin poligonlaştırılmış hallerinden elde edilen poligon kodları (PoC) içerisindeki 3'lü, 4'lü ve 5'li kod parçalarının, PoC içerisindeki bulunma sıklıklarını bilgisini de kullanarak öznitelik vektörleri elde edilmesidir.

Poli-Siluetlerin oluşturulabilmesi için öncelikle silüet görüntülerinden oluşan videolar elde edilmelidir. Bu adımın büyük veri setlerinde uzun zaman almasından dolayı bu işlemde hazırladığımız Yolact++ kullanılmıştır. Bu sayede Yolact++'a girdi olarak verilen her video, ismine Siluet-Videosu (SiVi) dediğimiz forma dönüşmektedir. SiVi olarak isimlendirdiğimiz videolardan bir örnek alt kısımda Şekil 3.19'da Kaya duvarı tırmanışı yapan bir kişinin videonun farklı zamanlarından alınmış kesitlerinde gördüğümüz vücut poz maskeleridir.

Şekil 3.19 Kaya Duvarı Tırmanşına ait Siluet-Videosu 1, 7, 11. saniyelerdeki görüntü.

Veri setindeki her videonun öncelikle SiVi'leri oluşturulur. Önerilen çözüm adımları içerisinde birbirini takip eden birden fazla algoritma bulunmakta olup, algoritmaların genel akış şeması Şekil 3.20'de yer almaktadır.

Birçok insan eylemi hızlı değildir, eklemlerin hareketi yavaştır ve vücudun uzuvları da eklemlerin serbestlik derecesiyle sınırlıdır. Bu nedenle, birbirini izleyen video kareleri arasında silüetlerde önemli bir değişiklik gözlemlenemez. Bu durumun üstesinden gelmek ve çerçeveler arasında bir boşluk oluşturmak için, SiVi videolarından bir PoS görüntüsü oluşturmadan önce Denklem 3.1'de gösterilmiş bir ön işleme tabi tutulur ve ortalama kare sayısı değeri hesaplanır. Ortalama kare sayısı değerinin yarısı, PoS adı verilen harekete özel ayrık çokgen görüntüler oluşturmak için ardışık kareler arasında boşluk bırakmak için kullanılmıştır.



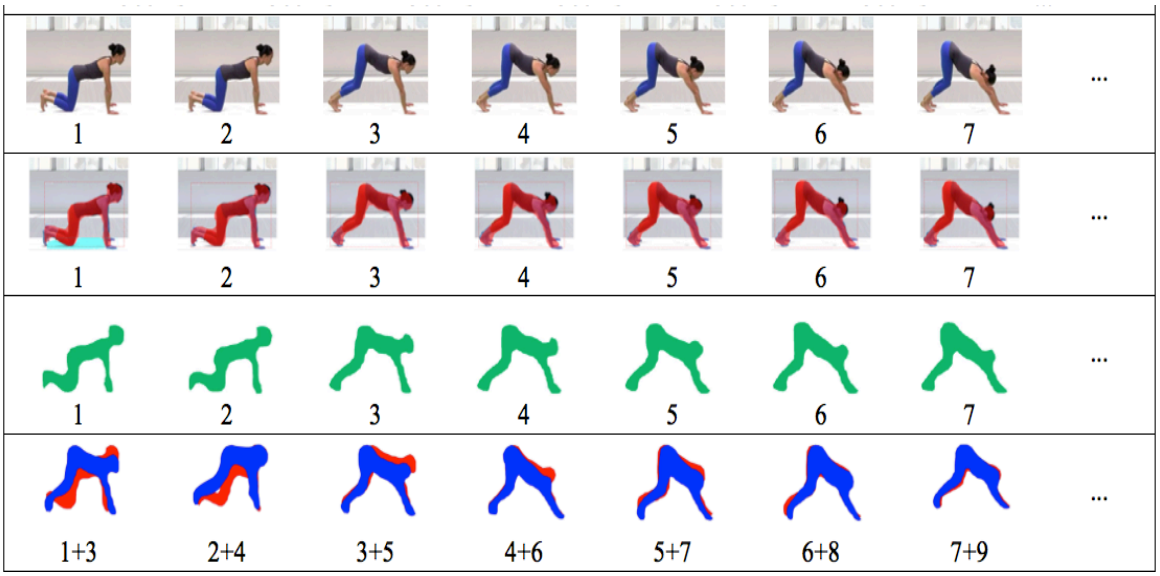
Şekil 3.20 Çalışmada önerilen çözüm adımlarının genel akış şeması

Literatürde kullanılan büyük video veri setleri olan HMDB51 ve UCF101 veri setlerinde saniyedeki kare sayısı (fps) değeri 29, oluşturulan yoga veri seti için bu değer 60'tır. FPS değerinin yarısında alınan karelerin toplamından elde edilen görüntüyü kullanmak, her kare için segmentasyon yapan derin ağ modelinin işlem hızını kare sayısının yarısı misli arttırmak anlamına gelmektedir. Sonuç olarak, işlem süresi azaltılmış ve genel sistem performansı makineden bağımsız iyileştirilmiş olur.

$$AvrgFPS = \left(\sum_{i=1}^{TVC} fps\ of\ SiVi(i) \right) / TVC \qquad (3.1)$$

Denklem (3.1), Ortalama Kare Sayısı (AvrgFPS) hesaplanmasını açıklar. SiVi veri kümesindeki her video için o videonun saniyedeki kare sayısı (fps) değeri toplanır ve sonuç toplam video sayısına (TVC) bölünür.

PoS üretimi için, önce SiVi video veri kümesi oluşturulmalıdır. Algoritma 1 Adım 5'teki kare sayısı güncelleme adımında ortalama fps'nin yarısını kullanmak, Şekil 3.20'de gösterildiği gibi Algoritma 3'te açıklanan öznitelik çıkarmada kullanılabilecek daha spesifik görüntüler üretir. Şekil 3.21'nin 4. satırında gösterildiği gibi, 3. sıradaki görüntüler boşluk değerine göre toplanır ve bu toplamlar Algoritma 2'nin çıktısından PoS görüntülerini oluşturmak için kullanılır.



Şekil 3.21. Yukarıdan aşağıya, 1. sıra: orijinal video kareleri. 2. sıra: Arka plan ve Yolact++ sonuçları. 3. sıra: Kodları yeniden şekillendirilmiş Yolact++ sonuçları (maskeler yeniden boyutlandırılır ve arka plan kaldırılır). 4. sıra: Algoritma 1 adım 10 tarafından renklendirilen ve toplanan ardışık olmayan maskeler. Ortaya çıkan görüntüler poligonlaştırma için Algoritma 2'ye gönderilir.

Algoritma-1. Poli-Siluet (PoS) Algoritması

• TfC, Videonun toplam kare sayısı.

• TvC, Video veri kümesinin toplam video sayısı.

Girdi: SiVi, Siluet-Videosu. (Kodları yeniden şekillendirilmiş Yolact++ çıktısı)

Çıktı: PoS, Poli-Silüet görüntüler.

01: Ön işlem adımı:

$Denklem\ (3.1)\ ile\ video\ veri\ setinin\ ortalama\ kare\ sayısını\ bul.$

02: $for\ i = 1\ den\ TvC'e\ kadar$

03: $Gözcü\ kontrollü\ yapı\ içinde\ for\ j = 1\ den\ \ SiVi(i)'nin\ TfC'sine\ kadar$

04: $img1\ \leftarrow\ SiVi(i)'nin\ j^{th} karesini\ oku$

05: $j'yi güncelle,\ j \leftarrow (\ j\ + (AvrgFPS/2)\ )$

06: $eğer\ SiVi(i)'nin\ \ j^{th}.\ karesi\ varsa$

07: $img2 \leftarrow SiVi(i)'nin\ j^{th} karesini\ oku$

08: $değilse$

09: $img2\ \leftarrow\ SiVi(i)'nin\ son\ karesini\ oku$

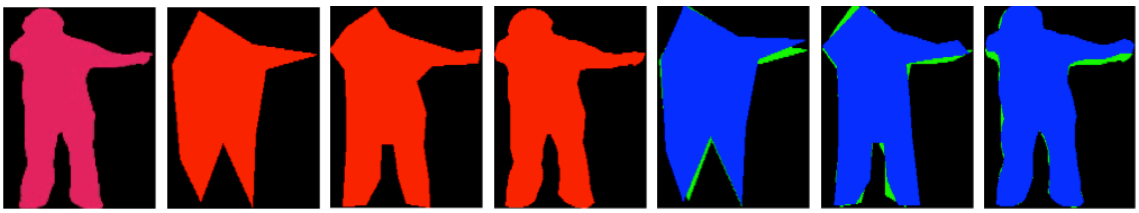
10: Toplam Görüntüsü Oluşturma Adımı:

img1 ve img2 içinde, silüeti çevreleyen en küçük kutuyu bul, sırasıyla mavi ve kırmızı renk ile doldur, görüntüleri ağırlık merkezlerine göre topla ve Tip-2 hareket geçmişi görüntüsünü (HGG) oluştur.

11: Oluşan Tip-1 HGG'yi, Algoritma 2'ye Poligonlaştırmak için gönder. Dönen resmi poli-silüet (PoS) olarak kaydet.

3.3. Poligonlaştırma Algoritması (PoG) (Algoritma-2)

Silüet'in poligonlaştırılması, Şekil 3.22'te (b-d) gösterildiği gibi, siluetin dış şeklinin daha az kenarla tanımlanması anlamına gelir.



Şekil 3.22 **(a)** SiVi 'den elde edilmiş minimum çevreleyen kutusu içerisindeki vücut silueti.
(b-d) Hedef Kenar Sayısı (TeC) değerleri 10, 27 ve 104 için örnek poligonlaştırma sonuçları
(e-g) Farklı TeC değerleri (10, 27 ve 104) için Algoritma 2'nin çıktıları.

Poligonlaştırma Algoritması (PoG), Algoritma-1'den girdi görüntüsü alır ve bunları poligonlaştırmadan önce, ön işlemden geçirir. Yanlış kontur algılamayı en aza indirmek için birçok morfolojik işlem ve renk dönüşümü gerçekleştirilmesine rağmen, ön işlemenin tespit edemeyeceği kusurlar kalabilir. Bazı durumlarda, özellikle büyük veri setlerinde (HMDB51 ve UCF101) geçerli olan görüntüde birden fazla kişi olma durumuyla karşılaşılabilir. Bu algoritma, bir seferde yalnızca tek silueti işler. Tespit edilen silüetin sınır değerlerinden çıkartılan kontur koordinat listesi (CcL) üzerinde eleme işlemi yaparak çalışan bu algoritmanın boyut indirgeme (eleman silme) adımında iki duyarlılığı vardır. CcL uzunluğu, hedef kenar sayısı (TeC) değerinin iki katından büyük ise benzer olmayan koordinatları, CcL uzunluğu TeC değerinin iki katından büyük değil ise, benzeyen koordinat değerini listeden eler. Bu eleme hedef koordinat sayısı elde edilinceye kadar devam eder. İşlem sonunda CcL listesinde TeC adedi kadar koordinat kalmış olur. Bu bilgi ile siluet çizimi yapılınca, siluetin genel duruşu Şekil 3.22 (b-d)'de görüldüğü gibi orijinal silueti andırır ve deforme olmaz.

$$k = \frac{CcL\ Uzunluğu}{TeC\ değeri} \qquad (3.2)$$

Algoritma-2. Poligonlaştırma Algoritması (PoG)

• TeC: Hedef Kenar Sayısı.

• CcL: Kontur Koordinat Listesi.

Girdi: TeC ve Algoritma-1'de üretilen HGG görüntüsü

Çıktı: Poligonlaştırılmış görüntünün kontur koordinat listesi

---

00: *Ön işlem adımı:*

01: *Girdi olarak gelen görüntünün kontur koordinat listesini (CcL) oluştur.*

02: $while\ CcL\ boyu\ \ TeC'den\ küçük\ olmadığı\ sürece$

03: $Eğer\ (CcL\ boyu)\ >\ (\ 2 * TeC)\ ise$

04: $i \leftarrow 1,\ \ k\ \leftarrow Adım\ Boyutu,\ \ y \leftarrow Boş\ Liste$

05: $while\ i < (CcL\ boyu)$

06: $y \leftarrow ekle\ (i + k)$

07: $i\ \leftarrow i + k$

08: *y listesindeki indis değerlerini silerek CcL listesini güncelle*

09: *Değilse*

10: $j\ \leftarrow 1$

11: *while j* < (*Güncel CcL boyu*)

12: $Eğer\ (CcL\ boyu) >\ TeC$

13: *CcL'i, (j + TeC / 2) indisine denk gelen elemanı silerek güncelle*

14: $j \leftarrow j + 1$

15: *Güncellenmiş CcL listesini döndür.*

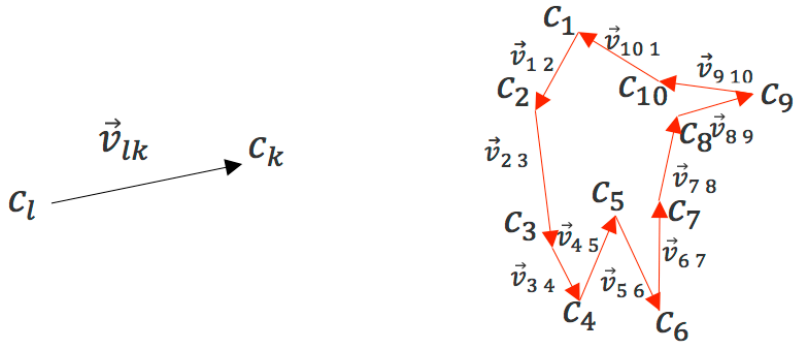
Bu algoritma ile işlem yapılırken, bir görüntüdeki tüm nesneler için kontur koordinat listeleri (CcL) oluşturulur. Bu CcL'ler, görüntüdeki her nesnenin sol üst köşesinden başlanarak ayrı ayrı yeniden sıralanır. Daha sonra en büyük CcL seçilir. Bunun anlamı, görüntüdeki en baskın silüeti seçmek olup, devamında kontur azaltma işlemi yani poligonlaştırma işlemi başlar. Bu süreçte ilk olarak, Denklem (3.2) ile adım boyutu, k hesaplanır. Bu işlem, kontur koordinat listesi (CcL) boyutunun hedef kenar sayısına (TeC) bölünmesiyle gerçekleşir. CcL'nin k ile artan indeks değerine sahip bir gözcü döngüde,

başlangıçta boş olan y listesi eleme için aday kontur noktalarıyla doldurulur. Oluşturulan y listesi ve Algoritma-2'deki 10 ila 14 aralığındaki koşullar ile silinmeye aday konturlar silinerek nihai CcL elde edilir. Şekil 3.22'de (b-g) indirgenmiş kontur'lu görüntüler görülebilir. İndirgenmiş konturun son şekli, orijinalin poza benzer. Poligon tabanlı insan silueti oluşturulduğunda, köşe sayısı farklı olan tüm poligonlar, Şekil 3.22'de görüldüğü gibi aynı vücut pozisyonuna benzer.

3.4. Poligon Kodlama (PoC) (Algoritma-3)

C kümesinin, $C = \{c_1, c_2, , c_n\}$ şeklinde n noktadan oluştuğu varsayılırsa ve C kümesindeki her noktanın $c_l = (x_{c_l}, y_{c_l})$ gibi iki boyutta koordinatları vardır. C'den bir c_l noktası alarak ve C'nin geri kalan noktalarıyla, Şekil 3.23 b'deki gibi birleştirerek, ardışık noktalar arasında Şekil 3.23 a'daki gibi bir $\vec{v}_{lk}$ kenar vektörü oluşturulur ve Denklem (3.3)'e göre bir V_m vektör kümesi elde edilebilir. Bu vektör kümesi, poligon şeklinin konturunun kenar vektörlerini oluşturur. SiVi'nin, poli-siluet oluşturan video kare adedi N tane ise, toplam m = 1'den N'e kadar V_m vektör kümesi olacaktır. Yani video veri tabanından elde edilen her silüet videosunun her HGG görüntüsüne ait bir V_m vektör kümesi oluşturulur.

$$V_m = \begin{cases} \vec{v}_{lk} = \overrightarrow{c_l c_k}, & l = 1, 2, , n-1, \; k = l+1 \\ \vec{v}_{lk} = \overrightarrow{c_l c_k}, & l = n, \qquad\quad k = 1 \end{cases} \tag{3.3}$$



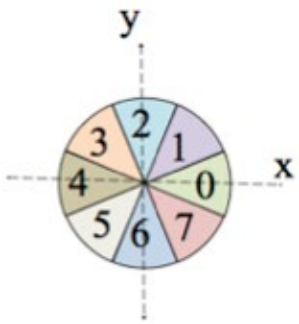
Şekil 3.23 **(a)** Örnek kenar vektörü $\vec{v}_{l\,k}$, **(b)** $\vec{v}_{1\,2}$'den $\vec{v}_{10\,1}$'e kadar 10 kenar vektörü ile tanımlanan örnek poligonlaştırılmış silüet görüntüsü.

V_m kümesindeki her $\vec{v}_{lk}$ vektörünün büyüklüğü ve açısı vardır. Büyüklükler, Denklem (3.4) ile hesaplanır.

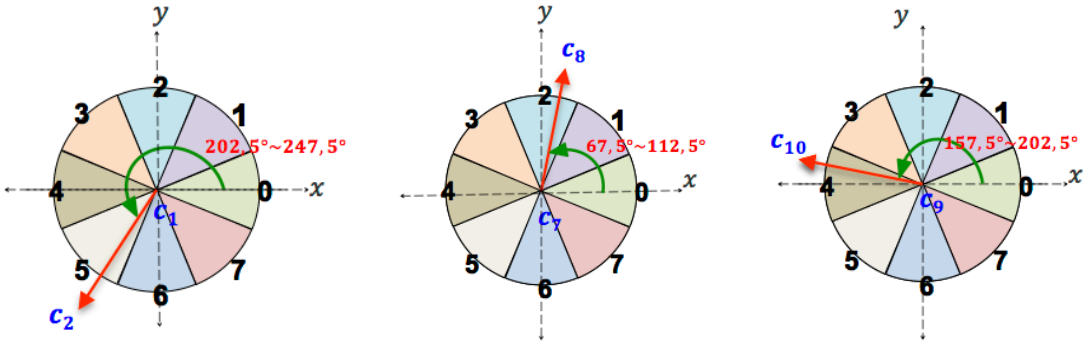
$$\|\vec{v}_{lk}\| = \sqrt{(x_{c_k} - x_{c_l})^2 + (y_{c_k} - y_{c_l})^2} \qquad (3.4)$$

V_m kümesindeki her $\vec{v}_{lk}$ vektörü için vektör kodu, Tablo 3.1'deki referans açısına (Φ) göre hesaplanır. Tablo 3.1'de görülen kutupsal koordinat açı aralığı, birim daire üzerinde bir dilim oluşturur. Bu açı aralığına giren $\vec{v}_{lk}$ vektörü o bölgeye atanmış kod ile eşleştirilir. V_m kümesindeki her $\vec{v}_{lk}$ için belirlenen kod bir dizi, bu dizinin içindeki sayıların tamamı da, poligonun kodunu belirler.

Tablo 3.1. Açısal alanlar ve bölge kodları.

	KUTUPSAL KOORDİNAT AÇISI (Φ)	KOD
	(-22,5° - 22,5°]	0
	(22,5° - 67,5°]	1
	(67,5° - 112,5°]	2
	(112,5° - 157,5°]	3
	(157,5° - 202,5°]	4
	(202,5° - 247,5°]	5
	(247,5° - 292,5°]	6
	(292,5° - 337,5°]	7

Verilen poligon şeklinin poligon kodunu (PoC) elde etmek için her kenar vektörünün kodu ayrı ayrı hesaplanması gerekir. Bunu yapmak için, $\vec{v}_{lk}$ kenar vektörünün başlangıç noktasının Kartezyen koordinat sisteminin orijininde olduğu varsayılır ve x-ekseni ile vektör arasındaki süpürme açısı hesaplanır. Şekil 3.23 b'de görülen poligon şekline ait bir kaç vektörün kodunun görsel oluşturuluş şekli, Şekil 3.24'te gösterilmiştir.



(a) $\vec{v}_{12} = \overrightarrow{c_1c_2}$ $Kod = 5$ **(b)** $\vec{v}_{78} = \overrightarrow{c_7c_8}$ $Kod = 2$ **(c)** $\vec{v}_{9\,10} = \overrightarrow{c_9c_{10}}$ $Kod = 4$

Şekil 3.24 Şekil 3.23'de görülen bir kaç vektör ve kodları

(a) vektör $\vec{v}_{12}$, (b) vektör $\vec{v}_{7\,8}$, (c) vektör $\vec{v}_{9\,10}$

Algoritma-3. Poligon Kodlama Algoritması (PoC)

Girdi: PoS görüntüsünün Kontur Koordinat Listesi (CcL)

Çıktı: PoS görüntüsünün Poligon Kod Listesi

00: *Poly Code Listesini boş liste olarak ilklendir*

01: while (*i < CcL boyu)*

02: *CcL'in i'inci indis* (x, y) *koordinatını* c_l *olarak ata*

03: *CcL'in* $(i+1)'$*inci indis* (x, y) *koordinatını* c_k *olarak ata*

04: Δx *ve* Δy *değerlerini,* c_l *ve* c_k *değerlerini kullanarak hesapla*

05: c_l *ve* c_k *arasındaki Öklid mesafesini hesapla ve vektörün boyunu bul*

06: $\vec{v}_{lk}$ *vektörünün kodunu Tablo 3.1'e göre bul*

07: *PolyCodeList* $\leftarrow$ *ekle* ($\vec{v}'_{lk}$*nın kodu*)

08: i'*inci* $\vec{v}_{lk}$ *vektörüne ait hesaplanan tüm özellikleri kaydet*

09: $i \leftarrow (i+1)$

11: *Eğer (i+1), CcL boyuna eşit ise*

12: *CcL'in i'inci indis* (x, y) *koordinatını* c_l *olarak ata*

13: *CcL'in* $0'$*inci indis* (x, y) *koordinatını* c_k *olarak ata*

14: *Son kodu bulmak için* $4'$*den* $8'$*e kadar adımları tekrarla ve özellikleri kaydet*

15: *PolyCodeList* $\leftarrow$ *ekle* (*Son Kodu*)

16: *PolyCodeListesini döndür*

Örnek olarak, Şekil 3.23 b'deki 10-kenarlı poligon şeklinde, kodlar sol üst köşeden ($\boldsymbol{c_1}$) saat yönünün tersine tekrar başlangıç noktasına ($\boldsymbol{c_1}$) ulaşana kadar bir listeye peşi sıra eklenir. Böylece Tablo 3.2'de görülen 10 adet kenar vektörü için, 10 basamaklı kod 5661722043 olarak elde edilmiş olur. PoC'nin uzunluğu TeC ile doğrusal orantılıdır ve TeC, CcL boyutundan her zaman küçük seçilmelidir.

Tablo 3.2. Şekil 3.23 b'de görülen 10 kenarlı poli-silüet için kenar vektör kodları

Vektör ($\vec{v}_{lk}$)	KUTUPSAL AÇI (Φ)	PoC
$\vec{v}_{1\,2}$	(202,5° - 247,5°]	5
$\vec{v}_{2\,3}$	(247,5° - 292,5°]	6
$\vec{v}_{3\,4}$	(247,5° - 292,5°]	6
$\vec{v}_{4\,5}$	(22,5° - 67,5°]	1
$\vec{v}_{5\,6}$	(292,5° - 377,5°]	7
$\vec{v}_{6\,7}$	(67,5° - 112,5°]	2
$\vec{v}_{7\,8}$	(67,5° - 112,5°]	2
$\vec{v}_{8\,9}$	(-22,5° - 22,5°]	0
$\vec{v}_{9\,10}$	(157,5° - 202,5°]	4
$\vec{v}_{10\,1}$	(112,5° - 157,5°]	3

3.5. Öznitelik Vektörü Oluşturma

Çok sınıflı sınıflandırma yöntemlerinin başarılı sonuçlar üretmesi için, sistemleri eğitmekte kullanılan veri hacmine bağlı olarak süreç günler, haftalar ve hatta aylar alabilir. Karmaşık veriler içeren büyük veri kümelerinde çalıştırıldıklarında, sistem kaynaklarını tüketebilirler ve uzun çalışma süreleri gerektirebilirler. Aynı zamanda işlem maliyetleri yüksek olan pahalı donanımlar gerekmektedir.

Bu çalışmada önerilen öznitelik çıkarma ve öznitelik vektörü oluşturma yöntemi, karmaşık insan hareket sınıflandırması için kullanılabilecek bir yöntemdir. Yöntem gelişmelere açıktır. Hassasiyeti işlem hızına göre uyarlanabilir. Bu da yöntemin özgünlüğüne önemli bir katkıdır. Vücut ve uzuvları içeren hareketler, farklı kişiler

tarafından yapılsa bile büyük ölçüde benzerdir. Farklı kişilerin yürürken, koşarken, zıplarken vb. hareketleri vücut siluetleri dikkate alındığında benzerlikler taşır. Bir kolu kaldırdığınızda önemli olan kolun ne kadar kaldırıldığı değil, o hareketin duruşu için kolun kaldırılıp kaldırılmadığıdır. Ancak, farklı açılardan bakıldığında aynı hareket, vücudun siluetinde bir farklılık yaratır, bu gerçek, veri tabanlarının zenginliği ile telafi edilebilir. Aynı hareketin farklı videolarda farklı açılardan çekilmiş versiyonları olduğu için ve hepsi aynı sınıf etiketi altında işaretleneceğinden, sistemin poz bazlı sınıflandırmadaki başarısı azalmaz. Bu duruma bir örnek vermek gerekirsek, Şekil 3.22'deki farklı kenar sayısında poligonlaştırılmış siluetlerin hepsinde "kol kaldırmış insan" görüntüsü olduğu söylenebilir. Farklı zamanlarda video karelerinden elde edilen silüetler, Algoritma-1'de açıklandığı gibi, onları çevreleyen en küçük kutuya alınır ve daha sonra silüetlerin daha ayrık bir formu olan PoS görüntüleri oluşturmak için kütle merkezlerine göre toplanır. Bu görüntüler PoC verilerini oluşturmak için kullanılır.

Klasik sınıflandırıcılarla sınıflandırma için, öznitelik vektörleri tüm örnekler için aynı uzunlukta olmalıdır. Tez çalışmasında önerilen poligonlaştırma adımı, sınıflandırıcıların ihtiyaç duyduğu eşit uzunluktaki öznitelik vektörlerinin oluşturulmasında öne çıkan bir yenilik olup, poligon kodlaması, poligon şekli'nin özelliğini belirtmektedir. PoC verilerini, genetikte uygulanan genlerin k-mer gruplamasına [96], [97] benzer bir kurguda gruplandırırsak, uzun kod dizisinin normalizasyonunu gerçekleştirebiliriz. Aynı zamanda bu gruplandırma, sınıflandırma işlem süresini gereksinime göre ayarlamak amacıyla da kullanılabilir. Bu adımlar deneysel sonuç ve analiz bölümünde tablolarda gösterilmektedir.

Videodaki insan görüntüsü ve bu görüntüden oluşturulan silüet büyüklüğü, morfolojik dönüştürmede bir hata olmadıkça büyük ölçüde eşleşir. Bazı durumlarda Algoritma-1 ile işlenen görüntüler küçük insan figürleri içerdiğinde CcL boyutu belirlenen TeC değerinden büyük olmamaktadır. Eğer, Algoritma-2 için büyük bir TeC değeri seçilirse, Algoritma-1 tarafından oluşturulan görüntüler kullanılamayacaktır. İndirgemenin yapılabilmesi için TeC değerinin Algoritma-1 ile oluşturulan görüntülerden elde edilecek CcL sayısından her zaman küçük olması gerekmektedir. Ayrıca k-mer işlemi bir gruplama işlemidir ve k-değerleri gruplanacak eleman sayısını belirler. Örneğin, 100 farklı veri 5-mer'e gruplandırılırsa, 20 grup elde edilir; 4'lü gruplar halinde gruplandırılırsa, 25 grup elde edilir. Bu yaklaşım, çözüm sürecimizde 3-mer, 4-mer ve 5-mer gibi farklı değerler için denenmiş ve sonuçlar deneysel sonuçlarda listelenmiştir. K-mer derecesi arttıkça grup

sayısı ve öznitelik vektörünün boyutu azalmakta dolayısıyla sınıflandırmanın işlem hızı artmaktadır. Hatasız gruplama yapılabilmesi için TeC değeri k-mer'e tam bölünebiliyor olmalıdır. Bu yüzden tez çalışmasında TeC değeri 360 olarak seçilmiştir. Bu değer k-mer grupları için belirlediğimiz 3, 4, ve 5 sayısına tam bölünebilmektedir. Bunlara bağlı olarak oluşan farklı gruplamalara karşı farklı uzunluklu öznitelik vektörleri elde edilmiştir. Her grup ile elde edilen sonuçlar diğer gruplardan elde edilmiş sonuçlar ile sınıflandırmadaki hız ve başarı performansı konusunda birbirleri ile karşılaştırılmıştır. Bulunan sınıflandırma başarım sonuçları'da, literatüde benzer veri setlerini inceleyen ve farklı öznitelik çıkartma yöntemleri kullanan diğer çalışmaların başarımları ile kıyaslanmış ve bulunan sonuçlar bir sonraki bölümde tablolar halinde verilmiştir.

TeC'nin 360 değerine göre, öznitelik vektör boyutlarımız, 240, 180 ve 144 arasında değişebilir. Boyutun yarısı PoC'nin seçilen k-mer değerine göre gruplanması sonucu elde edilen grup sayısıdır, diğer yarısı da PoC'de yer alan her bir grubun, PoC değeri içerisinde bulunma sıklık frekansıdır. Sınıflandırma işleminden önce verilerin normalleştirilmesi sınıflandırma başarısını arttıracağı bilinmektedir. 3-mer gruplandırmasına göre kodlama sisteminden üretilebilecek en büyük 3 haneli sayı 777'dir. 4-mer, dört hane ve 5-mer için beş hane sırasıyla 7777 ve 77777'dir. Bu nedenle, karşılık gelen öznitelik vektöründeki tüm k-mer grupları, normalizasyon adımında 777 veya 7777 veya 77777'ye bölünür. Önerilen yöntemde, vektör boyutu tüm örnekler için aynıdır ve öznitelik vektörleri için ek bir boyut indirgeme algoritmasının yürütülmesine gerek yoktur. Video veri setinde işlenen her karenin tüm PoC'leri aynı boyuta sahip olduğundan, çıktı öznitelik vektörü herhangi bir sınıflandırma algoritması kullanılarak sınıflandırılabilir veya herhangi bir sinir ağı bu bilgi ile beslenebilir.

4. DENEYSEL SONUÇLAR VE ANALİZ

Tez çalışmasında önerilen öznitelik çıkartma yönteminin başarısını ispatlayabilmek adına, sadece kendi hazırladığımız Yoga veri seti üzerinde önerdiğimiz yöntemi uygulamak yerine daha zengin içeriği olan ve akademik çalışmada da sıkça kullanılan farklı sahnelerin ve hareket tiplerinin olduğu, değişik kamera açıları ve arka planların yer aldığı veri setleri üzerinde de önerilen yöntemimiz denenmiştir.

Büyük veri setleri üzerinde bu çalışmalar yapılırken, Algoritma-1'in son adımında elde edilen ve Yoga veri seti için derin ağ mimarilerinin eğitimi amacı ile kullanılabileceği önerilen PoS görüntüleri ile değil, yöntem sonucunda elde edilen öznitelik vektörü ve klasik sınıflandırıcılar tercih edilmiştir. Bu seçimi yapmamızı zorlayan ilk unsur mevcutta iyi bir donanıma sahip (grafik kartı, işlemci, RAM vb.) olmayışımız ve bu işlemlerin Colab gibi çevrim içi sistemler ile yapılamıyor olmasıdır. Fakat bu durum şöyle bir sonucu da beraberinde doğurmuştur. Klasik yöntemler daha ilkel bilgi ya da somut bilgi ile işlem yaptıklarından, aslında önerilen yöntemin yalın başarısının da değerlendirilebildiği, gözle görülür olduğu bir durum ortaya çıkmıştır.

Özetlemek gerekirse, diğer akademik çalışmalarında kullandığı büyük veri setleri üzerinde, çalışmada önerilen yöntem ile çıkartılan özniteliklerin klasik sınıflandırma algoritmaları ile sınıflandırılması, hazırlanan yoga veri seti üzerinde ise hem klasik algoritmaların, hem de derin ağ modelleri ve eğitim metotlarının uygulanarak sınıflandırılması gerçekleştirilmiştir.

4.1. Yoga Veri Seti Üzerinde Yapılan Uygulamalar ve Elde Edilen Sonuçlar

YouTube üzerinden çalışma için özgün olarak hazırlanan 8 sınıfa sahip yoga veri setinin, 3.bölümde anlatılan ve çalışmanın belkemiğini oluşturan öznitelik çıkartma yöntemlerinden geçirilmesi sonucu elde edilen öznitelik vektörleri ve bu aşamada üretilen renkli HGG görüntüleri ile ayrı ayrı uygulamalar yapılmıştır. Öznitelik vektörü tasarlanan bir MLP ile sınıflandırılırken, HGG görüntüleri de Vgg16 ve Vgg19 gibi derin ağ mimarilerinde kullanılarak sınıflandırma sonuçları karşılaştırılmıştır.Ayrıca çalışma için özelleştirilen Vgg19 ağında da sonuçlar alınmıştır.

4.1.1. Çok Katmanlı Algılayıcı Uygulaması ve Alınan Sonuçlar

Genel tasarım formu Şekil 2.14'de gösterilmiş olan çok katmanlı algılayıcı mimarisinin 4 katmanlı bir modeli tez çalışmasında metin tabanlı sınıflandırma yapmak amacı ile oluşturulmuştur. Bu modeli oluştururken başvurulan Keras kütüphanesi; neredeyse her tür derin öğrenme modelini tanımlamak ve eğitmek için uygun bir alt yapı sağlar. Keras, Tensorflow, Theano ve CNTK üzerinde çalışabilen Python ile yazılmış bir üst düzey sinir ağları API'sidir.

MLP'nin uygulaması yapılırken, öznitelik çıkartma algoritmamız ile elde ettiğimiz öznitelik vektörü text dosyalarında yer alan öznitelikler, kurulan model eğitiminde 50 epoch ve 32 batch boyu parametreleri ve stokastik gradyen azalmalı optimizer muadili olan Keras'ın Adam optimizer'ının ön tanımlı değerlerine dokunulmadan, Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False), çalıştırılmıştır. Tablo 4.1'de dört katmanlı algılayıcı modelimizin parametreleri yer almaktadır.

Tablo 4.1. Tasarlanan 4 katmanlı MLP modeli ve model parametreleri.

Layer (type)	Output Shape	Param#
dense84 (Dense)	(None, 256)	64512
dense_85 (Dense)	(None, 128)	32896
dense_86 (Dense)	(None, 32)	4128
dense_87 (Dense)	(None, 8)	264
Total params: 101,800		
Trainable params: 101,800		
Non-trainable params:0		

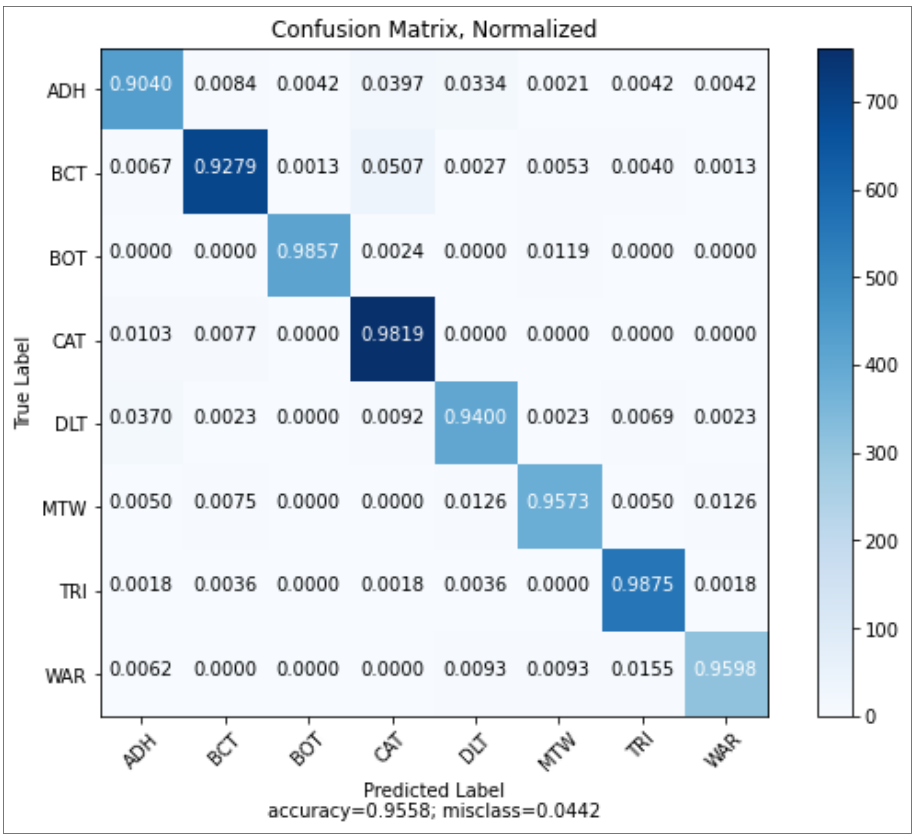
Eğitimde 50 itarasyonda model maksimum doğrulama değerine ulaşmasından dolayı eğitim bu aşamada otomatik durdurulmuştur. Önerdiğimiz yöntem ile oluşturulan öznitelik vektörlerinin her birinde 4139 adet instance olup, 3-mer 240 öznitelik, 4-mer 180 öznitelik, 5-mer 144 öznitelik barındırmaktadır. Bunların yarısı kenarların poligon kodları, diğer yarısı da k-mer gruplarının poligon kodu içerisinde bulunma sıklıklarıdır. Öznitelik vektörlerinin en son sütunu sınıf etiketini içerir. Veri seti, 80% eğitim, 20% doğrulama olarak bölünmüş ve buna göre aşağıdaki alınmıştır. Test, eğitim ve doğrulama setlerindeki örnekler birbirlerinden farklı videolardan oluşturulmuştur.

3-mer sonuçlarında, MLP modeli Tablo 4.2'de görüleceği üzere, yüzde 95,57 düzeyinde ortalama doğruluk ile sınıflandırma işlemini başarabilmiştir.

Tablo 4.2 Kmer gruplaması 3 olarak yapıldığında model sınıflandırma sonuçları.

	precision	recall	f1-score	support
0	0.93	0.90	0.92	479
1	0.98	0.93	0.95	749
2	0.99	0.99	0.99	420
3	0.92	0.98	0.95	775
4	0.94	0.94	0.94	433
5	0.96	0.96	0.96	398
6	0.97	0.99	0.98	562
7	0.97	0.96	0.96	323
accuracy			0.96	4139
macro avg	0.96	0.96	0.96	4139
weighted avg	0.96	0.96	0.96	4139

Average Accuracy: 0.9557864218410244



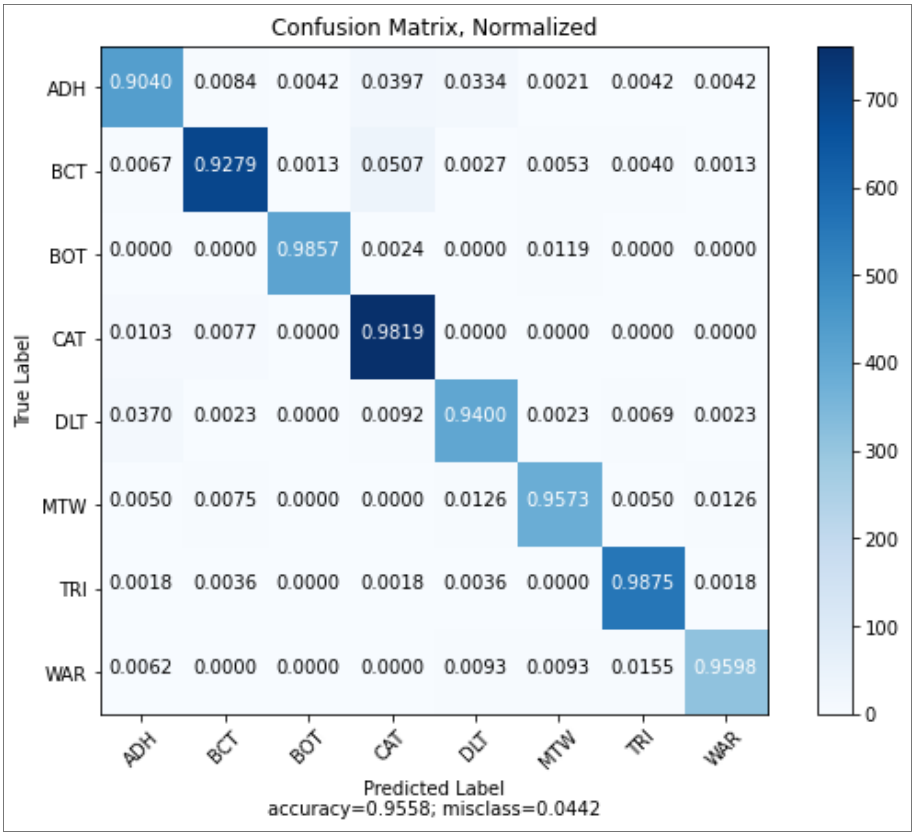
Şekil 4.1 MLP modeli - Kmer gruplaması 3

4-mer sonuçlarında, MLP modeli Tablo 4.2'de görüleceği üzere, yüzde 94,80 düzeyinde ortalama doğruluk ile sınıflandırma işlemini başarabilmiştir.

Tablo 4.3 Kmer gruplaması 4 olarak yapıldığında model sınıflandırma sonuçları.

	precision	recall	f1-score	support
0	0.90	0.93	0.91	479
1	0.95	0.94	0.94	749
2	0.97	0.99	0.98	420
3	0.93	0.97	0.95	775
4	0.95	0.90	0.92	433
5	0.95	0.92	0.93	398
6	1.00	0.96	0.98	562
7	0.96	0.97	0.96	323
accuracy			0.95	4139
macro avg	0.95	0.95	0.95	4139
weighted avg	0.95	0.95	0.95	4139

Average Accuracy: 0.9480550857695096



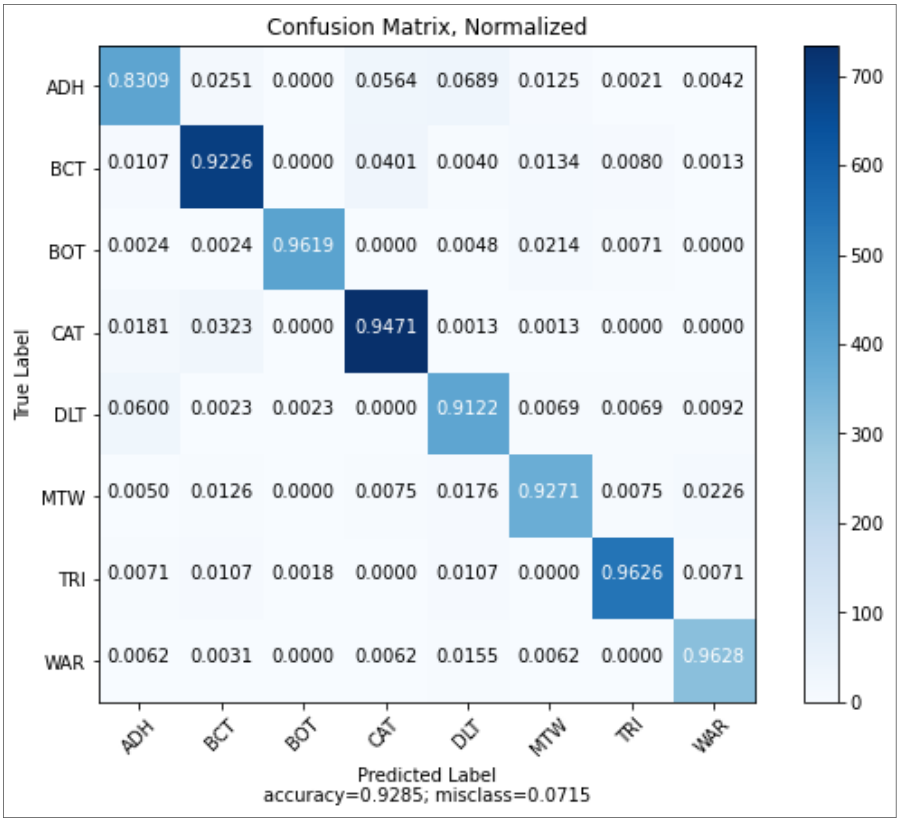
Şekil 4.2 MLP modeli - Kmer gruplaması 4

5-mer sonuçlarında, MLP modeli Tablo 4.2'de görüleceği üzere, yüzde 92,84 düzeyinde ortalama doğruluk ile sınıflandırma işlemini başarabilmiştir.

Tablo 4.4 Kmer gruplaması 5 olarak yapıldığında model sınıflandırma sonuçları.

	precision	recall	f1-score	support
0	0.87	0.83	0.85	479
1	0.93	0.92	0.93	749
2	1.00	0.96	0.98	420
3	0.92	0.95	0.93	775
4	0.87	0.91	0.89	433
5	0.92	0.93	0.92	398
6	0.97	0.96	0.97	562
7	0.94	0.96	0.95	323
accuracy			0.93	4139
macro avg	0.93	0.93	0.93	4139
weighted avg	0.93	0.93	0.93	4139

Average Accuracy: 0.9284851413384876



Şekil 4.3 MLP modeli - Kmer gruplaması 5

Buna göre; farklı gruplandırma için sınıflandırma başarısı sonuçları Tablo 4.2, Tablo 4.3 ve Tablo 4.4'de görülürken, modelin sınıflandırma karışıklık matrisleri, Şekil 4.1, Şekil 4.2 ve Şekil 4.3'de görülmektedir.

Beklendiği üzere öznitelik vektörünün boyu azalınca başarım yüzdesi de %95, %94 ve %92 şeklinde düşmüştür. Bu değerler birbirine çok yakın olsa da TeC'değeri 360'dan farklı bir büyüklüğe ayarlandığında ve gruplama boyları değiştirildiğinde fark da artabilir. Denemeler sonunda en yüksek başarım değerleri uygulama sonucunun alındığı değerler ile sağlanmıştır.

4.1.2. DVM ve k-NN Sınıflandırıcıları Uygulaması ve Alınan Sonuçlar

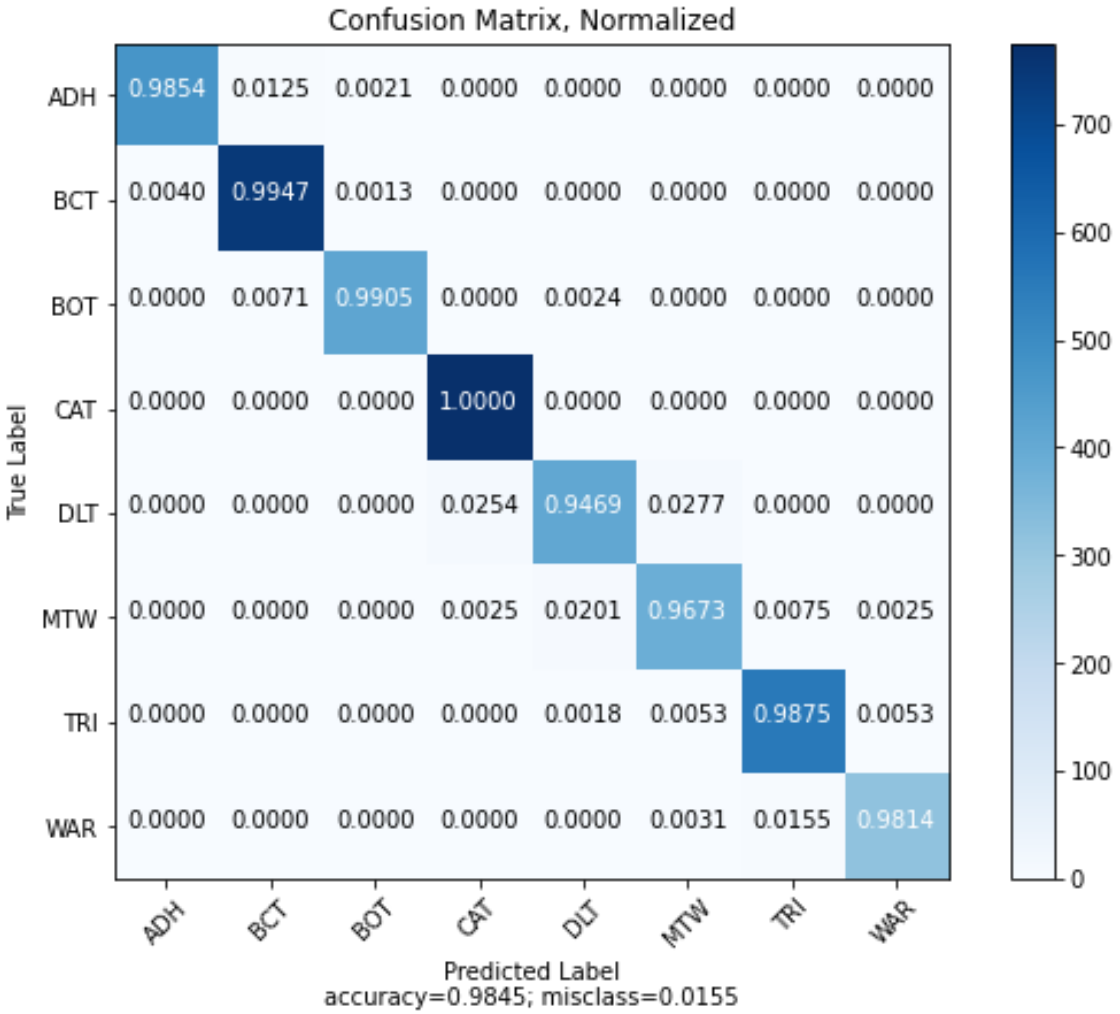
DVM ve k-NN farklı parametreler ile ayarlanarak yoga veri setinden elde edilen 3-mer, 4-mer ve 5-mer öznitelik vektörleri ile çalıştırılmış ve Tablo4.5 ve Tablo 4.6'da görülen sonuçlar alınmıştır.

Tablo 4.5'deki alınan sonuçlara bakıldığında, DVM sınıflandırıcısı en iyi performansı doğrusal çekirdekte vermiştir. Sınıflandırıcı ile 3'lü gruplandırılmış poligon kodlarından elde edilen öznitelik vektörü 5 katlamalı çapraz doğrulamaya tabi tutulduğunda sınıflandırıcının ortalama başarımı yüzde 98,45 olduğu görülmüştür. Toplamda 4139 örnek bulunan yoga sınıfında, sınıflar arası veri dağılımına bakıldığında örnek sayıları arasında farklılık olsa da sınıflandırma başarımı yüksektir. 8 sınıfın örnek dağılımı 479, 749, 420, 775, 433, 398, 562, 323 şeklindeki gibidir.

Tablo 4.6'da alınan sonuçlara bakıldığında, k-En yakın komşuluk sınıflandırıcısı en iyi performansı en yakın 3 komşuluk için vermiştir. Sınıflandırıcı ile 3'lü gruplandırılmış poligon kodlarından elde edilen öznitelik vektörü 5 katlamalı çapraz doğrulamaya tabi tutulduğunda sınıflandırıcının ortalama başarımı yüzde 93,81 olduğu görülmüştür. Toplamda 4139 örnek bulunan yoga sınıfında, sınıflar arası veri dağılımına bakıldığında örnek sayıları arasında farklılık olsa da sınıflandırma başarımı yüksektir. 8 sınıfın örnek dağılımı 479, 749, 420, 775, 433, 398, 562, 323 şeklindeki gibidir.

Tablo 4.5 Yöntem + DVM sınıflandırıcı (değişik çekirdek fonksiyonları ve k-mer'ler)

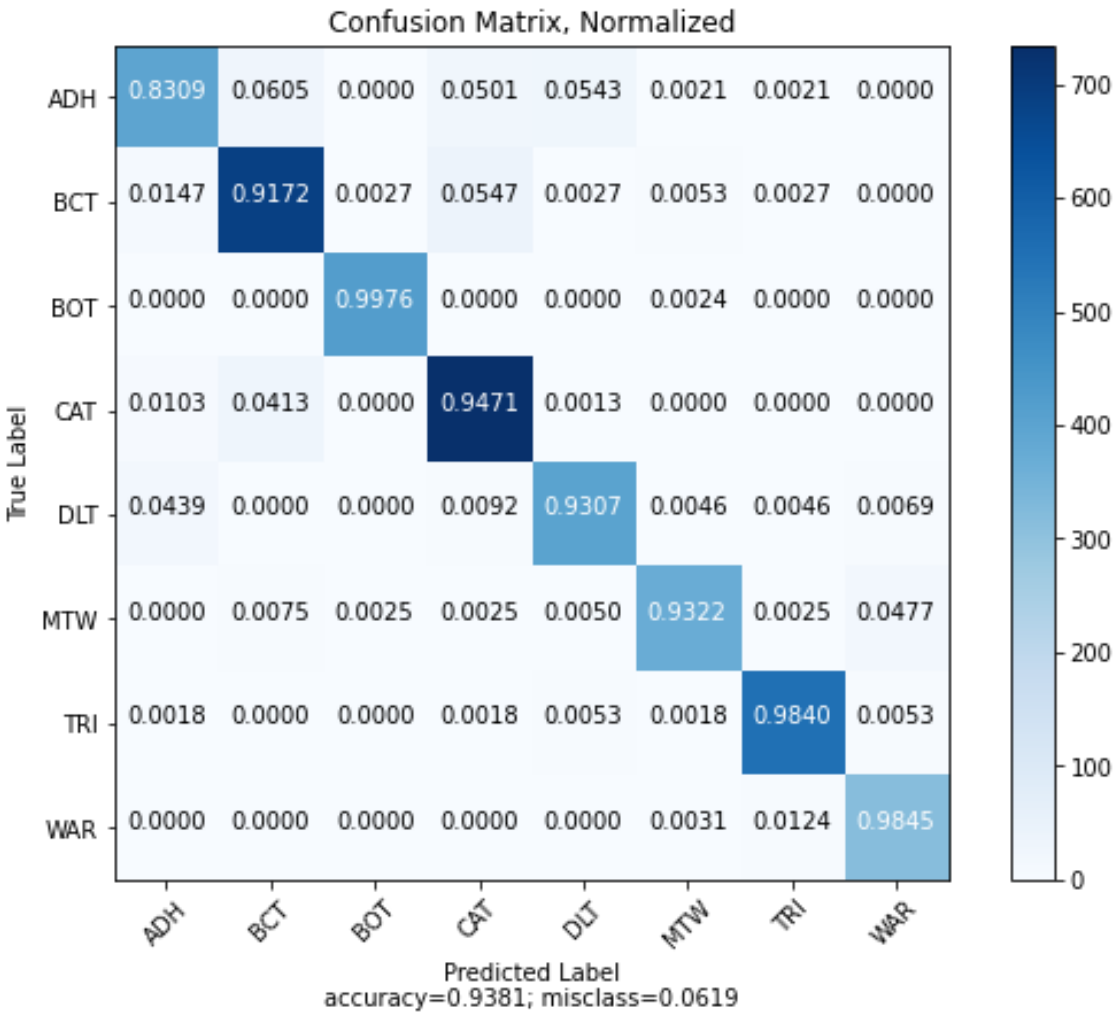
3-mer (poly= 96.69, **linear= 98.45**, rbf= 95.84)
4-mer (poly= 95.69, linear= 96.39, rbf= 94.61)
5-mer (poly= 94.91, linear= 95.58, rbf= 94.41)



Şekil 4.4 Gruplama 3-mer, Sınıflandırıcı DVM, Çekirdek Doğrusal

Tablo 4.6 Yöntem + k-NN sınıflandırıcı farklı Komşuluk (Nn) değerleri ve k-mer'ler

3-mer (**Nn3= 93.81**, Nn5= 92.13, Nn7= 91.13)
4-mer (Nn3= 91.87, Nn5= 91.60, Nn7= 91.10)
5-mer (Nn3= 91.63, Nn5= 91.54, Nn7= 91.05)



Şekil 4.5 Karşıtlık Matrisi, Gruplama 3-mer, Sınıflandırıcı k-NN, Komşuluk 3

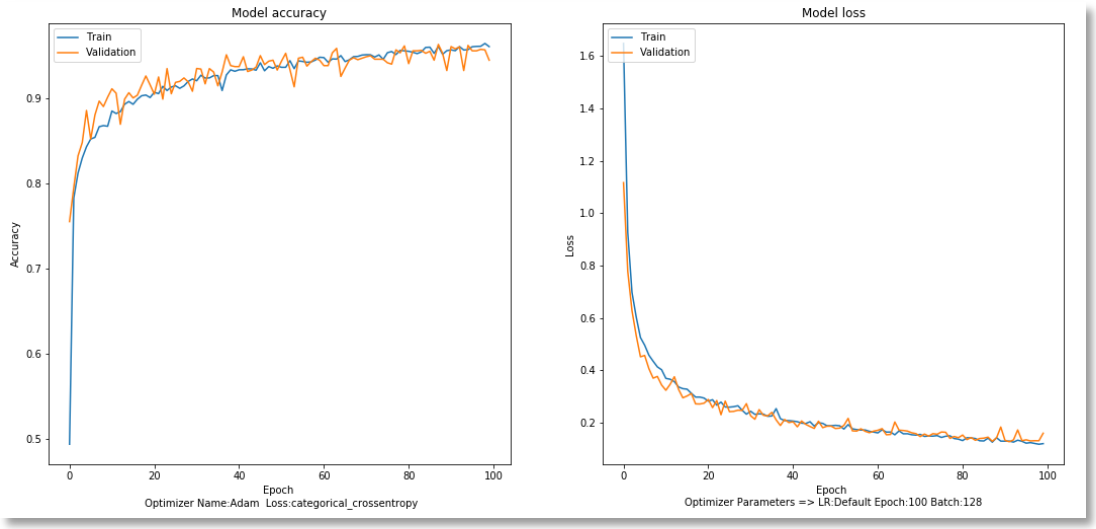
4.1.3. HGG görüntüleri ile Vgg16 Ağ Modeli Üzerinde Öğrenmenin Aktarımı Uygulaması ve Sonuçların Analizi

Çalışmanın bu aşamasında yoga veri setinin öznitelik vektörü oluşturulurken, ara adımda elde edilen renkli HGG görüntüleri kullanılarak derin ağ modellerinden biri olan Vgg16, öğrenmenin aktarımı yöntemi ile eğitilmiştir. Vgg16'nın Keras API'sinde bulunan kütüphanesi, ImageNet veri seti ile eğitilmiş bir pre-trained model ağırlık dosyası barındırır. Uygulamada bu kütüphaneyi kullanarak ve elde edilen Tip-2 HGG verileri çoğaltılarak Vgg16 modeli ile sınıflandırma gerçekleştirilmiştir. ImageNet veri seti renkli görüntülerden oluştuğu için eğitimde de bu görüntülere benzer olan renkli tip seçilmiştir. Eğitime başlanmadan önce hazır modelin son katmanı çıkartılarak, geri kalan tüm katmanları oluşturulan yeni sıralı modele eklenir. En son katman yoga setinin sınıf sayısı olan 8 olacak şekilde belirlenip eklenir. `len(CLASS_NAMES)` değeri model sınıf sayısı olan 8 değerini döndürmektedir. Son katman dışındaki tüm katmanların ağırlıkları bu eğitimden engellenmemesi ve değerlerinin değişmemesi için kilitlenir. Bu kilitlemenin amacı, modelin daha önce öğrendiği öznitelikleri kaybetmemesini sağlamaktır. Uygulama örneğine ait kod parçası alttaki gibidir.

```
from keras.applications import VGG16
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D

vgg16_model = VGG16(weights='imagenet', include_top=True, input_shape=IMAGE_DIMS)
vgg16_model.layers.pop()    # en son katmanı çıkarttık

model = Sequential()        # baslangıcta bos bir sıralı model yarattık
for layer in vgg16_model.layers:
   model.add(layer)        # vgg16 nın tum layerlarını kendı modelımıze ekledık

for  layer in  vgg16_model.layers:
   layer.trainable = False #eğitime kilitleyerek ağırlıklarının bozulmasını
engelledik

model.add(Dense(len(CLASS_NAMES), activation='softmax')) # son katman 8 sınıf
model.summary()

---------------------------
Total params: 134,293,320
Trainable params: 32,776 ,
Non-trainable params: 134,260,544
```

Modelin kurulumu ve ilklendirme aşamasında başlangıçta ne olması gerektiği bilinmeyen parametrelerine hiper-parametreler denir. Bu hiper parametrelerin seçimi veya optimizasyon algoritmasının seçimi tamamen sezgiseldir. Bunların olması gereken değerleri kesin olarak başlangıçta bilinememektedir. Bu çalışmada Tip-2 HGG

görüntülerinin sayısı 9000 civarında çıkmıştır. Veri çoğaltma tekniği ile bu sayı 10 katına çıkartılarak 90.000'e ulaşılmıştır. Veri sayısı çok fazla olmadığı için eğitiminde ezberleme olmaması adına 100 epoch'da tamamlanması ayarlanmış ve optimizasyon algoritması olarak da en iyi sonuç veren Adam optimizasyonu belirlenmiştir. Derin öğrenme uygulamalarında sıklıkla kullanılan optimizatörlerin hepsi tek tek denenmiştir. İçlerinde en iyi sonuç veren Adam optimizatörü olduğu tespit edilmiştir. Bu optimizatörler arasında başarım ve hız farklılıkları olabilir. Model eğitilirken RAM yetersiz olması durumunda tüm veriler aynı anda modele verilmez, bu durumda küçük bloklar seçileren model eğitilmeli ve geriye dönük yayılım birden fazla kez tekrarlatılarak modelin ağırlıklarının sınıflandırmayı yapabileceği en optimum değerleri alması sağlanmalıdır. Çalışmada bu blokların sayısı 128 olarak seçilmiştir. Bloklar ne kadar küçük olursa, model eğitim grafiğinde o kadar fazla zigzaglar görülecektir. Bunun anlamı eğitim için seçilen bloğun o anki değerleri modele tam uymamış olması durumunda bu sıçramalar görülür. Loss fonksiyonu tahmin edilen değer ile gerçek değer arasındaki uzaklığı hesaplar ve ardından optimizasyon ile model kendini günceller. Seçilen categorical crossentropy de iki olasılık dağılımı arasındaki mesafeyi ölçerek modelin optimizasyon yapmasını sağlamaktadır. Buna göre elde edilen model eğitim grafiği ve doğrulaması alt kısımda görülmektedir. Model sınıflandırma sonucunda %92.96 başarı elde etmiştir.



Şekil 4.6 Öğrenmenin aktarımının uygulandığı Vgg16 model eğitim sonuçları.

Öğrenmenin transferi yönteminin zorunlu olarak kullanılmasının nedeni, mevcut donanımın kullanılan bu derin ağ mimarisinin tüm ağırlıklarını değiştirebilecek kadar işlem gücüne sahip olmamasıdır.

Bu uygulamalar esnasında kullanılan modellerin bazıları sıralı katman yapısında inşa edilmişken, bazıları daha karmaşık modellenmiştir.

Fonksiyonel tanımlama gerektiren karmaşık modeller için uygulama ilklendirmeleri alt kısımda sergilenmiştir. Keras'da tanımlı karmaşık modellerden biri olan ResNet50 için öğrenmenin aktarımı yöntemini kullanmak amacı ile derin ağ modeli yapılandırması görülmektedir. Vgg16'da uygulanan adımlara ek olarak çıkış katmanının önüne ***global ortalama havuzlama katmanı*** yerleştirilmiş ve katmanlar fonksiyonel bir şekilde çağırım yapılarak tanımlanmıştır. Yine burada da çıkış sınıf sayımız kadar belirlenmiştir.

```
from keras.applications import ResNet50
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
base_model = ResNet50(include_top=True, weights='imagenet')
for layer in base_model.layers:
    layer.trainable = False
x = base_model.output
x = GlobalAveragePooling2D()(x)
predictions = Dense(len(CLASS_NAMES), activation='softmax')(x)
model = Model(input=base_model.input, output=predictions)
model.summary()
```

Keras'da tanımlı birçok eğitilmiş model ile denemeler yapılmıştır. Bu denemelerde 70 farklı hazır ve kurgulanan model üzerinde çalışılmıştır. Bu modelleri kullanmak isteyen diğer araştırmacılara yol göstermesi adına modellerin eklenme parametreleri aşağıdaki listede gösterilmiştir.

VGG16 224x224.

- `keras.applications.vgg16.VGG16(include_top=True,weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)`

VGG19 224x224.

- `keras.applications.vgg19.VGG19(include_top=True,weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)`

Xception 299x299.

- `keras.applications.xception.Xception(include_top=True, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)`

ResNet, ResNetV2 224x224.

- keras.applications.resnet.ResNet50(include_top=**True**,weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.resnet.ResNet101(include_top=**True**,weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.resnet.ResNet152(include_top=**True**,weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.resnet_v2.ResNet50V2(include_top=**True**,weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.resnet_v2.ResNet101V2(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.resnet_v2.ResNet152V2(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)

InceptionV3 299x299.

- keras.applications.inception_v3.InceptionV3(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)

InceptionResNetV2 299x299

- keras.applications.inception_resnet_v2.InceptionResNetV2(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)

MobileNet 224x224

- keras.applications.mobilenet.MobileNet(input_shape=**None**, alpha=1.0, depth_multiplier=1, dropout=1e-3, include_top=**True**, weights='imagenet', input_tensor=**None**, pooling=**None**, classes=1000)

MobileNetV2 224x224

- keras.applications.mobilenet_v2.MobileNetV2(input_shape=**None**, alpha=1.0, include_top=**True**, weights='imagenet', input_tensor=**None**, pooling=**None**, classes=1000)

DenseNet 224x224

- keras.applications.densenet.DenseNet121(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.densenet.DenseNet169(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)
- keras.applications.densenet.DenseNet201(include_top=**True**, weights='imagenet', input_tensor=**None**, input_shape=**None**, pooling=**None**, classes=1000)

NASNet NASNetLarge : 331x331 NASNetMobile : 224x224

- keras.applications.nasnet.NASNetLarge(input_shape=**None**, include_top=**True**, weights='imagenet', input_tensor=**None**, pooling=**None**, classes=1000)

- keras.applications.nasnet.NASNetMobile(input_shape=**None**, include_top=**True**, weights='imagenet', input_tensor=**None**, pooling=**None**, classes=1000)

Bu modellerde uygulamalar yapılırken hazırlanan verinin 20%'si doğrulama için 80%'i eğitim için kullanılmıştır. Her farklı model aynı veri seti üzerinde aynı eğitim ve test bölümleri ile denenmiş ve sonuçlar alınmıştır.

Evrişimli ağlarda girişe yakın ara katmanlar kenar, şekil, şeklin bölümleri vb. alt seviye özniteliklerin çıkartılmasında görevli iken, modellerin çıkışa yakın katmanları veri kümesinin daha spesifik üst özniteliklerini öğrenir. Bunda dolayı bu tür modellerin son bir kaç katmanı veya son katmanı çıkartılarak kendi sınıf sayımıza göre yoğunluk katmanı eklenerek softmax aktivasyonu ile modeller tekrar eğitilmiş ve içlerinden en iyisi olan Vggnet serisi seçilmiştir.

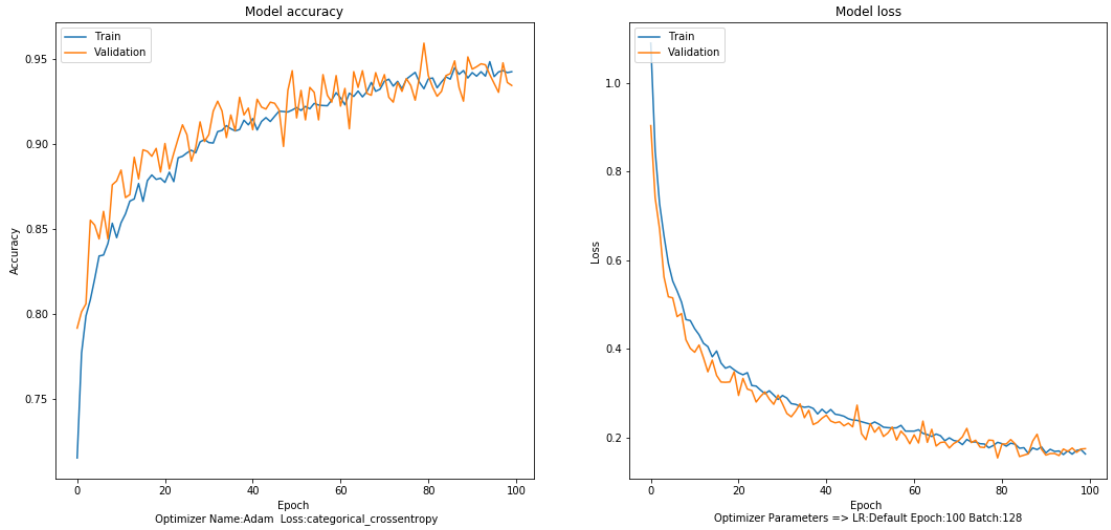
Ayrıca, öğrenmenin aktarımı yöntemi ile eğitilen 16 ve 19 katmanlı sıralı ağ modelleri olan Vgg16 ve Vgg19 ağları yakın başarım yüzdelerine erişirken 50 ve üzeri katmana sahip karmaşık ve daha derin ağları düzgün çalışabilecek kadar veri ile eğitemediğimizden alınan sonuçlar beklendiği gibi olmamıştır. Çünkü bu ağların parametre sayıları basit olan modellere göre çok daha fazladır.

4.1.4. HGG Görüntüleri İle Vgg19 Ağ Modeli Üzerinde Öğrenmenin Aktarımı Uygulaması ve Sonuçların Analizi

Vgg19 modeli'de öğrenmenin transferi yöntemi tekniğiyle aynı çoğaltılmış görüntüler kullanılarak eğitilmiş ve sınıflandırma performansı test edilmiştir. Vgg16'da yapılan gibi aşağıda görülen adımlarda Vgg19'un da son katmanına kadar olan katmanları dondurulup, son katmanı 8 sınıfı ayrıştıracak şekilde ayarlanmıştır. Şekil 4.7 model eğitim grafiği olup, alınan doğruluk başarısı, %91.40 seviyesinde çıkmıştır.

```
from keras.applications import VGG19
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D, Dropout

vgg19_model = keras.applications.vgg19.VGG19()
vgg19_model.layers.pop()
model = Sequential()
for layer in vgg19_model.layers:
    layer.trainable = False          # eğitime kilitleyerek
                                     # ağırlıklarının bozulmasını
                                     # engellenmiştir

    model.add(layer)          # vgg19'un tüm katmanlarını
                              # bos modele eklenmiştir

model.add(Dense(len(CLASS_NAMES), activation='softmax')) ))
                              # son katman 8 sınıf
model.summary()
```

```
-------------------------
Total params: 139,603,016
Trainable params: 32,776
Non-trainable params: 139,570,240
--------------------------------
```



Şekil 4.7 Öğrenmenin aktarımının uygulandığı orjinal Vgg19 model eğitim sonuçları.

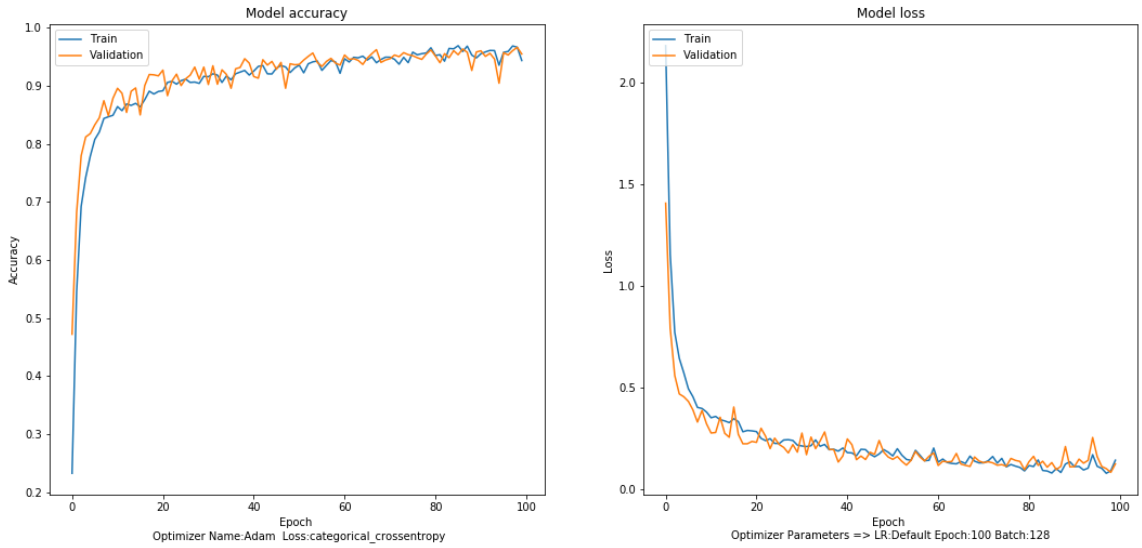
Vgg19, Vgg16 ağına göre daha derin bir model olduğu için öğrenmenin transferi yöntemi uygulansa dahi elde bulunan 90.000'lik veri seti bu derin modeli iyi eğitecek çoklukta olmadığı için sonuç %1.67 oranında Vgg16'ya göre daha düşük çıkmıştır.

4.1.5. HGG Görüntüleri ile Kodları Yeniden Şekillendirilmiş Vgg19 Ağ Modeli Üzerinde Öğrenmenin Aktarımı Uygulaması

Vgg19 mimarisine bir modifikasyon denenerek çıkış katmanından önce çalışmalarda görülen şişe boğazı (bottle neck) modeli ve %50'lik seyreltme katmanı yerleştirilmiştir. Bu modifikasyon başarım sonucunu %4.93 daha artırarak, %95,31'e yükseltmiştir. Yapılan eklemeler aşağıda kırmızı renk ile gösterilmiş satırlar gibidir. Model eğitim grafikleri Şekil 4.8'de görülmektedir.

```
from keras.applications import VGG19
from keras.models import Model
from keras.layers import Dense,GlobalAveragePooling2D,Dropout
```

```
vgg19_model = keras.applications.vgg19.VGG19()
vgg19_model.layers.pop()

model = Sequential()
for layer in vgg19_model.layers:
    layer.trainable = False
    model.add(layer)
model.add(Dense(2048, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(CLASS_NAMES), activation='softmax'))
model.summary()

-------------------------
Total params: 150,587,976
Trainable params: 11,017,736
Non-trainable params: 139,570,240
 ---------------------------------
```



Şekil 4.8 Öğrenmenin aktarımının uygulandığı yapısı yeniden şekillendirilmiş Vgg19 ağ modelinin eğitim sonuçları.

Vgg16, Vgg19'un orijinal hali ve yeniden şekillendirilmiş Vgg19 derin ağlarında yapılan denemeler sonucu Algoritma-1 sonunda elde edilen Tip-2 HGG görüntüleri bir derin ağın eğitiminde kullanılabilecek yeterlikte görüntüler olduğu görülmüştür. Harekete özgü farklılık gösteren bu görüntüler kullanılarak ve öğrenmenin transferi yöntemi ile derin ağ modellerinin eğitiminin gerçekleştirilebileceği söylenebilir.

4.2. Büyük Veri Setleri (HMDB51 ve UCF101) Üzerinde Yapılan Çalışmalar

Yapılan çalışmanın literatüre katkısının gerçek anlamda gözlemlenebileceği bu bölümde, önerilen yöntem diğer akademik çalışmalar ile karşılaştırılarak ve kendi içerisinde hız, performans testine tabi tutularak incelenmiş ve sonuçlar tablo ve grafiklerle verilmiştir.

4.2.1. Diğer Akademik Çalışmalar İle Yöntem Sonuçlarının Karşılaştırılması

Tablo 4.7 ve Tablo 4.8'teki bilgiler, kNN ve DVM sınıflandırıcıları için farklı çekirdek ve komşuluk uzaklığında bulunan sonuçları ve diğer çalışmalarda elde edilen sonuçları listelemektedir. UFC101 ve HMDB51 veri setleri ile elde edilen sonuçlar hem çalışma içinde hem de literatürde kullanılan diğer yöntemlerle karşılaştırılmıştır. Bu yöntemler arasında uçtan uca derin ağ mimarisine dayalı yöntemler veya geleneksel sınıflandırma yöntemleri kullanan çalışmalar vardır. Klasik sınıflandırıcılar ile yapılan çalışmalar esnasında sınıflandırıcıların parametreleri diğer akademik çalışmalar ile aynı tutulmuş ve hem DVM hem de k-NN sınıflandırma algoritmalarında diğer birçok çalışmada olduğu gibi 5 tekrarlı çapraz doğrulama (5-fold cross validation) uygulanmıştır.

Sonuçların bulunduğu Tablo 4.7 ve Tablo 4.8'de saf derin öğrenmeyi kullanan mevcut yöntemler (Derin) olarak işaretlenmiştir. Genel olarak, derin ağ mimarisini kullanan çalışmalar, klasik yöntemlere göre daha iyi sonuçlar verse de yeterli örnek olmadığında ya da sınıflandırılmak istenen veri, eğitim için kullanılmış veri ile çok fazla benzemediğinde derin modeller daha kötü sonuçlar üretebilir. Bu çalışmada önerilen öznitelik çıkarma yönteminin etkinliğini her alanda gösterebilmek için rakipler arasına derin öğrenme mimarileri kullanan çalışmalarda sokulmuştur.

Tablo 4.7. Farklı k-mer grupları, komşu değerleri ve çekirdek türleri için derin yöntemlere ve klasik (uzay-zamansal) özniteliklere göre HMDB51 veri kümesi için doğruluk sonuçları.

METOT	YIL	HMDB51 Acc. (%)
RBM + DVM [99]	2018	55.2
OFCM + DVM rbf kernel [100]	2016	56.9
(Deep) Inception3D [102]	2018	56.9
GBH + MBH + DVM [98]	2015	62.0
(Deep) ActionVLAD [103]	2017	66.9
(Deep) TS-LSTM [104]	2019	69.0
(Deep) TSM [105]	2020	72.7
ViCC-R-F+k-NN [111]	2022	76.2
Önerilen Yöntem + DVM sınıflandırıcı (değişik çekirdek fonksiyonları ve k-mer'ler)		
For 3-mer (**poly= 75.01, linear= 70.29, rbf= 67.90**)		
For 4-mer (poly= 74.47, linear= 69.82, rbf= 63.33)		
For 5-mer (poly= 73.96, linear= 69.94, rbf= 64.19)		
Önerilen Yöntem + k-NN sınıflandırıcı farklı Komşuluk (Nn) değerleri ve k-mer'ler		
For 3-mer (Nn3= **79.98**, Nn5= **77.50**, Nn7= **75.58**)		
For 4-mer (Nn3= 79.31, Nn5= 76.58, Nn7= 74.64)		
For 5-mer (Nn3= 78.27, Nn5= 75.90, Nn7= 74.35)		

Tablo 4.7 içerisinde, HMDB51 veri setini kullanarak ve önerilen yöntem ile elde edilen sonuçlarla beraber, diğer akademik çalışma sonuçları yer almaktadır. Tabloda yöntemimizin ve kullanılan sınıflandırıcı sekmesinin birden fazla satır değeri bulunmaktadır. Her bir satırda, farklı k-mer gruplama şekli ve ilgili sınıflandırma yönteminde kullanılabilen farklı çekirdek fonksiyonlarını ya da komşuluk ilişkilerini barındırmaktadır. Görülen tüm sonuçlar, sınıflandırma sonucunun başarısına yönelik sayısal değerlerdir. Diğer akademik çalışmalarda kullanılan donanımların konfigürasyon düzeneği, hep birbirleri ile farklılık gösterdiğinden dolayı ya da ilgili çalışmada bildirilmediğinden dolayı, tez çalışmasında bulunan ve alınan sonuçlarla hız-performans karşılaştırılmasının yapılabildiği durumlar, diğer akademik çalışmalar için verilememiştir.

Tablo 4.8. Farklı k-mer grupları, komşu değerler ve çekirdek türleri için derin yöntemlere ve klasik (uzay-zamansal) özniteliklere göre UCF101 veri seti için doğruluk sonuçları.

METOT	YEAR	UCF101 Acc. (%)
(Deep) PoseBase [107]	(2019)	60.9
Scene+Object+Pose + Nn=3 [108]	(2020)	62.1
(Deep) PoTion [106]	(2018)	65.2
(Deep) OPN [109]	(2017)	71.8
(Deep) VCOP [110]	(2019)	72.4
iDT w/BOW+DVM [112]	(2017)	76.2
ViCC-RGB + Nn=5 [111]	(2022)	77.1
Önerilen Yöntem + k-NN sınıflandırıcı farklı Komşuluk (Nn) değerleri ve k-mer'ler		
For 3-mer (Nn3= **82.02**, Nn5= **80.80** , Nn7= **79.98**)		
For 4-mer (Nn3= 81.44, Nn5= 80.34 , Nn7= 79.14)		
For 5-mer (Nn3= 80.44, Nn5= 79.36 , Nn7= 77.76)		
Önerilen Yöntem + DVM sınıflandırıcı (değişik çekirdek fonksiyonları ve k-mer'ler)		
For 3-mer (poly= **77.44**, linear= **59.78**, rbf= **73.04**)		
For 4-mer (poly= 77.18, linear= 59.60, rbf= 71.40)		
For 5-mer (poly= 75.26, linear=58.64, rbf= 69.54)		

Tablo 4.8 içerisinde, UCF101 veri setini kullanarak ve önerilen yöntem ile elde edilen sonuçlarla beraber, diğer akademik çalışma sonuçları yer almaktadır. Tabloda yöntemimizin ve kullanılan sınıflandırıcı sekmesinin birden fazla satır değeri bulunmaktadır. Her bir satırda, farklı k-mer gruplama şekli ve ilgili sınıflandırma yönteminde kullanılabilen farklı çekirdek fonksiyonlarını ya da komşuluk ilişkilerini barındırmaktadır. Görülen tüm sonuçlar, bsınıflandırma sonucunun başarısına yönelik sayısal değerlerdir. Diğer akademik çalışmalarda kullanılan donanımların konfigürasyon düzeneği, hep birbirleri ile farklılık gösterdiğinden dolayı ya da ilgili çalışmada bildirilmediğinden dolayı, tez çalışmasında bulunan ve kendi sonuçlarımız içerisinde hız-performans karşılaştırılmasını yapabildiğimiz durumlar, diğer çalışmalar için verilememiştir.

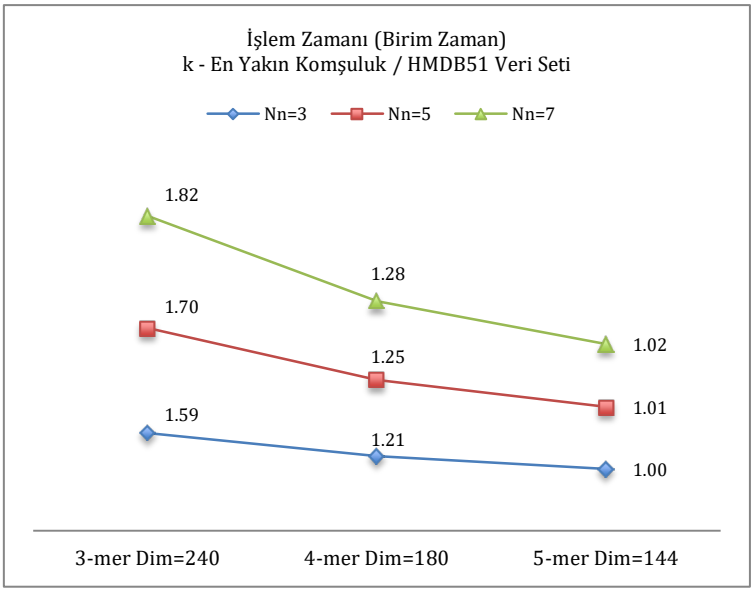
4.2.2. Yöntemin Kendi İçerisinde Hız-Performans Testleri ve Sonuçlarının Karşılaştırılması

Bu bölümde, yöntemin HMDB51 ve UCF101 veri setleri üzerinde farklı k-mer gruplamalarına göre hesaplama süreleri ve doğruluk değerleri grafikler halinde sunulup, karşılaştırmalı olarak yorumlanmıştır.

4.2.2.1. HMDB51 Veri Seti ve k-NN Sınıflandırıcı Sonuçları

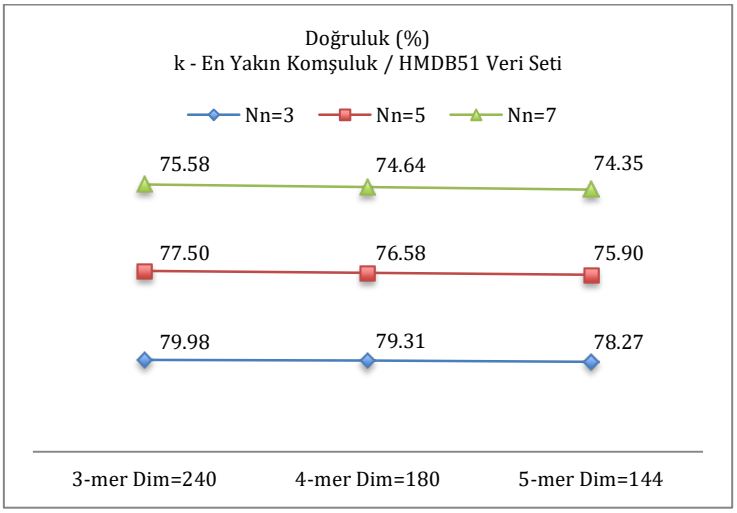
Şekil 4.9, HMDB51 veri seti üzerinde uygulanan öznitelik çıkartma yöntemimizin sonucunda elde edilen poligon kodlarının, farklı k-mer değerleri olan 3, 4 ve 5'e göre gruplandırıldığında, k-NN sınıflandırma algoritmasında, işlem hızı karşılaştırmasını göstermektedir. Bu grafik, makinenin donanımından bağımsız birim zaman cinsinde değerleri barındırmaktadır. İlgili grafik, yöntemin ile elde edilmiş sonuçların gruplandırmaya göre başarı grafiği olan Şekil 4.10 ile de bağıntılıdır.

Şekil 4.9'da gösterilen dokuz sonucun tümü için, beş katlı çapraz doğrulama arka arkaya yedi kez çalıştırılmış ve ortalaması alınmıştır. Ortalama değerler alınarak grafiğe yerleştirilmiş ve grafikteki değerler test sonuçlarında elde edilen en küçük zamana bölünerek donanımdan bağımsız olarak 1 birim zaman olarak ifade edilmiştir. Şekil 4.9'da görüldüğü gibi, k-mer değeri arttıkça, vektör boyutunun küçülmesi nedeniyle sınıflandırma hesaplaması için harcanan süre azalmaktadır. Aynı k-mer değeri için sınıflandırıcının 3, 5 ve 7 komşularının işlem sürelerinin ortalaması alınarak 3-mer için 1.703 birim zaman, 4-mer için 1.246 birim zaman ve 5-mer için 1.01 birim zaman elde edilmiştir.



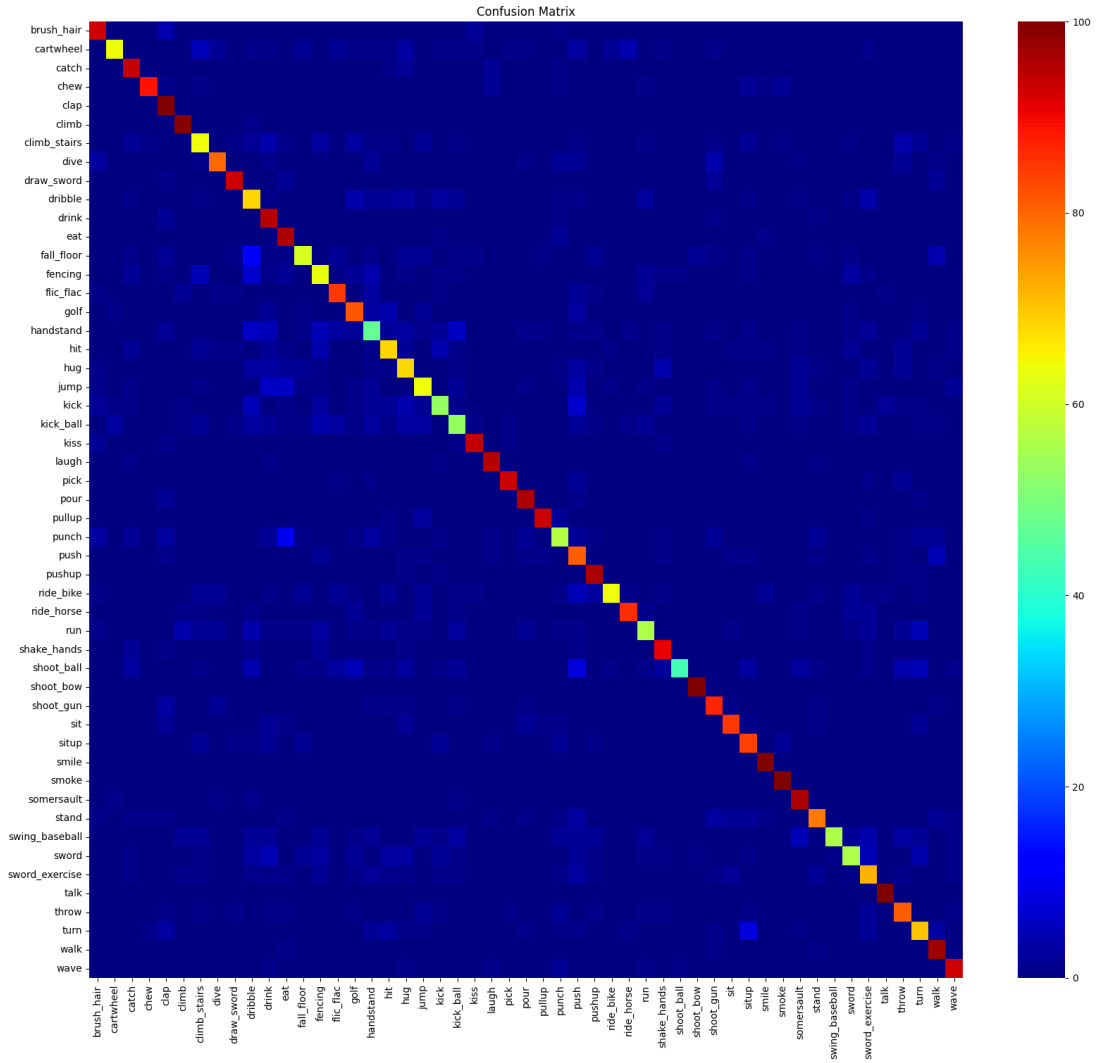
Şekil 4.9 HMDB51 veri seti için farklı k-mer grupları ve farklı k-en yakın komşu değerleri (Nn) ile k-NN sınıflandırma yönteminin işlem süresi birim zaman değerleri.

Bu ortalamalar, hız olarak yorumlandığında, 3-mer ve 4-mer grupları arasında %36,68, 3-mer ve 5-mer grupları arasında %68,61 ve 4-mer ve 5-mer grupları arasında %23,37 hesaplama süresi farkı olduğu göstermektedir. Daha düşük birim zaman, işlemin daha hızlı yapıldığı anlamına gelir.



Şekil 4.10 HMDB51 veri seti için farklı k-mer grupları ve farklı komşu (Nn) değerleri ile k-NN sınıflandırma yönteminin doğruluk yüzdesini göstermektedir.

Farklı komşu değerlere sahip k-NN sınıflandırıcısının performansı Şekil 4.10'da verilmiştir. 3-mer Nn=3 ile 5-mer Nn=7 arasında %7,04 doğruluk kaybı olsa da sınıflandırma sonucuna %68,61 daha hızlı ulaşılabildiği görülmektedir. En iyi sınıflandırma başarısı isteniyorsa Nn=3 olan 3-mer seçilmelidir. En hızlı sınıflandırma sonucu isteniyorsa Nn=3 olan 5-mer tercih edilebilir. HMDB51 veri seti doğruluğunun karışıklık matrisi, Şekil 4.11'de ısı haritası olarak gösterilmiştir. 51 sınıf üzerinden ortalama doğruluk, Nn=3 olan 3-mer için %79,98'dir. Bu sonuç, derin ağ mimarisi kullanan [101], [102], [103], [104] ve [105]'dan ve diğerlerinden [111]'den daha iyi doğruluk başarısına sahiptir.

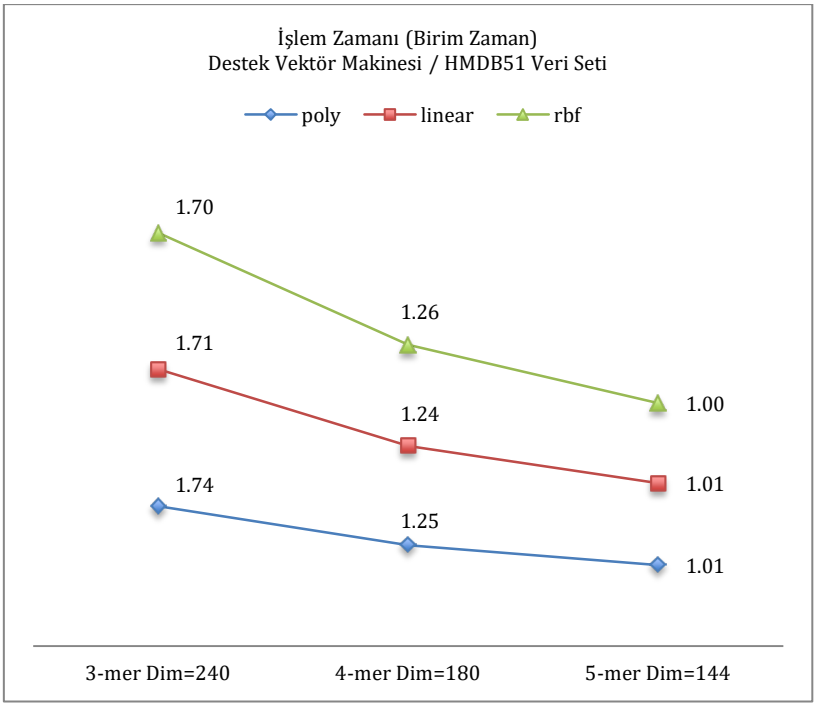


Şekil 4.11 Önerilen yöntem için karışıklık matrisi + HMDB51 veri kümesi için k-en yakın komşu değeri Nn=3 olan 3-mer gruplama için k-NN sınıflandırıcı. Ortalama doğruluk yaklaşık olarak %79,98'dir.

4.2.2.2. HMDB51 Veri Seti ve DVM Sınıflandırıcı Sonuçları

HMDB51 veri seti için k-mer değerleri 3, 4 ve 5'e göre farklı gruplanan öznitelik vektörü ile DVM sınıflandırma hızı sonuçları Şekil 4.12'deki grafiklerde gösterilmiştir. DVM sınıflandırıcısının farklı çekirdekler için farklı k-mer grupları arasındaki işlem süresini birim zamanda ifade etmek amacı ile oluşturulmuştur.

Grafikte gösterilen dokuz sonucun tümü için, beşli çapraz doğrulama arka arkaya yedi kez çalıştırılmış ve ortalaması alınmıştır. Ortalama değerler alınarak grafiğe yerleştirilmiş ve grafikteki değerler test sonuçlarında elde edilen en küçük zamana bölünerek donanımdan bağımsız olarak birim zaman cinsinden ifade edilmiştir. Şekil 4.12'de görüldüğü gibi, k-mer değeri arttıkça vektör boyutunun küçülmesi nedeniyle, sınıflandırma hesaplaması için harcanan süre azalmaktadır.

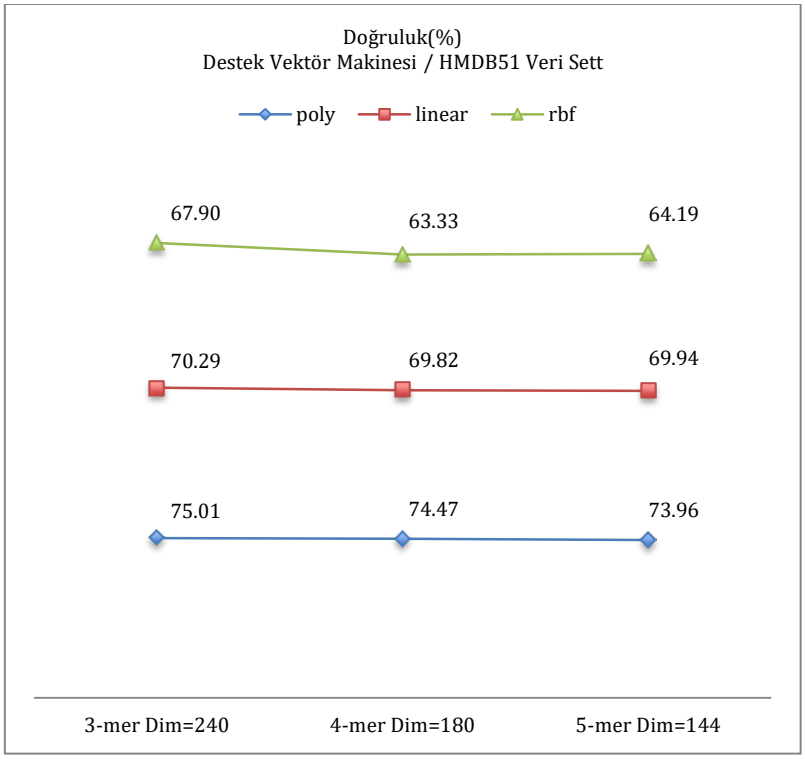


Şekil 4.12 HMDB51 veri seti için farklı k-mer grupları ve farklı çekirdekler ile DVM sınıflandırma yönteminin işlem süresi birim zaman değerleri.

Aynı k-mer değeri için, sınıflandırıcının farklı çekirdeklerdeki işlem sürelerinin ortalaması 3-mer için 1.716, 4-mer için 1.250 ve 5-mer için 1.006 zaman birimi verir. Bu, gruplar arasında ortalama hız olarak 3-mer ile 4-mer arasında %37,28, 3-mer ile 5-mer

arasında %70,58 ve 4-mer ile 5-mer arasında %24,25'lik bir zaman farkı verir. Daha düşük birim zaman, işlemin daha hızlı yapıldığı anlamına gelir.

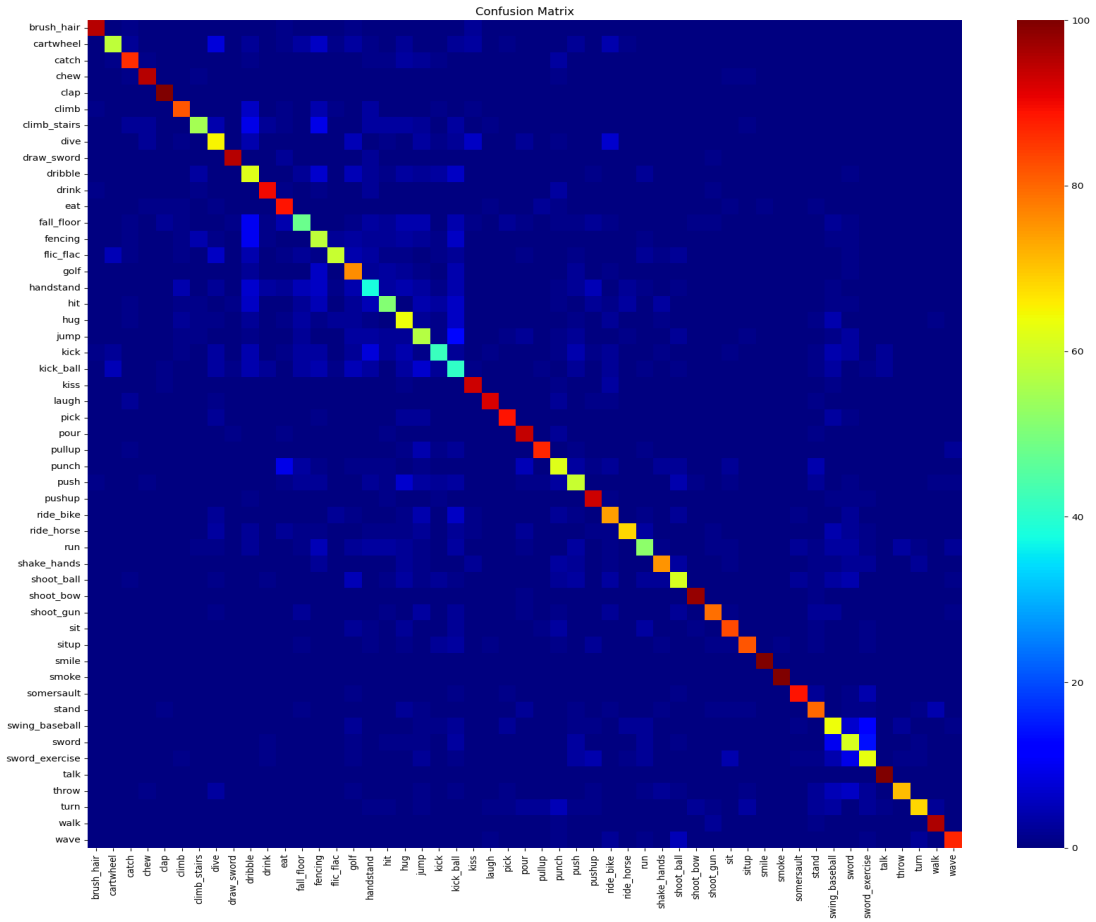
DVM sınıflandırıcının farklı çekirdeklere sahip performansları Şekil 4.13'de verilmiştir. Şekil 4.12'de görüldüğü gibi tüm çekirdeklerde sınıflandırma sonucuna %70.58 daha hızlı ulaşılabilmektedir. Poly kernel ile 3-mer ve 5.mer arasında doğrulukta %1,40'lık bir kayıp kabul edilebilir.



Şekil 4.13 HMDB51 veri seti için farklı k-mer grupları ve farklı çekirdeklerle DVM sınıflandırma yönteminin doğruluğu.

En iyi sınıflandırma sonucu isteniyorsa 3-mer gruplandırmalı poly kernel, en hızlı sınıflandırma sonucu isteniyorsa 5-mer gruplandırmalı poly kernel tercih edilebilir. Bu seçim %70.58'lik hız artışına karşı, doğrulukdan %1.40 kadarlık bir kaybı kabüllenmeyi gerektirir.

HMDB51 veri seti doğruluğunun karışıklık matrisi, Şekil 4.14'da ısı haritası olarak gösterilmektedir. 51 sınıf üzerinden ortalama doğruluk, 3-mer ve poli çekirdekli DVM için en yüksek %75,01'dir. Bu sonuç, derin sürümlerden [102], [103], [104], [105] ve diğerlerinden [98], [99], [100] daha iyi bir yüzdeye sahiptir.



Şekil 4.14 HMDB51 veri seti üzerinde önerilen yöntem ile elde edilen karışıklık matrisi. DVM poly çekirdek + 3-mer gruplama. Ortalama doğruluk başarısı %75,01'dir.

4.2.2.3. UCF101 Veri Seti ve k-NN Sınıflandırıcı Sonuçları

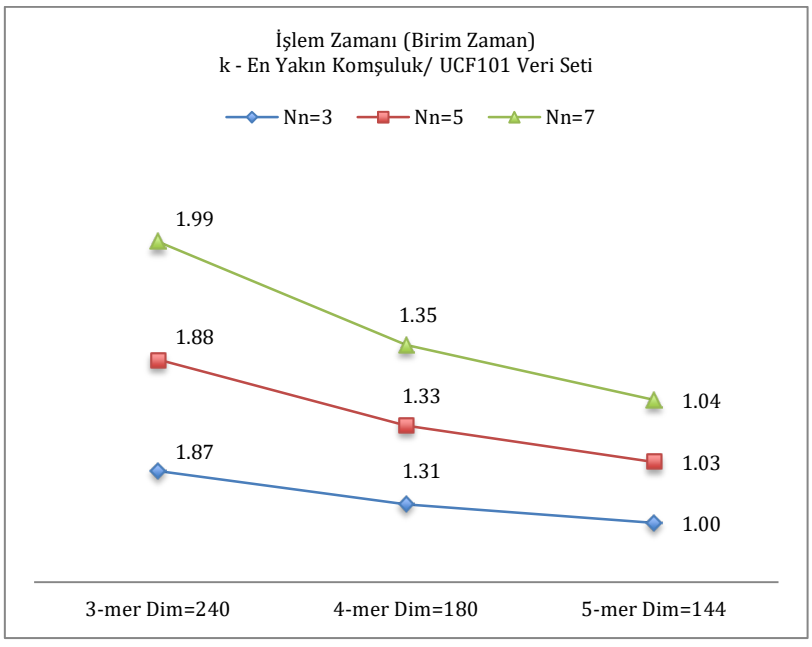
Şekil 4.15, UCF101 veri seti üzerinde uygulanan öznitelik çıkartma yöntemimizin sonucunda elde edilen poligon kodlarının, farklı k-mer değerleri olan 3, 4 ve 5'e göre gruplandırıldığında, k-NN sınıflandırma algoritmasında, işlem hızı karşılaştırmasını göstermektedir. Bu grafik, makinenin donanımından bağımsız birim zaman cinsinde

değerleri barındırmaktadır. İlgili grafik, yöntemimiz ile elde edilmiş sonuçların gruplandırmaya göre başarı grafiği olan Şekil 4.16 ile de bağıntılıdır.

Grafikte gösterilen tüm sonuçlar için, beşli çapraz doğrulama arka arkaya yedi kez çalıştırılmış ve ortalaması alınmıştır. Ortalama değerler alınarak grafiğe yerleştirilmiş ve grafikteki değerler test sonuçlarında elde edilen en küçük zamana bölünerek donanımdan bağımsız olarak birim zaman cinsinden ifade edilmiştir.

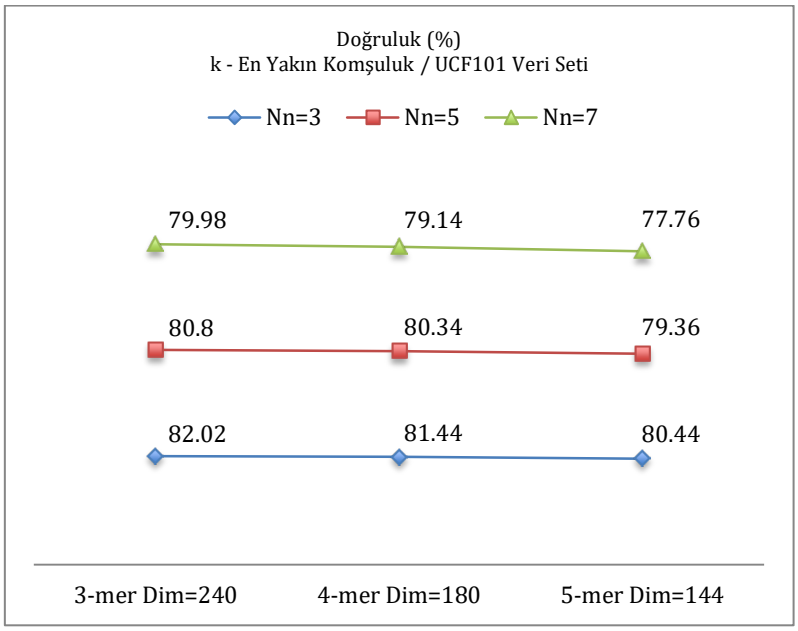
Şekil 4.15'de görüldüğü gibi, k-mer değeri arttıkça, vektör boyutunun küçülmesi nedeniyle sınıflandırma hesaplaması için harcanan süre azalmaktadır. Aynı k-mer değeri için sınıflandırıcının 3, 5 ve 7 komşularının işlem sürelerinin ortalaması alınarak 3-mer için 1.913 birim, 4-mer için 1.330 birim ve 5-mer için 1.023 birim zaman elde edilmiştir.

Bu ortalamalar hız olarak yorumlandığında, 3-mer ve 4-mer grupları arasında %43,83, 3-mer ve 5-mer grupları arasında %87,00 ve 4-mer ve 5-mer grupları arasında %30,01 hesaplama süresi farkı olduğunu göstermektedir. Daha düşük birim zaman, işlemin daha hızlı yapıldığı anlamına gelir.



Şekil 4.15. UCF101 veri seti için farklı k-mer grupları ve farklı en yakın komşuluk (Nn) değerleri ile k-NN sınıflandırma yönteminin işlem süresi birim zamanı değerleri.

Aynı sınıflandırıcının doğruluk performans yüzdeleri Şekil 4.16'de gösterilmektedir. Görüldüğü gibi k-NN sınıflandırma algoritması, komşuluk değeri Nn=3'den 5'e değiştirildiğinde, işlem sonucuna %1,96 doğruluk kaybı ile %87,00 daha hızlı ulaşılabilmektedir. En iyi sınıflandırma başarısı isteniyorsa, Nn=3 ve 3-mer gruplaması seçilmelidir. En hızlı hesaplama hızı isteniyorsa Nn=3 ve 5-mer gruplaması seçilmelidir.

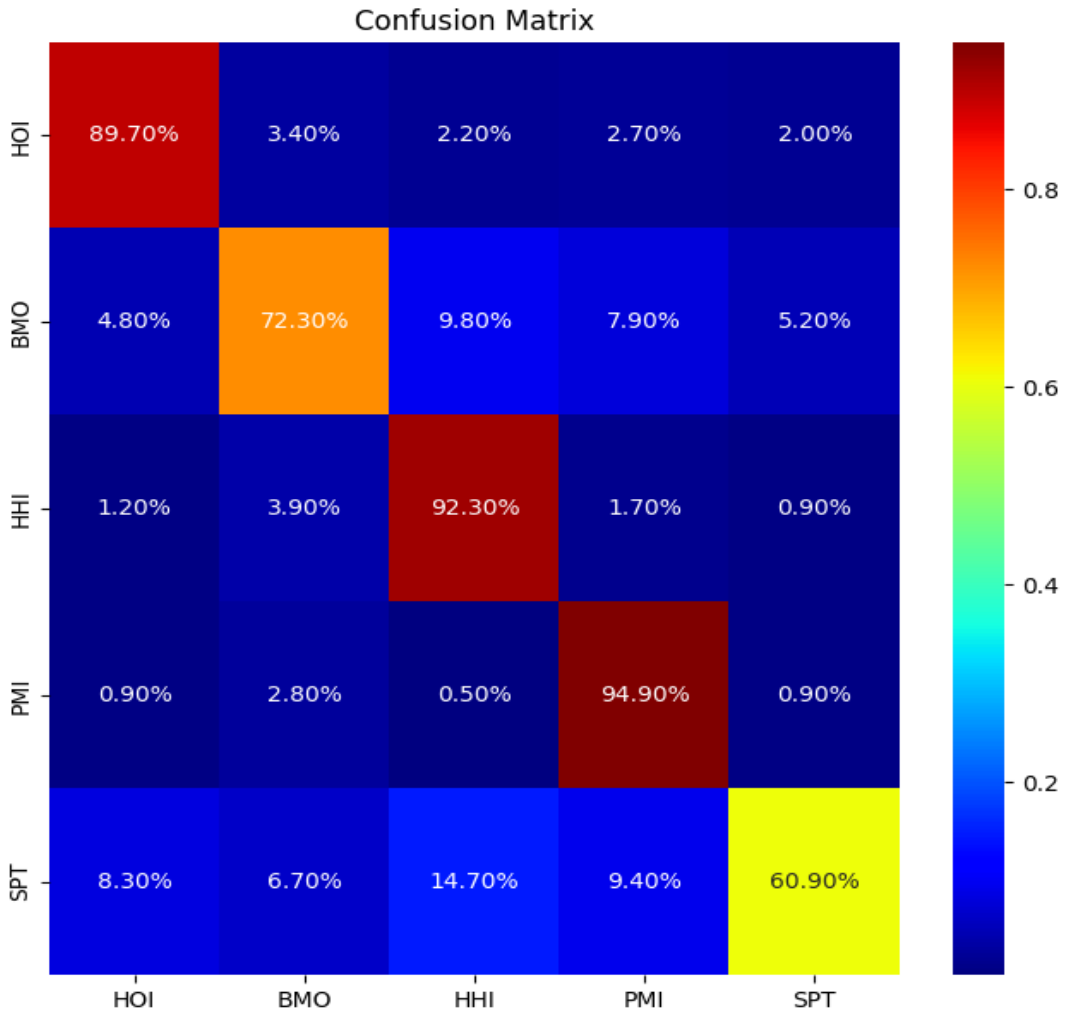


Şekil 4.16 UCF101 veri seti için farklı k-mer grupları ve farklı komşu (Nn) değerleri ile k-NN sınıflandırma yönteminin doğruluk yüzdesini göstermektedir

UCF101 veri seti doğruluğunun karşıtlık matrisi Şekil 4.17'da ısı haritası olarak gösterilmiştir. Veri seti, literatürdeki diğer çalışmalarda olduğu gibi 5 sınıfa ayrılmıştır. Bunlar;

- İnsan Nesnesi – Etkileşim (HOI)
- Gövde - Yalnızca Hareket (BMO)
- İnsan - İnsan Etkileşimi (HHI)
- Müzik Aletleri Çalma (PMI)
- Spor (SPT).

Nn=3 ile 3-mer ve k-NN sınıflandırıcı için 5 sınıf üzerinden ortalama doğruluk %82.02'dir. Sonuç, [106], [107], [109], [110] ve diğer derin sürümlerinden [108],[111] daha başarılıdır.



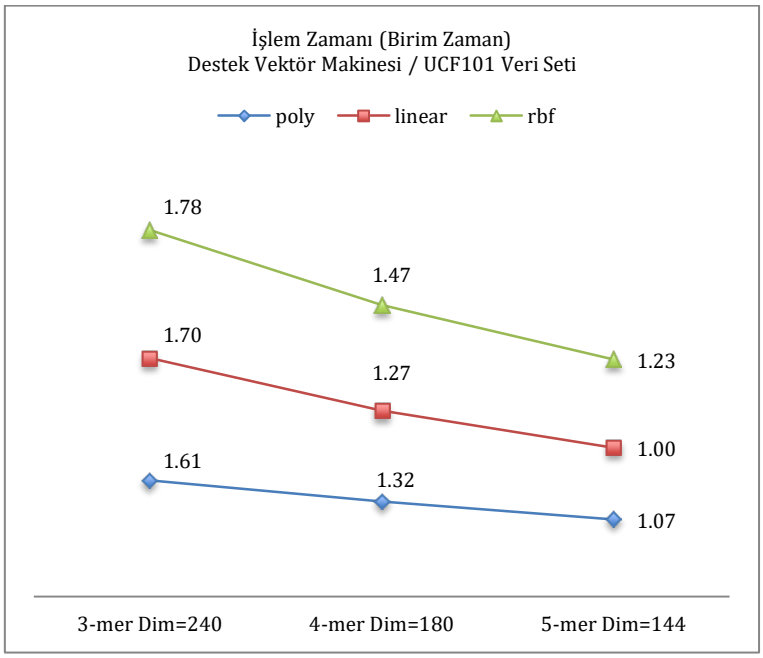
Şekil 4.17 Önerilen yöntem için UCF101 veri seti üzerinde k-NN sınıflandırma algoritmasının komşuluk değeri (Nn=3) ve gruplamanın 3-mer olarak yapıldığı karışıklık matrisi. Ortalama doğruluk %82,02'dir.

4.2.2.4. UCF101 Veri Seti ve DVM Sınıflandırıcı Sonuçları

Şekil 4.18, UCF101 veri setindeki 5 sınıftan çıkarılan ve k-mer değerleri 3, 4 ve 5'e göre gruplanan öznitelik vektörleri kullanılarak oluşturulmuştur. DVM sınıflandırıcısının işlem süresini farklı çekirdekler için ve birim zamanda farklı k-mer grupları arasında yüzde olarak ifade etmek amacı ile bu grafik sunulmuştur. Grafikte gösterilen dokuz sonucun tümü için aynı test arka arkaya yedi kez çalıştırılmış ve ortalaması alınmıştır. Ortalama değerler alınarak grafiğe yerleştirilmiştir ve grafikteki değerler test sonuçlarında elde edilen en küçük zamana bölünerek donanımdan bağımsız olarak birim zaman cinsinden ifade edilmiştir.

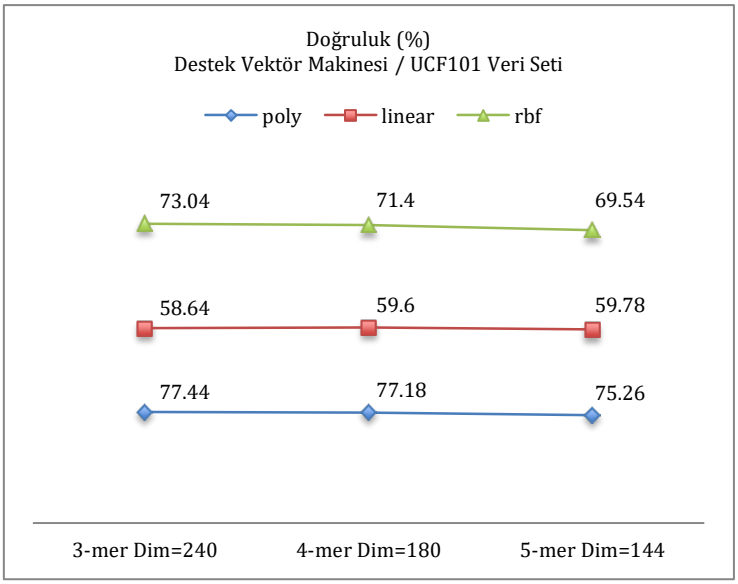
Şekil 4.18'de görüldüğü gibi, k-mer değeri arttıkça, vektör boyutunun küçülmesi nedeniyle, sınıflandırma hesaplaması için harcanan süre azalmaktadır. Aynı k-mer değeri için, sınıflandırıcının 3, 5 ve 7 komşularının işlem sürelerinin ortalaması alınarak 3-mer için 1,70 birim zaman, 4-mer için 1,35 birim zaman ve 5-mer için 1,10 birim zaman elde edilmiştir.

Bu ortalama hız olarak 3-mer ve 4-mer grupları arasında %25,93, 3-mer ve 5-mer grupları arasında %54,55 ve 4-mer ve 5-mer grupları arasında %22,73 hesaplama süresi farkı olduğunu göstermektedir. Daha düşük bir birim zamanın daha iyi olduğuna dikkat edilmelidir.



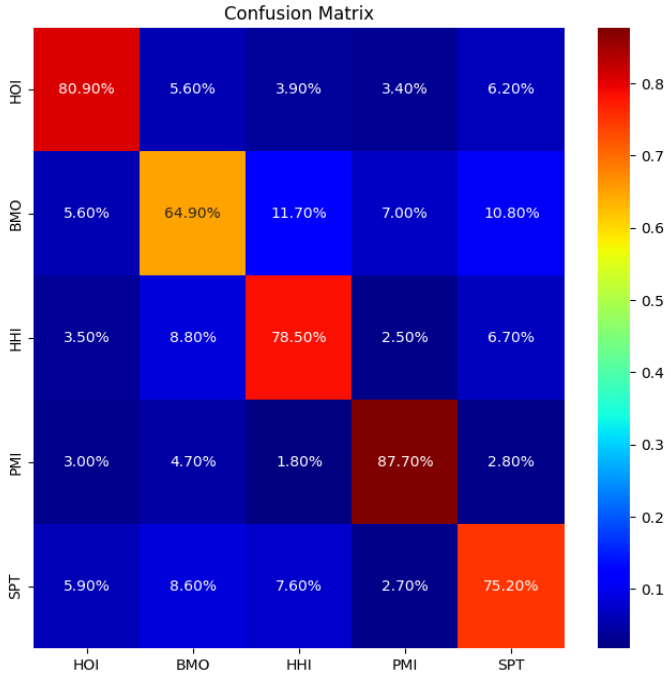
Şekil 4.18 UCF101 veri seti için farklı k-mer grupları ve farklı çekirdekler ile DVM sınıflandırma yönteminin işlem süresi birim zaman değerleri.

Aynı sınıflandırıcının doğruluk yüzdeleri Şekil 4.19'da gösterilmiştir. 3-mer ve poly kernel değeri ile sınıflandırma sonucuna %2,9 performans kaybı ile %50,47 daha hızlı ulaşılabildiği görülmektedir. En iyi ve en hızlı sınıflandırma isteniyorsa poly kernel ve 3-mer gruplama seçilebilir.



Şekil 4.19 UCF101 veri seti için farklı k-mer grupları ve farklı çekirdeklerle DVM sınıflandırma algoritmasının, doğru sınıflandırma başarı yüzdeleri.

UCF101 veri seti üzerinde DVM'nin sınıflandırma doğruluğunun karışıklık matrisi, Şekil 4.20'de ısı haritası olarak gösterilmiştir. Poli çekirdekli DVM sınıflandırıcı için ortalama doğruluk %77'dir. Bu sonuç, derin versiyonlardan [106], [107], [109], [110] ve diğerlerinden [108], [111] daha yüksektir.



Şekil 4.20 Önerilen yöntem için UCF101 veri seti üzerinde, poly çekirdekli DVM sınıflandırıcı sonuçları karışıklık matrisi. Ortalama doğruluk %77,44'dür.

5. SONUÇ VE TARTIŞMA

Bu tez çalışmasında, insan eylemlerini sınıflandırmak için kullanılabilecek yeni bir siluet tabanlı öznitelik çıkarma yöntemi önerilmiştir. Önerilen öznitelik çıkartma ve öznitelik vektörü oluşturma yöntemi, diğer siluet öznitelikleri veya yöntemlerle birleştirilmeden yalın olarak kullanılmış ve performansı farklı sınıflandırıcılar ve derin modellerle sergilenmiştir. Aynı veri setlerini, sınıflandırma algoritmalarını ve benzer parametreleri kullanan diğer çalışmalarla, sınıflandırma sonucunun doğruluğu üzerinden yapılan karşılaştırmalarda, başarılı sonuçlar elde edilmiştir. Buna göre Bölüm 4'de detaylı ele alınan sonuçlar özet tablolar olan Tablo 5.1 ve Tablo 5.2 halinde alt kısımda verilmiştir.

Çalışmanın bilime katkıları kısaca maddeler halinde özetlenecek olursa;

- Videolardan elde edilen siluetlerin ekstra yöntemler kullanmadan, gövde bütünlüğü korunan ve vücut uzuvlarının eksiksiz elde edilmesini sağlayan yenilikçi yaklaşımımız, arka planda modern derin ağ mimarilerini kullanarak kusursuz silüet görüntüleri elde edebilmektedir. Bu da beraberinde çalışma sonuçlarında elde edilen doğruluk yüzdesini arttırmaktadır.

- Elde edilen silüet görüntüleri ile hareketi daha ayrık tanımlayabilecek HGG görüntüleri ve bunların özel tiplerini oluşturmamız, Tip-1 modelinin hem klasik sınıflandırıcılar ile kullanımını, Tip-2 modelinin de modern derin ağ yapılarının eğitiminde kullanımının olası olduğunu ispatlamamızı sağlamıştır. Derin ağ uygulamalarımızın sonuçlarında %95'lere kadar çıkan doğruluk başarısı, bu özel görüntülerin, hareketi ne denli ayrık ifade edilebileceği sonucunu bizlere göstermiştir. Bu da literatüre yeni bir "anahtar görüntü" normunu katmıştır.

- Silüetlerin poligonlaştırılması aşamasında önerdiğimiz algoritma, büyük veri setlerinde çalışırken bile performansdan düşmemiştir. Çalışmada işlenen veri setlerinin hepsinde toplamda 40 saate yakın görüntü mevcuttur. Bu videoların her birinin her bir saniyesi ve her saniyedeki kare sayısı düşünülürse milyonlarca görüntüden bahsedildiği ve bu kadar çok görüntüdeki her bir insan figürünün poligonlaştırılmasında algoritmanın kusursuz çalışması ve silüetle ilgili çalışmalarda literatürde böyle bir yaklaşımın olmayışı, sağlanan katkılardan bir başkası olarak düşünülebilir. Algoritma sadece insan silueti üzerinde değil, herhangi bir görüntüden elde edilecek kontur koordinatları bilgisi üzerinde çalışabilecek evrensel bir yapıya sahiptir. Kodlanması ve fonksiyonel planlanması her

tipteki derin ağ mimarisi veya modeline kolay adapta edilebilir yapıda olmasını sağlamıştır.

- Çalışmada önerilen poligon tabanlı öznitelik, çalışma ile beraber öne sürülen yeni bir yöntemdir. Poligonun kodlanması ve özniteliğinin çıkartılması, çıkarılan bu bilgi ile de sınıflandırma yapılabileceğinin gösterilmesi bilime sunulan katkılar arasında yer almaktadır. Bu öznitelikler ile kurulan özellik vektörleri düşünüldüğünde, bu vektörler üzerinde ek bir boyut indirgeme algoritmasının kullanılmasına gerek olunmayışı, bilime olan katkılar arasında en önemli olandır.

- Poligon kodlarının içlerinde yön bilgisinin bulunması, kodun k-mer uzunluğunda parçalara ayrılması ve bu parçaların birbiri peşi sıra kod içerisinde bulunma sıklıkları ile oluşan özellik vektörü aslında şekil içerisinde şekil arama işlemini metin içerisinde metin arama işlemine evrilmesini sağlamıştır. Bu da beraberinde hızı getirmiştir.

- Farklı k-mer boylarının kullanılması yöntemin adaptif çekirdek boyuna göre uyarlanabilir performansının oluşmasını sağlamış, bu da hız ve performans teslerini yapılabilir kılmıştır.

- K-mer gruplama mantığının çalışmada kullanılması, özellik vektörünün normalizasyonunun yapılmasını sağlamış, bu da sınıflandırma algoritmaları ve çok katmanlı mimariler ile yapılan sınıflandırma işlemlerinin doğruluk oranının olumlu yönde yükselmesini sağlamıştır.

- Yaratılan özellik vektörlerinin, numerik ve alfanumerik bilgileri, farklı tip sınıflandırma algoritmaları ve metin girdisi ile işlem yapan derin ağ modelleri ile uygumluluğunu doğurmuş bu da kullanıcıya zenginlik sağlamıştır.

- Öznitelik çıkartma yönteminin, gerçek zamanlı maskeler üretebilecek derin ağ modellerinin ara katmanlarına adapte edilebilecek nitelikte olması, bu alanda çalışmalar yapan kişilerin ilgisini çekebilecek düzeyde olduğu söylenebilir.

- Yöntem ile elde edilen poligonun sadece poligon kodu ile çıkartılan özniteliği değil, bunun yanı sıra farklı özniteliklerinin de çıkartılabilmesine müsait yapıda olması bilimde kullanılabilirliğini arttıracak bir artı olarak görülmektedir.

Tablo 5.1. HMDB51 veri seti üzerinde önerilen öznitelik çıkartma yönteminin kullanılması sonucu, farklı sınıflandırıcılarda elde edilen en iyi doğruluk yüzdeleri.

METOT	DOĞRULUK (%)
ViCC-R-F+k-NN Nn=5 [111]	54,6
RBM + DVM [99]	55,2
OFCM + DVM [100]	56,9
(Deep)Inception3D [102]	56,9
GBH+MBH+DVM [98]	62,0
(Deep)ActionVLAD [103]	66,9
(Deep)TS-LSTM [104]	69,0
(Deep)TSM [105]	72,7
ViCC-R-F+k-NN Nn=20 [111]	76,2
Önerilen Metot + DVM (Poly + 3-mer)	**75,01**
Önerilen Metot + k-NN (Nn=3, 3-mer)	**79,98**

Tablo 5.1'de görüldüğü gibi, önerilen öznitelik çıkarma yöntemi, geleneksel yöntemlere [98], [99], [100], [111] ve bazı derin modellere [102], [103], [104], [105] dayalı yöntemlerden daha iyi sonuçlar vermektedir. k-NN sınıflandırıcısı ile aldığımız sonuç, tablodaki en iyi yöntem [111] ile karşılaştırıldığında, %4,96 geliştirilmiştir. DVM sınıflandırıcılı en iyi model sonucu olan [98], önerilen metot + DVM sınıflandırıcı ile %20.98 geliştirilmiştir. Ayrıca tablodaki en iyi sonuca sahip derin model [105], en iyi sonucumuz olan %79.98 ile %10.01 aşılmıştır.

İncelenen çalışmalardan biri olan [111]'de, örnek grupları üzerinde çalışan hem RGB hem de optik akış görünümlerinden tutarlı prototip atamalarını tahmin eden yeni bir yöntem olan "Video Çapraz Akış Prototipik Kontrastı" önerilmiş. Spesifik olarak, optimizasyon sürecini değiştirmişler; akışlardan biri optimize edilirken, tüm görünümler bir dizi akış prototip vektörüne eşlemişlerdir. Sonuç olarak, çıkarım sırasında optik akış hesaplamasına açık bir ihtiyaç olmadan, yerleşik hareket bilgileriyle daha verimli video yerleştirmelerinin öğrenilmiş olduğunu göstermişlerdir. Çalışmada HMDB51 ve UCF101 veri seti üzerinde k-NN sınıflandırıcı sonuçları verilmiştir.

Tablo 5.2. UCF101 veri seti üzerinde önerilen öznitelik çıkartma yönteminin kullanılması sonucu, farklı sınıflandırıcılarda elde edilen en iyi doğruluk yüzdeleri.

METOT	DOĞRULUK (%)
(Deep)PoseBase [107]	60,9
Sc.+Obj.+Pose +k-NN[108]	62,1
(Deep)PoTion [106]	65,2
(Deep)OPN [109]	71,8
(Deep)VCOP [110]	72,4
iDT w/BOW+DVM[112]	76,2
ViCC-RGB+k-NN Nn=5 [111]	77,1
Önerilen Metot + DVM	**77,44**
Önerilen Metot + k-NN	**82,02**

Tablo 5.2'dan görüldüğü gibi, önerilen öznitelik çıkarma yöntemi, geleneksel yöntemlere [108], [111], [113] ve bazı derin modellere [106], [107], [109], [110] dayalı yöntemlerden daha iyi sonuçlar vermektedir. k-NN sınıflandırıcısı ile aldığımız sonuç, tablodaki en iyi yöntem [111] ile karşılaştırıldığında, önerilen metot + k-NN ile %6,38 geçilmiştir. DVM sınıflandırıcılı en iyi model [112], önerilen metot + DVM sınıflandırıcı ile %1,63 geçilmiştir. Ayrıca tablodaki en iyi sonuca [110] sahip derin model, en iyi sonucumuz olan %82.02 ile %13.29 geçilmiştir.

İncelenen çalışmalardan biri olan [98]'de, Gradyan Sınır Histogramları (Gradient Boundary Histograms) adı verilen bir uzay-zamansal tanımlayıcıyı tanıtılmaktadır. Amaç, hesaplama maliyeti yüksek olan optik akış yönteminden kaçınmaktır. SIFT ve HOG tanımlayıcılarındaki θ ve r değerlerinin yön histogramlarını tutmuşlardır. Hareketli kenar sınırlarında vurgulanan görüntünün gradyanlarının zaman türevlerini kullanarak çözüm üretmişlerdir. LPM özelliklerine PCA ile boyut indirgeme yapılmış HMDB51 lineer DVM sınıflandırıcısı ile özellikler sınıflandırılarak sonuçlar alınmıştır. İncelenen çalışmalardan bir diğeri olan [105]'deki "Temporal-Spatial Mapping", isimli çalışmada tüm videoların karelerinin evrişim özelliklerini dönüştürerek bunları VideoMap olarak adlandırıp, 2B özellik haritası çıkarmıştır. Bir görüntünün her satırı vektörleştirilmiş

özellik gösterimi olduğundan, zamansal-uzaysal özellikler kompakt bir şekilde çalışmada temsil edilmiş ve sınıflandırma derin ağ mimarileri ile gerçekleştirilmiştir.

Tez çalışmasında sonuçların literatürle kıyaslanmasının dışında Bölüm 4 içerisinde çalışmanın kendi içinde ürettiği sonuçların hesaplanma süresi ve doğruluğu testleri yapılmıştır. Bu sayede parametrelerin sonuçları nasıl etkilediği netlik kazanmıştır. Donanımsal olarak akademik çalışmalarda belirtilen aynı test ortamları sağlanamadığı için diğer sistemlerle işlem süresi ve doğruluk açısından karşılaştırma yapmak mümkün değildir.

Poligonizasyon algoritmasının temel avantajlarından biri, problem konusu ne olursa olsun her türlü görüntü tabanlı çalışmada kullanılabilir olmasıdır. Görüntülerin ikilik veya renkli olmalarından bağımsız olarak, görüntüdeki nesne için kontur koordinat bilgileri bir liste veri yapısı içerisinde elde edilebiliyorsa, önerilen poligonlaştırma algoritması ile istenen sayıda köşeye poligonlaştırılabilir. Herhangi bir boyuttaki herhangi bir kapalı kontur için uygun hedef kenar değeri belirlenmesi yeterlidir. Poligonlaştırma algoritması kontur uzunluğu ve hedef kenar sayısına göre adım değeri belirleyip, bu adım değerine göre indis değerlerinde atlamalar yaparak veriyi işlediği için logaritmik bir asimtotik notasyona sahip olduğu söylenebilir. Bu algoritma sadece maske görüntüleri için değil, herhangi bir görüntü için kullanılabilir. Hızı adım hedef kenar sayısı ve buna bağlı olarak adım değeri gibi parametrelerle oynayarak ayarlanabilir.

Yapılan literatür çalışmasında, “silüetleri poligonize eden ve onlardan öznitelik çıkaran çalışmalar olmadığı” görülmüştür. Çalışmada önerilen öz nitelik çıkartma yöntemi kullanıldığında devamında boyut sorunundan kaçınmak için ek boyut indirgeme algoritmaları uygulamak gerekli değildir. Çalışmada önerilen poligonlaştırma algoritması ve poligon kodlaması, öznitelik vektörünün her görüntü boyutu için aynı uzunlukta olmasını garanti eder. K-mer gruplaması yapılan ve normalizasyon işlemine tabi tutulan her bir öznitelik grubu ile oluşturulmuş öznitelik vektörü herhangi bir sınıflandırma algoritmasında direkt olarak kullanılabilir. Algoritmaya girdi olarak verilecek öznitelik vektörünün kolonunda bulunan değer sayısı, k-mer değeri ya da hedef kenar sayısı değiştirilerek ayarlanabilir. Bu kurgu, sistemin genel hızının ve performansının bir parametre hilesi ile ayarlanmasına olanak tanır.

Yöntem, maskeleme yapan ve gerçek zamanlı sınıflandırmaya uyarlanabilen herhangi bir derin ağ mimarisinin ara katmanında kullanılabilir nitelikte kodlanmıştır. Bu fikir, donanım kaynaklarımızı iyileştirdikten sonra, gelecekteki çalışma planımızın bir parçasıdır. Tüm algoritmanın yürütülmesi sırasındaki ara adımlarda elde ettiğimiz PoS görüntü formu, ileride ele almayı planladığımız derin mimarinin eğitim ve test aşamalarında da kullanılacaktır. Oluşturulan poligon şekillerinden vektör kodlarının çıkarılmasına ek olarak, vektör uzunlukları, poligonun alanı, çevresi, köşe koordinatlarının merkezden uzaklığı, Hu momentleri, iç ve dış bükey kenar sayısı, minimum çevreleyen daire, yıldız iskelet, diferansiyel kod vb. ek öznitelikler çıkarılarak çalışmalara dahil edilebilir.

K-En yakın komşuluk sınıflandırma parametreleri veya DVM sınıflandırıcısının çekirdek parametrelerindeki oynamalarla, sınıflandırma performansının sonuçları daha da iyileştirilebilir. Çalışmada kullanılan k-NN sınıflandırıcısına ait parametreler *KNeighborsClassifier(weights = 'distance', algorithm='ball_tree', leaf_size = 100, p=1, n_neighbors = n_neighbors)* şeklinde iken DVM sınıflandırıcı parametreleri *SVC(kernel = kernel, degree=7, gamma="scale")* olarak seçilmiştir. Kernel değeri "poly", "lineer" ve "rbf" olacak şekilde sırasıyla değiştirilmiş ve her farklı k-mer için ayrı ayrı sonuçlar alınmıştır. Tablo 4.7 ve Tablo 4.8'in alt satırlarında kullanılan kernel ve komşuluk değerine karşı alınan sonuçların tamamı gösterilmiştir.

En yakın komşuluk sınıflandırma algoritması kullanılırken denenen 3, 5, 7 gibi komşulukların üzerine çıkıldıkça, sınıflandırıcının başarımında bozulmalar olduğu, bunun sebebinin de karşılaştırma yapılan öznitelikler arası girişimin daha fazlalaştığı bununda sonuca olumsuz etki gösterdiği görülmüştür. Yanlız şu unutulmamalıdırki adaptif çekirdek değeri gibi davranan k-mer değeri büyüdükçe oluşan grup adedi küçüleceğinden öznitelik vektörü de küçülecektir. Bu da belli bir noktadan sonra sınıflandırma başarısını olumsuz yönde etkiler. O yüzden uygulamalarda optimum k-mer değeri denemeler yapılarak belirlenmelidir.

Bu tez çalışmasında kullandığımız veri setleri üzerinde 3, 4, 5 gibi değerler en iyi aralıkta sonuçlar almamızı sağlamış olsa da, veri seti değiştirildiğinde optimum değerler olarak bulunan bu sayıların değiştirilmesi gerekebilir. Bu yüzden yöntem veri seti üzerinde uygulandıktan sonra denemeler yapılması önerilmektedir.

Çalışmada önerilen çözüm yöntemi önerildiği formdan daha farklı noktalara evrilebilir, geliştirmelere sonuna kadar açıktır. Bu yöntem kullanılarak, siluet görüntülerinin farklı öznitelikleri çıkartılabilir veya öznitelik çıkartma adımları bir derin ağ modelinin ara katmanlarına dâhil edilerek, o model ile füzyonlanıp, tümleşik bir sistem olarak da kullanılabilir. Çalışma gerçekleştirilirken yeterli donanım alt yapısının olmayışı, derin ağlarla yapılan işlemler için büyük veri setlerinde çevrimiçi sistemlerin kullanılmak zorunda oluşu, araştırma kapsamında düşünülen bu fikrin, imkânların iyileştirildiği bir sürece bırakılmasına neden olmuştur.

KAYNAKLAR

[1] Wikipedia, “Closed Circuit Television - CCTV”, en.wikipedia.org/wiki/CCTV (Son Erişim 25.11.2022)

[2] R. U. Shekokar and S. N. Kale, "Deep Learning for Human Action Recognition," 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, 2021, pp. 1-5. doi: 10.1109/I2CT51068.2021.9418080

[3] J. Adolf, J. Dolezal, M. Macas and L. Lhotska, "Remote Physical Therapy: Requirements for a Single RGB Camera Motion Sensing," 2021 International Conference on Applied Electronics (AE), 2021, pp. 1-4, doi: 10.23919/AE51540.2021.9542912.

[4] A. K. Mishra, M. Skubic and C. Abbott, "Development and preliminary validation of an interactive remote physical therapy system," 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015, pp. 190-193, doi: 10.1109/EMBC.2015.7318332.

[5] F. Ullah, A. Iqbal, A. Khan, R. G. Khan, L. Malik and K. S. Kwak, "An Image-based Human Physical Activities Recognition in an Indoor Environment," 2020 International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 588-593,doi: 10.1109/ICTC49870.2020.9289314.

[6] B. Kwon, J. Kim, K. Lee, Y. K. Lee, S. Park and S. Lee, "Implementation of a Virtual Training Simulator Based on 360° Multi-View Human Action Recognition," in IEEE Access, vol. 5, pp. 12496-12511, 2017 doi:10.1109/ACCESS.2017.2723039.

[7] K. Sharma, V. Agarwal and A. K. Rajpoot, "Digital Yoga Game with Enhanced Pose Grading Model," 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), 2022, pp. 1-6, doi: 10.1109/ICCSEA54677.2022.9936591.

[8] J. W. Park, Y. C. Lee, B. S. Jo and C. -W. Lee, "Virtual playing ground interface using upper-body gesture recognition," 2011 17th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), 2011, pp. 1-5, doi: 10.1109/FCV.2011.5739762.

[9] Y. Sha, P. Shi, D. Pan, S. Zhou, "Human pose estimation combined with depth information," 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016, pp. 663-667, doi: 10.1109/IMCEC.2016.7867292.

[10] H. C. Shih, "A Survey of Content-Aware Video Analysis for Sports," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 5, pp. 1212-1231, May 2018, doi: 10.1109/TCSVT.2017.2655624.

[11] R. Xu, "Sports Video Extraction and Analysis Method Based on Cyclic Random Walk Algorithm," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 1005-1008, doi: 10.1109/ICOSEC51865.2021.9591877.

[12] G. Mori and J. Malik, "Recovering 3D human body configurations using shape contexts," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 7, pp. 1052-1062, July 2006, doi: 10.1109/TPAMI.2006.149.

[13] J. Cheng, H. Liu, F. Wang, H. Li and C. Zhu, "Silhouette Analysis for Human Action Recognition Based on Supervised Temporal t-SNE and Incremental Learning," in IEEE Transactions on Image Processing, vol. 24, no. 10, pp. 3203-3217, Oct. 2015, doi: 10.1109/TIP.2015.2441634.

[14] L. Zhang and Y. Liang, "Motion Human Detection Based on Background Subtraction," 2010 Second International Workshop on Education Technology and Computer Science, 2010, pp. 284-287, doi: 10.1109/ETCS.2010.440.

[15] L. Bin, L. Huanxia and Z. Qiang, "The research of moving object detection based on complex background," 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), 2011, pp. 1366-1369, doi: 10.1109/MEC.2011.6025724.

[16] S. Kumari and S. K. Mitra, "Human Action Recognition Using DFT," 2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, 2011, pp. 239-242, doi: 10.1109/NCVPRIPG.2011.58.

[17] Z. Y. Lin, J. L. Chen and L. G. Chen, "A 203 FPS VLSI Architecture of Improved Dense Trajectories for Real-Time Human Action Recognition," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 1115-1119, doi: 10.1109/ICASSP.2018.8461988.

[18] D. Yang, Y. Wang, A. Dantcheva, L. Garattoni, G. Francesca and F. Brémond, "Self-Supervised Video Pose Representation Learning for Occlusion- Robust Action Recognition," 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021), 2021, pp. 1-5, doi: 10.1109/FG52635.2021.9667032.

[19] M. Ahmad, L. Seong-Whan, "Hmm-Based Human Action Recognition Using Multiview Image Sequences" Pattern Recognition, 2006. Icpr 2006. 18th International Conference On Volume 1, Page(S):263 – 266

[20] C. Orrite, F. Mart´Inez, E. Herrero, H. Ragheb,S. Velastin, "Independent Viewpoint Silhouette-Based Human Action Modelling And Recognition", The 1st International Workshop On Machine Learning For Vision-Based Motion Analysis - MLVMA'08, Marseille : France (2008)

[21] M. Ahmad, L. Seong-Whan, "Recognizing Human Actions Based On Silhouette Energy İmage And Global Motion Description" Automatic Face & Gesture Recognition, 2008. Fg '08. 8th Ieee International Conference On 17-19 Sept. 2008 Page(S):1 – 6

[22] L. Wang And David Suter, "Informative Shape Representations For Human Action Recognition", Pattern Recognition, 2006. Icpr 2006. 18th International Conference On Volume 2, Page(S):1266 – 1269

[23] R. Myung-Cheol, S. Ho-Keun, L. Seong-Whan, "View-Independent Human Action Recognition With Volume Motion Template On Single Stereo Camera" CJKPR 2009 - The 1st CJK Joint Workshop On Pattern Recognition

[24] A. Yang, R. Jarafi, P. Kuryloski, S. Iyengar, S. Sastry, R. Bajcsy, "Distributed Segmentation And Classification Of Human Actions Using A Wearable Motion Sensor Network", Workshop On Human Communicative Behavior Analysis, CVPR 2008.

[25] M. Ahmad, L. Seong-Whan, “Human Action Recognition Using Shape And CLG-Motion Flow From Multi-View İmage Sequences” Pattern Recognition (2008), Pp. 2237–2252.

[26] C. Chuang, J. Hsieh, L. Tsai, K. Fan, “Human Action Recognition Using Star Templates And Delaunay Triangulation”, International Conference On Intelligent Information Hiding And Multimedia Signal Processing

[27] H. Meng, N. Pears, C. Bailey , “A Human Action Recognition System For Embedded Computer Vision Application”, Computer Vision And Pattern Recognition 2007

[28] X. Li, “Hmm Based Action Recognition Using Oriented Histograms Of Optical Flow Field” Electronıcs Letters 10th May 2007 Vol. 43 No. 10

[29] C. Canton-Ferrer, J.R.Casas, M.Pard`As, M.E.Sargin, A.M.Tekalp, “3d Human Action Recognition In Multiple View Scenarios”, ICIP, 2006

[30] F. Niu And Mohammed Abdel-Mottaleb, [2004] “View-Invarient Human Acivity Recognition Based On Shape And Motion Features”, Dept. Of Electr. & Comput. Eng., Miami Univ., USA

[31] V. Veeraraghavan, [2004], “Role Of Shape And Kinematics İn Human Movement Analysis.” IEEE Computer Society Conf. On Computer Vision And Pattern Recognition, CVPR, Vol. 1, Pp. 730-737

[32] K. Hatun, P. Duygulu, “A New Representation For Action Recognition Using Sequence Of Posewords”, 19th International Conference On Pattern Recognition, 2008.

[33] L. Gorelick, M. Blank, E. Shechtman, M. Irani and R. Basri, “Actions as Space-Time Shapes,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 12, pp. 2247-2253, Dec. 2007, doi: 10.1109/TPAMI.2007.70711.

[34] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 257-267, March 2001, doi: 10.1109/34.910878.

[35] K. K. Htike, O. O. Khalifa, H. A. Mohd Ramli and M. A. M. Abushariah, “Human activity recognition for video surveillance using sequences of postures,” The Third International Conference on e-Technologies and Networks for Development (ICeND2014), 2014, pp. 79-82, doi: 10.1109/ICeND.2014.6991357.

[36] S. Carlsson and J. Sullivan, “Action recognition by shape matching to key frames,” presented at the Int. Workshop on Models Versus Exemplars in Computer Vision, 2001, doi:10.7202/1026736ar

[37] S. S. Kumar and M. John, "Human activity recognition using optical flow based feature set," 2016 IEEE International Carnahan Conference on Security Technology (ICCST), 2016, pp. 1-5, doi: 10.1109/CCST.2016.7815694.

[38] L. Zhang, W. Zhou, J. Li, J. Li and X. Lou, "Histogram of Oriented Gradients Feature Extraction Without Normalization," 2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2020, pp. 252-255, doi: 10.1109/APCCAS50809.2020.9301715.

[39] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[40] S. Jain, B. L. S. Kumar and R. Shettigar, "Comparative study on SIFT and SURF face feature descriptors," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), 2017, pp. 200-205, doi: 10.1109/ICICCT.2017.7975187.

[41] B. M. Nair and V. K. Asari, "Regression Based Learning of Human Actions from Video Using HOF-LBP Flow Patterns," 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 4342-4347, doi: 10.1109/SMC.2013.741.

[42] D. G. Lowe, (2004), "Distinctive Image Features from Scale-Invariant Keypoints" (PDF), International Journal of Computer Vision, 60 (2): 91–110, CiteSeerX 10.1.1.73.2924, doi:10.1023/B:VISI.0000029664.99615.94

[43] D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller and M. C. Benfield, "Combining Local and Global Image Features for Object Class Recognition," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops, 2005, pp. 47-47, doi: 10.1109/CVPR.2005.433.

[44] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio and T. Serre, "HMDB: A large video database for human motion recognition," 2011 International Conference on Computer Vision, 2011, pp. 2556-2563, doi: 10.1109/ICCV.2011.6126543.

[45] K. Soomro, A. Roshan Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, arXiv:1212.0402. Linkten Ulaşılabilir: arxiv.org/abs/1212.0402

[46] U. Aftab and G. F. Siddiqui, "Big Data Augmentation with Data Warehouse: A Survey," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 2775-2784, doi: 10.1109/BigData.2018.8622182.

[47] L. Perez, J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning", Computer Vision and Pattern Recognition, 2017.doi: 10.48550/arXiv.1712.04621

[48] F. Gimenez, Z. Hussain, D. Yi "Differential Data Augmetation Techniques for Medical Imaging Classifcation Tasks," 2018, AMIA Annu Symp Proceedings, pp. 979984

[49] J. Shijie, W. Ping, J. Peiyi and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," 2017 Chinese Automation Congress (CAC), 2017, pp. 4165-4170, doi: 10.1109/CAC.2017.8243510.

[50] M. Sajjad, S. Khan, K. Muhammad, W. Wu, A. Ullah, S. W. Baik, "Multi-grade brain tumor classification using deep CNN with extensive data augmentation", 2019, Journal of Computational Science Volume 30, pp.174-182, doi: 10.1016/j.jocs.2018.3

[51] P. König, A. Hernández-García, "Further Advantages of Data Augmentation on Convolutional Neural Networks", 2018, Artificial Neural Networks and Machine Learning ICANN Lec. Notes in Computer Sci., vol 11139. Springer, Cham.

[52] S. Noguchi, M. Nishio, M. Yakami, K. Nakagomi, K. Togashi, “Bone Segmentation on Whole-Body CT Using CNN With Novel Data Augmentation Techniques”, 2020, Computers in Biology and Medicine, Vol. 121.

[53] E. Taşçı, A. Onan, “K-En Yakın Komşu Algoritması Parametrelerinin Sınıflandırma Performansı Üzerine Etkisinin İncelenmesi” , XVIII. Akademik Bilişim Konferansi – 2016

[54] T. Mitchell, “Machine Learning”,McGraw Hill, New York, (1997).

[55] T. Mladenova and I. Valova, "Analysis of the KNN Classifier Distance Metrics for Bulgarian Fake News Detection," 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2021, pp. 1-4, doi: 10.1109/HORA52670.2021.9461333.

[56] Wikipedia “Euclidean Distance”, en.wikipedia.org/wiki/Euclidean_distance (Son Ziyaret: 17/12/2019)

[57] Wikipedia “Manhattan Distance”, en.wikipedia.org/wiki/Manhattan _distance (Son Ziyaret: 17/12/2019)

[58] Wikipedia “Chebyshev Distance”, en.wikipedia.org/wiki/Chebyshev_distance (Son Ziyaret: 17/12/2019)

[59] Wikipedia “Minkowski Distance”, en.wikipedia.org/wiki/Minkowski_distance (Son Ziyaret: 17/12/2019)

[60] C. Kuzey, “Veri Madenciliğinde Destek Vektör Makinaları ve Karar Ağaçları Yöntemlerini Kullanarak Bilgi Çalışanlarının Kurum Performansı Üzerine Etkisinin Ölçülmesi ve Bir Uygulama”, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü, İşletme A.B.D.Doktora Tezi, 2021

[61] Wikipedia, “Kernel Methods”, en.wikipedia.org/wiki/Kernel_methods (Son Ziyaret: 17/12/2019)

[62] E. Güldoğan, “Çeşitli Çekirdek Fonksiyonları İle Oluşturulan Destek Vektör Makinesi Modellerinin Performanslarının İncelenmesi: Bir Klinik Uygulama”, 2017, Doktora Tezi, İnönü Üniversitesi ve Mersin Üniversitesi Biyoistatistik ve Tıp Bilişimi Anabilim Dalı Ortak Doktora Programı.

[63] G. Nur Calayır, M. Kaba, “Bakım için makine öğrenme tekniklerinin analizi ve bir uygulam”, JTOM(5)1, 662-675, 2021

[64] T. CURA, “Karar Verme Aracı Olarak Yapay Sinir Ağları ve Yapay Sinir Ağı İle Portföy Optimizasyonu”, 2004, Doktora Tezi, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı Sayısal Yöntemler Bilim Dalı

[65] M. Ali Seven, A. Osman Engin, “Öğrenmeyi Etkileyen Faktörler”, (2010), Atatürk Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, Cilt 12, Sayı 2, Sayfa 189-212, dergipark.org.tr/tr/pub/ataunisosbil/issue/2822/38073

[66] D. N. Perkins, G. Salomon, (1992) "Transfer of Learning. In T. Husén, & T. N. Postlethwaite (Eds.)", The International Encyclopedia of Education (2nd ed., pp. 425-441). Oxford: Pergamon.

[67] B. Büyükarıkan, E. Ülker, “ Aydınlatma Özelliği kullanılarak Evrişimsel Sinir Ağı Modelleri İle Meyve Sınıflandırma”, Uludağ Üniversitesi, Mühendislik Fakültesi Dergisi, Cilt 25, Sayı 1, 2020, doi:10.17482/uumfd.628166

[68] He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. 2017 IEEE International Conference On Computer Vision (ICCV).

[69] F. Bayram, “Derin öğrenme tabanlı otomatik plaka tanıma”, Politeknik Dergisi, 23(4), 955-960. 2020.

[70] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, P. Ramanan, “Microsoft COCO: Common Objects in Context”, Computer Vision – ECCV 2014, 740-755, 2014

[71] Pinheiro, P., Lin, T., Collobert, R., & Dollár, P. (2016). Learning to Refine Object Segments. Computer Vision – ECCV 2016, 75-91.

[72] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. 2018 IEEE/CVF Conference On Computer Vision And Pattern Recognition.

[73] Fu, C. Y., Shvets, M., & Berg, A. C. (2019). RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. arXiv preprint arXiv:1901.03353.

[74] M. Everingham, S. Eslami, L. Van Gool, C. Williams, J. Winn, A. Zisserman, (2014) "The Pascal Visual Object Classes Challenge: A Retrospective." International Journal Of Computer Vision, 111(1), 98-136.

[75] G. Neuhold, T. Ollmann, S. Bulo, P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes." 2017, IEEE International Conference On Computer Vision (ICCV).

[76] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler,R. Benenson, "The Cityscapes Dataset for Semantic Urban Scene Understanding." 2016, IEEE Conference On Computer Vision And Pattern Recognition (CVPR).

[77] X. Xiao and W. Wan, "Human pose estimation via improved ResNet50," 4th International Conference on Smart and Sustainable City (ICSSC 2017), 2017, pp. 1-5, doi: 10.1049/cp.2017.0126.

[78] A Zhang, Z.C. Lipton, M. Li, A.J. Simola, “Dive Into Deep Learning”, E-Book, tr.d2l.ai/index.html, (Son Ziyaret: 22/12/2022)

[79] I. Goodfellow, Y. Bengio, A. Courville “Deep Learning”, MIT Press Book, (2016) , deeplearningbook.org/ , (Son Ziyaret : 1/12/2022)

[80] Y. Li, Z. Chen, “Performance evaluation of machine learning methods for breast cancer prediction” Appl Comput Math, 7(4), 212-216, (2018)

[81] M. E. Balaban, E. Kartal, “Veri Madenciliği ve Makine Öğrenmesi Temel Algoritmaları ve R Dili ile Uygulamaları” (Birinci Baskı.), İstanbul: Çağlayan Kitabevi, (2015)

[82] S. Kumar, D. Manocha, A. Lastra, “Interactive display of large NURBS models,” in IEEE Transactions on Visualization and Computer Graphics, vol. 2, no. 4, pp. 323-336, Dec. 1996, doi: 10.1109/2945.556501.

[83] D. J. Harbinson, R. J. Balsys, K. G. Suffern, "Polygonisation of Non-manifold Implicit Surfaces Using a Dual Grid and Points," 2010 Seventh International Conference on Computer Graphics, Imaging and Visualization, 2010, pp. 26-31, doi: 10.1109/CGIV.2010.35.

[84] R. J. Balsys and K. G. Suffern, “Adaptive polygonisation of non-manifold implicit surfaces,” International Conference on Computer Graphics, Imaging and Visualization (CGIV'05), 2005, pp. 257-263, doi: 10.1109/CGIV.2005.13.

[85] C. E. V. Muniz, A. A. Montenegro and E. W. G. Clua, “Fast Polygonization and Texture Map Extraction from Volumetric Objects Based on Surface Fairing Using a Modified Discrete Laplacian Operator,” 2010 Brazilian Symposium on Games and Digital Entertainment, 2010, pp. 146-155, doi: 10.1109/SBGAMES.2010.33.

[86] S. Saha and A. Biswas, “Polygonization of 3D Objects using Norm Similarity,” 2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2020, pp. 1-6, doi: 10.1109/ICIEVicIVPR48672.2020.9306612.

[87] L. Hermes and J. M. Buhmann, “A minimum entropy approach to adaptive image polygonization,” in IEEE Transactions on Image Processing, vol. 12, no. 10, pp. 1243-1258, Oct. 2003, doi: 10.1109/TIP.2003.817240.

[88] M. Pang, Z. Pan, M. Zang, F. Zang, “An Adaptive and Efficient Algorithm for Polygonization of Implicit Surfaces,” Computational Science and Its Applications – ICCSA 2005. ICCSA 2005. Lecture Notes in Computer Science, vol. 3482. Springer, Berlin, Heidelberg, doi: 10.1007/11424857_27

[89] B. Yang, G.-L. Chen and M.-Y. Pang, “Parallel Polygonization of Implicit Surfaces,” 2010 International Symposium on Intelligence Information Processing and Trusted Computing, 2010, pp. 220-223, doi: 10.1109/IPTC.2010.137.

[90] B. R. de Araujo and J. A. P. Jorge, "Curvature dependent polygonization of implicit surfaces," Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing, 2004, pp. 266-273, doi: 10.1109/SIBGRA.2004.1352970.

[91] M. Marji, P. Siy, “A new algorithm for dominant points detection and polygonization of digital curves”, Pattern Recognition, Volume 36, Issue 10, 2003, Pages 2239-2251, ISSN 0031-3203.

[92] C. Coniglio, C. Meurie, O. Lézoray, M. Berbineau, “People silhouette extraction from people detection bounding boxes in images”, Pattern Recognition Letters, Volume 93, 2017, Pages 182-191,ISSN:0167-8655, doi: 10.1016/j.pac.2016.12.014

[93] A. Akula, N. Khanna, R. Ghosh, S. Kumar, A. Das, H.K. Sardana, “Adaptive contour-based statistical background subtraction method for moving target detection in infrared video sequences”, Infrared Physics & Technology, Volume 63, 2014, Pages 103-109, ISSN 1350-4495, doi: 10.1016/j.infrared.2013.12.012

[94] B. Garciaa, T. Bouwmans, A. J. R. Silva, “Background subtraction in real applications: Challenges, current models and future directions”, Computer Science Review, Volume 35, 2020, ISSN 1574-0137, doi: 10.1016/j.cosrev.2019.04

[95] M. Chapel, T. Bouwmans, “Moving objects detection with a moving camera: A comprehensive review”, Computer Science Review, Volume 38, 2020, 100310, ISSN 1574-0137, doi: 10.1016/j.cosrev.2020.100310

[96] C. L. Biji, S. N. Achuthsankar, M. K. Madhu, R. Vijayakumar, “A Bio-Sequence k-mer Frequency Counter (kFC)”, Proceedings of International Conference on Circuits, Communication, Control and Computing (I4C 2014)

[97] A. G. Borodinov, V. V. Manoilov, I. V. Zarutskiy, A. I. Petrov and V. E. Kurochkin, "Methodology for Assessing the Quality of Genomic Assembly Based on the Analysis of K-Mers Frequency in a Parallel Sequencing Sequencer," 2021 XV International Scientific-Technical Conference on Actual Problems Of Electronic Instrument Engineering (APEIE), 2021, pp. 512-515, doi: 10.1109/APEIE52976.2021.9647640.

[98] F. Shi, R. Laganiere and E. Petriu, “Gradient Boundary Histograms for Action Recognition,” 2015 IEEE Winter Conference on Applications of Computer Vision, 2015, pp. 1107-1114, doi: 10.1109/WACV.2015.152.

[99] F. P. An, “Human Action Recognition Algorithm Based on Adaptive Initialization of Deep Learning Model Parameters and Support Vector Machine,” in IEEE Access, vol. 6, pp. 59405-59421, 2018, doi: 10.1109/ACCESS.2018.2874022.

[100] C. Caetano, J. A. dos Santos and W. R. Schwartz, “Optical Flow Co-occurrence Matrices: A novel spatiotemporal feature descriptor,” 2016 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 1947-1952, doi: 10.1109/ ICPR.2016. 7899921.

[101] J. Carreira and A. Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4724-4733, doi: 10.1109/CVPR.2017.502.

[102] A. Diba, V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss, “Spatio-temporal Channel Correlation Networks for Action Classification”, Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol. 11208. Springer, Cham., doi: 10.1007/978-3-030-01225-0_18

[103] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic and B. Russell, “ActionVLAD: Learning Spatio-Temporal Aggregation for Action Classification,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3165-3174, doi: 10.1109/CVPR.2017.337.

[104] M. Chih-Yao, C. Min-Hung, K. Zsolt, A. Ghassan, “TS-LSTM and temporal inception: Exploiting spatiotemporal dynamics for activity recognition,” Signal Process., Image Common., vol. 71, pp. 76–87, Feb. 2019. arXiv:1703.10667

[105] X. Song, C. Lan, W. Zeng, J. Xing, X. Sun, and J. Yang, “Temporal– spatial mapping for action recognition,” IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 3, pp. 748–759, Mar. 2020. arXiv:1809.03669

[106] V. Choutas, P. Weinzaepfel, J. Revaud and C. Schmid, “PoTion: Pose MoTion Representation for Action Recognition,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7024-7033, doi: 10.1109/CVPR.2018.0074.

[107] F. Angelini and S. M. Naqvi, “Joint RGB-Pose Based Human Action Recognition for Anomaly Detection Applications,” 2019 22th International Conference on Information Fusion (FSION) 2019, pp. 1-7, doi: 10.23919/FUSION43075.2019.90177.

[108] Y. Ji, Y. Zhan, Y. Yang, X. Xu, F. Shen and H. T. Shen, “A Context Knowledge Map Guided Coarse-to-Fine Action Recognition,” in IEEE Transactions on Image Processing, vol. 29, pp. 2742-2752, 2020, doi: 10.1109/TIP.2019.2952088.

[109] H. Y. Lee, J. B. Huang, M. Singh, M. H. Yang, “Unsupervised Representation Learning by Sorting Sequences,” 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 667-676, doi: 10.1109/ICCV.2017.79.

[110] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie and Y. Zhuang, “Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10326-10335, doi: 10.1109/CVPR.2019.01058.

[111] M. Toering, I. Gatopoulos, M. Stol and V. T. Hu, “Self-supervised Video Representation Learning with Cross-Stream Prototypical Contrasting,” 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022, pp. 846-856, doi: 10.1109/WACV51458.2022.00092.

[112] Y. Qin, L. Mo and B. Xie, "Feature fusion for human action recognition based on classical descriptors and 3D convolutional networks," 2017 Eleventh International Conference on Sensing Technology (ICST), 2017, pp. 1-5, doi: 10.1109/ICSensT.2017.8304460.

[113] D. Bolya, C. Zhou, F. Xiao and Y. J. Lee, “YOLACT++ Better Real-Time Instance Segmentation,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 2, pp. 1108-1121, 1 Feb. 2022, doi: 10.1109/TPAMI.2020.3014297.