GÖRÜNTÜ İŞLEME TEKNİKLERİ VE EVRİŞİMSEL SİNİR AĞLARI KULLANARAK YÜZ İFADESİNDEN DUYGU TESPİTİ

Fatih ALTEKİN

Yüksek Lisans Tezi

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı
Danışman: Dr. Öğr. Üyesi Hasan DEMİR

2021

T.C.

TEKİRDAĞ NAMIK KEMAL ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

GÖRÜNTÜ İŞLEME TEKNİKLERİ VE EVRİŞİMSEL SİNİR AĞLARI KULLANARAK YÜZ İFADESİNDEN DUYGU TESPİTİ

Fatih ALTEKİN

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Hasan DEMİR

TEKİRDAĞ-2021

ÖZET

Yüksek Lisans Tezi
GÖRÜNTÜ İŞLEME TEKNİKLERİ
VE EVRİŞİMSEL SİNİR AĞLARI KULLANARAK
YÜZ İFADESİNDEN DUYGU TESPİTİ

Fatih ALTEKİN
Tekirdağ Namık Kemal Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Hasan DEMİR

Bu tez çalışmasında, yüzdeki duygu ifadelerini tespit etmek için görüntü işleme teknikleri ve makine öğrenmesi teknikleri incelenmiştir. Öznitelik vektörlerini elde etmek ve yüzdeki duygu ifadedelerini tespit etmek için yapılan çalışmalar aşamalar halinde verilmiştir. Yapılan uygulamalarda ve kategorilerde, 7 duygu kategorisinde, insan yüzleri içeren 981 adet imgeden oluşan CK+ imge seti kullanılmıştır. CK+ veri setinden HOG, LBP, Geometrik Öznitelikler ve Dalgacık Dönüşümü yöntemleri kullanarak öznitelik veri seti oluşturulmuştur. Öznitelik veri seti Destek Vektör Makinaları, K-En Yakın Komşuluk Algoritması ve Lojistik Regresyon sınıflandırma yöntemleri ile sınıflandırılarak yüz ifade tespiti başarı durumları karşılaştırmalı olarak verilmiştir. Veri setindeki imgelerin orijinal hali ve HOG, LBP ve Dalgacık Dönüşümü öznitelik imgelerinin kullanıldığı durumlardaki başarı oranları karşılaştırılmıştır. Evrişimsel Sinir Ağları (CNN) yönteminin duygu ifadeleri tespitindeki başarısı ele alınmıştır.

Anahtar Kelimeler: Evrişimsel Sinir Ağları, HOG, LBP, Geometrik Öznitelikler, CK+ veriseti, Yüz duygu ifadeleri tanımlama

2021, 138 sayfa

ABSTRACT

MSc. Thesis
FACIAL EXPRESSION RECOGNITION
USİNG IMAGE PROCESSING TECHNIQUES
AND CONVOLUTIONAL NEURAL NETWORKS

Fatih ALTEKİN
Tekirdağ Namık Kemal University
Graduate School of Natural and Applied Sciences
Department of Electronics and Communication Engineering

Supervisor: Assist. Prof. Dr Hasan Demir

In this thesis, image processing techniques and machine learning techniques have been examined to detect facial emotions. The studies carried out to obtain the feature vectors and determine the emotion expressions on the face are given in stages. In the applications and categories, CK + image set consisting of 981 images containing human faces in 7 emotion categories was used. The feature data set was created from the CK + data set by using HOG, LBP, Geometric Features and Wavelet Transform methods. The feature data set is classified with Support Vector Machines, K-Nearest Neighborhood Algorithm and Logistic Regression classification methods, and face expression detection success cases are given comparatively. The original state of the images in the data set and the success rates in cases where HOG, LBP and Wavelet transform feature images were used were compared. The success of the convolutional neural networks (CNN) method in detecting emotion expressions is discussed.

Keywords: Convolutional Neural Network, HOG, LBP, Geometric Features, CK+ datasets, Facial Emotion Expression Recognition

2021, 138 pages

İÇİNDEKİLER

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

SİMGELER ve KISALTMALAR

LBP	: Local Binary Pattern - Yerel İkili Örüntüler
HOG	: Histogram of Oriented Gradients - Yönlendirilmiş Gradyanların Histogramı
CNN	: Convolutional Neural Networks - Evrişimsel Sinir Ağları
HKS	: Haar Cascade Classifier - Haar Kaskad Sınıflandırıcı
DWT	: Discrete Wavelet Transform - Ayrık dalgacık dönüşümü
SVM	: Support Vector Machines - Destek Vektör Makinalarıdır
KNN	: K-Nearest Neghbors - K-En Yakın Komşuluk Algoritması
LR	: Logistic Regression - Lojistik Regresyon
FACS	: Facial Action Coding System - yüz Eylem Kodlama Sistemi
AU	: Action Units - Eylem Birimleri
AFA	: Automatic Face Analysis - Otomatik Yüz Analizi

TEŞEKKÜR

Yüksek Lisans öğrenimlerim boyunca bana her zaman yol gösteren saygıdeğer danışmanım Sayın Dr. Öğr. Üyesi Hasan DEMİR'e, bana her türlü olanağı sağlayıp, çalışmalarıma katkı sağlayan değerli üniversite hocalarıma, arkadaşlarıma ve maddi ve manevi her türlü desteği sağlayan değerli eşime ve aileme sonsuz teşekkürlerimi sunarım

Haziran 2021 Fatih ALTEKİN

Elektronik ve Haberleşme Mühendisi

1. GİRİŞ

Yüz ifade analizi ile insanların duygusal durumlarının tespiti, uzaktan veya online satış işlemlerinde müşteri davranışı ve duygusuna göre ürün önerilmesi, iş başvurularında kişilerin duygularına göre işe alım kararlarının verilmesinde makine öğrenmesi yöntemlerinin kullanılabilmesine, güvenlik amacıyla mobese kayıtları vasıtasıyla kişilerin yüzlerinden duyguların tanınması ile olası suçların erken tespiti ve önlenmesi, sağlık alanında hastalıkların belirlenmesi gibi konularda etkili çözümler sunabilecek sonuçlar sağlanmaktadır.

Problemin çözümünde kullanılacak görüntü işleme, makine öğrenmesi, evrişimsel sinir ağları yöntemleri elektrik-elektronik sektöründe ve sanayinin ihtiyaç duyduğu alanlarda çalışılması gerektiği ifade edilen veri bilimi, veri modellemesi ve işlenmesi, yapay zeka, yapay zeka yöntemleri, insan-makine etkileşimi, makine öğrenmesi, otonom sistemler ve derin öğrenme gibi başlıklar bu tez kapsamında irdelenmiştir.

Tez kapsamında öncelikle yüz ifadelerinden hareketle belirli duyguları tespit etmek için temelde kullanılan görüntü işleme teknikleri incelenecektir. Ardından sırası ile öznitelik vektörlerinin elde edilmesi ve sınıflandırma yöntemleri ele alınacaktır.

Görüntü işlemenin amacı, görüntüyü iyileştirme ve analiz etme şeklinde temel olarak ikiye ayrılmaktadır [1] . Görüntü analizi, görüntüdeki nesneleri tespit etmek ya da ayırt etmek amacıyla uygulanmaktadır. Görüntü işleme çalışmalarında opencv, c++, python, matlab gibi çeşitli bilgisayar programları kullanılabilmektedir.

Görüntü analiz işlemlerinde, herhangi bir nesneyi ya da objeyi tespit etmek için kullanılan farklı yöntemler bulunmaktadır.

Bu çalışmada, bir görüntüden çeşitli görüntü işleme teknikleri kullanılarak insan yüzünün tespit edilip buna bağlı olarak yüz ifadelerinin belirlenmesine çalışılacaktır. Görüntü işleme teknikleri ile ilgili daha önce yapılan çalışmalar incelenecek olup, yüz tanıma gerçekleştirilecek ve duyguların yüz ifadelerine nasıl yansıdığının tespiti konusunda yapılan çalışmalar incelenecek ve yeni yöntemler geliştirilmeye çalışılacaktır. Elde edilen bulgular evrimsel sinir ağları sonuçları ile karşılaştırılacaktır.

Bu tez çalışmasında imgeler üzerinde yüz tespiti yapılacaktır. Daha sonra yüzdeki duyguların oluştuğu gözler, kaşlar, burun ve ağız gibi yüz bölgelerinden nirengi noktaları belirlenecektir. Belirlenen nirengi noktalarının öznitelikleri çıkarılacaktır. Son aşamada belirlenen özniteliklere göre sınıflandırma işlemi ile duygu analizi yapılacak ve "mutluluk", "üzüntü", "sürpriz", "korku", "küçümseme", "öfke", "şaşkınlık" gibi 7 temel duygu tespit edilmeye çalışılacaktır.

Yüzdeki duygu ifadesini tespit etmek için python programlama dili ve opencv [2], numpy [3], pandas [4] matplotlib [5], dlib [6], scikit-learn [7], tensorflow [8] ve keras [9] kütüphaneleri kullanılacaktır.

2. KAYNAK ARAŞTIRMASI

Bugüne kadar yapılmış olan, görüntü işleme , yüz bölgesi tespiti , yüz bölgesinden duygu tanıma ile ilgili yapılmış olan çalışmalar araştırılmış ve bu bölümde incelenmiştir.

Bayrakdar ve arkadaşları [10] yaptıkları çalışmada yüzdeki duygu ifadelerinin tespit edilmesi ile ilgili literatürde yapılan çalışmaları incelemiş ve yüz ifadelerinin analizinde kullanılan bazı yöntem ve tekniklere yer vermişlerdir. Bu yöntemlerin ve insan-makine etkileşiminin hızla gelişmesi sonucunda, insanlara ait duygusal ifadelerin bilgisayarlar tarafından tespit edilebileceği sonucuna varmışlardır. Yüz ifadelerinin hızlı analiz edilip doğru olarak tanımlanabilmesinin farklı uygulama alanlarındaki birçok yazılım sistemi için çok önemli olduğu sonucuna varmışlardır.

Hareketli ve sabit görüntülerdeki insan yüzlerinin duygusal ifadelerinden psikolojik durumlarını belirlemek amacıyla bazı çalışmalar yapılmıştır. Tenekeci ve arkadaşları [11] tarafından duygusal ifadelerin tespitinde büyük bir önem arz eden göz ve ağız bölgesindeki duygusal değişimlerin belirlenmesi için haar kaskad fonksiyonları kullanılmıştır. Görüntü analizi için C++ programlama dili ile geliştirilen bazı özel algoritmalar kullanılmıştır. Sonuç olarak, yapılan çalışmalar ile bir görüntü içerisinde bulunan yüz ifadelerinin tespit edilebileceği ortaya konmuştur.

Ait Younes ve arkadaşları [12] yaptıkları çalışmada görüntü indeksleme ve alma için bir yazılım geliştirmişlerdir. Bu çalışmada, görüntülerin baskın renklerine dayanan bir sınıflandırma yöntemi önerilmiştir. HLS uzayında (Hue, Lightness, Saturation - Ton, Işık, Doygunluk) resme renkölçer (colorimetric) bir profil atanmıştır. Öncelikle, renk dağılımının düzensizliğini göz önünde bulunduran bulanık bir temsil vasıtasıyla renk tonu tanımı yapılmış, ardından profil görüntünün farklı sınıflara bağlı üyelik derecesini temsil eden bulanık işlevlerle oluşturulmuştur. Performansı artırmak ve daha doğru profiller tanımlayabilmek için, ayrı ayrı pikseller yerine piksel bölgelerinin ön plana çıkarılarak değerlendirilmesi önerilmiştir. Bu bölgeler, bir kenar algılama algoritması aracılığıyla oluşturulmuştur. Bölgenin rengini belirleyebilmek için, bölge içinde bir piksel numunesi seçilmiştir. Çalışma neticesinde, tespit edilen baskın renklere göre, böyle bir yazılımın uyumlu / uyumsuz görüntüleri sınıflandırma vb için kullanılabileceği sonucuna ulaşılmıştır.

Pantic ve Rothkrantz [13] çalışmalarında insanların kolaylıkla bir görüntü içerisindeki yüzleri ve yüz ifadelerini algılamalarına ve yorumlamalarına karşın bu görevi yerine getirebilecek otomatik bir sistemin geliştirilmesinin ciddi manada zor olacağını ifade etmişlerdir. Görüntü içerisindeki bir yüzün otomatik olarak algılanması, yüz ifadesi bilgisinin çıkarılması ve ifadenin sınıflandırılması (örneğin, duygu kategorilerinde) gibi bazı problemlerin var olduğunu ortaya koymuşlardır. Bu işlemleri doğru ve gerçek zamanlı olarak gerçekleştiren bir sistemin tasarımı konusunu da ele almışlardır. Bu sistemin tasarımıyla ilgili problemlerin çözülmesi amacıyla yapılmış geçmiş çalışmaları araştırmışlardır. Mevcut durumu insan görme sisteminin çalışma mekanizması ile karşılaştırmışlardır. İlgili çalışmayı, otomatik bir yüz ifadesi analizörünün geliştirilmesine yönelik tavsiyelerin belirlenmesinde nihai bir amaç ve rehber olarak kullanılmak üzere sunmuşlardır.

Comaniciu ve arkadaşları [14] yaptıkları çalışmada Nonrigid nesnelerin görsel olarak izlenmesinde merkezi bileşen olan hedef gösterimi ve yerelleştirmeye yönelik yeni bir yaklaşım önermektedir. Çalışma doğrultusunda bir histogram tabanlı özellik hedef gösterimleri, izotropik bir çekirdekli mekânsal maskeleme ile düzenlenmiştir. Maskelemenin gradyan bazlı optimizasyonu için, uygun olan ve uzamsal açıdan pürüzsüz benzerlik fonksiyonlarını indüklemekte olduğu görülmüş; bu nedenle hedef lokalizasyon problemi, yerel maksimanın çekim alanı kullanılarak formüle edilebildiğinden benzerlik ölçüsü olarak Bhattacharyya katsayısından türetilen bir metrik kullanılmış ve optimizasyonu gerçekleştirmek için ortalama kayma prosedüründen yararlanılmıştır. Hareket filtreleri ve veri ilişkilendirme teknikleri ile entegrasyon konusu da tartışılmıştır. Arka plan bilgisinin kullanılması, hareket modellerini kullanarak Kalman izleme ve yüz izleme şeklinde ifade edilebilen az sayıda potansiyel uygulama tanımlanmıştır. Çalışmalar sonucunda sunulan izleme örneklerinde; yeni yöntem kamera hareketleri, kısmi tıkanmalar, dağınıklık ve hedef ölçek değişimleriyle alakalı problemlerin başarılı bir şekilde çözüldüğü belirtilmiştir.

Turk ve arkadaşları [15] çalışmalarında, insan yüzünü tespit edip belirleme ve bilinen kişilerin yüz karakteristiklerini karşılaştırarak kişinin baş hareketlerini izleyen ve ardından kimliğini belirleyen yakın gerçek zamanlı yüz tanıma(doğrulama) sistemi şeklinde bir yaklaşım geliştirmişlerdir. İlgili yaklaşımda; yüzlerin normal koşullarda dik pozisyonda olmasından dolayı küçük bir 2B karakteristik görünüm kümesi ile tanımlanabildikleri gerçeğinden yararlanarak yüz tanıma, iki boyutlu bir nesne tanıma problemine indirgenerek çözümlenmiştir.

Bilinen yüz görüntüleri arasındaki farkı çözümleme kabiliyeti en yüksek olan kodlardan oluşan bir özellik alanına ("face space") yansıtmışlardır. Bu yüz alanını, "eigen faces" olarak tanımlamışlardır. Bunların, aynı zamanda yüz setlerinin eigen vektörleri olduğu ve gözler, kulaklar ve burun gibi izole edilmiş özelliklere karşılık gelemeyeceklerini belirtmişlerdir. Yapılan çalışmaların sonucunda, denetimsiz (unsupervised) bir şekilde öğrenme yeteneğinin sağlandığı sonucuna varmışlardır.

Belhumeur ve arkadaşları [16] çalışmalarında, aydınlatma yönündeki ve yüz ifadesindeki büyük değişikliklere duyarlı olmayan bir yüz tanıma algoritması geliştirmişlerdir. Desen sınıflandırma yaklaşımı kullanılarak görüntüdeki her piksel yüksek boyutlu bir uzayda koordinat değeri olarak değerlendirilmiştir. Eğer yüz gölgesiz bir Lambertian yüzeyi ise bu durumda farklı bir ışıklandırma sistemi altındaki sabit pozlamada ancak belirli yüz görüntülerinin, yüksek boyutlu görüntü uzayının 3B doğrusal bir alt uzayında yer aldığı gözleminden faydalanmışlardır. Ancak yüzlerin gerçekte Lambert yüzeyleriyle tam olarak uyuşmadığından ve gerçekte kendi kendisi üzerinde gölgeleme ürettiğinden, görüntülerin bu doğrusal alt alandan saptığını tespit etmişlerdir. Bu sapmanın açıkça modellenmesinden ziyade yüzün bu bölgelerini büyük sapmalara indirgeyecek şekilde görüntüyü bir alt uzaya doğrusal olarak yansıtmışlardır. Projeksiyon yöntemi olarak Fisher'ın Linear Discriminant'ı yöntemini temel almışlar ve aydınlatma ve yüz ifadelerinde ciddi değişiklikler olsa bile düşük boyutlu bir alt alanda bulunan, iyi biçimde gruplanmış sınıflar oluşturmuşlardır. Görüntü alanını düşük boyutlu bir alt alana doğrusal olarak yansıtmaya dayanan başka bir yöntem olan Eigenface tekniğinin de benzer hesaplama gereksinimlerine sahip olduğu sonucuna varmışlardır. Fakat yapmış oldukları kapsamlı deneysel sonuçlar, önerilen "Fisherface" yönteminin, Harvard ve Yale Yüz Veritabanı sistemleri üzerinde yapılan testler için Eigenface tekniğinden daha düşük hata oranına sahip olduğu sonucuna varmışlardır.

Tian ve arkadaşları [17] çalışmalarında neredeyse önden görünüşlü bir yüz görüntüsü dizisindeki kalıcı yüz özelliklerine (kaşlar, gözler, ağız) ve geçici yüz özelliklerine (yüz üzerinde oluşan kırışıklıklar) dayanan yüz ifadelerini analiz etmek için otomatik bir yüz analiz (Automatic Face Analysis- AFA) sistemi geliştirmişlerdir. AFA sisteminin, yüz ifadesindeki ince kırışıklıklarda meydana gelen değişiklikleri, birkaç prototipik ifadenin yerine yüz eylem kodlama sisteminin (Facial Action Coding System-FACS) eylem birimlerine (action units-AUs) dönüştürdüğünü belirlemişlerdir. Dudaklar, gözler, kaşlar, yanaklar ve kırışıklıklar dahil

olmak üzere çeşitli yüz özelliklerinin izlenmesi ve modellenmesi için çok aşamalı yüz ve yüz bileşen modelleri önermişlerdir. İzleme sırasında, yüz özelliklerinin ayrıntılı parametrik tanımlamalarını ortaya koymuşlardır. Sistem genelleştirmesinin sağlanabilmesi için farklı araştırma ekipleri tarafından toplanan bağımsız görüntü veri tabanları kullanmış olup, temel gerçekliğin test edilmesinde FACS kodlarından yararlanmışlardır. Sonuç olarak, otomatik yüz ifadesi analiz sistemlerinin çoğunun; mutluluk, öfke, sürpriz ve korku gibi küçük bir prototipik ifade setini tanımaya çalıştığı ifade etmişlerdir. Bununla birlikte, bu gibi prototipik ifadelerin oldukça seyrek görüldüğü dile getirilmiştir. İnsani duygu ve niyetlerin, genellikle bir veya birkaç ayrı yüz özelliğindeki değişikliklerle yansıtılabileceği sonucuna varmışlardır.

Sim ve arkadaşları [18] yaptıkları çalışmalar ile 2000 yılının sonbaharında, 68 kişiden 40.000'den fazla yüz imgesinin bulunduğu bir veritabanı meydana getirmiştirler. Carnegie Mellon Üniversitesi 3D Odası kullanılarak, her kişi 13 farklı poz, 43 farklı aydınlatma koşulu altında ve dört farklı ifadeyle görüntülenmiştir. Buna CMU Poz, Aydınlatma ve İfade (Pose, Illumination, and Expressio- PIE) veritabanı ismi verilmiştir. Çalışma sonucunda; görüntüleme donanımına, toplama prosedürüne, görüntülerin düzenine, birkaç olası kullanıma ve veritabanının nasıl elde edileceğine dair bilgiler elde edilmiştir.

Viola ve Jones [19] çalışmalarında görüntüleri son derece hızlı bir şekilde işleyebilen ve yüksek saptama oranlarına ulaşabilen, fiziksel nesne algılaması için bir makine öğrenme yaklaşımında bulunmuşlardır. Bu çalışmalar üç temel katkı ile ayırt edilmiştir. İlk olarak, dedektör ile bulunan özelliklerin hızlı bir şekilde hesaplanmasını sağlayan "Integral linage" adı verilen yeni bir imge gösterimi tasarlanmıştır. Ardından, daha büyük bir kümeden küçük sayıda kritik görsel özellik seçebilen ve son derece etkili bir sınıf verimi sağlayan AdaBoost'a dayalı bir öğrenme algoritması geliştirilmiştir. Son olarak, nesne benzeri bölgeler üzerinde hesaplama yaparken, imgenin arka plan bölgelerinin hızla atılmasını sağlayan bir "cascade" içinde giderek daha karmaşık sınıflandırıcıların birleştirilmesine yönelik bir yöntem uygulaması yapılmıştır. Cascade'in önceki yaklaşımlardan farklı olarak, atılan bölgelerin ilgilenilen hedefi içerme olasılığı düşük olan istatistiksel garantiler sağlayan spesifik bir dikkat odağı mekanizması olarak görülebildiği tespit edilmiştir. Yapılan çalışma sonucunda yüz algılama alanında, sistem tespit oranları karşılaştırılabilir olarak verilmiştir. Gerçek zamanlı uygulamalar kapsamında kullanıldığında, dedektör görüntü farkına veya ten rengini algılamaya başvurmadan saniyede 15 kare hızında çalıştığı ortaya konmuştur.

Ahonen ve arkadaşları [20] yaptıkları çalışmalarında yerel ikili desen (local binary pattern - LBP) doku özelliklerine dayanan yeni ve etkili bir yüz görüntü sunumu ortaya koymuşlardır. Yüz görüntüsü, yüz tanımlayıcısı olarak kullanılacak gelişmiş bir özellik vektörü kontrolünde ayıklanmış ve birleştirilmiş olup LBP özellik dağılımları kullanılarak birkaç bölgeye bölünmüştür. Sonuç olarak önerilen yöntemin performansı hakkında, yüz tanıma konusunda farklı zorluklarla karşılaşıldığı sonucuna varılmıştır.

Lucey vd. [21] yaptıkları çalışmada 2000 yılında, bireysel yüz ifadelerini otomatik olarak tespit etmeye yönelik araştırmaların teşviki doğrultusunda yayınlanan Cohn-Kanade (CK) veri tabanını incelemişlerdir. CK veritabanının hem AU hem de duygu tespiti için kullanıldığını ancak kıyaslama algoritmalarıyla karşılaştırıldığında eksikleri olduğunu belirtmişlerdir. Orijinal veritabanının rastgele alt kümelerinin kullanımının, meta-analizleri zorlaştırdığı ifade edilmiştir. Bu ve diğer kaygıları gidermek için genişletilmiş Cohn-Kanade (CK +) veritabanı geliştirilmiştir. Bu veritabanında sıralama sayısının %22, proje sayısının %27 artmış olduğu görülmüştür. Her sekans için hedef ifadesi tamamen FACS kodlu yapılmış, duygu etiketleri revize edilmiş ve doğrulanmıştır. Buna ek olarak, çeşitli gülüş tipleri ve bunlarla ilişkili metaveriler için pozlanmamış diziler eklenmiştir. Yapılan çalışma sonucunda elde edilen Aktif Görünüm Modelleri (AAM'ler) ve hem AU hem de pozlanmış veriler için duygu tespiti yapmak üzere bir dizi destek vektör makine (DVM) sınıflandırıcısı kullanılarak temel sonuçlar açıklanmıştır.

Ofislerde, bilgisayarla iş yapan kişilerin duygu durumlarının tespit edilebilmesi, bu kişilerin moral, motivasyon durumu ile işteki performansına dair anlamlı veriler sağlayabilmektedir. Bu fikirden hareketle, Ayvaz ve Gürüler [22] çalışmalarında; bilgisayar kullanıcısının yüz ifadelerinden anlık duygusal değişim tespitini gerçekleştiren prototip bir sistem geliştirmişlerdir. Çalışma sonucunda, doğrusal olmayan DVM algoritması ile elde edilen yüksek doğruluk değerinin, belirlenen öznitelik verileri için oldukça ayırt edici olduğu ve duygusal ifadeleri sınıflandırmak maksadıyla bu öznitelik verilerine dayalı kural tabanlı bir algoritmanın kullanılabileceği yönündeki savları destekler nitelikte olduğu sonucuna varılmıştır. Bu yapılan çalışma ile farklı öznitelik verilerinin ve sınıflandırıcı yöntemlerin araştırılması için açık kapı bırakılmıştır.

Zubair ve arkadaşları [23] çalışmalarında bir makine vizyonu (MV) yaklaşımı kullanılarak duygu temelli yüz ifadesi tanıma çerçevesi önermişlerdir. Çalışmalarında yerel

anket Bahawalpur şehir veri setinden mutlu, üzgün ve kızgın 3 sınıfa bölünmüş yüz duygu veri seti oluşturmuşturlar. Toplam 600 boyutta görüntüyü (256 x 256) bir gri seviye formatına dönüştürmüşler ve parazit giderimi için bir medyan filtre kullanmışlardır. Doku, histogram ve ikili özellikler olarak adlandırılan toplam 45 istatistiksel özellik belirlemişlerdir. Korelasyona dayalı özellik bölümü tekniğini kullanarak bu özellikleri optimize etmişlerdir. MV sınıflandırıcılarından olan rastgele orman (RF), lojistik (Lg) ve J48'den yararlanarak optimize edilmiş veri seti ile değerleri sırasıyla %96,33, %95,67 ve %95,33 olan son derece umut verici doğruluk analizi sonuçları elde etmişlerdir.

Battini Sönmez [24] çalışmasında, az sayıda örnekle CNN kullanımı açık bir sorun olduğundan 2 boyutlu yüzlerin küçük veritabanlarında otomatik yüz ifadesi tanıma konusuna odaklanmaktadır. Klasik makine öğrenimi yaklaşımını takiben, farklı özellik çıkarma ve sınıflandırıcı kombinasyonlarını değerlendirmiştir. Ayrıca ilgili kombinasyonların performanslarını özel tasarlanmış CNN ile karşılaştırmıştır. Çalışmasında, CNN'nin "yakın sistem" deneyinde diğer sınıflandırıcılardan daha iyi performans gösterdiği sonucuna ulaşmıştır. Ayrıca daha zorlu "açık sistem" deneysel kurulumunda Seyrek Temsil tabanlı Sınıflandırıcının daha başarılı olduğunu dile getirmiştir.

Thacker ve Makwana [25] çalışmalarında duyguların davranışlarımızla güçlü bir ilişkisi olduğunu, insan duygularının bazı önemli anlamlara sahip olan, iç veya dış olaylara yönelik ortaya konan farklı tepkiler olduğunu belirtmişlerdir. Neşe, üzüntü, öfke, korku, şaşkınlık, nefret ve doğal durumu içeren yedi temel duygudan bahsetmişlerdir. FER sistemlerinde yeterli aydınlatma ve baş duruşu gibi diğer sorunlara rağmen duyguları tanımlamak için eğitim verisinin eksik olduğunu öne sürmüşlerdir. Bu makalelerinde, farklı veri kümeleri ile kullanılan farklı sinir ağı algoritmalarını ve verimlilik sonucunu içeren derin öğrenme yöntemleriyle yüz ifadesi tespitinin kapsamlı bir öğrenimini sunmuşlardır. Ayrıca, derin öğrenmeyi kullanarak sağlam bir FER geliştirmek için bu alandaki mevcut zorlukları ve fırsatları ortaya koymaya çalışmışlardır. Makalelerinde, önemli yüz özelliklerini belirtmek için kullanılan farklı yüz ifadesi tanıma teknikleri ve mimarileri hakkında bir anket sunmuşlardır. Yüz ifadesi tanımada kullanılan farklı veri setlerine dair detaylı bilgiler istenilen şekilde açıklanmıştır. Diğer araştırmacıların mevcut yöntemlerin sorunlarının üstesinden gelmesine ve sonuçları doğruluk açısından iyileştirmesine yardımcı olacak, karşılaştırmalı son özellik çıkarma tekniklerini ve son zorlukları ele almışlardır.

Abdulsalam ve arkadaşları [26] çalışmalarında yüzdeki duyguyu tanımanın arka planını açıklamaktadırlar. Bu bağlamda, görsel yöntemi kullanarak duygusal ifadeyi tanımlamanın yollarını araştırmaktadırlar. Çalışmaları, performans değerlendirmesinde kullanılan kamuya açık bazı veri kümelerini içermektedir. 2013'ten 2018'e kadar olan yıllarda duyguları görsel modelleme kullanarak sınıflandırmaya yönelik bazı araştırmaların bir özetini tablo halinde sunmuşlardır. Bazı araştırmacıların bireysel olarak kullandığı, özellik çıkarma ve sınıflandırma için kullanılan birçok farklı teknik olduğunu gösterdiğine ve diğerlerinin, birden fazlasından fayda sağlamak için bu tekniklerin bir kombinasyonunu kullandıklarını ifade etmişlerdir. Bu alanda tanımlanmış birleşik yöntem olmadığının ve son araştırmalardaki eğilimin, derin öğrenmede özellikle CNN kullanımına yönelik olduğu sonucuna varmışlardır.

3. MATERYAL VE METOT

Yüz ifadesinden duygu tanımada bilinmesi gereken temel kavram görüntü kavramıdır. Görüntü, bir boyutu görüntünün yüksekliği, diğer boyutu ise görüntünün genişliği olan iki boyutlu bir fonksiyon olarak tanımlanabilmektedir [27]. Bu fonksiyonun matematiksel karşılığı iki boyutlu bir matristir. 1920x1080 boyutlarındaki bir görüntü, 1920 sütundan ve 1080 satırdan oluşan bir matris şeklinde matematiksel olarak ifade edilebilmektedir.

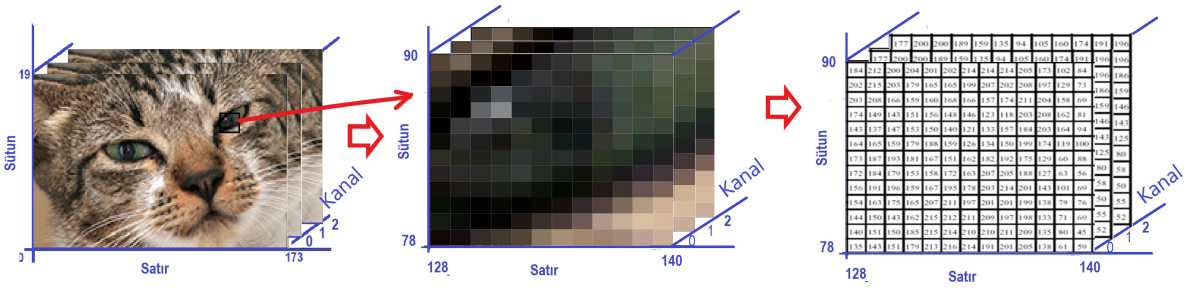
En temel görüntü pikselleri sadece siyah ya da beyazdan oluşan ikili (binary) görüntüdür. Gri tonlamalı görüntülerde ise pikseller, siyah için '0' ve beyaz için '255' değerleri arasındaki gri tonları da görsel olarak ifade edebilmektedir. Renkli bir görüntü, kırmızı, yeşil ve mavi (RGB) şeklinde üç adet katmandan meydana gelmektedir. Ayrıca renkli görüntü kavramı, üç boyutlu bir matris olarak da ifade edilmektedir [1] .

İkili (binary) resimde, her bir piksel 1 bit yer kaplamaktadır. Gri tonlamalı resimlerde ise genellikle her piksel 8 bit yer kaplarken; bu değer renkli resimlerde, her renk kanalı için 8 bit, toplamda 24 bit olarak karşılık bulmaktadır.

Görüntü işleme; herhangi bir görüntü yakalama aygıtı aracılığıyla oluşturulmuş görüntüler üzerinde görüntünün netliğini artırma, görüntü üzerinde belirli özellikleri vurgulama veya değiştirme, görüntüdeki herhangi bir nesneyi tespit etme gibi işlemler yapabilmeyi sağlayan tekniğe verilen isimdir. Şekil 3.1'de herhangi bir imgenin görüntü işleme teknikleri ile işlenebilmesi için sayısallaştırılması görülmektedir. Sayısallaştırma, görüntü içerisindeki renkleri ve resimleri oluşturan piksel olarak adlandırılan her bir noktanın sayısal değerlerle ifade edilmesi şeklinde tanımlanmaktadır [28].

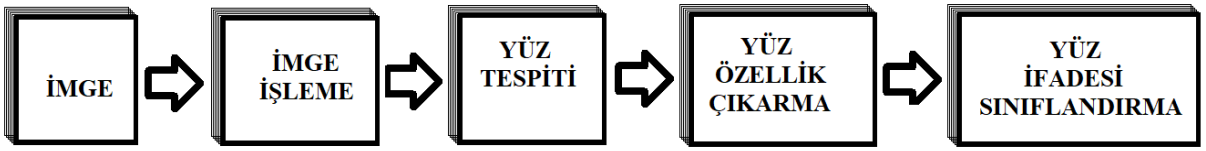
Görüntünün sayısallaştırılması ile matematiksel olarak 2 boyutlu (satır,sütün) bir matris elde edilmektedir. Renkli görüntünün sayısallaştırılmasından satır, sütün ve kanal boyutlarından oluşan 3 boyutlu bir matris elde edilmektedir. Bu matrislerle görüntü işleme teknikleri uygulanarak imge iyileştirme, imge pekiştirme, görüntüden nesne tespiti gibi işlemler yapılabilmektedir.

Bir yüz ifadesinden duygu analizi yapmak için işlenecek görüntünün sayısallaştırılmasından sonra ilk aşama imgedeki yüz bölgesinin tespit edilmesidir. Sonraki aşama, yüzdeki



Şekil 3.1. Görüntünün sayısallaştırılması

ağız, burun, göz, kaş gibi bölümlerin farklı hareketlerinde meydana gelen yüz ifadelerinin tespit edilmesine yardımcı olacak temel noktaların belirlenmesi ve belirlenen noktaların analizi yapılarak, yüz özniteliklerinin çıkarılmasıdır. Son aşama ise sınıflandırma yöntemleri kullanılarak, yüzdeki duygusal ifadeyi tanıma işleminin gerçekleştirilmesidir. Şekil 3.2'de bu aşamalar gösterilmektedir.



Şekil 3.2. Yüz ifadelerinden duygu tanıma diyagramı

3.1 Yüz Tespiti İçin Kullanılan Yöntemler

Yüz tespiti işlemi, bir görüntü içerisinde bulunan yüz bölgelerinin belirlenmesidir. Yüz tespitinde kullanılan çeşitli yöntemler bulunmaktadır. Bunlardan bazıları aşağıda incelenmiştir.

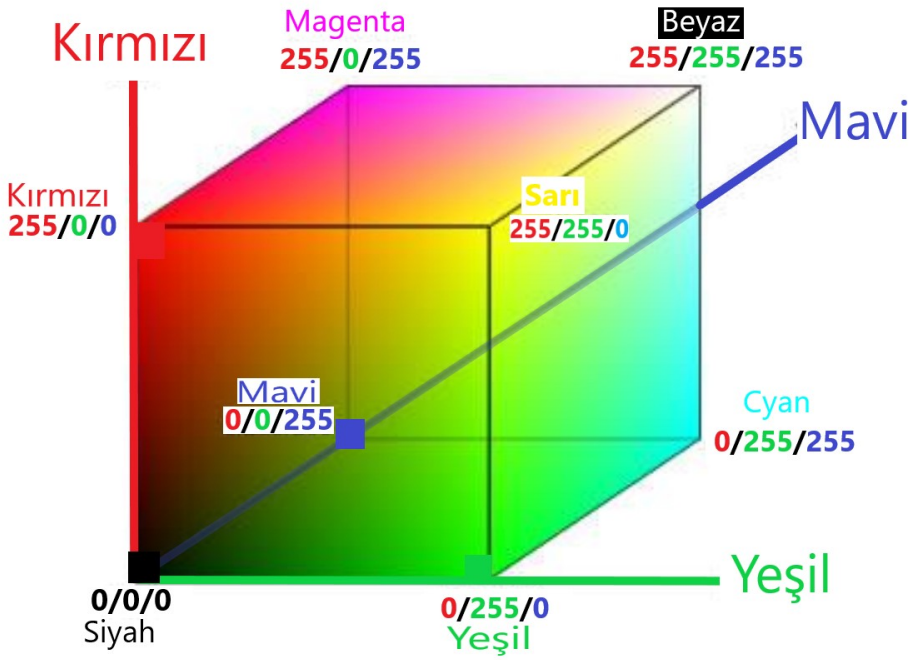
3.1.1 Renk Uzayı Yöntemi

Yaygın olarak kullanılan ve mevcut çalışma kapsamında irdelenen renk uzayları RGB, HSV ve YCrCb renk uzaylarıdır.

3.1.1.1 RGB Renk Uzayı

RGB ismi, bütün renklerin elde edildiği en temel üç renk olan Red, Green, Blue (Kırmızı,Yeşil,Mavi) ana renklerinin baş harfleri ile oluşturulmuştur. Günlük hayatta da en yaygın olarak kullanılan renk uzayıdır. Cilt tespiti için çok uygun değildir.

Bütün renkler kırmızı, yeşil ve mavi rengin 0-255 değerleri arasında belirli oranda alınması ile elde edilmiştir. Örneğin, kırmızı rengi elde etmek için kırmızı renk değeri ’255’ , yeşil renk değeri ’0’ ve mavi renk değeri ’0’ şeklinde olmak zorundadır. Bütün renk değerleri ’0’ alındığında siyah renk, ’255’ alındığında ise beyaz renk oluşmaktadır. Ara renklerden sarı rengi elde etmek için kırmızı ve yaşil renk değerleri ’255’ ve mavi renk değeri ise ’0’ olarak seçilmektedir. Şekil 3.3’te RGB renk uzayı gösterilmektedir.



Şekil 3.3. RGB renk uzayı

3.1.1.2 HSV Renk Uzayı

HSV; Hue, Saturation ve Value kelimelerinin baş harflerinden oluşmaktadır. Burada Hue (renk tonu) renk uzayıda 0-360 değerleri aralığında renklerinin ayarlandığı kısımdır. OpenCV'de ise 0-180 değer aralığında seçilmektedir. Bu değer aralığında baskın renk (Red,Yellow, Green, Cyan,Blue, Magenta) tanımlanmaktadır.

Saturation (doygunluk) değeri ile siyah için %0 ile beyaz için %100 arasındaki renklerin grilik düzeyi ayarlanmaktadır. Value (değer) doygunluk ile birlikte çalışmakta olup rengin parlaklığını veya yoğunluğunu %0 ile %100 arasında ayarlanmasını sağlamaktadır.

RGB renk uzayından HSV renk uzayına geçiş için aşağıdaki 3.1, 3.2 ve 3.3 formülleri kullanılmaktadır [29] :

$$
\begin{aligned}
Hue \in \{0,360\}\, Saturation, Value, Red, Green, Blue \in \{0,1\} \\
max = max(Red, Green, Blue), min = min(Red, Green, Blue) \\
\text{Hue=tanimsiz, Eger max=min} \\
Hue = 60\frac{Green - Blue}{max - min} + 0, \text{ Eger max=Red ve Green>=Blue ise} \\
Hue = 60\frac{Green - Blue}{max - min} + 360, \text{ Eger max=Red ve Green<Blue ise} \\
Hue = 60\frac{Blue - Red}{max - min} + 120, \text{ Eger max=Green ise} \\
Hue = 60\frac{Red - Green}{max - min} + 240, \text{ Eger max=Blue ise}
\end{aligned}
\tag{3.1}
$$

$$
\begin{aligned}
Saturation = 0, \text{ Eger max=0 ise} \\
Saturation = 1 - \frac{min}{max}, \text{ Eger max=0 değilse}
\end{aligned}
\tag{3.2}
$$

$$Value = max \tag{3.3}$$

3.1.1.3 YCrCb Renk Uzayı

Burada Cb mavi fark kroma ve Cr kırmızı fark kroma bileşenleridir. Parlaklık olan Y ise, gama düzeltmeli RGB renklerine göre doğrusal olmayacak şekilde kodlanmış ışık yoğunluğu anlamına gelmektedir. YCbCr, kabaca kırmızı, yeşil ve maviye karşılık gelen ana renklerin algısal olarak anlamlı bilgilere dönüştürüldüğü pratik bir yaklaşımdır. Pratik bir kullanımına örnek verilecek olursa insanlar siyah-beyaz aralığındaki renk değerlerine daha duyarlı olduklarından dolayı "siyah-beyaz" ile karşılaştırıldığında "renkli" ye tahsis edilen bant genişliğini veya çözünürlüğünü azaltmaktır.

RGB renk uzayından YCrCb renk uzayına geçiş için 3.4'te verilen formül kullanılmaktadır.

$$\begin{aligned} Y &= 0.299xRed + 0.114xBlue - 0.587xGreen \\ Cr &= Red - Y \\ Cb &= Blue - Y \end{aligned} \tag{3.4}$$

3.1.1.4 Renk Uzayı İle Cilt Tespiti

Bu yöntemin ilk amacı, kişilerin görüntüsünün cilt bölgesini çıkarmaktır. Bunun için HSV ve YCrCb renk modelini kullanılmıştır.

Cilt renk tonunu tespit etmek için aşağıdaki formül (3.4) ve (3.5) temel alınarak iki parametre, yani x ve y tanımlanmıştır [30].

$$x = 0.148H - 0.291S + 0.439V + 128 \tag{3.5}$$

$$y = 0.439H - 0.368S - 0.071V + 128 \tag{3.6}$$

Görüntüdeki bir pikselin bir cilt pikseli olarak tespit edilebilmesi için x, y ve H parametre değerlerin 140 <y <165, 140 <x <195 ve 0.02 <H <0.1 eşitsizlikleri karşılaması

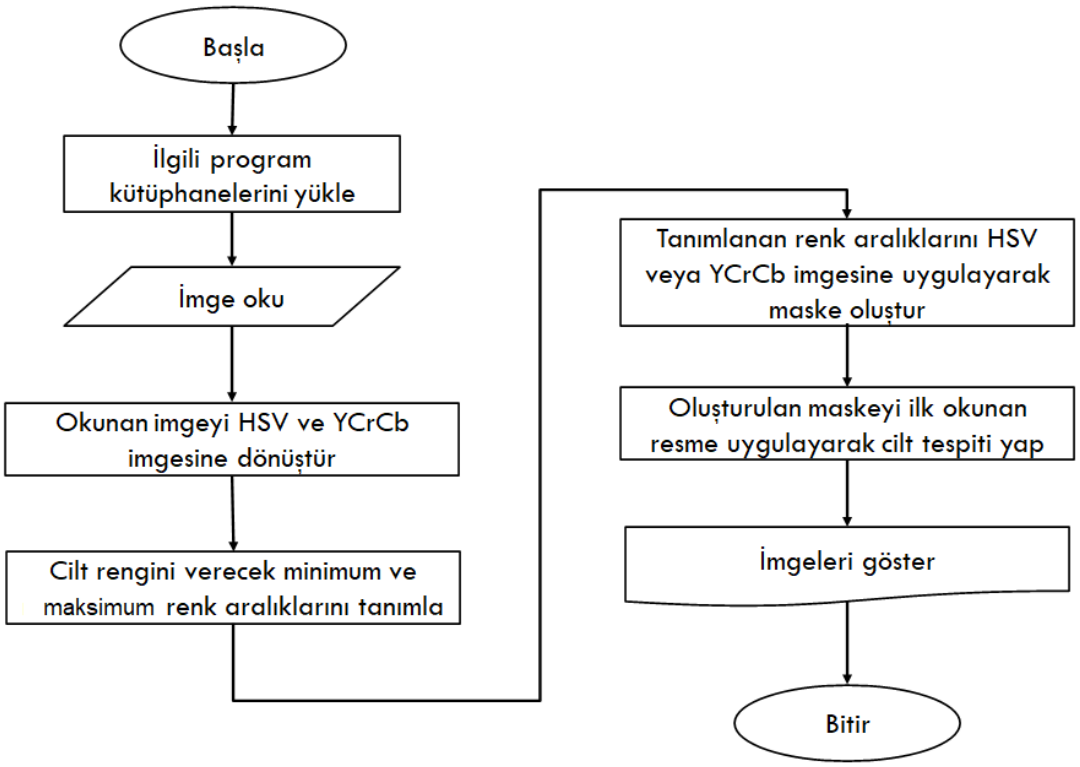
gerekmektedir. Cilt bölgesi tespiti tamamen renk değeri eşleşmesine dayandığından, yüz ve boyun bölümü dışında görüntüde bulunan vücudun diğer kısımları da cilt bölgelerine dahil edilerek yüz bölgesi, tespit edilmiş olur.

Parlaklık ve renklilik bileşenlerinin dönüşümünün kolay olması ve açık bir şekilde ayrılabilmesinden dolayı, YCrCb renk uzayı da ten rengi modellemesinde kullanılmaktadır.

```
    # Resim okunuyor..
    imge=cv2.imread("binsanlari.jpg")
    # HSVrenk uzayi ile...
    minHSV = np.array([0, 70, 90], dtype = "uint8")
    maxHSV = np.array([20, 255, 255], dtype = "uint8")
    imgeHSV = cv2.cvtColor(imge, cv2.COLOR_BGR2HSV)
    skinRegionHSV = cv2.inRange(imgeHSV, minHSV, maxHSV)
    skinHSV = cv2.bitwise_and(imge, imge, mask=skinRegionHSV)
    # YCrCb renk uzayi ile...
    minYCrCb = np.array([0,143,97],np.uint8)
    maxYCrCb = np.array([235,173,117],np.uint8)
    imgeYCrCb = cv2.cvtColor(imge,cv2.COLOR_BGR2YCR_CB)
    skinRegionYCrCb =cv2.inRange(imgeYCrCb,minYCrCb,maxYCrCb)
    skinYCrCb =cv2.bitwise_and(imge,imge,mask=skinRegionYCrCb)
```

Şekil 3.4. Renk uzayı ile yüz tespit programı

Yüz ve cilt bölgesi tespiti için Şekil 3.5'te verilen algoritma kullanılarak oluşturulmuş olan EK-A bölümdeki Şekil A.1'de verilen python programı ve opencv [2], numpy [3] ve matplotlib [5] kütüphaneleri kullanılmıştır. Şekil 3.4.'de verilen python program parçasında minHSV ve maxHSV değeri ile HSV renk uzayı için minYCrCb ve maxYCrCb değeri ile de YCrCb renk uzayı için tespit edilmek istenen cilt renk tonu ayarlanmıştır. Bu renk tonları ile oluşturulan maske ile görüntü maskelenerek cilt bölgeleri tespit edilmiştir. Şekil 3.6'de bilim adamlarının bir arada olduğu resimdeki [31] tepit edilen yüz ve cilt bölgeleri görülmektedir.



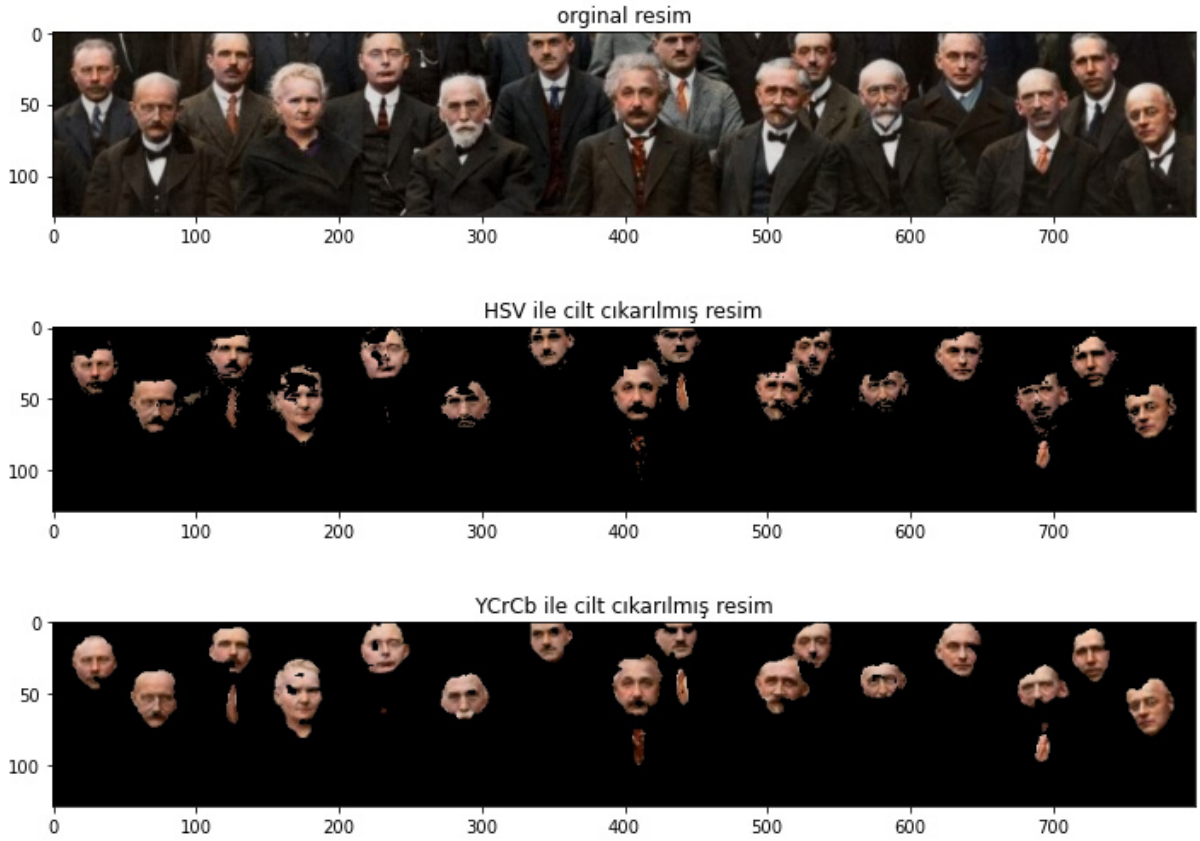
Şekil 3.5. Renk uzayı ile yüz tespit algoritması

3.1.2 Yazılım Algoritması veya Kütüphaneleri Kullanarak Yüz Tespti Yöntemi

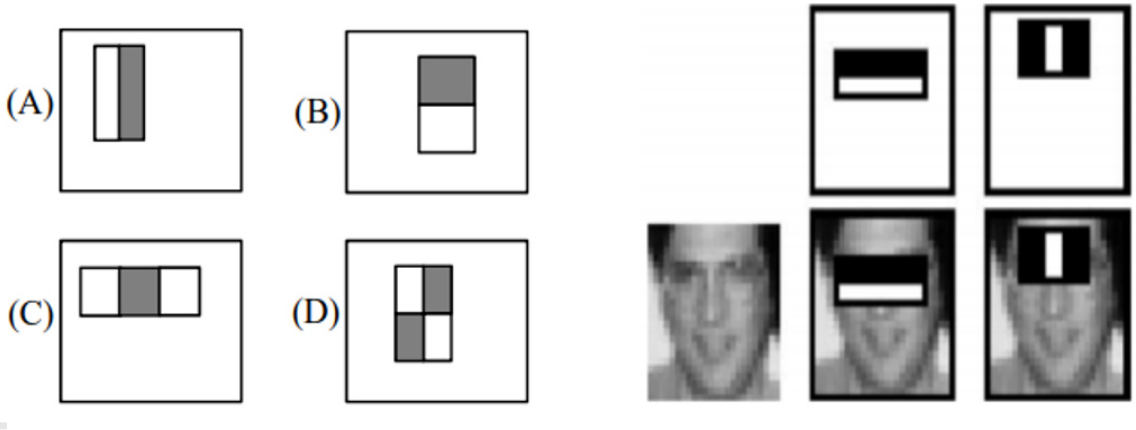
3.1.2.1 Haar Kaskad Sınıflayıcıları

Yüz tespitinde kullanılan çok yüksek performanslı yöntemlerden birisi de Viola-Jones tarafından geliştirilen algoritmadır [19]. Haar özellikli dalgacık şablonlarına benzeyen dikdörtgen çerçeveler ile yüz tespiti yapılmıştır. Zayıf sınıflayıcılar olan bu çerçeveler bir araya gelerek Kaskad Sınıflayıcıları (Haar Cascade Classifier, HKS) oluştururlar. Bu zayıf sınıflayıcılar görüntünün yüz olması muhtemel bölgelerde daha fazla işlem yapılmasını sağlayarak dedektör hızını artırmaktadır. HKS öğrenme temelli bir algoritmaya sahip olduğu için oldukça doğru sonuçlar vermektedir.

Sınıflayıcıları eğitmek için insan yüzü içeren resimler ile içermeyen resimler verilir. Sınıflayıcı verilen resimlerden insan yüzü içeren bölgelerde “1” içermeyen bölgelerde “0” değeri oluşturur. Şekil 3.7’de dikdörtgen özellikler ve bunların yüz görüntüsüne uygulanması



Şekil 3.6. Renk uzayı kullanarak cilt bölgesi çıkarma



Şekil 3.7. Dikdörtgen özellikler ve yüz görüntüsüne uygulanması[19]

görülmektedir. İlk olarak yatay olarak üst siyah ve alt beyaz dikdörtgen cerçeve yüze uygulandığında kaşlar siyah ve onun altında yüz bölgesi daha açık renk olacağından yüz tespiti gerçekleştirilmektedir. Dikey siyah-beyaz-siyah dikdörtgen çerçeve yüz görüntüsüne

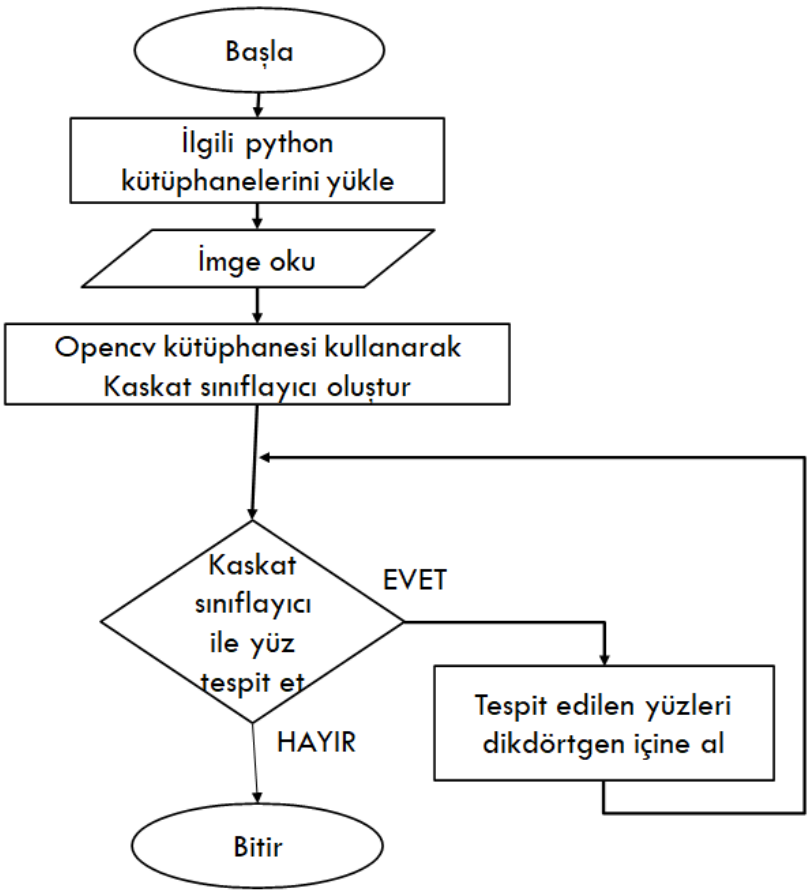
uygulandığında iki kaş ve göz bölgesi siyah ve iki kaş arası ve burun bölgesi beyaz olduğundan eşleşme olmaktadır.

Haar öznitelik vektörleri resim tamamı üzerindeki karanlık ve aydınlık bölge üzerindeki piksellerin toplamlarının farkını alarak ilerlemektedir. İşlemi daha hızlı hesaplamak için görüntüdeki piksel değerlerinin toplamından oluşan "integral görüntü" olarak adlandırılan bir ara görüntü temsili tanımlanmaktadır.

HKS kullanarak yüz tespit etmek için Şekil 3.9'daverilen algoritma kullanılarak oluşturulmuş olan Şekil 3.8'de görülen python programı ve opencv [2], numpy [3], matlotlib [5] kütüphaneleri kullanılmıştır. Program ile bir imgedeki tespit edilen yüz bölgeleri Şekil 3.10'da verilmiştir. Programdaki "cv2.CascadeClassifier ("hksFrontalFaceDefault.xml")" komutu ile insan yüzlerini ön cepheden tanımak için Viola-Jones algoritması ile oluşturulmuş olan xml dosyası çalıştırılmaktadır. Daha sonra "yuzcasc.detectMultiScale" komutu ile seçilen resim taranarak resimdeki yüzler tespit edilmektedir. Bir döngü oluşturularak tespit edilen bütün yüzler dikdörtgen içine alınmaktadır.

```
import cv2
import numpy as np
import matplotlib.pyplot as mplt
yuz_casc=cv2.CascadeClassifier("hksFrontalFaceDefault.xml")
resim=cv2.imread("biliminsanlari1.jpg")
gri_ton = cv2.cvtColor(resim,cv2.COLOR_BGR2GRAY)
yuzler=yuz_casc.detectMultiScale(gri_ton,1.1,4)
for (x,y,w,h) in yuzler:
    cv2.rectangle(resim, (x,y),(x+w,y+h),[0, 255, 0], 2)
mplt.figure(figsize=(12, 9))
mplt.imshow(cv2.cvtColor(resim, cv2.COLOR_BGR2RGB))
mplt.show()
```

Şekil 3.8. HKS İle Yüz Tespiti Programı

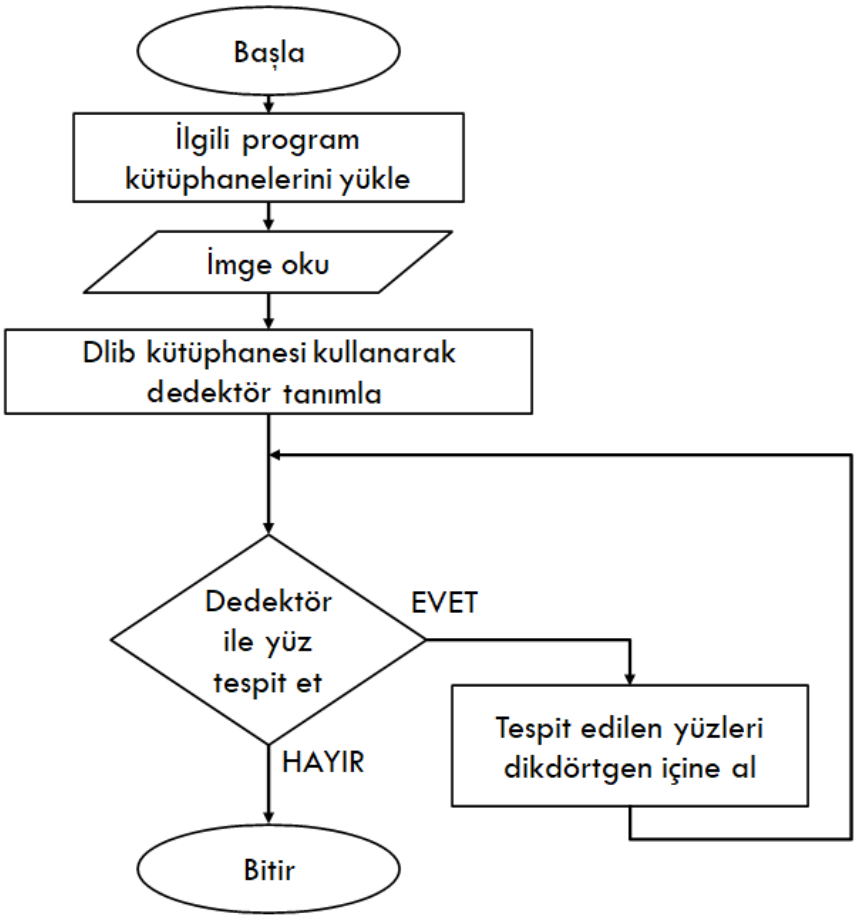


Şekil 3.9. HKS ile yüz tespit algoritması

Şekil 3.10. HKS İle Yüz Tespiti

3.1.2.2 Dlib kütüphanesi

Dlib, makine öğrenmesi (machine learning) ve yapay öğrenme algoritmalarını içeren C++ tabanlı bir kütüphanedir. Bu kütüphane ile yüz tanıma (face recognition) işlemi, yüz tespiti (face detection) işlemi, yüz modelleme (facial landmarks) işlemi gibi uygulamalar kolaylıkla gerçekleştirilebilmektedir.



Şekil 3.11. Dlib kütüphanesi ile yüz tespit algoritması

Dlib, Python programlama dili ile de kullanılabilmektedir. Bu calışmada yapılan uygulamada dlib python kütüphanesi kullanılmıştır.

Şekil 3.11'de verilen algoritma kullanılarak python programı Dlib [6] ve openvc [2] kütüphaneleri kullanılarak oluşturulmuştur. Program Ekler bölümünde Program A.2'de

Şekil 3.12. Dlib kütüphanesi ile yüz tespiti

verilmiştir. Bu programda oluşturulan detector ile imgedeki yüz bolgeleri tespit edilmiştir . Program ile elde edilen sonuç Şekil 3.12'de görülmektedir.

3.2 Yüz Özellik (Öznitelik) Çıkarma

Yüz tespitinden sonraki aşama yüz özelliklerinin çıkarılmasıdır. Yüzdeki kalıcı öznitelikler olan göz, kaş, burun, ağız, yüz çizgilerinin tespit edildiği aşamadır. Yüz özniteliklerini çıkarmak için özellik tanımlayıcıları kullanılmaktadır.

Özellik tanımlayıcı, yararlı bilgileri alarak ve gereksiz bilgileri eleyerek bu şekilde görüntüyü basitleştirerek oluşturulmuş bir görüntünün temsilidir. Bu işlemi yapabilmek için bir görüntü içerisindeki gerekli olan ve olmayan bölümlerin bilinmesi gerekmektedir.

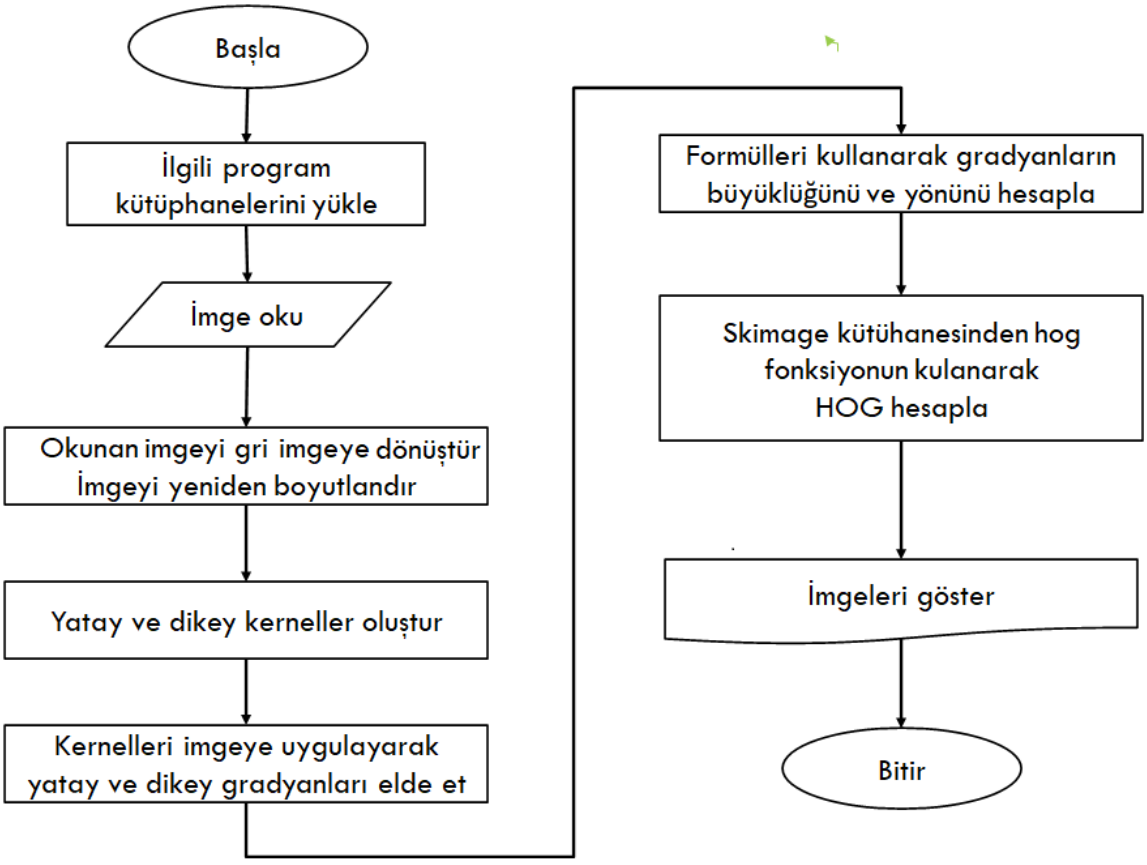
Bir özellik tanımlayıcısı ile genişlik x yükseklik x 3 (kanallar) boyutundaki bir görüntü, n uzunluklu bir özellik vektörüne ya da dizisine dönüştürülmektedir.

3.2.1 Yönlendirilmiş Gradyan Histogramı - HOG

HOG (Histogram of Oriented Gradient - Yönlendirilmiş Gradyan Histogramı), görüntü verilerinden öznitelikleri çıkarmak için sıklıkla kullanılan bir özellik tanımlayıcısıdır. Nesne algılama ve bilgisayarla görme görevlerinde yaygın olarak kullanılmaktadır.

HOG tanımlayıcısı, bir nesnenin yapısına veya şekline odaklanır. Kenar özellikleri yöntemleriyle yalnızca pikselin bir kenar olup olmadığı belirlenmektedir. HOG, kenarların gradyanını ve yönünü çıkarma işlemi yaparak kenarları belirleyebildiği gibi kenar yönünü de belirleyebilmektedir.

HOG ile bölgelerin her biri için ayrı ayrı histogramlar oluşturulmaktadır. Histogramlar, piksel değerlerinin gradyanları ve yönleri kullanılarak oluşturulduğundan bu yönteme "Yönlendirilmiş Gradyanların Histogramı" adı verilmiştir. HOG işlem adımlarını gösteren bir algoritma Şekil 3.13'te verilmiştir.

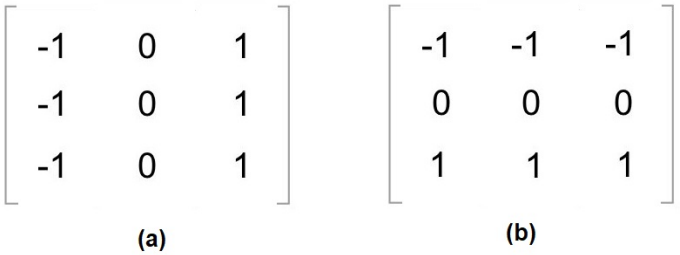


Şekil 3.13. HOG algoritması

Yönlendirilmiş Gradyanların Histogramı'nı hesaplayabilmek için ilk olarak işlenmek istenen görüntünün sabit bir en-boy oranına sahip olması sağlanmaktadır. Hesaplama kolaylığı için 1:2 en-boy oranı seçilebilir.

İkinci adımda Bir HOG tanımlayıcısını hesaplamak için önce yatay ve dikey gradyanların hesaplanması gerekmektedir. Görüntüyü Şekil 3.14'deki kernellerle filtreleyerek

gradyanlar kolayca elde edilmektedir. Yatay ve dikey gradyanların elde edildiği örnek bir python programı Şekil 3.15'de gösterilmiştir. Elde edilen imgeler Şekil 3.16 da verilmiştir.



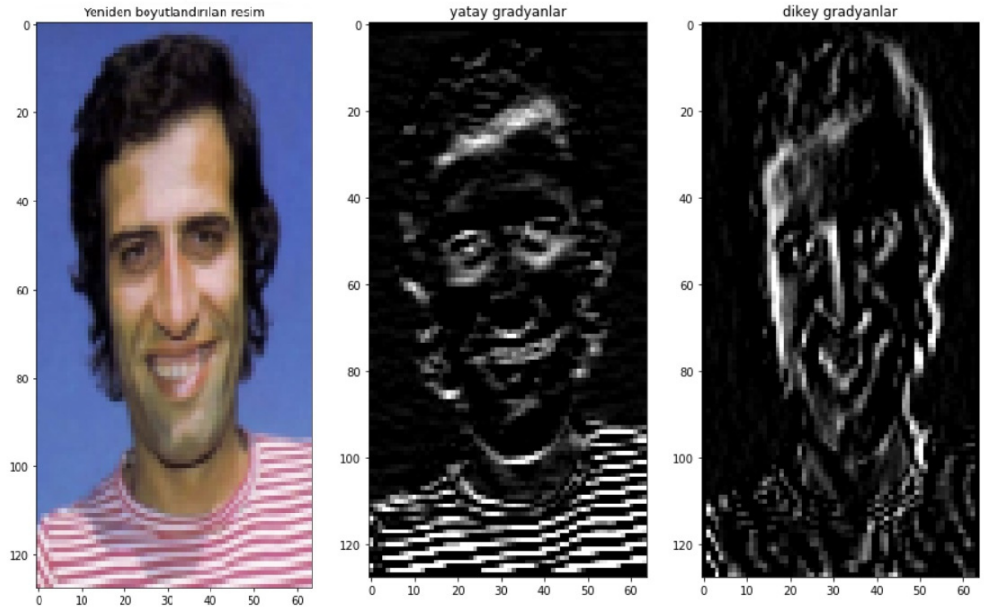
Şekil 3.14. Gradyan kernellleri: a) Yatay , b) Dikey

```python
import matplotlib.pyplot as mplt
import numpy as np
import cv2
 # imge okuma
imge = cv2.imread('ksunal.jpg')
imge_gri=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
imge = cv2.resize(imge_gri, (64,128))

yatayKernel=np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
dikeyKernel=np.array([[-1,0,1],[-1,0,1],[-1,0,1]])

 # Yatay ve dikey gradyanlarin elde edilmesi
yatayGradyan=cv2.filter2D(imge,-1,yatayKernel)
dikeyGradyan=cv2.filter2D(imge,-1,dikeyKernel)

fig = mplt.figure(figsize=(15,10))
ax1 = mplt.subplot(121)
mplt.imshow(yatayGradyan,cmap=mplt.cm.gray)
mplt.title('yatay gradyanlar')
ax2 = mplt.subplot(122)
mplt.imshow(dikeyGradyan,cmap=mplt.cm.gray)
mplt.title('dikey gradyanlar')
```

Şekil 3.15. Yatay ve dikey gradyanların elde edilmesi python programı

Şekil 3.16'da görüleceği üzere yatay gradyanlar hesaplandığında resimdeki yatay kenarlar belirginleşirken dikey kenarlar kaybolmaktadır. Aynı şekilde dikey gradyanlar hesaplandığında ise dikey kenarlar belirginleşirken yatay kenarlar kaybolmaktadır.

Formül 3.7'yi kullanarak gradyanın büyüklüğü ve yönü hesaplanabilmektedir. Şekil 3.17'deki python programı kullanılarak gradyanların yönü ve büyüklüğü hesaplanmıştır. Program ile elde edilen görüntüdeki gradyanların büyüklüğü Şekil 3.18'de bulunan soldaki imgede, gradyanların yönü ise sağdaki imgede verilmiştir. Eğimin yönünü gösteren Şekil



Şekil 3.16. Yatay ve dikey gradyanlar

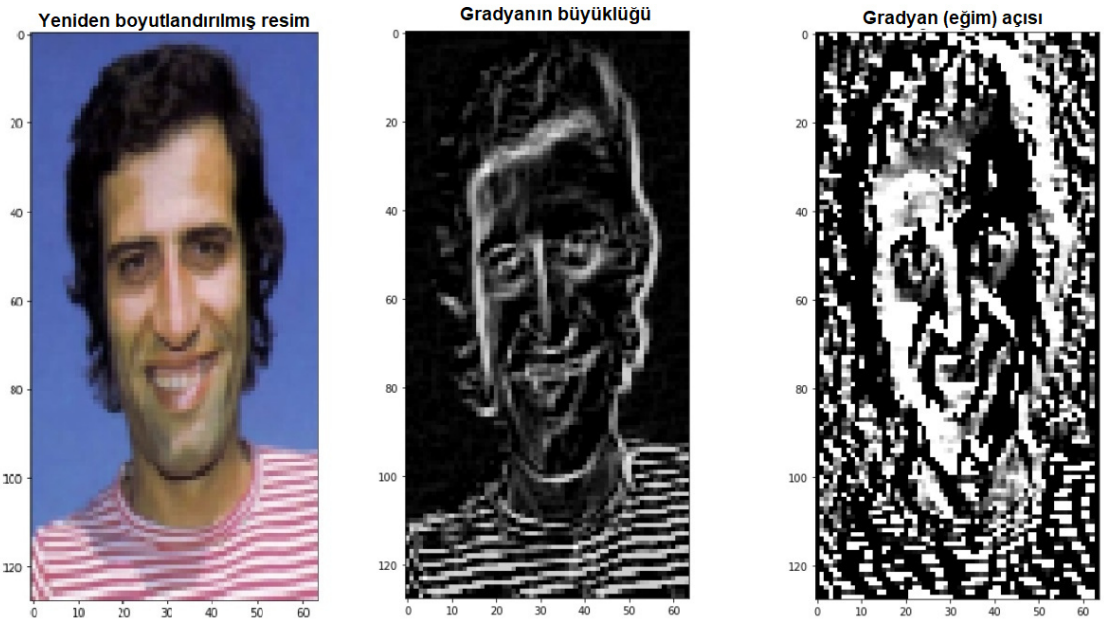
3.18'deki yön (Angle of gradient) imgesine bakıldığında yön olarak belirtilen imgenin önemli bir anlam ifade etmediği görülmektedir. Degrade (Magnitude of gradient) görüntüde ise önemli olmayan birçok bilginin kaldırıldığı, ancak ana hatların vurgulandığı görülmektedir. Bir başka deyişle, gradyan resmine bakılarak imgenin içerisinde bir kişi olduğu kolaylıkla söylenebilir.

$$\begin{aligned} buyukluk &= \sqrt{g_x^2 + g_y^2} \\ acisal_{yon} &= \arctan(\frac{g_y}{g_x}) \end{aligned} \tag{3.7}$$

Üçüncü adımda, görüntü 8 × 8 hücreye bölünmektedir ve her 8 × 8 hücre için bir gradyan histogramı hesaplanmaktadır. 8 x 8 hücreye bölünmesinin nedeni başta 64 X 128 olarak ölçeklendirilen imgedeki dikkat çekici özellikleri (yüz, göz vb.) yakalamak için 8 x 8 hücrelerin yeterince büyük olmasıdır. Her 8 × 8 hücrenin bir histogram için 9 × 1 matrisi olduğunu hesaplanmıştır. Histogram aslında 0, 20, 40, 60... 160 açılarına karşılık gelen 9 adet kutudan (sayılardan) oluşan bir vektördür (veya dizidir). Histagramda negatif değerler kullanılmak istenmediği için 0-360 derece yerine 0-180 derecelik aralık alınmıştır. Bu 9 bin değerinde histogramın hesaplanması bir örnek üzerinden anlatılırsa; bir imgedeki piksel değeri

```
#gradyanin buyuklugunu hesaplama
buyukluk=np.zeros([imge.shape[0],imge.shape[1]])
for i in range (imge.shape[0]):
    for j in range (imge.shape[1]):
        buyukluk[i][j]=np.sqrt(yatayGradyan[i][j]**2+
                               dikeyGradyan[i][j]**2)

#gradyanin yonunu hesaplama
acisal_yon=np.zeros([imge.shape[0],imge.shape[1]])
for i in range (imge.shape[0]):
    for j in range (imge.shape[1]):
        acisal_yon[i][j]=np.arctan((dikeyGradyan[i][j])/
                           (yatayGradyan[i][j]+0.0001))

fig = mplt.figure(figsize=(15,10))
ax1 = mplt.subplot(121)
mplt.imshow(buyukluk,cmap=mplt.cm.gray)
mplt.title('gradyanin buyuklugu')
ax2 = mplt.subplot(122)
mplt.imshow(acisal_yon,cmap=mplt.cm.gray)
mplt.title('gradyanin yonu')
```

Şekil 3.17. Grandyanların büyüklüğü ve yönünü hesaplanma programı



Şekil 3.18. Gradyanların büyüklüğü ve yönü

200 olsun. Bu pikselin yatayda bir önceki piksel değerinin 180, sonraki piksel değerinin ise 210 ve dikeyde ise altındaki piksel değeri 165, üzerindeki piksel değeri ise 125 olsun. Bu durumda gx=210-180=30 ve gy=165-125=40 olarak hesaplananır. 3.7'de verilen formül

kullanılarak büyüklük ve açı değerleri hesaplandığında, büyüklük=50 ve $acisal_{yon} = 53$ derece elde edilir. 53 derece 9 binlik histogramımızda 40 ve 60 derecelere karşılık gelecektir. Yakınlık değerine göre 50 olan büyüklük değerinden (60-53)/2050=17,5 ve (60-53)/2050=32,5 şeklinde iki değer hesaplanır. Histogramımızda 40 dereceye 17,5 ve 60 dereceye ise 32,5 değeri yazılarak histogram değerlerimiz bulunmuş olur. Bu işlem imgedeki bütün pikseller için tekrar edilerek 9X1 histogram vektörlerimiz elde edilmiş olur.

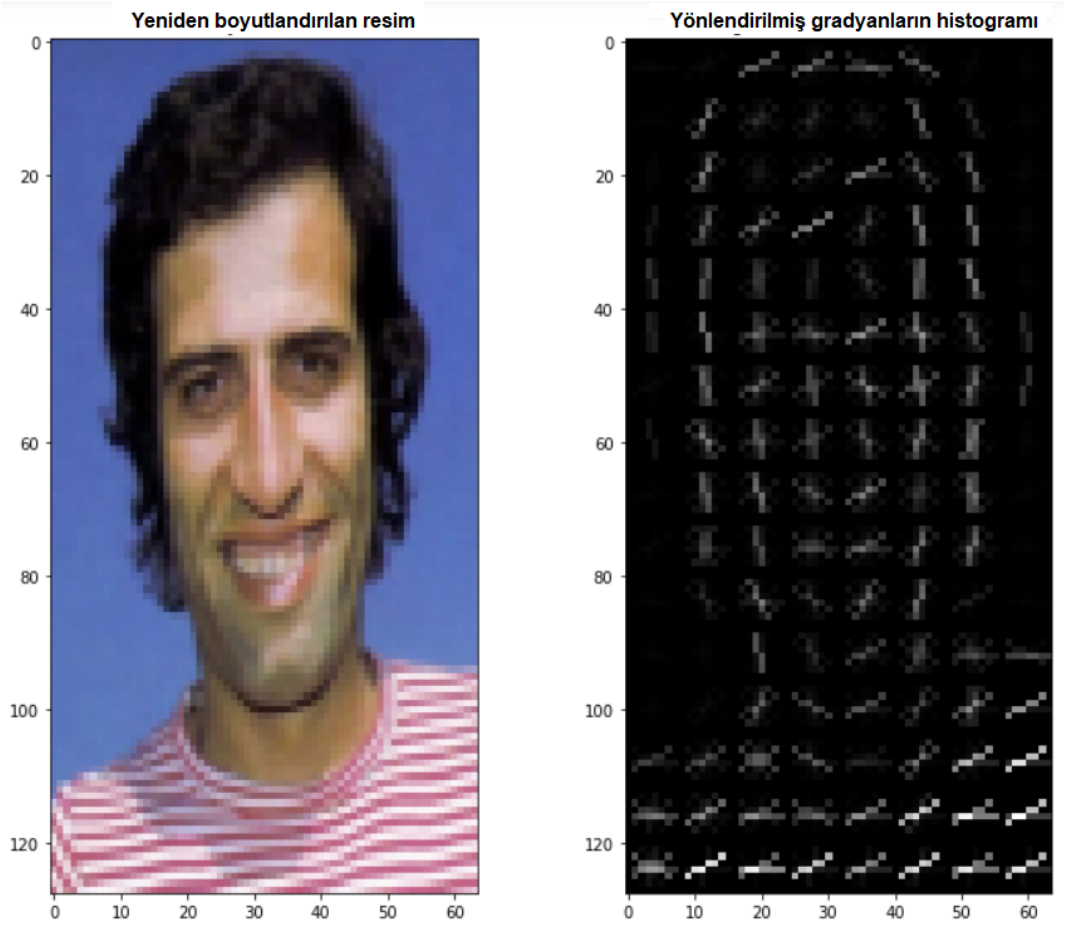
16 × 16 boyutlu bir blok, 36 x 1 boyutunda bir eleman vektörü oluşturmak için birleştirilebilen 4 histograma sahiptir. Pencere daha sonra 8 piksel hareket ettirilir ve bu pencere üzerinden 36 × 1 boyutunda bir $V = [a_1, a_2, a_3, \ldots, a_{36}]$ vektörü hesaplanır ve işlem tekrarlanır.

```
imge = imread('ksunal.jpg') #imge okuma
re_imge = resize(imge, (128,64)) #yeniden boyutlandirilmis imge
from skimage.feature import hog

#HOG oznitelikler olusturuluyor
f_d, hog_imge = hog(re_imge, orientations=9,
                    pixels_per_cell=(8, 8),cells_per_block=(2, 2),
                    visualize=True, multichannel=True)

#imgeler ciziliyor
fig = mplt.figure(figsize=(12, 6))
ax1 = mplt.subplot(131);mplt.imshow(imge)
mplt.title('Orjinal resim')
ax2 = mplt.subplot(132)
mplt.imshow(re_imge, cmap=mplt.cm.gray)
mplt.title('Yeniden boyutlandirilan resim')
ax2 = mplt.subplot(133);mplt.imshow(hog_imge, cmap=mplt.cm.gray)
mplt.title('Histogram of Oriented Gradients')
mplt.show()
```

Şekil 3.19. Gradyanların büyüklüğü ve yönünün elde edilmesi programı

Dördüncü adımda, ışık değişimlerinden etkilenmemeleri için histogram normalleştirilmektedir . Bir önceki adımda elde edilen V matrisini normalleştirmek için, 3.8'de verilen formülde görüleceği üzere matrisdeki değerlerin her biri, değerlerin karelerinin toplamının karekökune bölünür. Böylece normaleştirilmiş olan V vektrü hesaplanmış olur.

$$k = \sqrt{(a_1^2 + a_2^2 + a_3^2 + \ldots + a_{36}^2)}$$
$$V_{normal} = \left[\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \ldots, \frac{a_{36}}{k}\right] \tag{3.8}$$



Şekil 3.20. Yönlendirilmiş gradyanların histogramı

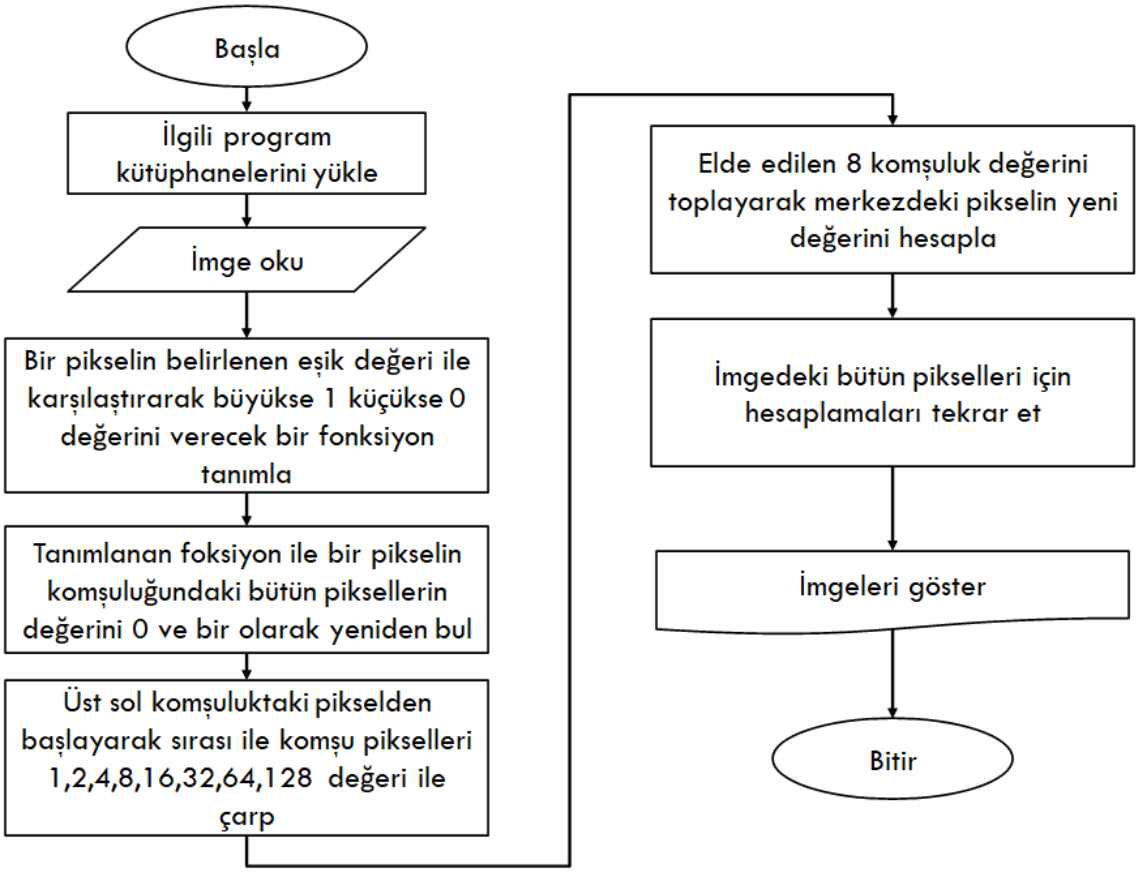
Beşinci adımda HOG özellik vektörü hesaplanır. Başlagıçtaki imgeninin son özellik vektörünü elde etmek için hesaplanmış olan 36 × 1 boyutundaki vektörler, birleştirilrek tek vektör haline getirilir. Bu vektör şu şekilde hesaplanmaktadır; 16 × 16'lık bir blok için, 7 yatay ve 15 dikey konum olmak üzere toplamda 7 x 15 = 105 tane konumu vardır. Her 16 × 16'lık bir blok 36 × 1'lik vektör ile temsil edilir. Hepsi tek bir vektörde birleştirildiğinde 36 × 105 = 3780 boyutlu bir vektör elde edilmektedir. [32].

Bu adımlardaki işlemler Şekil 3.19'daki python programı ile gerçekleştirilmiştir. Elde edilen HOG resmi Şekil 3.20'de verilmişir.

3.2.2 Yerel İkili Örüntüler - LBP

Yerel İkili Desen (Local Binary Pattern - LBP), bir görüntünün piksellerini her pikselin komşuluğunu eşleyerek işlem yapan ve sonucu bir ikili sayı şeklinde veren ve basit olmasının yanında çok verimli bir doku tanımlayıcısıdır.

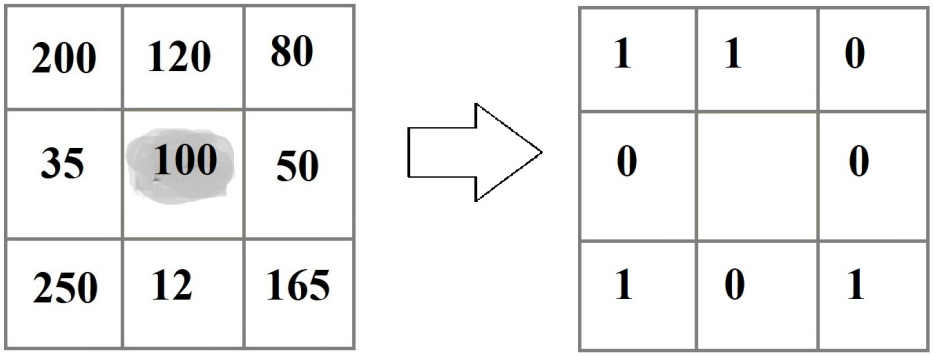
LBP operatörünün en önemli özelliğine, aydınlatma varyasyonlarının neden olduğu monoton gri ölçekli değişikliklere karşı dayanıklılığı örnek olarak gösterilmektedir. Başka bir özelliği ise gerçek zamanlı durumlarda, zorlu imgelerin analiz edilmesini mümkün kılan hesaplama basitliğidir. LBP işlem adımlarını gösteren bir algoritma Şekil 3.13'te verilmiştir.



Şekil 3.21. LBP algoritması

İlk olarak, işlenecek imgeyi gri tonlamaya dönüştürerek LBP doku tanımlayıcısının ilk adımını atmış oluyoruz. Gri tonlamalı imgedeki her piksel için hesaplanacak pikseli çevreleyen, istenen boyutta bir komşuluk seçilmektedir. Sonra bu piksel için bir LBP değeri hesaplanmaktadır. Giriş imgesi ile aynı genişlik ve yükseklikte olan çıktı 2 boyutlu dizide saklanmaktadır. Eğer yerel komşu piksel değeri, merkez piksel değerlerinden büyükse veya bunlara eşitse 1 olarak ayarlanmaktadır. Diğer piksel değerlerine 0 değeri girilmektedir [33].

Örnein 3 x 3'lük komşulukta çalışan bir LBP oluşturmak için değeri 100 olan merkez pikseli çevreleyen 8 piksellik komşuluk alınmıştır. Bu komşuluktaki 100'e eşit ve büyük değerler için '1' küçük değerler için ise '0' değeri yazılarak Şekil 3.22'de görüldüğü gib bir set oluşturulmuştur. Bu işlem Şekil 3.23'deki Python program fonksiyonu ile gerçekleştirilmektedir [34].



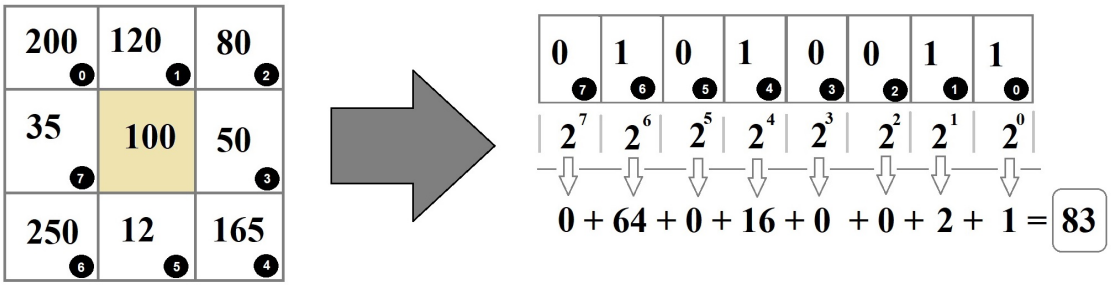
Şekil 3.22. Bir LBP oluşturmanın ilk adımı, bir merkez pikseli çevreleyen 8 piksel komşuluk alınması ve bir eşik değerine göre ondan binary set oluturulması

```python
import cv2
import numpy as np
from matplotlib import pyplot as mplt

def piksel_al(img, merkez, x, y):
    yeni_deger = 0
    try:
    # Eger yerel komsu piksel degeri, merkez piksel degerlerinden
    # buyukse veya bunlara esitse, 1 olarak ayarlanir.
        if imge[x][y] >= merkez:
            yeni_deger = 1
    except:
    # Bir merkez piksel degerinin komsuluk degeri bos oldugunda,
    # yani sinirlarda mevcut degerler oldugunda istisna gereklidir.
        pass
    return yeni_deger
```

Şekil 3.23. Python 3 x 3 komşuluk için binary set oluşturulma programı

Sonraki adımda, merkez piksel için LBP değerininin hesaplanması gerekmektedir. Komşu herhangi bir pikselden başlanabilir ve yönteme bağlı olarak saat yönünde veya saat yönünün tersine işlem yapılabilmektedir. Ancak yapılacak sıralama görseldeki tüm pikseller ve veri setindeki tüm imgeler için tutarlı olmak zorundadır [33].

3 x 3'lük komşuluk verildiğinde, üzerinde ikili test yapılması gereken 8 komşu vardır. Bu ikili testin sonuçları, 8 bitlik bir dizide saklanmaktadır. Bu dizi Şekil 3.24'de görüldüğü şekilde ondalık sayıya dönüştürerek LBP değeri hesaplanmıştır. Bu hesaplamayı yapan python program fonksiyonu Şekil 3.25'de verilmiştir [34].



Şekil 3.24. Merkez pikselin 8 bitlik ikili komşuluğunu alıp ondalık gösterime dönüştürülmesi

```python
# LBP'yi hesaplama fonksiyonu:
def lbp_hesaplanan_piksel(imge, x, y):
    merkez = imge[x][y]
    deger_ar = []
    deger_ar.append(piksel_al(imge, merkez, x-1, y-1))      #ust sol
    deger_ar.append(piksel_al(imge, merkez, x-1, y))        #ust
    deger_ar.append(piksel_al(imge, merkez, x-1, y + 1))    #ust sag
    deger_ar.append(piksel_al(imge, merkez, x, y + 1))      #sag
    deger_ar.append(piksel_al(imge, merkez, x + 1, y + 1)) #alt sag
    deger_ar.append(piksel_al(imge, merkez, x + 1, y))      #alt
    deger_ar.append(piksel_al(imge, merkez, x + 1, y-1))    #alt sol
    deger_ar.append(piksel_al(imge, merkez, x, y-1))        #sol
    # ikili degerleri ondalik sayiya cevirmek icin;
    guc_degeri = [1, 2, 4, 8, 16, 32, 64, 128]
    deger = 0
    for i in range(len(deger_ar)):
        deger += deger_ar[i] * guc_degeri[i]
    return deger
```

Şekil 3.25. Python LBP'yi hesaplama fonksiyonu

Bu eşikleme işlemi daha sonra giriş imgesindeki her piksel için tekrarlanarak bütün piksel değerleri için LBP hesaplanlmaktadır. Şekil 3.26'daki python programı ile oluşturulmuş fonksiyon ile her bir piksel değerleri için LBP hesaplama işlemi gerçekleştirilmektedir.

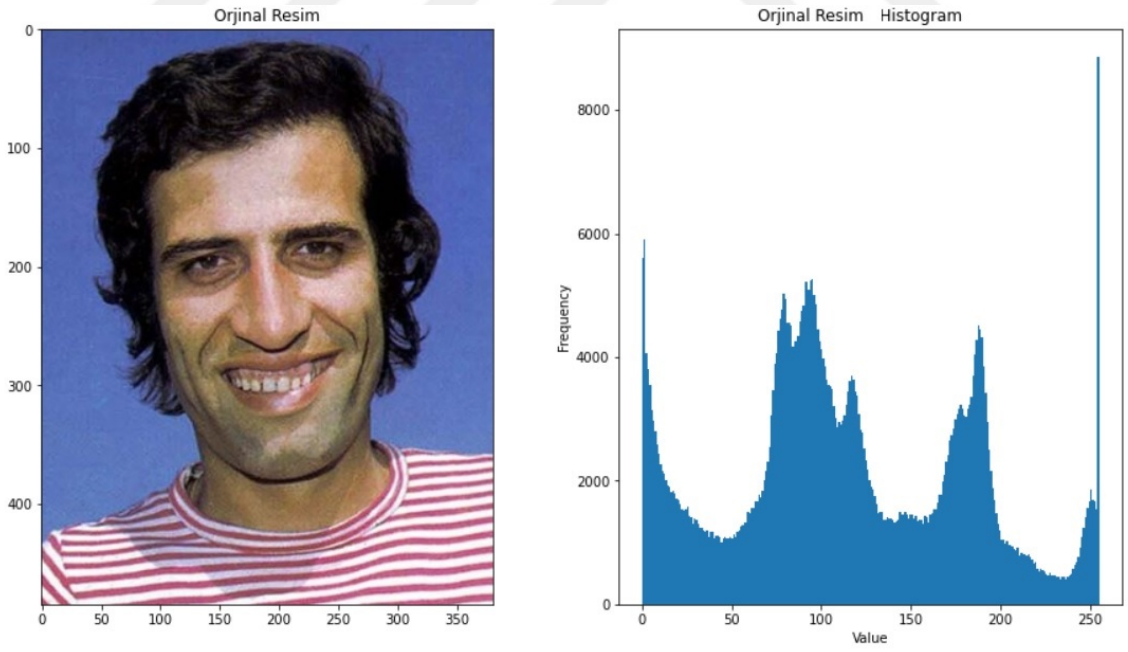
Son adımda elde edilen LBP dizisi üzerinden bir histogram elde edilmektedir. 3 x 3'lük komşulukta $2^8 = 256$ olası desen olduğundan, iki boyutlu LBP dizisi bu nedenle minimum 0 ve

```
# LBP fonksiyonunu kullarak LPB yi hesaplama:
imgeLBP = np.zeros((yukseklik, genislik),np.uint8)
for i in range(0, yukseklik):
    for j in range(0, genislik):
        imgeLBP[i, j] = lbp_hesaplanan_piksel(imge_gri, i, j)
```

Şekil 3.26. Her bir piksel değerleri için LBP hesaplanma programı

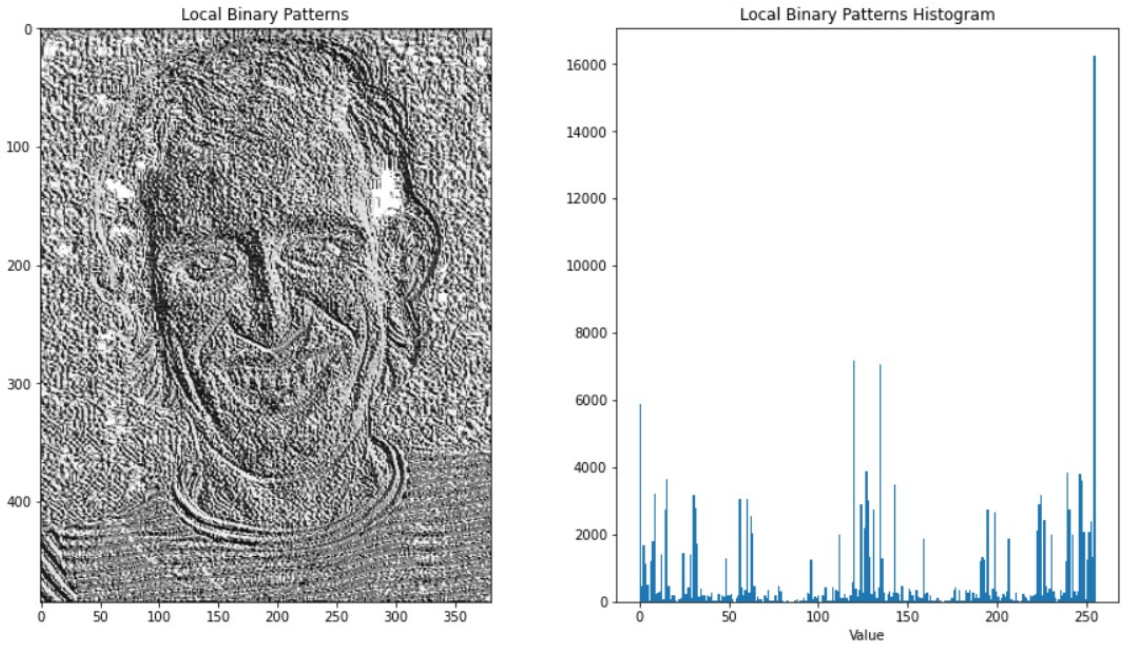
```
fig = mplt.figure(figsize=(15,8))
ax1 = mplt.subplot(121)
mplt.imshow(cv2.cvtColor(imge_bgr, cv2.COLOR_BGR2RGB))
mplt.title('Orjinal Resim')
ax2= mplt.subplot(122)
mplt.hist(imge_bgr.ravel(),256,[0,255]) ;mplt.xlabel('Value')
mplt.ylabel('Frequency'); mplt.title('Input image Histogram')
fig = mplt.figure(figsize=(15,8))
ax1 = mplt.subplot(121); mplt.imshow(imgeLBP, cmap=mplt.cm.gray)
mplt.title('Local Binary Patterns'); ax2= mplt.subplot(122)
mplt.hist(imgeLBP.ravel(),256,[0,255]) ;mplt.xlabel('Value')
mplt.title('Local Binary Patterns Histogram')
mplt.show()
```

Şekil 3.27. Asıl resim ve histogramı gösteren program



Şekil 3.28. Asıl giriş imgesi (solda) ve histogramı (sağda)

maksimum 255 değerlerine sahip olması beklenmektedir. Bu şekilde son özellik vektörü olarak LBP kodlarının 256 binarilik histogramı oluşturulmaktadır.



Şekil 3.29. Hesaplanmış LBP imgesi (solda) ve histogramı (sağda)

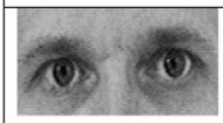
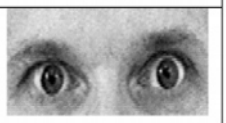
Şekil 3.27'deki python programı çalıştırılarak asıl resim ve histogramı Şekil 3.28'de, hesaplanmış LBP imgesi ve histogramı ise Şekil 3.29'da görüldüğü şekilde elde edilmiştir.

LBP resmine baktığımızda daha basit olduğu fakat asıl resimdeki insan imgesinin hala belirgin olduğu görülmektedir.

3.2.3 Yüz Eylem Kodlama Sistemi-FACS

Ekman ve Friesen [35], yüz ifadelerini tanımlamak için eylem birimlerini (AU) kullanmışlar ve bunlarla Yüz Eylem Kodlama Sistemi'ni (FACS) geliştirmişlerdir. Tanımladıkları 44 FACS AU'sundan, 30 AU, anatomik olarak belirli yüz kaslarının kasılmalarıyla ilişkilidir. Şekil 3.30'da görüldüğü gibi 12 tanesi göz ve kaşında bulunduğu üst yüz içindir ve Şekil 3.31'de görüldüğü gibi 18 tanesi ağız ve dudak durumlarını tanımlayan alt yüz içindir. AU'lar tek başlarına veya kombinasyon halinde kullanılabilmektedir.

FACS eylem birimlerinin (AU) otomatik olarak tanınması ciddi bir sorundur ve nispeten bu konuda az sayıda çalışma yapıldığı göze çarpmaktadır. AU'ların nicel tanımları yoktur ve belirtildiği gibi karmaşık kombinasyonlar da ortaya çıkabilmektedir.

NEUTRAL	AU 1	AU 2	AU 4	AU 5
				
Eyes, brow, and cheek are relaxed.	Inner portion of the brows is raised.	Outer portion of the brows is raised.	Brows lowered and drawn together	Upper eyelids are raised.
AU 6	AU 7	AU 1+2	AU 1+4	AU 4+5
				
Cheeks are raised.	Lower eyelids are raised.	Inner and outer portions of the brows are raised.	Medial portion of the brows is raised and pulled together.	Brows lowered and drawn together and upper eyelids are raised.
AU 1+2+4	AU 1+2+5	AU 1+6	AU 6+7	AU 1+2+5+6+7
				
Brows are pulled together and upward.	Brows and upper eyelids are raised.	Inner portion of brows and cheeks are raised.	Lower eyelids cheeks are raised.	Brows, eyelids, and cheeks are raised.

Şekil 3.30. Üst yüz eylem birimleri (AU) ve bazı kombinasyonlar [17]

Kalıcı ve geçici yüz özelliklerinin ortaya çıkmasında, yüz kaslarının kasılması ile cilt yüzeyindeki hareket ve bu hareketin büyüklüğü etkili olmaktadır. Kalıcı özelliklere örnek olarak dudaklar, gözler, kaşlar, burun ve ilerleyen yaş ile kalıcı hale gelen kırışıklıklar verilebilir. Geçici özellikler ise ancak yüz ifadeleriyle ortaya çıkan yüz çizgilerini ve kırışıklıkları içermektedir. Önden bir yüzde bile, yüz özelliklerinin görünümü ve yeri önemli ölçüde değişebilmektedir. Örneğin, gözler açık ve kapalı olduğunda niteliksel olarak farklı görünmektedir. Farklı bileşenler farklı ekstraksiyon ve tespit yöntemleri gerektirir. Bir yüz imgesindeki hem geçici hem de kalıcı özellikleri tespit etmek ve izlemek için çok bileşenli yüz bileşenleri modelleri tanıtılmıştır.

Bu çalışma içerisinde hem geçici hem de kalıcı yüz özelliklerini tespit etmek için Dlib kütüphanesi ile oluşturulan 68 yüz noktası kullanılmıştır.

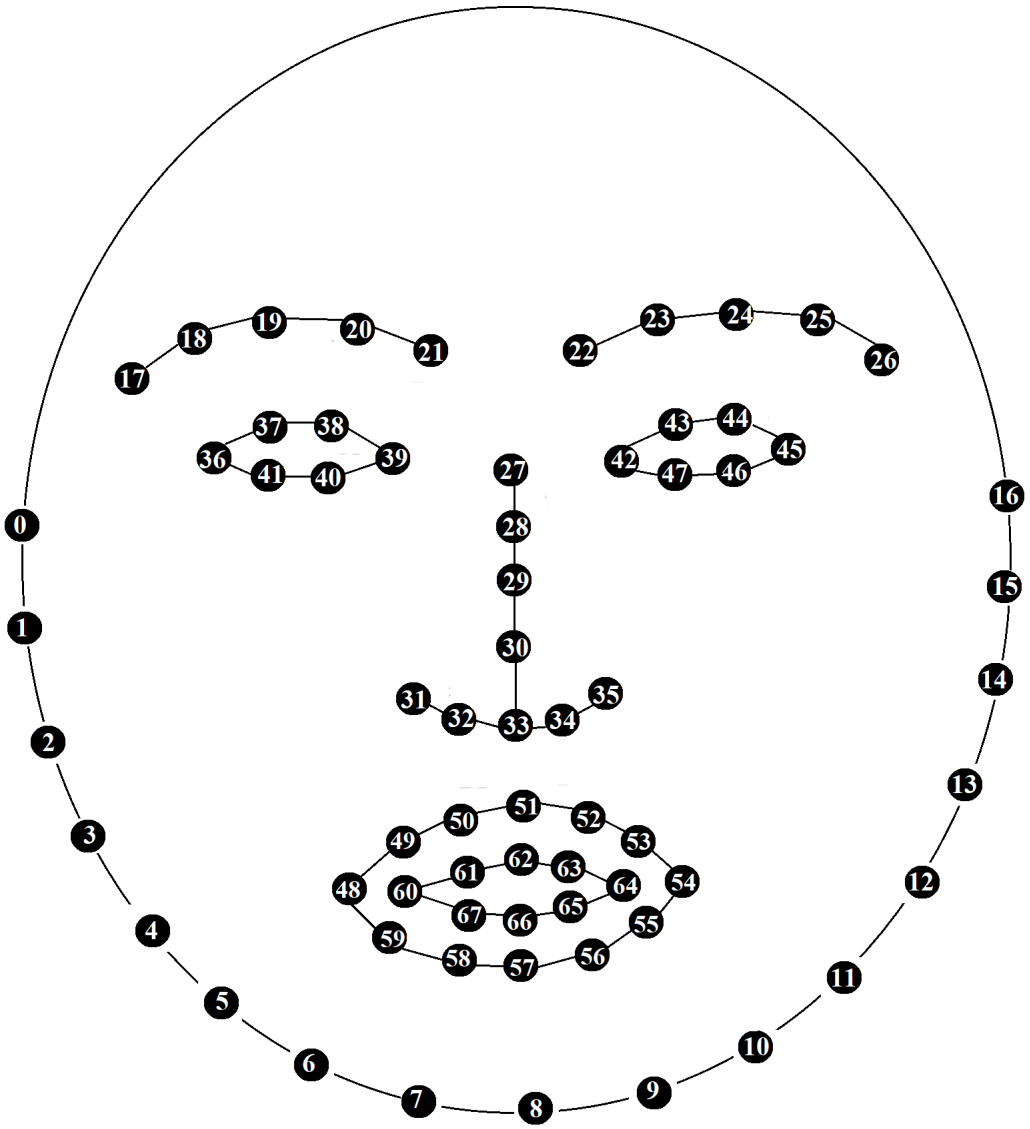
NEUTRAL	AU 9	AU 10	AU 12	AU 20
				
Lips relaxed and closed.	The infraorbital triangle and center of the upper lip are pulled upwards. Nasal root wrinkling is present.	The infraorbital triangle is pushed upwards. Upper lip is raised. Causes angular bend in shape of upper lip. Nasal root wrinkle is absent.	Lip corners are pulled obliquely.	The lips and the lower portion of the nasolabial furrow are pulled pulled back laterally. The mouth is elongated.
AU15	AU 17	AU 25	AU 26	AU 27
				
The corners of the lips are pulled down.	The chin boss is pushed upwards.	Lips are relaxed and parted.	Lips are relaxed and parted; mandible is lowered.	Mouth stretched open and the mandible pulled downwards.
AU 23+24	AU 9+17	AU9+25	AU9+17+23+24	AU10+17
				
Lips tightened, narrowed, and pressed together.				
AU 10+25	AU 10+15+17	AU 12+25	AU12+26	AU 15+17
				
AU 17+23+24	AU 20+25			
				

Şekil 3.31. Alt yüz eylem birimleri (AU) ve bazı kombinasyonlar [17]

3.2.4 Geometrik Öznitelikler

Yüz antropometrisi; insan yüzündeki kaş, göz, burun ve ağız gibi önemli yüz noktaları arasındaki uzaklıklığı ve bu uzaklıkların birbirleriyle oranları olarak tanımlanabilmektedir [36]. Yüzdeki önemli noktalar arasındaki bu oranlar ve mesafeler yüzün duygu durumunu tespit etmek için kullanılabilmektedir. Bu çalışmada duygusal ifade tespiti için Şekil 3.32'de gösterilen insan yüzünde tespit edilmiş olan 68 nokta kullanarak elde edilmiş uzaklıklar ve bu uzaklıkların birbiriyle oranları kullanılmıştır.

Dlib kütüphanesinin [6] kullanıldığı ekler bölümünde verilmiş olan Program A2'de görüldüğü üzere önceden eğitilmiş yüz dönüm noktası dedektörü ile imgedeki yüz bölgeleri tespit edilmiştir. Programda kullanılan prediktör ile tespit edilmiş olan yüz yapılarına eşlenen ve yüzdeki önemli noktalara karşılık gelen 68 (x, y) koordinatın yerini tahmin etmek için kullanılmıştır.



Şekil 3.32. Dlib kütüphanesi ile oluşturulan 68 yüz noktası

Bu tez çalışmasında, insan yüzündeki kalıcı öznitelikler ve değişik duygusal ifade durumlarındaki şekilsel değişiklikler geometrik öznitelikler kullanılarak tanımlanmıştır. İnsan

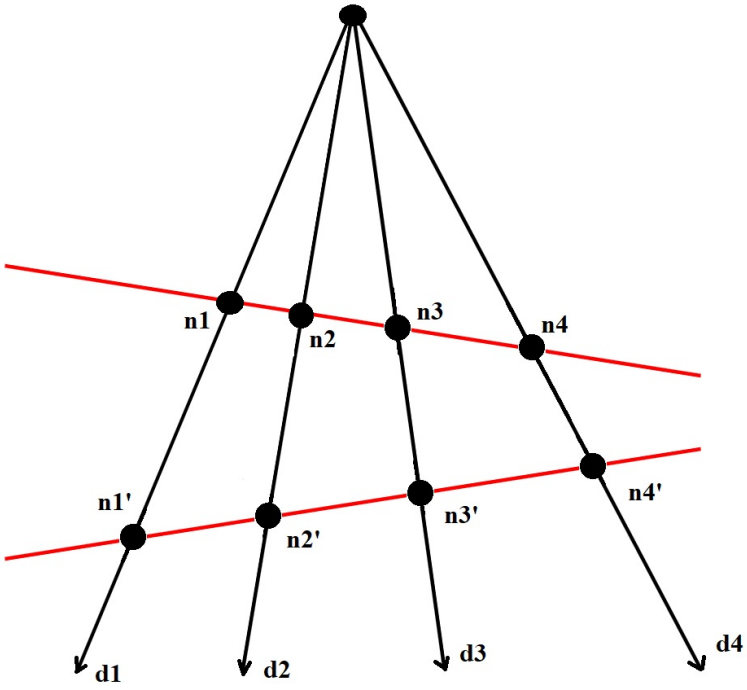
yüzünü geometrik olarak tanımlamak için Dlib kütüphanesi ile oluşturulan 68 önemli yüz noktası arasındaki öklid uzaklık değerleri ve bunların birbirleriyle oranları kullanılmıştır.

İnsan yüzünden elde edilen geometrik öznitelikler, aydınlatma miktarlarının neden olduğu değişikliklerden fazla etkilenmemektedirler. Buna karşın başın dikeyde ve yatayda aldığı pozisyondan ve kameranın çekim açısından etkilenmektedir. Bu problemlere çözüm olarak çapraz oran (cross ratio) tabanlı geometrik oranlar kullanılmıştır. Şekil 3.30'da verilmiş olan Oran 1, başın dikey pozisyonu için Oran 2 ise başın yatay pozisyonu için kullanılan geometrik özniteliklerdir.

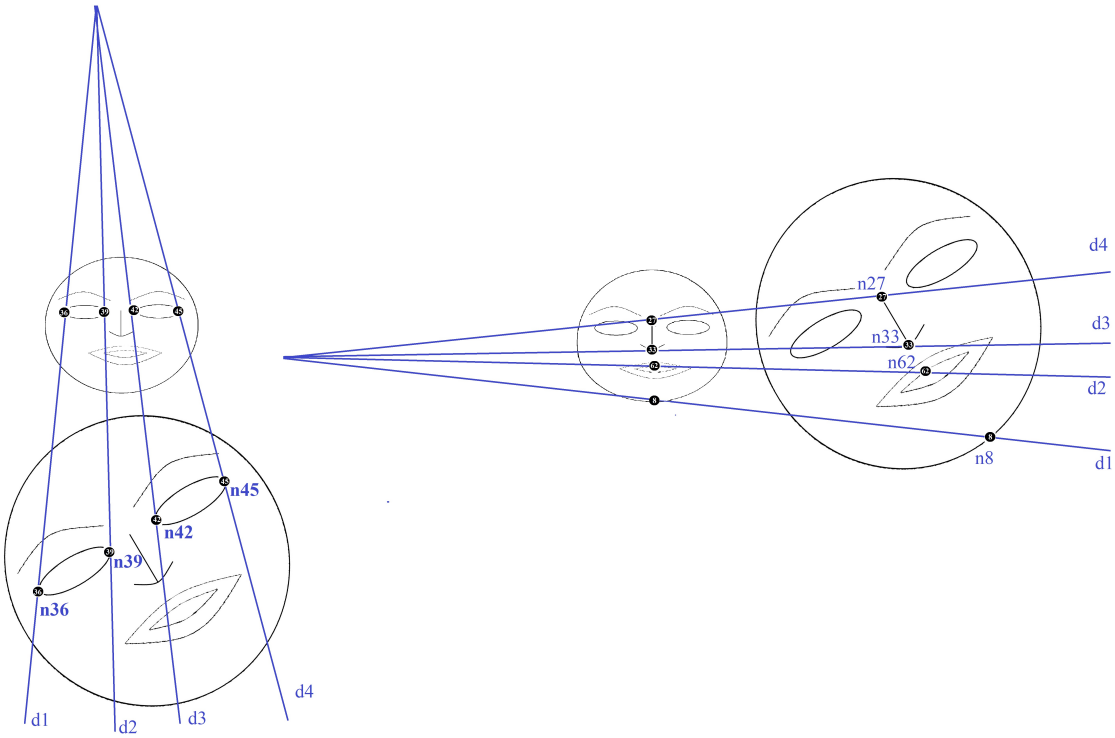
Çapraz oran, izdüşümsel değişimlerden etkilenmez. Şekil 3.33'de gösterildiği gibi aynı tek noktadan geçen, fakat birbirlerine paralel olmayan (aynı düzlemde bulunmayan) d1 , d2 , d3 ve d4 olmak üzere 4 adet doğru bulunmaktadır. Geometride, aynı doğru üzerinde bulunan n1, n2, n3 ve n4 4 farklı nokta olmak üzere, bu noktalar için çift oran ve harmonik olmayan oran olarak da adlandırılan çapraz oran formülü 3.9'da verilmiştir [37].

$$(n1,n2;n3,n4) = \frac{|n1-n3| * |n2-n4|}{|n2-n3| * |n1-n4|} \tag{3.9}$$

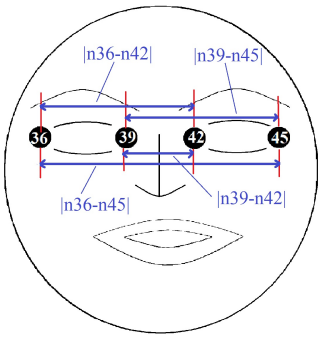
Bu çalışma kapsamında kullanılan d1 , d2 , d3 ve d4 doğruları kulanılarak Şekil 3.34'te görüleceği üzere, yüzdeki yatay pozisyon değişimlerini tespit etmek için Oran 1, yüzdeki dikey pozisyon değişimleri tespit etmek için ise Oran 2 tanımlanmıştır.



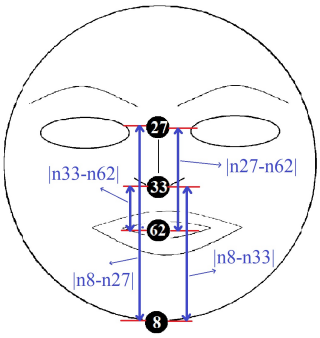
Şekil 3.33. Çapraz oran [37]



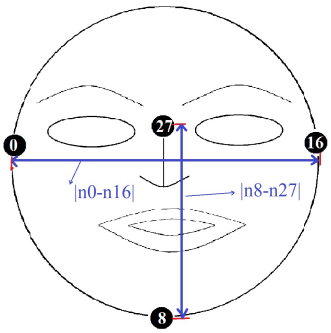
Şekil 3.34. Yüzdeki yatay (solda) ve dikey (sağda) pozisyon değişimleri göz önüne alınarak çapraz oran



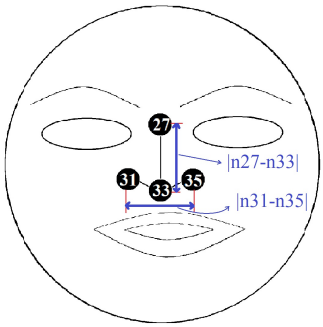
(a) $Oran1 = \frac{|n36-n42|*|n39-n45|}{|n39-n42|*|n36-n45|}$



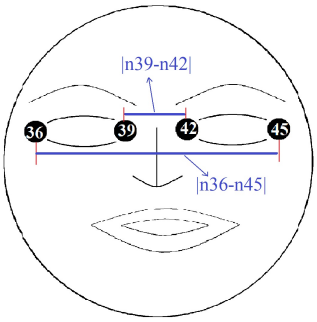
(b) $Oran2 = \frac{|n27-n62|*|n8-n33|}{|n33-n62|*|n8-n27|}$



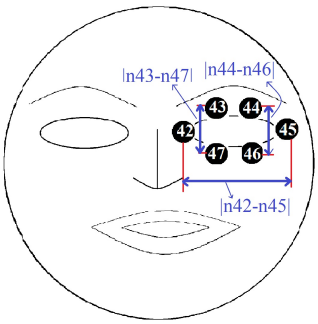
(c) $Oran3 = \frac{|n8-n27|}{|n0-n16|}$



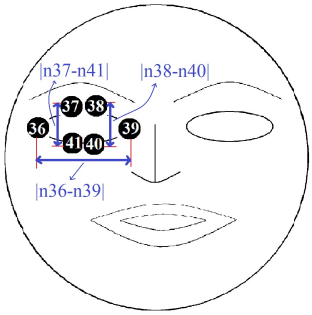
(d) $Oran4 = \frac{|n31-n35|}{|n27-n33|}$



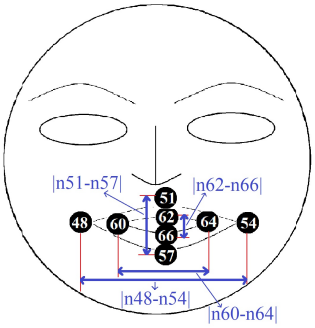
(e) $Oran5 = \frac{|n39-n42|}{|n36-n45|}$



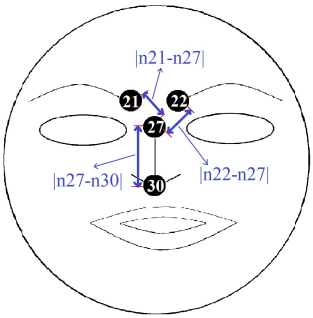
(f) $[Oran6 = \frac{(|n43-n47|+|n44-n46|)/2}{|n42-n45|}$



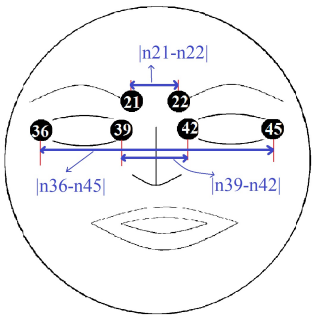
(g) $Oran7 = \frac{(|n37-n41|+|n38-n40|)/2}{|n36-n39|}$



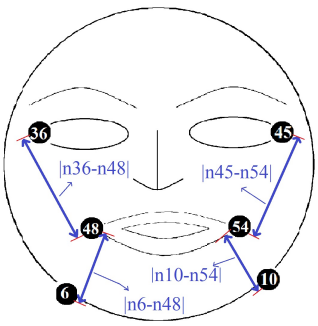
(h) $Oran8 = \frac{(|n51-n57|+|n62-n66|)/2}{(|n48-n54|+|n60-n64|)/2}$



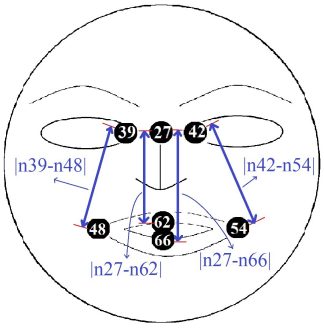
(i) $Oran9 = \frac{(|n21-n27|+|n22-n27|)/2}{|n27-n30|}$



(j) $Oran10 = \frac{|n21-n22|}{(|n39-n42|+|n36-n45|)/2}$

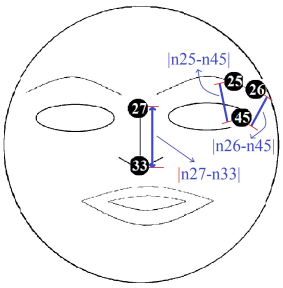


(k) $Oran11 = \frac{(|n6-n48|+|n10-n54|)/2}{(|n36-n48|+|n45-n54|)/2}$

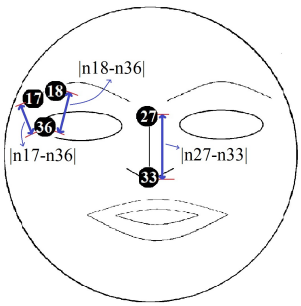


(l) $Oran12 = \frac{(|n27-n62|+|n27-n66|)/2}{(|n39-n48|+|n42-n54|)/2}$

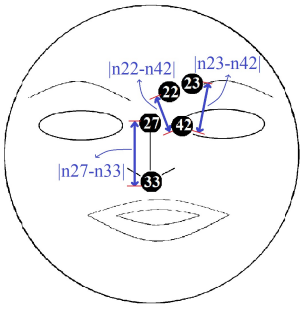
Şekil 3.35. Çalışmada kullanılan geometrik öznitelikler 1



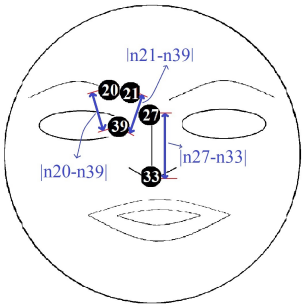
(a) $Oran13 = \frac{(|n25-n45|+|n26-n45|)/2}{|n27-n33|}$



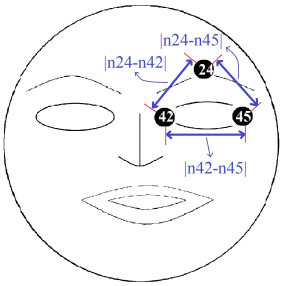
(b) $Oran14 = \frac{(|n18-n36|+|n17-n36|)/2}{|n27-n33|}$



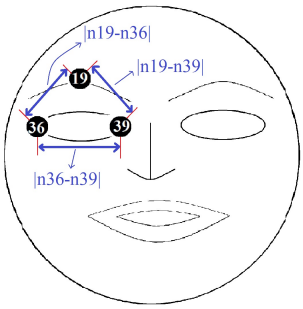
(c) $Oran15 = \frac{(|n22-n42|+|n23-n42|)/2}{|n27-n33|}$



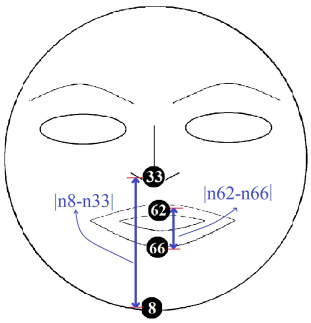
(d) $Oran16 = \frac{(|n20-n39|+|n21-n39|)/2}{|n27-n33|}$



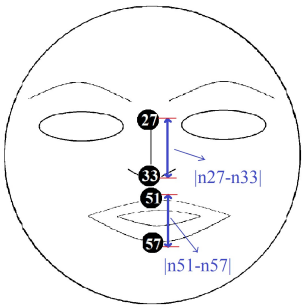
(e) $Oran17 = \frac{(|n24-n42|+|n24-n45|)/2}{|n42-n45|}$



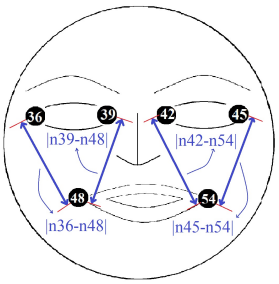
(f) $Oran18 = \frac{(|n19-n36|+|n19-n39|)/2}{|n36-n39|}$



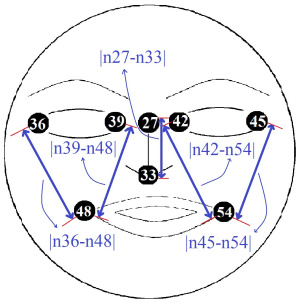
(g) $Oran19 = \frac{|n8-n33|}{|n62-n66|}$



(h) $Oran20 = \frac{|n51-n57|}{|n27-n33|}$



(i) $Oran21 = \frac{(|n36-n48|+n39-n48|)/2}{(|n42-n54|+n45-n54|)/2}$



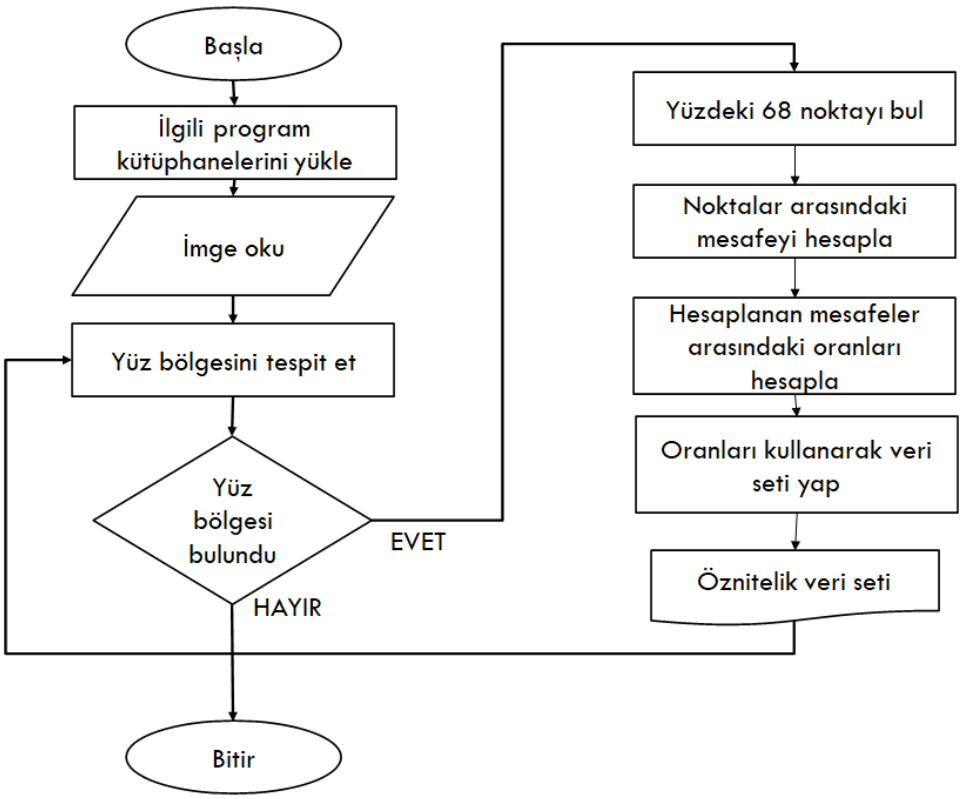
(j) $Oran22 = \frac{(\frac{|n36-n48|+n39-n48|}{+|n42-n54|+n45-n54|})/4}{|n27-n33|}$

Şekil 3.36. Çalışmada kullanılan geometrik öznitelikler 2

Bu tez çalımasında kullanılan geometrk öznitelik değerlerinden bazıları daha önce yapılmış olan çalışmalardan elde edilmiştir. Oran 3 (Facial index), Oran 4 (Nasal index), Oran 5 (Intercanthal index), Oran 6 (Eye fissure index) [36] diğerleri ise yüzdeki duygusal ifade tespitinde ayırt ediciliğinin yüksek olduğu düşünülerek kullanılmıştır. Şekil 3.35 ve Şekil 3.36'de geometrik öznitelikleri tespit etmek için bu çalışmada kullanılan 22 oran verilmiştir.

Tespit edilen oranlar Ekler bölümünde Program A.3'te verilen python programı ile yüz ifadesinden duygu durumu tespiti yapabilmek için kullanılmıştır. Programda öncelikle Dlib ktüphanesi kullanarak resimdeki yüz bölgesi ve yüze ait 68 nokta tespit edilmiştir.

Bu noktalar arasındaki mesafeler hesaplanmış daha sonra bu mesafelerin oranları bulunmuştur. Şekil 3.35 ve Şekil 3.36'da gösterilmiş olan bu oranlar ile geometrik öznitelikleri saptayacak fonksiyon oluşturulmuştur.



Şekil 3.37. Geometrik öznitelik algoritması

Daha sonra CK+ veri setinde bulunan her bir imgedeki öznitelikler Şekil 3.37'de verilen algoritma ile oluşturulmuş olan Ekler bölümünde Program A.4'teki python programı

kullanılarak tespit edilmiştir. Bu programla veriler ölçeklenmiş daha sonra her bir imgedeki duygusal durumu gösteren etiketler (labels) geometrik öznitelik veri setine eklenmiştir.

Out[19]:

	x0	x1	x2	x3	x4	x5	...	x17	x18	x19	x20	x21	x22	Duygu
0	1	1.246154	2.631416	0.838356	0.536293	0.384615	...	0.907403	0.925615	0.000000	0.199557	1.010094	1.299732	0
1	1	1.242878	2.207478	0.839273	0.532152	0.386248	...	0.907403	0.907403	0.000000	0.266076	1.011376	1.298912	0
2	1	1.247153	2.600653	0.841352	0.561405	0.384331	...	0.949519	0.907403	0.000000	0.187135	0.986491	1.233333	0
3	1	1.191521	2.322853	0.689077	0.618347	0.429783	...	0.845991	0.845991	0.000000	0.306786	0.957250	1.500602	0
4	1	1.216337	2.614695	0.703819	0.615385	0.407128	...	0.845991	0.806376	0.055556	0.307692	1.026139	1.482028	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
976	1	1.250000	2.380452	0.737886	0.641223	0.384615	...	1.207107	1.182229	0.176166	0.569976	1.000000	1.633711	6
977	1	1.218855	2.400000	0.697674	0.714286	0.407407	...	1.295012	1.318610	0.187500	0.571429	1.000000	1.645155	6
978	1	1.247153	2.139127	0.825121	0.498729	0.384331	...	1.071667	1.140388	0.263158	0.783718	0.984066	1.690137	6
979	1	1.218855	2.433315	0.756770	0.569976	0.407407	...	1.057381	1.071667	0.176166	0.574412	1.006236	1.646091	6
980	1	1.247153	2.377270	0.757444	0.498729	0.384331	...	1.071667	1.140388	0.234888	0.641223	0.990165	1.625677	6

981 rows × 24 columns

Şekil 3.38. Geometrik öznitelik veri seti

Elde edilen geometrik öznitelik veri setinin örnek bir bölmü Şekil 3.38'de gösterilmiştir. Burada Oran 1'den Oran 22'ye kadar olan tüm oranlar x1 den x22'ye kadar öznitelik olarak ifade edilmiştir.

3.2.5 Dalgacık Dönüşümü

Dalgacık dönüşümü (Wavelet Transform) tek boyutlu sinyallere ve imge gibi iki boyutlu sinyallere uygulanabilmektedir.

Fourier dönüşümünün dezavantajı olarak dönüşüm gerçekleştirilirken frekans bilgileri elde edilirken zaman bilgileri yok olmaktadır. Dalgacık dönüşümünde ise zaman bilgisi de vardır. Çok fazla değişmeyen bir sinyal için zaman bilgisinin kaybedilmesi önemli olmayabilir ancak sinyallerin çoğu değişkendir ve zaman bilgisi bu tür sinyaller için önemlidir.

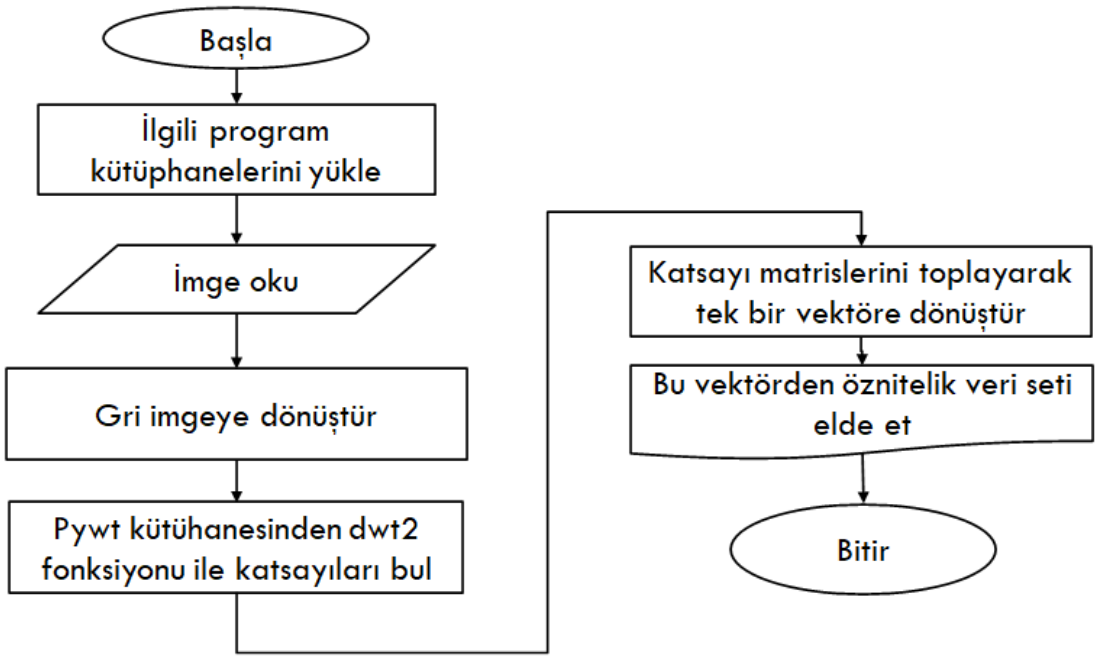
Fourier dönüşümü, gelen sinyali sinüs dalgalarına çevirirken, dalgacık dönüşümü ise gelen sinyali ölçekleyerek ve kaydırarak MOTHER WAVELET denilen başka bir forma dönüştürür. Dalgacık dönüşümü imge kaliteden çok kayıp vermeden daha yüksek sıkıştırma oranı sağladığı görülmektedir. Ayrıca imgedeki önemli bölgelerin belirlenerek daha yüksek kalitede, önemsiz bölgelerin daha düşük kalitede sıkıştırılmasına imkân vermektedir. (ROI: Region of Interest)

Sürekli işaretlerin ayrık işaret olarak analizinin gerçekleştirilmesi için ayrık dalgacık dönüşümü (Discrete Wavelet Transform - DWT) kullanılmaktadır. DWT kullanılması için ölçek ve dönüşüm parametrelerinin de ayrık olarak ele alınması gerekmektedir

DWT elde etmek için çoklu çözünürlük analizi (ÇÇA) kullanılmaktadır. ÇÇA ile ayrık işarete ardışık olarak alçak geçiren ve yüksek geçiren filtreler uygulanmaktadır ve ortaya çıkan işaretler 2 kat veri azaltmaya tabi tutulmaktadır.

Şekil 3.39'da verilen algoritma ile pywt [38], opencv [2], matplotlib.pyplot [5], kütüphaneleri kullanılarak oluşturulmuş python kodları Şekil 3.40 ta görülmektedir.

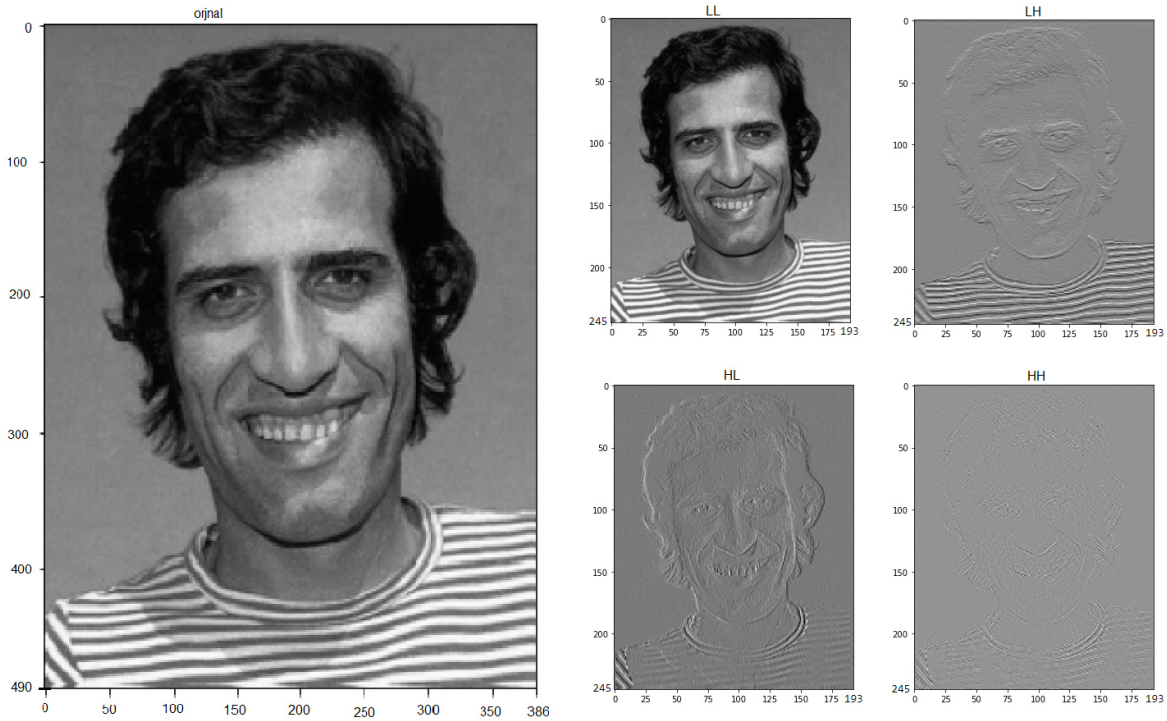
(485, 381) boyutlarındaki bir imgeye ayrık dalgacık dönüşümü uygulanmıştır. Dönüşüm sonucunda (245, 193) boyutlarında elde edilen dört adet resim Şekil 3.41'de verilmiştir. Bu resimler asıl resmi bize yaklaşık olarak veren yaklaşık resim (AA), yatay kenarların belirgin olduğu yatay detay (AY), dikey kanarların belirgin olduğu dikey detay (YA) ve köşegenlerin belirgin olduğu diagonal detay (YY) imgeleridir.



Şekil 3.39. Ayrık dalgacık dönüşümü algoritması

```
import pywt,cv2
import pywt.data
import numpy as np
import matplotlib.pyplot as mplt

imge = cv2.imread('....\\ksunal.jpg')
imge = cv2.cvtColor(imge, cv2.COLOR_BGR2GRAY)
katsayilar2=pywt.dwt2(imge,'bior1.3')
AA, (AY, YA, YY) = katsayilar2

fig = mplt.figure(figsize=(12, 15)) ;
ax1 = mplt.subplot(221);mplt.imshow(AA, cmap=mplt.cm.gray)
mplt.title('Yaklasik resim')
ax2 = mplt.subplot(222);mplt.imshow(AY, cmap=mplt.cm.gray)
mplt.title('Yatay detay ')
ax3 = mplt.subplot(223);mplt.imshow(YA, cmap=mplt.cm.gray)
mplt.title('Dikey detay')
ax4 = mplt.subplot(224);mplt.imshow(YY, cmap=mplt.cm.gray)
mplt.title('Diyagonal detay ');mplt.show()
```

Şekil 3.40. Ayrık dalgacık dönüşümü python programı



Şekil 3.41. Ayrık dalgacık dönüşümü ile elde edilen resimler

Dalgacık dönüşümü ile imgenin boyutu yatayda ve dikeyde yarısı oranında azaltılmıştır. Böylece çok daha rahat ve hızlı bir şekilde işlemler yapılabilir.

3.3 Yüz İfadesi Sınıflandırma

Yüz ifadelerinin belirlendiği son ve en önemli aşamadır. Önceki aşamada tespit edilen yüz özellikleri kullanılarak duygusal ifadeler tanımlanmaya çalışılmaktadır. Yüz öznitelikleri arasındaki benzerliklere göre sınıflandırma yapılmaktadır. Yüz ifadelerinden duygusal durum tanıma, scikit-learn [7] tensorflow-keras [9] ve opencv[2] kütüphaneleri aracılığıyla makine öğrenme tekniği olan SVM, KNN, lojistik regresyon ve CNN teknikleri ile de yapılabilmektedir.

3.3.1 Destek Vektör Makinaları - SVM

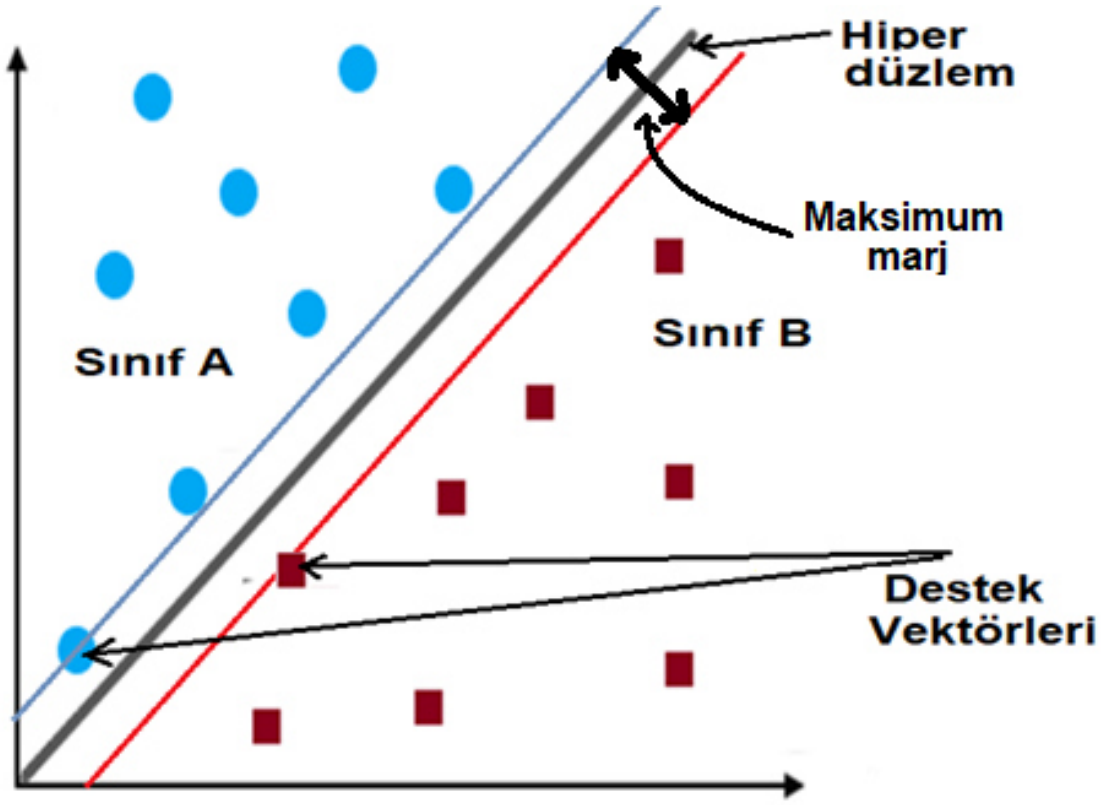
Destek Vektör Makinalarıdır (Support Vector Machines , SVM) eğitim verilerindeki noktaların herhangi birinin en uzağında olan iki sınıf arasında, vektör uzayında belirli bir karar sınırı tespit eden, makine öğrenme yöntemi olarak tanımlanabilir. En güçlü sınıflandırma algoritmalarından biridir.

Şekil 3.42'de görüleceği üzere bu yöntem ile iki sınıfı doğru bir şekilde bölebilmek için en uygun bir hiper düzlem bulmaya çalışılmaktadır. Buna ek olarak, iki sınıf arasında örtüşmeyi önlemek için her iki sınıf açısından da maksimum olması gereken bir marj kavramı bulunmaktadır.

Şekil 3.43'te gösterildiği üzere doğrusal ayrılabilen ve doğrusal ayrılamayan iki ayrı veri seti için farklı SVM çeşitleri kullanılmaktadır. Doğrusal olarak ayrılamayan verileri daha iyi sınıflandırabilmek için daha yüksek bir boyuta eşlenmektedir. 3.10'da verilen formül radyal temel işlevi (rbf) için ve 3.11'de verilen formül ise polinom gibi çekirdek işlevleri doğrusal olmayan veriler için kullanılmaktadır [39].

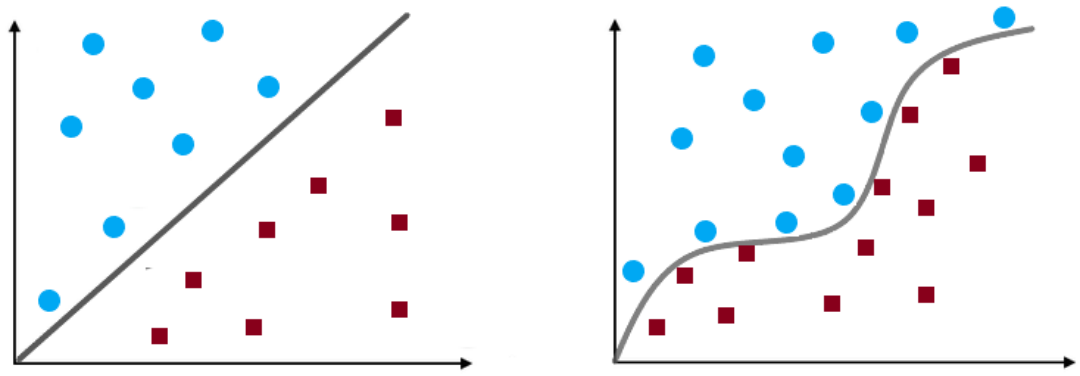
K-kat çapraz doğrulaması, veritabanındaki herhangi bir varyasyonu kaldırmak ve farklı makine öğrenme algoritmalarını karşılaştırmak için kullanılmaktadır. K-kat çapraz doğrulamasında, veri seti K kez K adet dilime bölünmekte ve tahmin sonuçlarının tüm yinelemelerde ortalaması alınmaktadır.

$$K(x,y) = e^{\gamma|x-y|^2} \tag{3.10}$$



Şekil 3.42. Destek vektör makinaları

$$K(x,y) = (x \cdot y + 1)^n \tag{3.11}$$



Şekil 3.43. Doğrusal ayrilabilen (solda) ve doğrusal ayrilamayan (sağda) iki veri seti

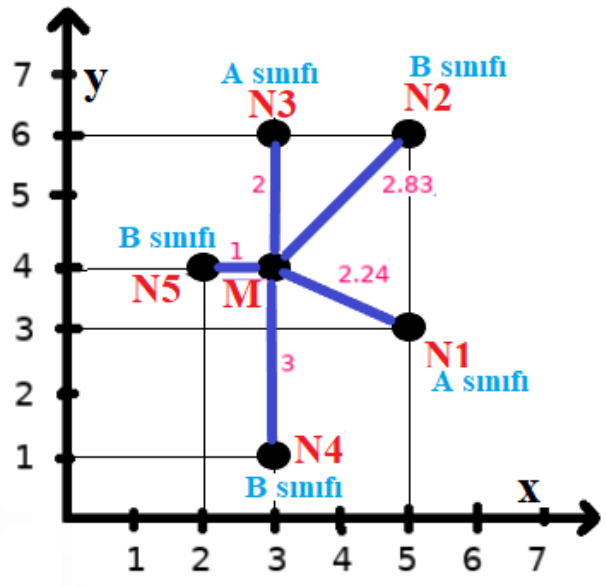
Yüz ifadesinden duygu tespiti yapmak için genellikle ikili yerine çok-sınıflı bir SVM kullanılmaktadır. Genel anlamda çok sınıflı bir SVM, sınıf sayısı kadar SVM'nin birbirine füzyonuyla elde edilmektedir. SVM ile oluşan her bir sınıf diğer sınıflarla karşılaştırılarak bir sonuç elde edilmektedir. Eğer N kadar sınıf varsa N sayıda SVM eğitilmekte ve bu SVM'ler birbirleriyle kıyaslanarak hangi sınıf için en güvenilir sonucun elde edildiği değerlendirilmekte ve bu sonuca bakılarak sınıflandırma yapılmaktadır.

$$f(x) = arg\max_{i} f_i(x) \tag{3.12}$$

3.12'de verilen formülde x girdi vektörü olmakla beraber i sınıfı temsil etmektedir [39].

3.3.2 K-En Yakın Komşu Algoritması - KNN

K-En yakın komşu algoritması (K-Nearest Neighbor - KNN) sınıflandırılmak istenen bir veriyi daha önce gelen verilerle olan yakınlık durumlarına göre sınıflandırmaktadır. Hem sınıflandırma hemde regresyon problemlerini çözmek için kullanılabilmekte olan denetimli bir öğrenme algoritmasıdır.



Şekil 3.44. KNN örneği

KNN algoritmasının çalışması, Şekil 3.44'de gösterildiği gibi üzerinde nokta olarak gösterilen veriler ile anlatılmak istenirse;

k=1 olarak alındığında, en yakın komşulukta 1 değer olmaktadır. Yeni gelen verinin önceki verilere olan uzaklıkları ölçüldüğünde 1 brim uzaklıktaki N5 noktasındaki verinin en yakın olduğu görülmektedir. Bu noktadaki verinin sınıfı B sınıfı olduğundan yeni gelen verinin sınıfının da B sınıfı olduğuna karar verilmektedir.

k=3 olarak alındığında, en yakın komşulukta 3 değer olmaktadır. Yeni gelen verinin eski verilere olan uzaklıkları ölçüldüğünde, en yakın 3 tane nokta olarak, 1 birim uzaklıktaki N5, 2 birim uzaklıktaki N3 ve 2.24 birim uzaklıktaki N1 noktaları belirlenmiştir. Bu noktalardan 2'sinin A sınıfında ve 1'nin B sınıfında olduğu, Şekil 3.36'da görülmektedir. En yakın komşulukta en çok A sınıfından veri olduğundan, algoritma yeni gelen verinin A sınıfında olduğuna karar vermektedir.

Eğer k=5 alırsak, 2 tane nokta A sınıfından 3 tane nokta B sınıfından olmak üzere, en yakın komşulukta 5 değer olmaktadır. Bu durumda algoritma yeni verinin B sınıfından olduğuna karar vermektedir.

Verilen örnekte öklid Uzaklığı (euclidean distance) kullanılarak uzaklık ölçümü yapılmıştır. Bundan başka yöntem olarak, Manhattan ve Minkowski uzaklıkları da kullanılabilmektedir.

3.13'de verilen formül öklid uzaklığının, iki boyutlu uzayı ele alındığında, aslında pisagor teoreminin uygulaması olduğu görülmektedir.

$$|M,N| = \sqrt{(x1-x2)^2+(y1-y2)^2} \tag{3.13}$$

3.14'te verilen formül Manhattan Uzaklığı, noktalar arasındaki mutlak uzaklıkların toplamı alınarak hesaplanmaktadır.

$$|M,N| = \sum_{i=1}^{k} |x_i - y_i| \tag{3.14}$$

3.15'te verilen formül Minkowski Uzakığı, q sayıda değişkene bağlı bir uzaklık hesaplanmak istendiğinde kullanılmaktadır. Formülde q yerine 2 değeri verildiğinde aslında Öklid bağlantısı elde edilmektedir.

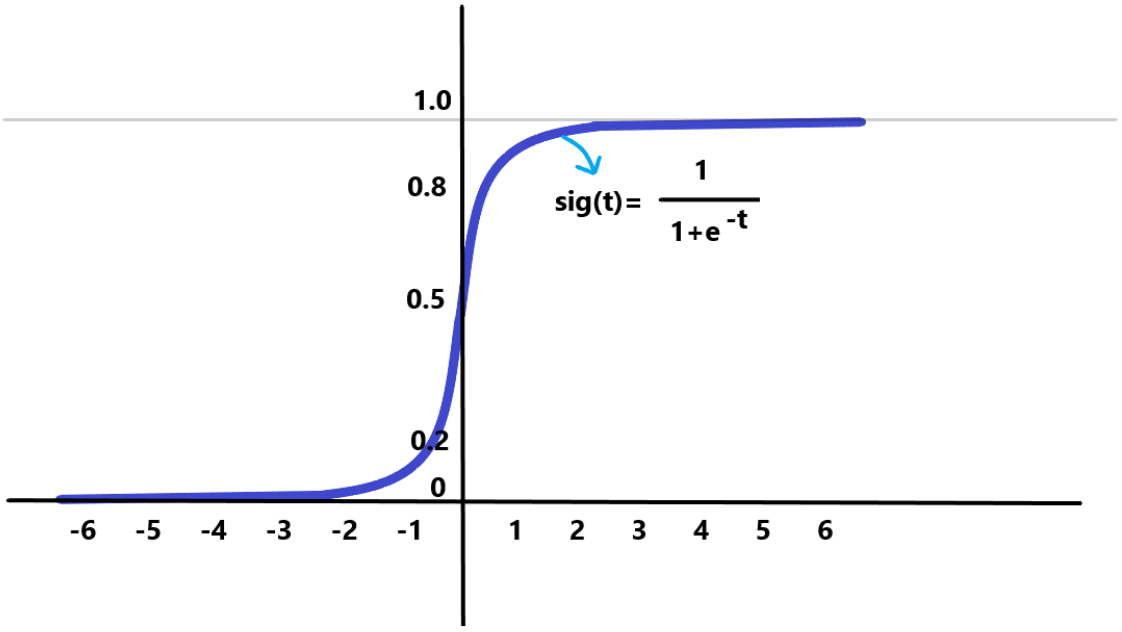
$$|M,N| = (\sum_{i=1}^{k}(|x_i - y_i|)^q)^{1/q} \tag{3.15}$$

3.3.3 Lojistik Regresyon

Lojistik regresyon, bağımlı değişken olarak kategorik bir değişkenin kullanıldığı, regresyon problemine benzemektedir. Her ne kadar regresyon şeklinde nitelendirilse de aslında bir sınıflandırma işlemi yapılmaktadır. Yaygın olarak doğrusal sınıflandırma problemlerinde kullanılmaktadır.

Lojistik regresyondan, cevap değişkeninin kategorik olarak ikili veya daha fazla kategoriler şeklinde olduğu durumlarda, açıklayıcı değişkenlerle birlikte neden-sonuç ilişkisini belirlemede yararlanılmaktadır. Bu regresyon yönteminde açıklayıcı değişkenlerin durumlarına göre cevap değişkeninin beklenen değerlerinin olasılıkları elde edilmektedir.

Elde etmek istediğimiz bir olasılık değerini [0,1] arasında olması için Şekil 3.45'te görülen lojistik fonksiyonundan (sigmoid fonksiyonu) yararlanıyoruz.



Şekil 3.45. Lojistik fonksiyon grafiği

Veri yapılarına göre lojistik model kurmak için 3.16'daki formüllerden yararlanılabilir [40]. Burada $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$, bağımsız değişkenlerin doğrusal kombinasyonunu

göstermektedir. P(X), karakteristik özelliğinin var olma olasılığıdır. " 1-P(X) " ise var olmama olasılığını vermektedir.

$$\ln\left(\frac{P(X)}{1-P(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n = t$$

$$olasilikOrani = \frac{P(X)}{1-P(X)} = e^{\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_n X_n} = e^t \tag{3.16}$$

$$P(X) = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}} = sig(t)$$

Formülde olasılık oranının (odds ratio) P(X)/1-P(X) olarak hesaplandığında her bir parametrenin e^{β} değerleri olasılık oranı olarak ele alınmaktadır. β_n katsayısının önem derecesi aynı zamanda n. olasılık oranını e^{β_n}'nın da önem derecesi olduğu görülmektedir.

3.4 Evrişimsel Sinir Ağları - CNN

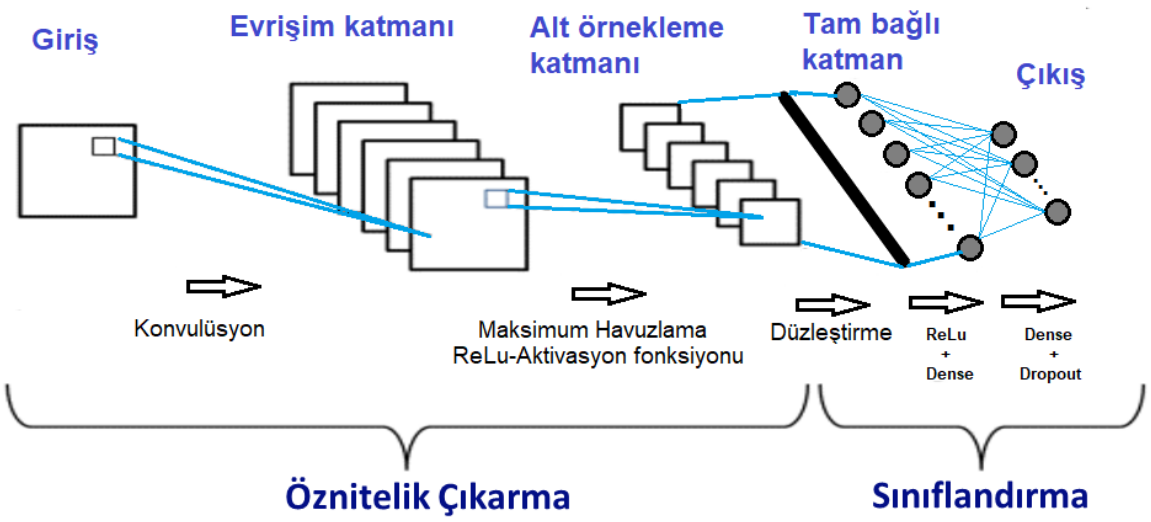
Evrişimli sinir ağı (Convolutional Neural Networks - CNN), imgeleri analiz etmek için en yaygın olarak uygulanan bir derin sinir ağları sınıfıdır. İmge ve video tanıma, imge sınıflandırma, tıbbi imge analizi, doğal dil işleme [41], beyin-bilgisayar arayüzleri [42], ve finansal zaman serileri [43] alanlarında CNN uygulamaları kullanılmaktadır.

CNN'ler, ileri beslemeli bir yapay sinir ağıdır ve çok katmanlı algılayıcıların düzenlenmiş versiyonlarıdır. CNN aynı zamanda çok katmanlı algılayıcılar (multilayer perceptrons -MLPs) olarak da adlandırılmaktadırlar [44]. Çok katmanlı algılayıcılar, tamamen bağlı ağlar olarakta bilinmektedir. Bunun anlamı, bir katmandaki her nöron, bir sonraki katmandaki tüm nöronlara bağlı olmasıdır. Bu ağların tamamen bağlı olmaları onları aşırı veri uydurmaya yatkın hale getirirmektedir. Bunu önlemek için eğitim sırasında cezalandırma parametreleri kullanma (ağırlıkları azalmak gibi), bazı bağlantının kırpılması (bağlantıları atlama, bırakma (dropout) vb.) işlemler yapılmaktadir.

CNN, Sinir Ağları ile karşılaştırıldığında, temel fark, genel matris çarpımı yerine, CNN'in katmanlarından en az birinde evrişim işlemini kullanmasıdır. Evrişim işlemi, üçüncü bir sinyal üretmek için iki sinyali veya işlevi birleştirmenin matematiksel bir yolu olarak

tanımlanabilmektedir. Evrişime ilişkin ilk argüman genellikle imgenin gri düzeyi olan ve işlenecek imgenin ham piksellerini ifade etmektedir ve girdi olarak adlandırılmaktadır. İkinci argüman aynı zamanda filtre veya işaret olarak da adlandırılan çekirdektir (kernel). Çıktı işlevi gelecekteki eşlemedir. Filtreleme işlemi, işlenmiş olanla aynı konumda, komşu piksellerin ağırlıklı toplamına eşit gri seviye değeri ile yeni bir piksel üretir.

Bir CNN'nin girdisi, bir dizi imgedir ve bir evrişimli katman birden fazla filtre kullanır. Her filtre giriş imgelerinin genişliği ve yüksekliği boyunca kaydırılarak bir aktivasyon haritası oluşturulur. Filtrenin giriş boyunca kıvrılması, giriş işlevi ve filtre arasında nokta çarpımını hesaplamaya eşdeğer olduğundan ağ, girişin belirli bir özelliğinin varlığında etkinleşen filtreleri arar. Bir evrişim düzeyinin çıktısı, tüm filtreler için tüm etkinleştirme haritalarının toplamıdır; çıktı hacminin her pikseli, girdinin sadece küçük bir bölgesine bakan ve aynı aktivasyon haritasında nöronlarla parametreleri paylaşan bir nöronun tepkisi olarak da yorumlanabilir. Filtreleme işlemi nedeniyle, küçük çekirdekler ve adımlarla elde edilen 'seyrek etkileşim' ve aynı çekirdeğin birden fazla işlev için kullanılmasını ifade eden 'parametre paylaşımı' şeklinde CNN'nin iki ana avantajı vardır. Sonuç olarak, evrişim matris çarpımından çok daha etkili olmaktadırlar [24].

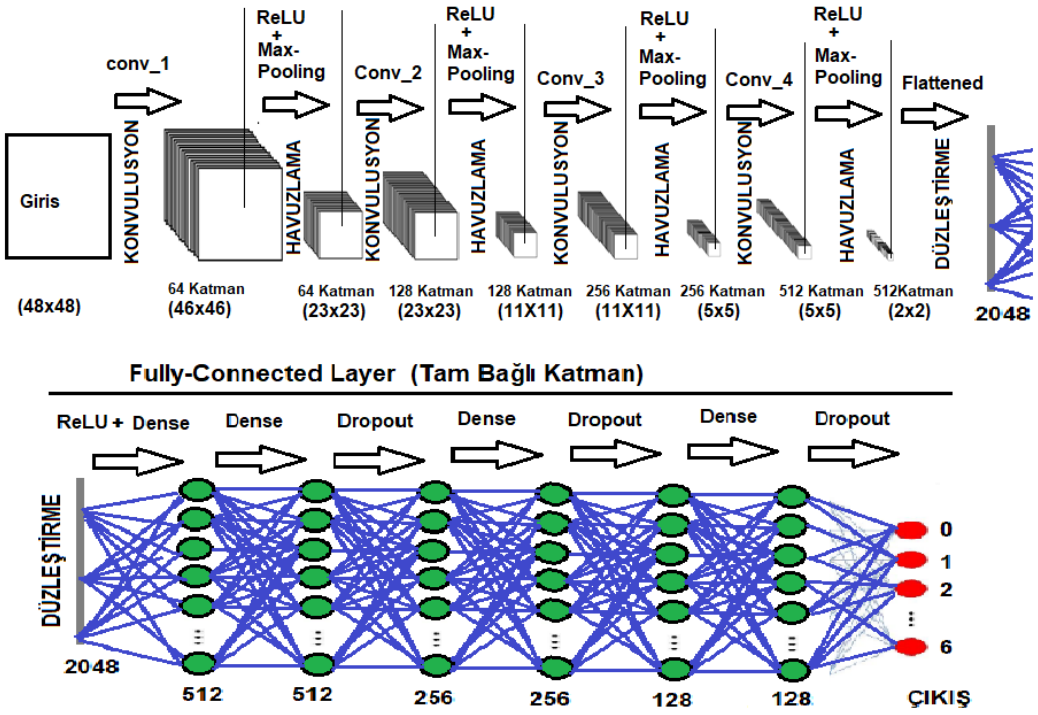


Şekil 3.46. Temel bir evrişimli sinir ağı (CNN) mimarisinin şematik diyagramı [45]

En genel haliyle CNN mimarisi blok diyagramı Şekil 3.46'da gösterilmiştir. En temel CNN mimarisi, evrişimsel katmana (convulation layer) , alt örnekleme katmanına (sub-sampling layer) ve tamamen bağlı katmana (full-connected layer) sahiptir. [46].

```
## Model olusturulma
cnnModel = Sequential()
cnnModel.add(Conv2D(oznitelikSayisi,kernel_size=(3, 3),
                    activation='relu', input_shape=girisBoyut,
                    data_format='channels_last'))
cnnModel.add(Conv2D(oznitelikSayisi,kernel_size=(3, 3),
                    activation='relu', padding='same'))
cnnModel.add(BatchNormalization())
cnnModel.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
cnnModel.add(Dropout(0.5))
cnnModel.add(Flatten())
cnnModel.add(Dense(2*2*2*oznitelikSayisi, activation='softmax'))
cnnModel.add(Dropout(0.4))
cnnModel.summary()

## Model derleme
cnnModel.compile(loss='categorical_crossentropy',
                 metrics=['accuracy'], optimizer='adam')
## Model egitilme
hist = cnnModel.fit(X_egitim,y_egitim,batch_size=7,epochs=100,
                    verbose=1,validation_data=(X_test, y_test) )
```

Şekil 3.47. Evrişimsel sinir ağı modeli python programı



Şekil 3.48. Oluşturulan evrişimsel sinir ağı modeli blok diyagramı

Şekil 3.47 'de CNN modeli oluşturmak için en temel komutlar gösterilmiştir.

Bu çalışmada, CNN katman sayıları artırılarak test edilmiş ve en iyi başarının elde edildiği katman sayısı bütün CNN uygulamalarında kullanılmıştır.

Ekler bölümünde verilen Program C.1 de verilen python programı ile tensorflow [8] ve keras [9] kütüphaneleri kulanılarak oluşturulmuş olan çok katmanlı bir CNN modeli Şekil 3.49'da gösterilmiştir. Modele ait blok diyagram ise Şekil 3.48'te verilmiştir.

CNN ile ilgili temel kavramlar şunlardır;

3.4.1 Evrişim katmanı (Convolution Layer)

Evrişim katmanı, boyutlarına göre giriş imgesi tarayarak evrişim işlemleri gerçekleştiren filtrelerin kullanıldığı katmandır. Hiperparametreleri, filtre boyutu ve adımları içerir. Ortaya çıkan çıktıya, öznitelikleri gösteren özellik haritası denir. Evrişim (Convolution) işlemi, Şekil 3.50'de görüleceği üzere bir imgeye seçilen bir kernel uygulanarak yapılan matematiksel bir işlemdir. Bu hesaplama ile, giriş imgesinden belirli bir özellik tespit edilmektedir ve o özellik hakkında bilgi veren sonucu elde ediyoruz. Bu sonuca "özellik haritası" denilmektedir.

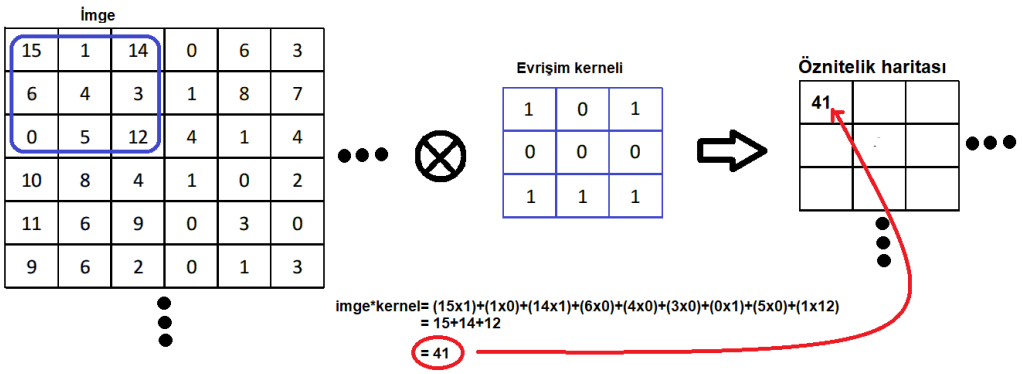
3.4.2 Dolgu (Padding)

Dolgu (Padding), verilerin sınırında ek pikseller eklemek anlamına gelmektedir. İmgedeki piksellere evrişimleri uygularken, köşedeki pikseller ortadakilerden daha az işlenmektedir. Bu, piksellerin aynı miktarda ağırlık almayacağı anlamına gelir. Bu çalışmada ilk evrişim işleminde giriş imgesi 48x48 piksele ve filtre 3x3 piksel boyutlarına sahiptir. Yapılan uygulamada başarı oranını artırdığı görüldüğünden giriş katmanı için dolgu parametresi 'valid' seçilerek dolgu (padding) kullanılmamıştır. 3.17'deki formül uygulandığında 46x46 piksel boyutlarında imge elde edilmiştir. Bu durumda imge boyutlarının küçüldüğü görülmektedir. Çalışmamızda kullanılan 2. evrişim katmanında ise giriş imgesi 46x46 piksele ve filtre 3x3 piksel boyutlarına sahiptir. İmgedeki bütün piksel değerleri işleme dahil edilmek istenildiği için dolgu parametresi 'same' seçilmiş böylece Şekil 3.51'de görüldüğü gibi imge sınırlarında 1'er piksellik dolgu kullanmıştır. 3.16'daki formül uygulandığında ikinci evrişim katmanındaki 46x46 piksel boyutlarında imge elde edilmiştir. Giriş ve çıkış imgelerinin boyutlarının değişmediği görülmektedir.

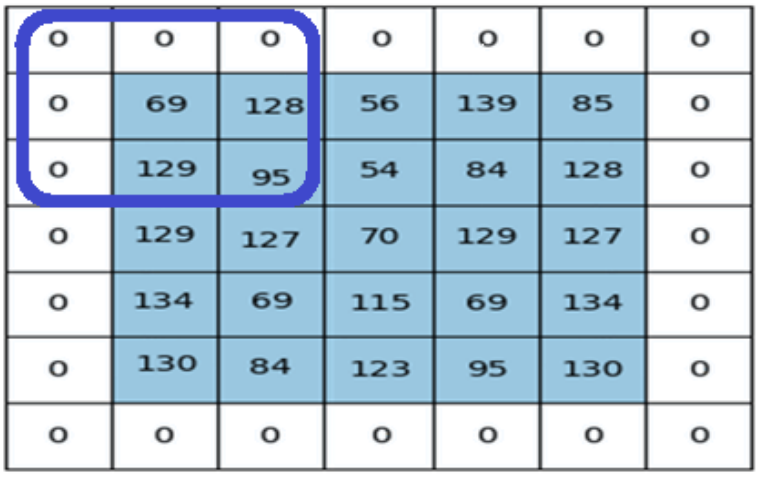
$$Sonuc_b = (giris_b + 2 * dolgu - filtre_b + 1) x (giris_b + 2 * dolgu - filtre_b + 1) \qquad (3.17)$$

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 64)        1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 46, 46, 64)        36928
_________________________________________________________________
batch_normalization (BatchNo (None, 46, 46, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 23, 23, 128)       73856
_________________________________________________________________
batch_normalization_1 (Batch (None, 23, 23, 128)       512
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 128)       147584
_________________________________________________________________
batch_normalization_2 (Batch (None, 23, 23, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 11, 11, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 256)       295168
_________________________________________________________________
batch_normalization_3 (Batch (None, 11, 11, 256)       1024
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 256)       590080
_________________________________________________________________
batch_normalization_4 (Batch (None, 11, 11, 256)       1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 256)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 5, 5, 512)         1180160
_________________________________________________________________
batch_normalization_5 (Batch (None, 5, 5, 512)         2048
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 5, 5, 512)         2359808
_________________________________________________________________
batch_normalization_6 (Batch (None, 5, 5, 512)         2048
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1049088
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 903
=================================================================
Total params: 5,907,015
Trainable params: 5,903,303
Non-trainable params: 3,712
_________________________________________________________________
```

Şekil 3.49. Oluşturulan evrişimsel sinir ağı modeli



Şekil 3.50. Evrişim (convolution) işlemi



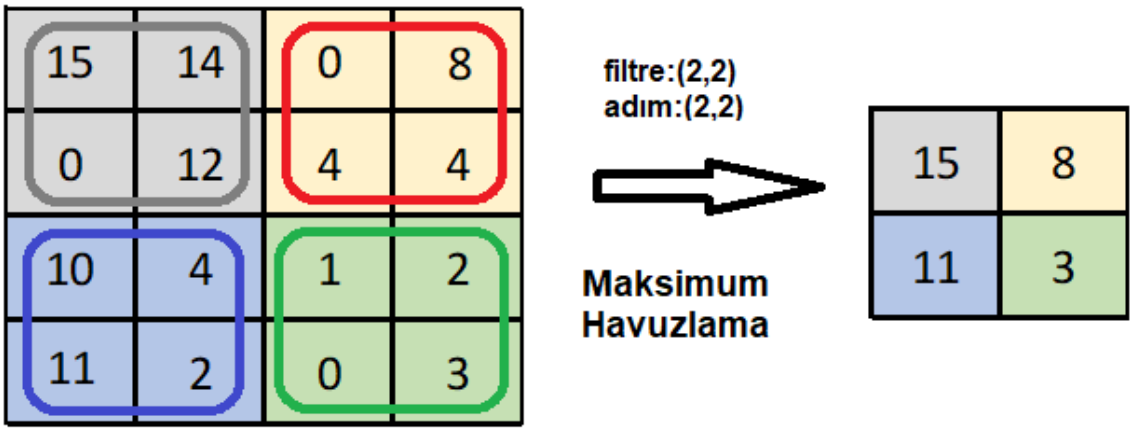
Şekil 3.51. Dolgu (Padding) işlemi

3.4.3 Havuzlama katmanı (Pooling Layer)

Havuzlama katmanı, "Alt Örnekleme Katmanı" olarak da adlandırılır. Bu katmanın temel işlevi, tipik olarak bir evrişim katmanından sonra uygulanan ve bir miktar uzamsal (spatial) değişmezlik yapan bir alt örnekleme işlemidir. Bu katmanda Şekil 3.43'de görüleceği üzere, sonraki evrişim katmanı için giriş boyutu (genişlik x yükseklik) azaltılmaktadır.

Bu katmanda da ortalama ve maksimum değerleri hesaplayan filtreler kullanılır. Bu filtreler evrişim katmanından gelen imgelerdeki piksellerin üzerinde aşağıda açıklandığı şekilde adım değerlerinde uygulanarak, uygulanan yerlerdeki piksellerin maksimum değerlerini veya

ortalama değerlerini alarak işlem yapmaktadır. Bu çalışmada en iyi performans gösterdiği düşünüldüğü için piksellerin maksimum değerini alan yöntem olan maksimum havuzlama kullanılmıştır. Havuzlama genellikle ReLU işleminden sonra gerçekleştirilmektedir.



Şekil 3.52. Maksimum havuzlama (maximum pooling) işlemi

3.4.4 Adımlama (striding)

Bir filtrenin havuzlama işlemi yapmak için her zaman bir piksel hareket etmesi zorunlu değildir. Hem yatay hem de dikey olarak aynı anda iki veya üç piksel atlanarak hareket ettirilebilir. Buna "adım (striding) " denir.

Bu çalışmada kullanılan havuzlama katmanındaki giriş imgesi 46x46 piksel ve poolsize parametresi (2, 2) seçilerek havuzlama katmanı filtresi 2X2 piksel boyutlarında ve strides parametresi (2, 2) seçilerek adım (striding) değeri yatay ve dikeyde 2 piksel atlanarak hareket ettirilmiştir. Dolgu (padding) olarak 1 piksel kullanılmıştır. 3.18'deki formül uygulandığında 23x23 piksel boyutlarında imge elde edilmiştir.

Ayrıca ikinci havuzlama katmanında 11x11, üçüncü havuzlama katmanında 5x5 ve son olarak dördüncü havuzlama katmanı çıkışında 2x2 boyutlarına imge elde edilmiştir.

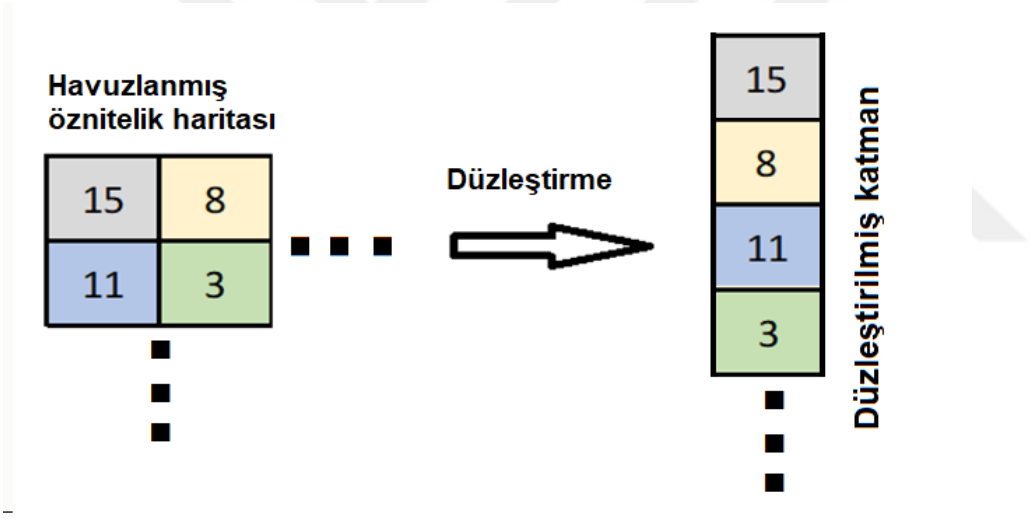
$$boyut = (\frac{giris + 2.dolgu - filtre}{adim} + 1)x(\frac{giris + 2.dolgu - filtre}{adim} + 1)$$

$$boyut = (\frac{46 + 2.1 - 2}{2} + 1)x(\frac{46 + 2.1 - 2}{2} + 1) \tag{3.18}$$

$$boyut = 23x23$$

3.4.5 Düzleştirilmiş Katman (Flattened Layer)

Düzleştirme (Flattening) katmanında , veriler bir sonraki katmana girmek için bir boyutlu bir diziye (tek bir uzun öznitelik vektörüne) dönüştürülmektedir. Kısaca tüm piksel verileri tek bir vektör haline getirilip sınıflandırma işleminin yapıldığı tamamen bağlı katmanla bağlantı kurulmaktadır. Şekil 3.53'te düzleştirme işlemi görülmektedir.



Şekil 3.53. Düzleştirme (Flattining) işlemi

3.4.6 Tamamen Bağlı Katman (Full Connected Layer)

Tamamen bağlı katman (FC), düzleştirilmiş bir giriş üzerinde çalışırak her bir girişin tüm nöronlara bağlandığı katmandır. Bu katman genellikle CNN mimarilerinin sonuna doğru kullanılmaktadır. Ve bulunmak istenen sınıf puanları gibi hedefleri en uygun hale getirmek için kullanılmaktadır.

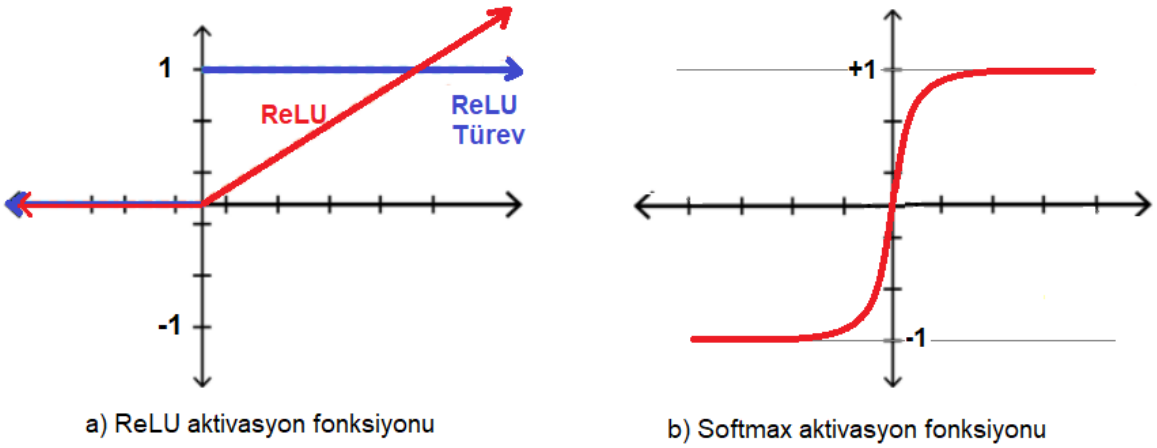
3.4.7 Toplu normalleştirme (Batch Normalization)

Toplu normalleştirme (Batch Normalization), ham veri yerine bir sinir ağının katmanları arasında yapılan bir normalleştirme tekniğidir. Tam veri seti yerine mini gruplar halinde yapılır. Eğitimi hızlandırır ve daha yüksek öğrenme oranları elde ederek öğrenmeyi kolaylaştırır.

3.4.8 Aktivasyon Fonksiyonu (Activation Function)

Aktivasyon fonksiyonu, sinir ağlarının sonuna veya arasında kullanılır. Nöronun çıkışının hangi değerlerde olacağına karar verir. Farklı aktivasyon fonksiyonları vardır. Bu çalışmada aktivasyon fonksiyonu olarak Şekil 3.44 a)'da gösterilmiş olan Düzeltilmiş Doğrusal Birim (Rectified Linear Unit - ReLU) kullanılmıştır. ReLU, aşağıda açıklandığı üzere, kaybolan gradyan probleminin üstesinden gelir ve modellerin daha hızlı öğrenmesine ve daha iyi performans göstermesine olanak tanır. Çalışmada kayıp fonksiyonu olarak ilerde anlatılmış olan kategorik çapraz entropi (categorical crossentropy) kullanılmıştır.Bu kayıp fonksiyon ile önerilen tek aktivasyon fonksiyonu olan Softmax olduğu için sinir ağının sonunda Şekil 3.54 b)'de gösterilmiş olan softmax aktivasyon fonksiyonu kullanılmıştır. Bu fonksiyon, belirli sınıfa ait olma ihtimalini 0–1 aralığında bir olasılık hesabı olarak döndürmektedir.

Kaybolan gradyan problemi, sinir ağlarına, belirli etkinleştirme işlevlerini kullanan daha fazla katman eklendiğinde, kayıp işlevinin gradyanları sıfıra yaklaşarak ağın eğitilmesini zorlaştırmasına denilmektedir.



a) ReLU aktivasyon fonksiyonu

b) Softmax aktivasyon fonksiyonu

Şekil 3.54. ReLU aktivasyon fonksiyonu

3.4.9 Bırakma (dropout)

Bırakma (dropout) işlemi, rastgele seçilen belirli nöron setinin eğitim aşaması sırasında birimleri (yani nöronları) göz ardı etmeyi ifade eder. Yani "görmezden gelerek", bu birimlerin belirli bir ileri veya geri geçiş sırasında dikkate alınmamasıdır. Tamamen bağlı katmandaki nöronlar, eğitim sırasında her bir nöronun bireysel gücünü sınırlayan ve eğitim verilerinin aşırı ayarlanmasına yol açarak birbirlerine bağımlılık geliştirirler. Bu nedenle sinir ağlarının parçalarını kelimenin tam anlamıyla kapatmamız gerekiyor. Bu işlemi gerçekleştirmek için bırakma işlemi kullanılmaktadır.

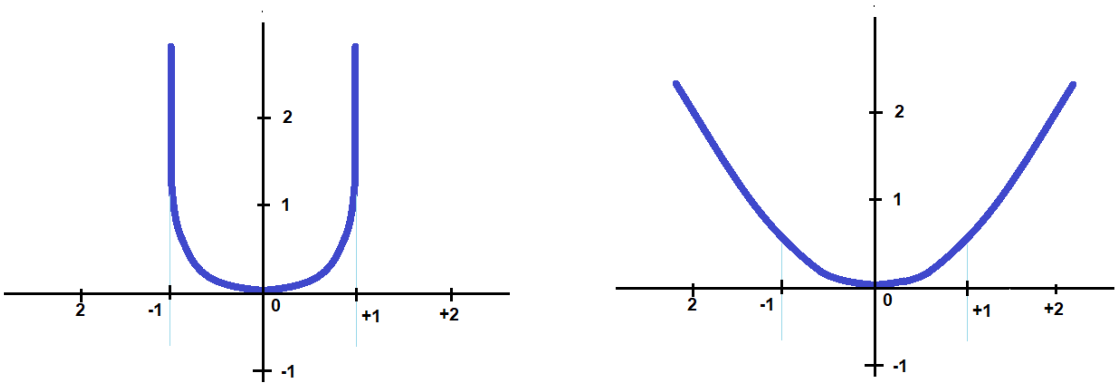
3.4.10 Kayıp Fonksiyonu (Loss Function)

Kayıp fonksiyonu, modelin bir görevi ne kadar iyi gerçekleştirdiğini ölçmektedir. Model tahminlerinde hata arttıkça kayıp yükselmekte, doğruluk arttıkça kayıp sıfıra yakın olmaktadır. Eğitim sırasında, optimize edici, eğitim örneklerindeki kaybı en aza indirmek için modeli ayarlamaktadır. Bir modelin eğitimi ile gradyanın eğimini belirlenmekte ve en alçak nokta nerede olduğu bulunmaktadır.

Model tahmin sonuçları, modelin hedefine eşit olduğunda tüm kayıp fonksiyonları minimum değerdedir. Diğer yandan farklı kayıp fonksiyonları, kesin eşitlik elde edilemediğinde (örneğin gürültü varsa, yeterli girdi bilgisi yoksa veya model mükemmel değilse) modelin nasıl davranacağını değiştirmektedir.

Uygulamada, kayıp fonksiyonunun seçimi çoğunlukla modelin yapması gereken görev tarafından belirlenmektedir. Çok çeşitli kayıp fonksiyonları vardır. Sınıflandırma uygulamalarında kullanılan kategorik çapraz entropi ve regresyon işlemlerinde kullanılan ortalama karesel hata (mean squared error) kayıp fonsiyonları grafiği Şekil 3.55'de gösterilmiştir.

Kategorik Çapraz Entropi (Categorical Crossentropy), iki olasılık dağılımı arasındaki farkı ölçmektedir. İki veya saha fazla sınıfı sınıflandırmak için kullanılan bir kayıp (loss) fonksiyonu olmasından dolayı çalışmamızda kullanılmıştır. Çapraz entropi kaybı fonksiyonu, logaritmik kayıp , log kaybı veya lojistik kayıp olarak bilinmektedir. Eğitim sırasında model



Şekil 3.55. Kayıp fonksiyonları

ağırlıklarını ayarlarken çapraz entropi kaybı kullanılır. Amaç, kaybı en aza indirmektir, yani kayıp ne kadar küçükse model o kadar iyidir. Mükemmel bir modelin çapraz entropi kaybı 0'dır.

$label_i$ doğruluk etiketi, p_i, i'nci sınıf için softmax olasılığı ve n sınıf etiket sayısı olmak üzere kategorik çapraz entropi kayıp fonksiyonu 3.18'deki formül kullanılarak hesaplanmaktadır.

$$loss_{CCE} = \sum_{i=1}^{n} (label_i).\log(p_i) \tag{3.19}$$

3.4.11 İyileştirici (Optimizer)

İyileştirici, python da bulunan keras [9] kütüphanesi ile oluşturulmuş olan CNN modelini derlemek için gereken iki bağımsız parametreden biridir. Diğer paretmeremiz bir önceki konuda anlatılmış olan loss parametresidir. Keras kütüphanesi ile birlikte, "Adam", "SGD", "RMSprop", "Nadam", "Ftrl", "Adamax", "Adagrad", "Adadelta", gibi iyileştiriciler (Optimizers) kullanılabilmektedir.

Adam optimizasyonu, birinci dereceden ve ikinci dereceden momentlerin uyarlanabilir tahminine dayanan stokastik bir gradyan iniş (stochastic gradient descent) yöntemidir. Kingma ve arkadaşlarına [47], göre bu yöntem hesaplama açısından verimlidir, az bellek gereksinimi duymaktadır, gradyanlar köşegen yeniden ölçeklendirilmesi ile değişmemektedir ve veri /

parametreler açısından büyük problemler için çok uygundur. Bu avantajlarından dolayı yapılan çalışmada "adam" optimizasyonu kullanılmıştır.

Gradyan iniş, türevlenebilir bir fonksiyonun yerel minimumunu bulmak için kullanılmaktadır. Gradyan inişi büyük bir veri kümesiyle kullanıldığında sorunlar oluşmaktadır. Bu sorunların üstesinden gelmek için gradyan inişinde, her bir yineleme için tüm veri seti yerine rastgele birkaç örnek kullanan stokastik gradyan inişi kullanılabilmektedir.

4. ARAŞTIRMA BULGULARI

4.1 İmge Veri Setinin Hazırlanması

Bu tez çalışmasında yapılan uygulamalarda python programlama dili kullanılmıştır. Yüz ifadesinden duygusal durumu tanımlamak için 981 adet yüz görüntüsünden oluşan CK+ imge seti kullanılmıştır. Öncelikle CK+ veri setindeki imgeler okunarak bir veri setine dönüştürülmüştür. Bu işlem için Şekil 4.1'de oluşturulmuş python fonksiyonu kullanılmıştır.

```
import os,cv2
import numpy as np

def imge_veriseti_yap(veri_yolu):
    veri_yolu_listesi = os.listdir(veri_yolu)
    img_veri_list=[]
    for veriseti in veri_yolu_listesi:
        imge_list=os.listdir(veri_yolu+'/'+ veriseti)
        for imge in imge_list:
            giris_imge=cv2.imread(veri_yolu+'/'+ veriseti+'/'+imge)
            #giris_imge=cv2.cvtColor(giris_imge,cv2.COLOR_BGR2GRAY)
            giris_imge_resize=cv2.resize(giris_imge,(48,48))
            img_veri_list.append(giris_imge_resize)

    img_veri = np.array(img_veri_list)
    return img_veri

veri_yolu = "...\\veri setleri\\CK+48"
img_veri=imge_veriseti_yap(veri_yolu)
img_veri[0].shape
```

Şekil 4.1. Yüz imgelerinden veri setinin elde edilmesi

Oluşturulmuş olan veri setine her bir görüntüye ait olan duygu ifadesini etiketlemek için yazılan python programı Şekil 4.2'de verilmiştir. Burada her imgedeki duygusal ifade önceden tespit edilerek etiketlenmiş durumdadır.

Bir imgedeki duygusal ifadeyi belirlemek için ilk aşamada imgeyi diğer imgelerden ayıran özellikler olan özniteliklerin tespit edilmesi gerekmektedir. Sonraki aşamada, tespit edilen öznitelikler, sınıflayıcı kullanılarak her bir imgeye ait duygusal ifade tahmin edilecektir. Bu çalışmada "öfke" duygusu için "0" etiketi,"aşağılama" duygusu için "1" etiketi,"tiksinti"

```
sinif_sayisi = 7
ornek_sayisi = img_veri.shape[0]
etiketler = np.ones((ornek_sayisi,),dtype='int64')
etiketler[0:134]  =0 #135
etiketler[135:188]=1 #54
etiketler[189:365]=2 #177
etiketler[366:440]=3 #75
etiketler[441:647]=4 #207
etiketler[648:731]=5 #84
etiketler[732:981]=6 #249
def getLabel(id):
    return ['ofke', 'asagilama', 'tiksinti', 'korku',
            'mutlu', 'uzuntu', 'supriz'][id]
```

Şekil 4.2. İmge setindeki her bir yüze ait duygu fadelerinin etiketlenmesi işlemi

duygusu için "2" etiketi,"korku" duygusu için "3" etiketi,"mutluluk" duygusu için "4" etiketi,"üzüntü" duygusu için "5" etiketi,"sürpriz" duygusu için "6" etiketi kullanılmıştır.

4.2 Kullanılacak Özniteliklerin Tespit Edilmesi

Bir imgeyi diğer imgelerden ayıran öznitelikleri tespit etmek için bir çok yöntem vardır. Bu çalışmada en çok bilinen 4 tanesi kullanılmıştır.

4.2.1 Yönlendirilmiş Gradyanların Histogramı (HOG) ile Özniteliklerin Tespit Edilmesi

Ekler bölümünde verilmiş olan Program B.1' deki python programı ile 981 adet her bir imge ait öznitelikler tespit edilmiştir. Her bir imge 48 yatay ve 48 dikey piksellerden oluşmaktadır.

HOG özellik vektörünü hesaplamak için 36×1'lik vektörler tek bir vektör haline getirilir. Bu vektörün boyutu; 16×16'lık blogun, 48/8-1=5 yatay ve 48/8-1=5 dikey konum, toplamda ise 5x5 = 25 adet konumu vardır. Her 16×16'lık blok 36×1'lik vektör ile temsil edilir. Yani hepsini tek bir vektöründe birleştirdiğimizde 36×25 = 900 boyutlu bir vektör elde edilir.

Böylece her bir imgeye ait 900 adet öznitelik elde edilmiş olur (Çizelge 4.1).

4.2.2 Yerel İkili Örüntüler (LBP) ile Özniteliklerin Tespit Edilmesi

Çizelge 4.1. 981 adet imge için hesaplanan 900 HOG öznitelik

İmge No	0	1	2	...	898	899	Duygu
0	0.143037	0.840238	0.193954	...	0.153133	0.263703	0
1	0.127814	0.852236	0.244243	...	0.011599	0.271402	0
2	0.117705	0.824939	0.135700	...	0.229273	0.308156	0
...	...	...	...	...	...	...	...
978	0.077421	0.047311	0.197834	...	0.338113	0.019889	6
979	0.051797	0.054441	0.072706	...	0.138008	0.017430	6
980	0.046402	0.096903	0.085410	...	0.000000	0.301931	6

Ekler bölümünde verilmiş olan Program B.2' deki python programı ile 981 adet her bir imgeye ait öznitelikler Yerel İkili Örüntü Yöntemi kullanarak tespit edilmiştir. Bu yöntemde son adımda çıktı LBP dizisi üzerinden bir histogram hesaplanmaktadır. 3 x 3'lük komşulukta $2^8 = 256$ olası desen olduğundan, LBP iki boyutlu dizimiz bu nedenle minimum 0 değerine ve maksimum 255 değerine sahiptir ve bu da son özellik vektörümüz olarak LBP kodlarının 256 binarilik histogramının oluşturulmasını sağlamaktadır.

Elde edilen histogramı kullanarak her bir resme ait 256 adet öznitelik tespit edilmiş olur (Çizelge 4.2).

Çizelge 4.2. 981 adet imgeye ait 256 LBP öznitelik

İmge No	0	1	2	...	254	255	Duygu
0	45.0	8.0	41.0	...	6.0	153.0	0
1	44.0	10.0	35.0	...	6.0	146.0	0
2	44.0	8.0	32.0	...	3.0	146.0	0
...	...	...	...	...	...	...	...
978	43.0	5.0	19.0	...	11.0	139.0	6
979	33.0	3.0	17.0	...	10.0	135.0	6
980	37.0	10.0	13.0	...	11.0	134.0	6

4.2.3 Geometrik Özniteliklerin Tespit Edilmesi

Bölüm 3.2.4' de anlatıldığı üzere insan yüzünde oluşan duygusal ifadeleri tespit etmek için kaş, göz, dudak ve burun gibi noktaların arasındaki belirlenmiş oranlar kullanılarak öznitelikler oluşturulmuştur. Bu oranları ve bunlara ait öznitelikleri tespit etmek için Ekler bölümünde verilen Program B.3'deki python programı kullanılmıştır.

Bu programda öncelikle her bir imgedeki yüz bölgeleri tespit edilmiştir. Daha sonra Dlib kütüphanesi kullanılarak yüzdeki göz, kaş, burun, ağız gibi önemli yerleri gösteren 68 nokta belirlenmiştir. Duygusal ifadelerin tespit edilebileceği bazı noktalar arasındaki mesafeler hesaplanmıştır.

Son aşamada hesaplanan mesafeler kullanılarak oranlar elde edilmiş ve bu oranlarla her bir imgeye ait olan 22 tane öznitelik tespit edilmiştir (Çizelge 4.3). Ekler bölümünde verilmiş olan Program B.4' deki python programı ile bu işlemler yapılmıştır.

Çizelge 4.3. 981 adet imgeye ait 22 adet geometrik öznitelikler

İmge No	x0	x1	x2	...	x21	x22	Duygu
0	1	0.574606	0.339014	...	0.480543	0.326465	0
1	1	0.558957	0.219749	...	0.486088	0.325461	0
2	1	0.579380	0.330359	...	0.378463	0.245113	0
...	...	...	...	...	...	...	...
978	1	0.579380	0.200520	...	0.367972	0.804791	6
979	1	0.444220	0.283283	...	0.463855	0.750825	6
980	1	0.579380	0.267516	...	0.394349	0.725814	6

4.2.4 Dalgacık Dönüşümü İle Özniteliklerin Tespit Edilmesi

Veri setindeki (48,48) boyutlarındaki imgelerin ayrık dalgacık dönüşümü alındığında dört adet (26,26) boyutlarında LL,LH,HL,HH matrisler elde edilmektedir. Bu matrislerden LH+HL+HH matrisleri toplandığında o imgeye ait ayırt edici bilgileri elde edilmektedir. Bu matris, tek bir dizi şekline dönüştürüldüğünde 26*26=676 boyutlu bir vektör elde edilmektedir.

Çizelge 4.4. 981 adet imgeye ait 676 adet dalgacık dönüşümü öznitelikler

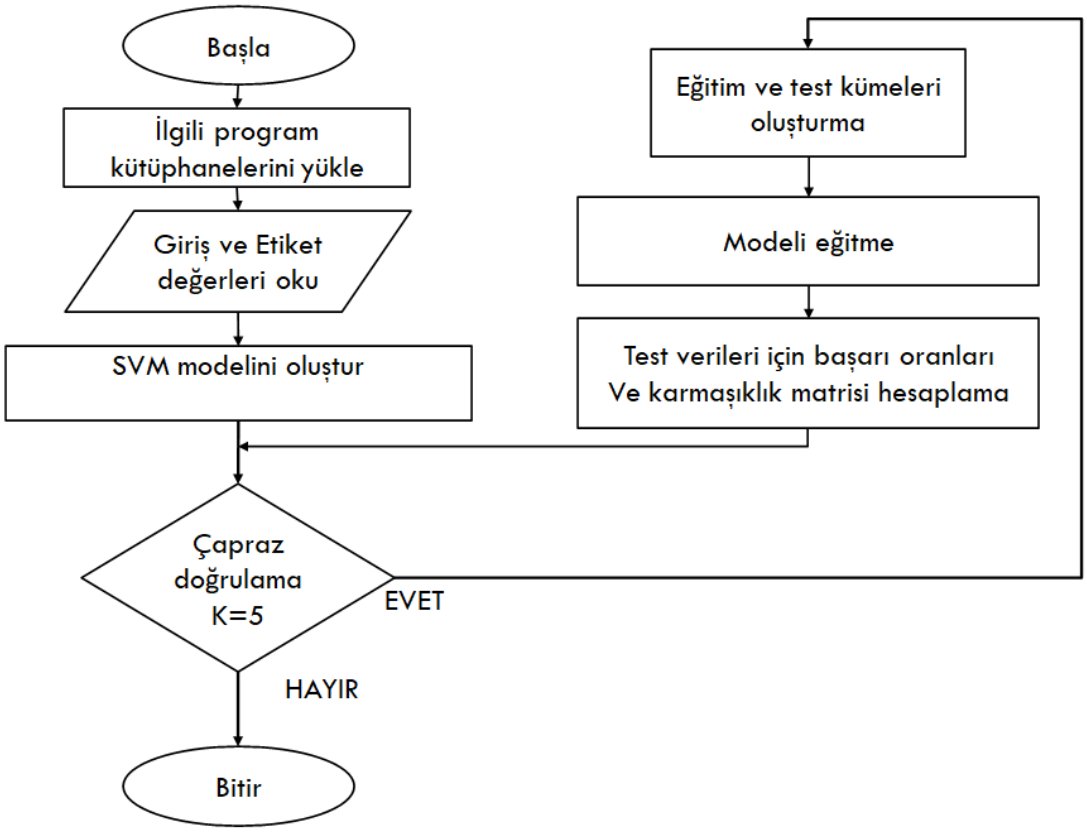
imge No	0	1	2	...	674	675	Duygu
0	-12.7500	4.8750	4.3125	...	-5.000e+03	1.0000	0
1	-2.3750	-0.7500	6.3750	...	-4.000e+03	2.0000	0
2	-16.8125	4.9375	0.8125	...	-4.000e+03	4.2500	0
...	...	...	...	...	...	...	...
978	-45.0625	-4.2500	0.200520	...	2.0625e+02	0.0625	6
979	-23.3750	-3.2500	-2.5000	...	-1.18e-16	0.1250	6
980	-38.7500	0.3750	-5.3750	...	-1.125e+03	-1.0000	6

Bu vektör her bir imgeye ait öznitelik vektörü olarak kullanıldığında her bir imgeye ait 676 adet öznitelik verisi elde edilmiş olur (Çizelge 4.4).

4.3 Sınıflayacılar İle Bulguların Elde Edilmesi

4.3.1 Destek Vektör Makinaları Kullanarak Elde Edilen Bulgular

Elde edilen öznitelik veri seti rastgele olarak karıştırılarak 5 gruba (fold) ayrılmaktadır. Her bir veri grubu %80 eğitim ve %20 test verisi olarak iki gruba ayrılmıştır. Destek Vektör Makinaları(SVM) sınıflayıcı modeli oluşturulmuştur. Bu model ile eğitim verileri kullanılarak model eğitilmiştir. Şekil 4.3'te gösterilen algoritma kullanılarak Şekil 4.4'te verilen python programı oluşturulmuştur.



Şekil 4.3. SVM model algoritması

Eğitilen modele test verileri girilerek modelin başarı oranı hesaplanmıştır. Ayrıca karmaşıklık matrisi hesaplanarak oluşturulan modelin her bir duyguyu tespit etmedeki başarı oranı hesaplanmıştır. Karmaşıklık matrisinde %20 test verisi kullanıldığından ve 981*0.2= 196 adet veri etiketi olacağı açıkça gözükmektedir.

```
from sklearn import svm
import matplotlib.pyplot as mplt
from sklearn.metrics import confusion_matrix
dvm = svm.SVC(kernel='linear') # Linear Kernel

#Capraz dogrulama (Cross-Validation)
skf = StratifiedKFold(n_splits=5,random_state=None, shuffle=False)
skf.get_n_splits(X, y)
print(skf)
skorlar = []
kms=[]
for n_kat,(egtm_indeks,test_indeks) in enumerate(skf.split(X,y)):
    x_egitim, x_test = X[egtm_indeks], X[test_indeks]
    y_egitim, y_test = y[egtm_indeks], y[test_indeks]
    dvm.fit(x_egitim,y_egitim)    # Makineyi egitiyoruz
    y_tahmin = dvm.predict(x_test)
    dog_skor = accuracy_score(y_test, y_tahmin)
    skorlar.append(dog_skor)
    print('\n Katlama'+str(n_kat+1)+'icin dogruluk skoru'+ '->'
                                        + str(dog_skor)+'\n')
 km = confusion_matrix(y_test,y_tahmin)    # Karmasiklik matrisi
    kms.append(km)
print(skorlar)
print('Ortalama dogruluk skoru  :' + str(np.mean(skorlar)))

# Karmasiklik matrisi ciziliyor
kms1=np.asarray(kms)
kmatris=(kms1[0]+kms1[1]+kms1[2]+kms1[3]+kms1[4])/5
df_km = pd.DataFrame(kmatris, ('ofke', 'kucumseme', 'tiksinti',
                     'korku', 'mutlu', 'uzuntu', 'supriz'),
                      ('ofke', 'kucumseme', 'tiksinti', 'korku',
                      'mutluluk', 'uzuntu', 'supriz'))
fig, ax = mplt.subplots(figsize=(12, 4))
ax.xaxis.tick_top()
sn.set(font_scale=1.5) # for label size
sn.heatmap(df_km, annot=True,fmt=".1f", annot_kws={"size": 18},
                 linecolor="k", linewidths=1,cmap="binary",
                 cbar=True) # font size
mplt.xlabel('Gercek durum',fontsize = 18)
mplt.ylabel('Tahmin edilen durum',fontsize = 18)
mplt.show()
```

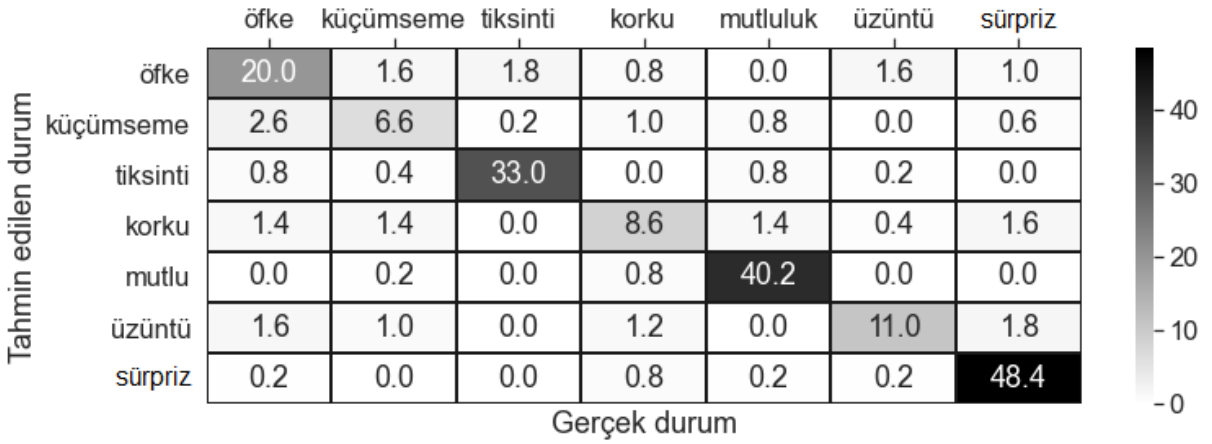
Şekil 4.4. Destek vektör makinaları programı

Öznitelik verisi olarak HOG ile elde edilen veriler kullanıldığında, bu verilerle oluşturulan Destek Vöktör Makinası (SVM) modelinin 5 katlama için ve ortalama olarak başarı oranları Çizelge 4.5'te verilmiştir.

Elde edilen 5 katlama için ortalama karmaşıklık matrisi ise Şekil 4.5'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir. Karmaşıklık matrisinde köşegenlerdeki değerler tam olarak doğru tahmin edilen

Çizelge 4.5. HOG-SVM modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.868020304568528	% 86.80
fold 2	0.8673469387755102	% 86.73
fold 3	0.8979591836734694	% 89.79
fold 4	0.8775510204081632	% 87.75
fold 5	0.7653061224489796	% 76.53
Ortalama	0.8552367139749301	% 85.52



Şekil 4.5. HOG-SVM modelinin karmaşıklık matrisi

duygu etiketi sayısını, yataydaki satırlara ait değerler başka duyguya ait olan, fakat yanlış olarak istenen duygu olarak bulunan etiket sayısını, dikeydeki sütunlara ait değerler ise istenen duygu olarak bulunması gerekirken başka duygu etiketi olarak bulunan etiket sayılarını göstermektedir.

Öfke duygu etiketlerinden 20 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 6,8 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 13,8 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 6,6 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 5,2 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 4,6 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 33 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 6,8 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 2 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 8,6 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 6,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 4,6 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 40,2 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 3,2 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 11 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 5,6 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 2,4 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 48,4 etiket doğru tahmin edilmiştir. HOG-SVM karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 1,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 5 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Öznitelik verisi olarak LBP ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Destek Vöktör Makinası (SVM) modelinin, 5 katlama için ve ortalama olarak başarı oranları Çizelge 4.6'da verilmiştir.

Çizelge 4.6. LBP-SVM modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.4263959390862944	% 42.63
fold 2	0.5306122448979592	% 53.06
fold 3	0.4846938775510204	% 48.46
fold 4	0.42857142857142855	% 42.85
fold 5	0.41836734693877553	% 41.83
Ortalama	0.45772816740909567	% 45.77

Şekil 4.6. LBP-SVM modelinin karmaşıklık matrisi

Elde edilen 5 katlama için ortalama karmaşıklık matrisi ise Şekil 4.6'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 7,6 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 19,2 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 29 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 3,4 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 8,4 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 9,4 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 14 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 21,2 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 21 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 3,6 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 11,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 4,6 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 24 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 17,2 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 8,2 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 1,8 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 14,8 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 13,4 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 35,4 etiket doğru tahmin edilmiştir. LBP-SVM karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 14,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 13,2 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Öznitelik verisi olarak geometrik öznitelikler ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Destek Vöktör Makinası (SVM) modelinin, 5 katlama (fold) için ve ortalama olarak başarı oranları Çizelge 4.7'de verilmiştir.

Çizelge 4.7. Geometrik-SVM modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.8578680203045685	% 85.78
fold 2	0.7908163265306123	% 79.08
fold 3	0.8214285714285714	% 82.14
fold 4	0.8112244897959183	% 81.12
fold 5	0.8061224489795918	% 80.61
Ortalama	0.8174919714078526	% 81.74

Şekil 4.7. Geometrik-SVM modelinin karmaşıklık matrisi

Elde edilen 5 katlama (fold) için ortalama karmaşıklık matrisi ise Şekil 4.7'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 20,2 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 6,6 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 10,8 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 5,2 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 6,6 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 6,2 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 29,6 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 5,6 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 5,6 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 8,6 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 11,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 4,6 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 39,6 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 1,6 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 3 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 9,6 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 7 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 5,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 47,6 etiket doğru tahmin edilmiştir. Geometrik-SVM karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 2,2 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 1,6 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Öznitelik verisi olarak dalgacık dönüşüm yöntemi (DWT) ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Destek Vöktör Makinası (SVM) modelinin, 5 fold için ve ortalama olarak başarı oranları Çizelge 4.8'de verilmiştir.

Çizelge 4.8. DWT-SVM modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.7918781725888325	% 79.18
fold 2	0.7908163265306123	% 79.08
fold 3	0.8112244897959183	% 81.12
fold 4	0.6377551020408163	% 63.77
fold 5	0.5867346938775511	% 58.67
Ortalama	0.723681756966746	% 72.36

Şekil 4.8. DWT-SVM modelinin karmaşıklık matrisi

Elde edilen 5 fold (fold) için ortalama karmaşıklık matrisi ise Şekil 4.8'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 18,2 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 8,6 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 10 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 6,2 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 5,6 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 17 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 26,8 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 8,4 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 7,2 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 4 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 10,8 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 2 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 35,2 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 6 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 8,4 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

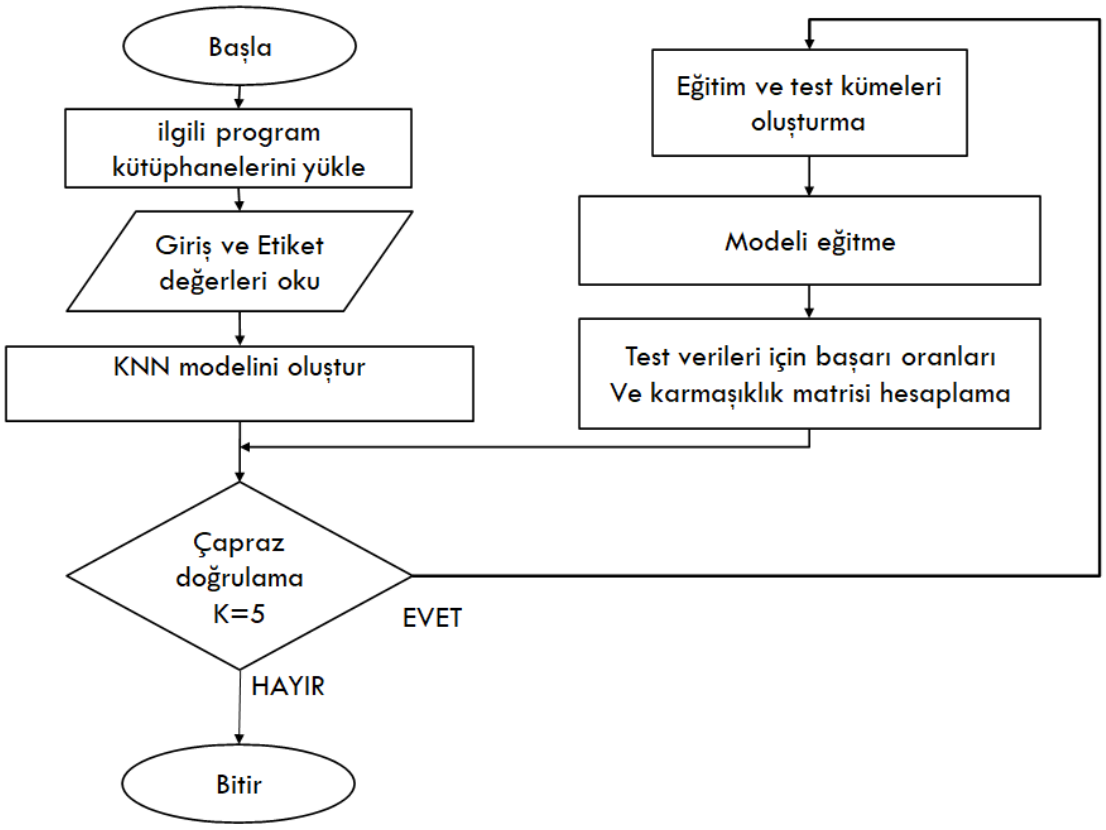
Üzüntü duygu etiketlerinden 6,8 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 9,8 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 9,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 44,8 etiket doğru tahmin edilmiştir. DWT-SVM karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 4,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 5,2 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

4.3.2 KNN Kullanarak Elde Edilen Bulgular

K-En Yakın Komşu Algoritması için Şekil 4.9'da verilen algoritma kullanılarak Şekil 4.10'daki program oluşturulmuştur. Bu programda ilk olarak kklearn kütüphanesinden [7] KNeighborsClassifier() fonksiyonu kullanılarak knn modeli oluşturulmuştur. StratifiedKFold()

fonksiyonu ile veri seti 5 katlama (fold) olarak eğitim ve veri setine bölünmüştür. Oluşturulan model eğitim veri seti ile eğitilmiştir. Eğitilen modele test verileri girilerek her bir katlama (fold) için modelin başarı oranı hesaplanmıştır. Ayrıca her bir katlama için ve ortalama olarak karmaşıklık matrisi hesaplanarak oluşturulan modelin her bir duyguyu tespit etmedeki başarısı test edilmiştir. Karmaşıklık matrisinde verinin %20 test verisi olduğundan dolayı ve 981*0.2= 196 formülasyonundan yola çıkılarak 196 adet veri etiketi üretilmiştir.



Şekil 4.9. KNN model algoritması

Çizelge 4.9. HOG-KNN modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.7918781725888325	% 79.18
fold 2	0.6224489795918368	% 62.24
fold 3	0.6581632653061225	% 65.81
fold 4	0.4642857142857143	% 46.42
fold 5	0.3826530612244898	% 38.26
Ortalama	0.5838858385993992	% 58.38

```
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as mplt
from sklearn.metrics import confusion_matrix
knn = KNeighborsClassifier(n_neighbors=1,p =2)

#Cross-Validation
skf = StratifiedKFold(n_splits=5,random_state=None, shuffle=False)
skf.get_n_splits(X, y)
skorlar = []
kms=[]
for n_kat, (egtm_indeks, test_indeks) in enumerate(skf.split(X,y)):
    x_egitim, x_test = X[egtm_indeks], X[test_indeks]
    y_egitim, y_test = y[egtm_indeks], y[test_indeks]
    knn.fit(x_egitim,y_egitim)    # Makineyi egitiyoruz
    y_tahmin = knn.predict(x_test)
    dog_skor = accuracy_score(y_test, y_tahmin)
    skorlar.append(dog_skor)
    print('\n Katlama'+str(n_kat+1)+' icin dogruluk skoru'+ ' --> '
                                            + str(dog_skor)+'\n')
    km = confusion_matrix(y_test,y_tahmin) # Karmasiklik matrisi
    kms.append(km)
print(skorlar)
print('Ortalama dogruluk skoru  :' + str(np.mean(skorlar)))

# Karmasiklik matrisi ciziliyor
kms1=np.asarray(kms)
kmatris=(kms1[0]+kms1[1]+kms1[2]+kms1[3]+kms1[4])/5
df_km = pd.DataFrame(kmatris, ('ofke', 'kucumseme', 'tiksinti',
                     'korku', 'mutlu', 'uzuntu', 'supriz'),
                      ('ofke', 'kucumseme', 'tiksinti', 'korku',
                      'mutluluk', 'uzuntu', 'supriz'))
fig, ax = mplt.subplots(figsize=(12, 4))
ax.xaxis.tick_top()
sn.set(font_scale=1.5) # for label size
sn.heatmap(df_km, annot=True,fmt=".1f", annot_kws={"size": 18},
                 linecolor="k", linewidths=1,cmap="binary",
                 cbar=True) # font size
mplt.xlabel('Gercek durum',fontsize = 18)
mplt.ylabel('Tahmin edilen durum',fontsize = 18)
mplt.show()
```

Şekil 4.10. KNN programı

Öznitelik verisi olarak HOG yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan KNN modelinin, 5 katlama için ve ortalama olarak başarı oranları Çizelge 4.9'da verilmiştir.

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.11'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Şekil 4.11. HOG-KNN modelinin karmaşıklık matrisi

Öfke duygu etiketlerinden 8,6 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 18,2 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 14 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 7 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 4,8 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 11,8 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 19,4 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 15,8 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 15,6 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 4,2 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 10,6 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 6,8 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 32,6 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 8,6 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 15,8 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 3,6 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 13 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 9,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 39,2 etiket doğru tahmin edilmiştir. HOG-KNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 10,6 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 9 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Çizelge 4.10. LBP-KNN modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.3197969543147208	% 31.97
fold 2	0.34183673469387754	% 34.18
fold 3	0.30612244897959184	% 30.61
fold 4	0.27040816326530615	% 27.04
fold 5	0.32142857142857145	% 32.14
Ortalama	0.31191857453641353	% 31.19

Öznitelik verisi olarak LBP yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan KNN modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.10'de verilmiştir.

Elde edilen 5 katlama için ortalama karmaşıklık matrisi ise Şekil 4.12'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Şekil 4.12. LBP-KNN modelinin karmaşıklık matrisi

Öfke duygu etiketlerinden 5,6 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 21,4 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 23,4 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 4,2 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 7,6 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 11,4 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 7,6 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 27,6 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 24,8 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 1,2 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 13,6 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 14,4 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 15,2 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 26 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 22,4 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 1 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 15,6 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 18,4 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 26,4 etiket doğru tahmin edilmiştir. LBP-KNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 23,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 20,4 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Çizelge 4.11. Geometrik-KNN modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.7715736040609137	% 77.15
fold 2	0.7397959183673469	% 73.97
fold 3	0.7448979591836735	% 74.48
fold 4	0.7653061224489796	% 76.53
fold 5	0.7244897959183674	% 72.44
Ortalama	0.7492126799958563	% 74.92

Öznitelik verisi olarak Geometrik Öznitelikler Yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan KNN modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.11'de verilmiştir.

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.13'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Şekil 4.13. Geometrik-KNN modelinin karmaşıklık matrisi

Öfke duygu etiketlerinden 20,2 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 6,6 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 10,8 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 5,2 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 6,6 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 6,2 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 29,6 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 5,6 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 5,6 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 8,6 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 6,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 2,8 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 39,6 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 1,6 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 3 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 9,6 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 7 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 5,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

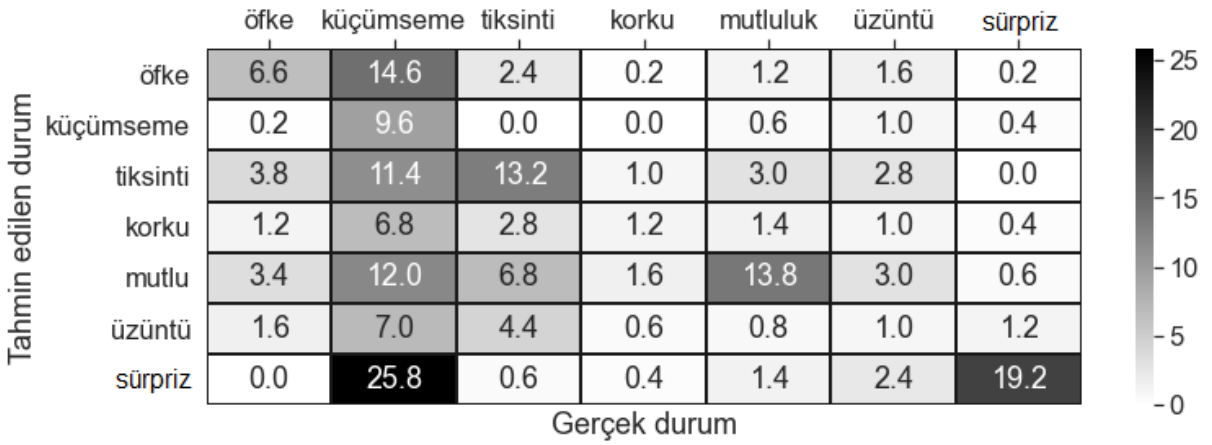
Sürpriz duygu etiketlerinden 47,6 etiket doğru tahmin edilmiştir. Geometrik-KNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 2,2 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 1,6 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Çizelge 4.12. DWT-KNN modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.3604060913705584	% 36.04
fold 2	0.5255102040816326	% 52.55
fold 3	0.3469387755102041	% 34.69
fold 4	0.25510204081632654	% 25.51
fold 5	0.15816326530612246	% 15.81
Ortalama	0.3292240754169688	% 32.92

Öznitelik verisi olarak dalgacık dönüşüm (DWT) yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan KNN modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.12'de verilmiştir.

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.14'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.



Şekil 4.14. DWT-KNN modelinin karmaşıklık matrisi

Öfke duygu etiketlerinden 6,6 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 20,2 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 10,2 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 9,6 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 2,2 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 77,6 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 13,2 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 22 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 17 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 1,2 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 13,6 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 3,2 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 13,8 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 27,4 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 8,4 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

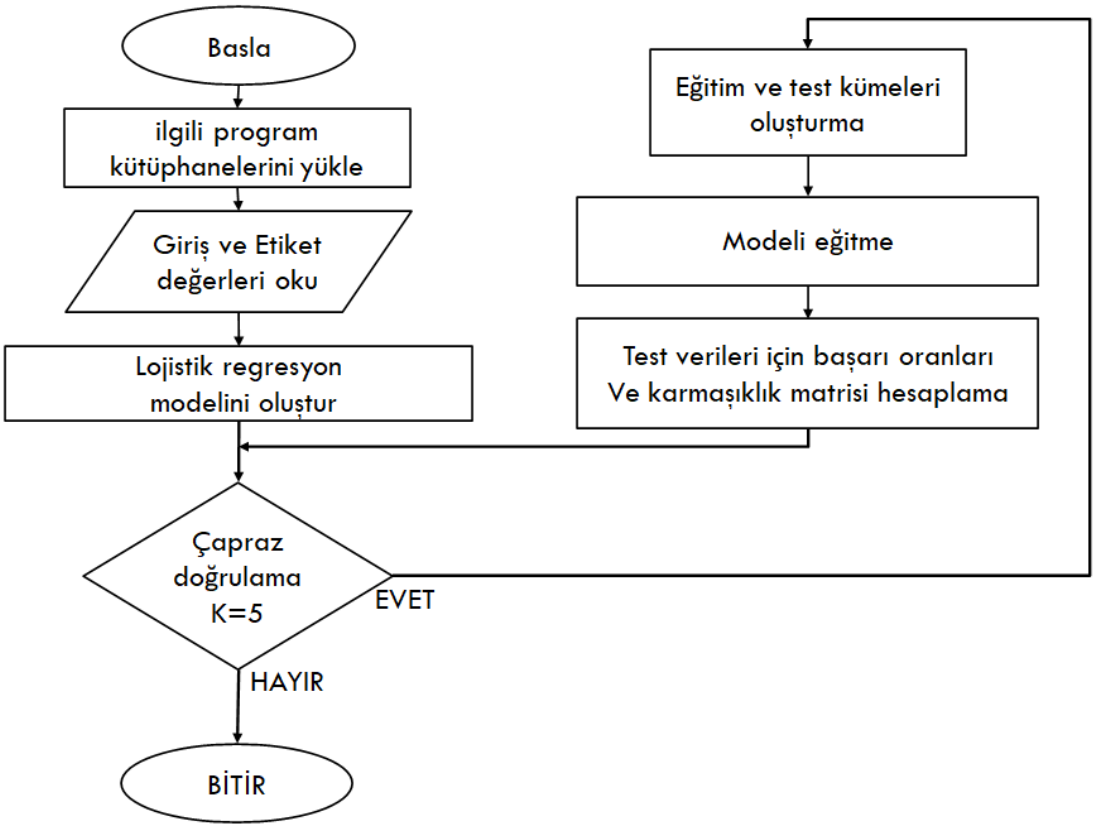
Üzüntü duygu etiketlerinden 1 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 15,6 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 11,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 19,2 etiket doğru tahmin edilmiştir. DWT-KNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 30,6 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 2,8 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

4.3.3 Lojistik Regresyon Kullanarak Elde Edilen Bulgular

Lojistik regresyon modeli için Şekil 4.15'de verilen algoritma kullanılarak oluşturulan Şekil 4.16'daki program kullanılmıştır. Bu programda önce kullanılacak olan öznitelik veri seti eğitim ve test olarak ikiye ayrılmıştır. Sklearn kütüphanesinden Logistic Regression (LR) fonksiyonu kullanılarak Lojistik Regresyon modeli oluşturulmuştur. Oluşturulan model eğitim veri seti ile eğitilmiştir. Eğitilen modele test verileri girilerek modelin başarı oranı hesaplanmıştır. Ayrıca karmaşıklık matrisi hesaplanarak oluşturulan modelin her bir duyguyu tespit etmedeki başarısı belirlenmiştir. Karmaşıklık matrisinde verinin %20 test verisi olduğundan dolayı 981*0.2= 196 formülü gereği 196 adet veri etiketi olacağı hesaplanmıştır.

Öznitelik verisi olarak HOG yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Lojistik Regresyon (LR) sınıflayıcı modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.13'de verilmiştir.



Şekil 4.15. Lojistik regresyon modeli algoritması

Çizelge 4.13. HOG-Lojistik regresyon modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.8730964467005076	% 87.30
fold 2	0.8928571428571429	% 89.28
fold 3	0.8979591836734694	% 89.79
fold 4	0.8877551020408163	% 88.77
fold 5	0.7653061224489796	% 76.53
Ortalama	0.8633947995441831	% 86.33

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.17'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

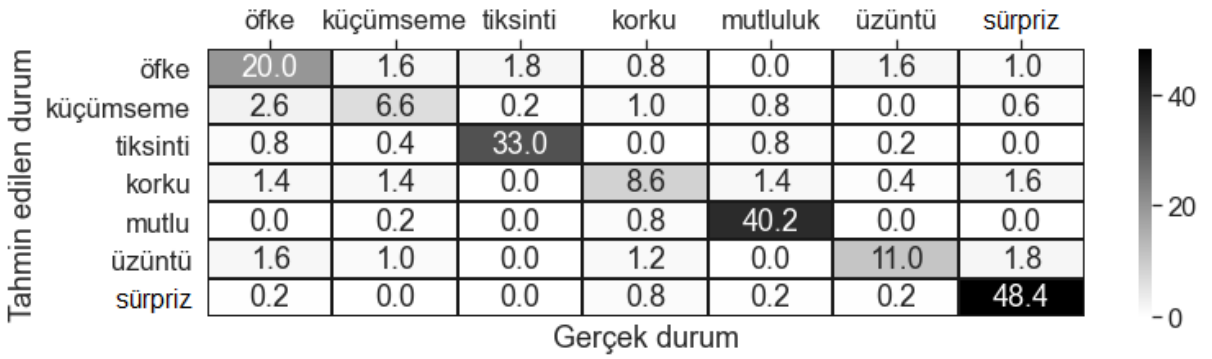
Öfke duygu etiketlerinden 20 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 6,8 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 6,6

```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as mplt
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=1e5,penalty='l1',tol=0.005,solver='saga')

#Cross-Validation
skf = StratifiedKFold(n_splits=5,random_state=None, shuffle=False)
skorlar = []
kms=[]
for n_kat, (egtm_indeks, test_indeks) in enumerate(skf.split(X,y)):
    x_egitim, x_test = X[egtm_indeks], X[test_indeks]
    y_egitim, y_test = y[egtm_indeks], y[test_indeks]
    LR.fit(x_egitim,y_egitim)    # Makineyi egitiyoruz
    y_tahmin = LR.predict(x_test)
    dog_skor = accuracy_score(y_test, y_tahmin)
    skorlar.append(dog_skor)
    print('\n katlama'+str(n_kat+1)+' icin dogruluk skoru'+ ' --> '
                                + str(dog_skor)+'\n')
    km = confusion_matrix(y_test,y_tahmin)   # Karmasiklik matrisi
    kms.append(km)
print(skorlar)
print('Ortalama dogruluk skoru :' + str(np.mean(skorlar)))

# Karmasiklik matrisi ciziliyor
kms1=np.asarray(kms)
kmatris=(kms1[0]+kms1[1]+kms1[2]+kms1[3]+kms1[4])/5
df_km = pd.DataFrame(kmatris, ('ofke', 'kucumseme', 'tiksinti',
                     'korku', 'mutlu', 'uzuntu', 'supriz'),
                      ('ofke', 'kucumseme', 'tiksinti', 'korku',
                      'mutluluk', 'uzuntu', 'supriz'))
fig, ax = mplt.subplots(figsize=(12, 4))
ax.xaxis.tick_top()
sn.set(font_scale=1.5) # for label size
sn.heatmap(df_km, annot=True,fmt=".1f", annot_kws={"size": 18},
                 linecolor="k", linewidths=1,cmap="binary",
                 cbar=True) # font size
mplt.xlabel('Gercek durum',fontsize = 18)
plt.ylabel('Tahmin edilen durum',fontsize = 18)
plt.show()
```

Şekil 4.16. Lojistik regresyon programı

adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 6,6 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 5,2 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 4,6 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.



Şekil 4.17. HOG-Lojistik regresyon modelinin karmaşıklık matrisi

Tiksinti duygu etiketlerinden 33 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 2,2 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 2 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 8,6 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 6,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 4,6 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 40,2 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 3,2 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 11 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 5,6 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 2,4 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 48,4 etiket doğru tahmin edilmiştir. HOG-LR karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 1,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 5 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Çizelge 4.14. LBP-Lojistik regresyon modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.48223350253807107	% 48.22
fold 2	0.6326530612244898	% 63.26
fold 3	0.5408163265306123	% 54.08
fold 4	0.5357142857142857	% 53.57
fold 5	0.5102040816326531	% 51.02
Ortalama	0.5403242515280223	% 54.03

Şekil 4.18. LBP-Lojistik regresyon modelinin karmaşıklık matrisi

Öznitelik verisi olarak LBP yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Lojistik Regresyon (LR) sınıflayıcı modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.14'te verilmiştir.

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.18'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 6 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 20,8 adet başka bir duyguya ait etiket yanlış

olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 15 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 4,6 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 7,2 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 6,6 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 18,8 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 16,4 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 22,4 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 2 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 10,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 6,8 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

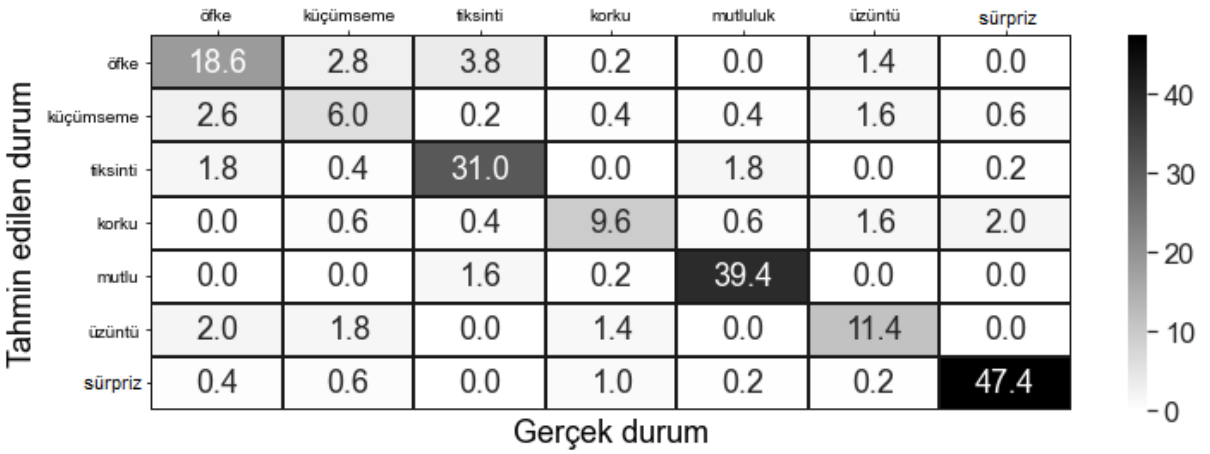
Mutluluk duygu etiketlerinden 32 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 9,2 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 13 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 1,6 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 15 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 41 etiket doğru tahmin edilmiştir. LBP-LR karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 8,8 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 18,4 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Çizelge 4.15. Geometrik-Lojistik regresyon modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.8527918781725888	% 85.27
fold 2	0.826530612244898	% 82.65
fold 3	0.8214285714285714	% 82.14
fold 4	0.8418367346938775	% 84.18
fold 5	0.8214285714285714	% 82.14
Ortalama	0.8328032735937014	% 83.28



Şekil 4.19. Geometrik-Lojistik regresyon modelinin karmaşıklık matrisi

Öznitelik verisi olarak Geometrik Öznitelik yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Lojistik Regresyon (LR) sınıflayıcı modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.15'te verilmiştir.

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.19'da verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 18,6 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 8,2 adet başka bir duyguya ait

etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 6,8 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 6 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 6,6 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 5,8 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 31 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 4,2 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 6 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 9,6 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 5,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 3,2 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

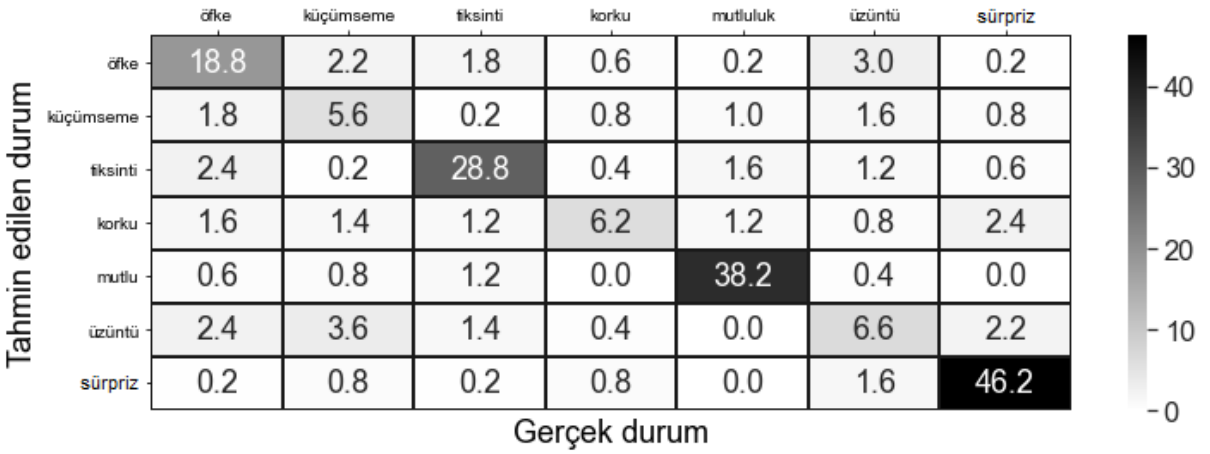
Mutluluk duygu etiketlerinden 39,4 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 1,8 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 3 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 11,4 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 5,2 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 4,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 47,4 etiket doğru tahmin edilmiştir. Geometrik-LR karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 2,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 2,8 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Çizelge 4.16. DWT-Lojistik regresyon modelinin doğruluk skoru

SONUCLAR	Modelin doğruluk skoru	Başarı Oranı
fold 1	0.7868020304568528	% 78.68
fold 2	0.8214285714285714	% 82.14
fold 3	0.826530612244898	% 82.65
fold 4	0.75	% 75.00
fold 5	0.6479591836734694	% 64.79
Ortalama	0.7665440795607583	% 76.65



Şekil 4.20. DWT-Lojistik regresyon modelinin karmaşıklık matrisi

Öznitelik verisi olarak Dalgacık Dönüşüm (DWT) Öznitelik yöntemi ile elde edilen veriler kullanıldığında bu verilerle oluşturulan Lojistik Regresyon (LR) sınıflayıcı modelinin, 5 katlama için ve ortalama olarak başarı oranı Çizelge 4.16'da verilmiştir.

Elde edilen 5 katlama için ortalama olarak karmaşıklık matrisi ise Şekil 4.20'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 18,8 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 8 adet başka bir duyguya ait etiket yanlış

olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 9 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 5,6 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 6,2 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 9 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 28,8 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 6,4 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 6 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 6,2 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 8,6 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 3 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 38,2 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 3 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 4 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 6,6 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 10 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 8,6 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 46,2 etiket doğru tahmin edilmiştir. DWT-LR karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 3,6 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 6,2 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

4.4 Evrişimsel Sinir Ağları Yöntemi Kullanarak Elde Edilen Bulgular

Bu çalışmada ekler bölümünde verilen Program C.1'de görüldüğü üzere ilk olarak CK+ imge setinden elde edilen veri seti kullanılarak yeni bir veri seti oluşturulmuştur. Bu veri seti Evrişimsel Sinir Ağları (CNN) ile oluşturulmuş model kullanılarak eğitilmiştir.

Eğitilen modele test verileri girilerek modelin başarı oranı hesaplanmıştır. Ayrıca karmaşıklık matrisi hesaplanarak oluşturulan modelin her bir duyguyu tespit etmedeki başarısı tespit edilmiştir. Karmaşıklık matrisinde verinin %20 test verisi olduğundan dolayı ve 981*0.2= 196 formulü gereği 196 adet veri etiketi elde edilmiştir.

Çizelge 4.17. Asıl imgeler kullanılan CNN modelinin doğruluk skoru

SONUCLAR	Eğitim doğruluğu	Eğitim kaybı	Test doğruluğu	Test kaybı
fold 1 için	0,987245	0,059818	0,959390	0,135441
fold 2 için	0,975796	0,072221	0,943877	0,321693
fold 3 için	0,960509	0,145279	0,877551	1,324558
fold 4 için	0,994904	0,017702	0,979591	0,859405
fold 5 için	0,971974	0,102025	0,959183	0,135367
Ortalama	0,978085	0,079409	0,943918	0,555293

Şekil 4.21. Asıl imgeler kullanılan CNN modelinin karmaşıklık matrisi

Asıl CK+ imge seti kullanılarak oluşturululan veri setinin CNN modeli uygulanması sonucunda 5 katlama için ve ortalama olarak elde edilen başarı oranları Çizelge 4.17' de verilmiştir.

Elde edilen karmaşıklık matrisi ise Şekil 4.21'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 25,4 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 1,4 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 2,6 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 10,2 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 1,8 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 0,8 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 34,4 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 0,8 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 0,8 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 13,6 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 1,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 2,4 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 40,6 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 0,6 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan

görüleceği üzere; 1 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 14,4 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 2,2 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 2,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 46,6 etiket doğru tahmin edilmiştir. Asıl-CNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 3 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 1 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

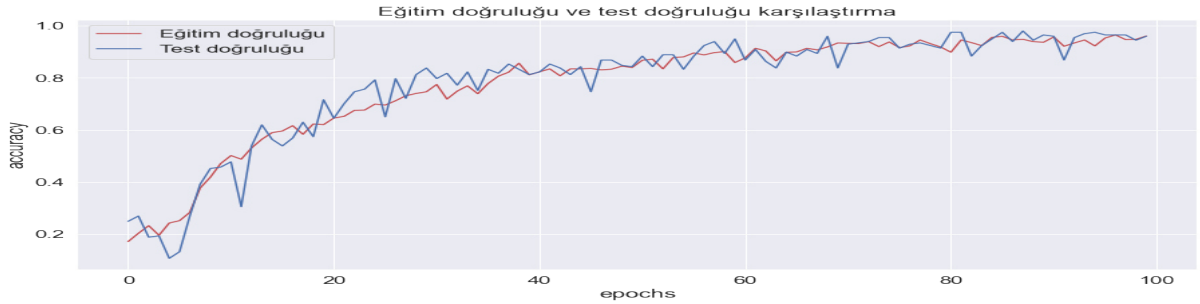
CK+ imge seti kullanılarak elde edilen veri seti vasıtasıyla CNN sınıflayıcı modeli eğitildiğinde, bu modelin doğruluk grafiği Şekil 4.22'de görüldüğü gibi olmaktadır.

Ekler bölümünde verilen Program C.2'de görüldüğü üzere asıl CK+ imge setindeki her imgenin HOG versiyonları bulunmuştur. HOG imgelerinden oluşturulan veri setinin CNN modeli uygulanması sonucunda elde edilen 5 katlama için ve ortalama olarak elde edilen başarı oranları Çizelge 4.18' de verilmiştir.

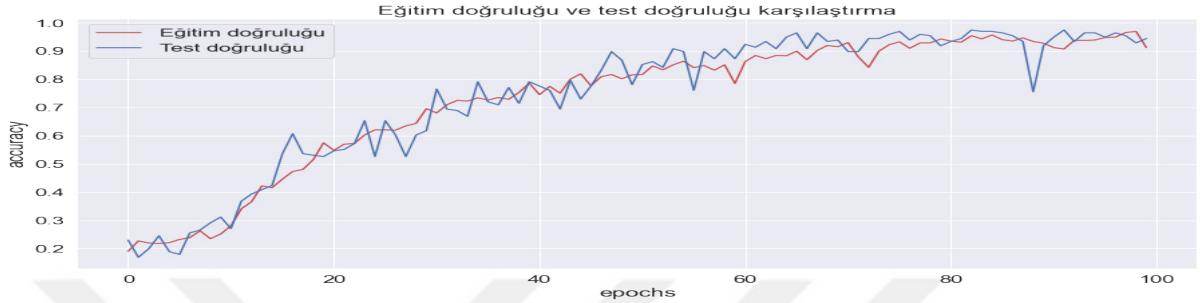
Elde edilen karmaşıklık matrisi ise Şekil 4.23'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 25,6 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 1,2 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 2,2 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 10,6 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 1,4 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey



(a) fold 1



(b) fold 2

(c) fold 3

(d) fold 4

(e) fold 5

Şekil 4.22. Asıl imgeler kullanıldığında elde edilen eğitim doğruluğu ve test doğruluğu grafiği

küçümseme sütunundan görüleceği üzere; 0,6 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 34,6 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 0,6 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 0,2 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 13,8 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 0,6 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 41 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 0,2 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 0,6 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

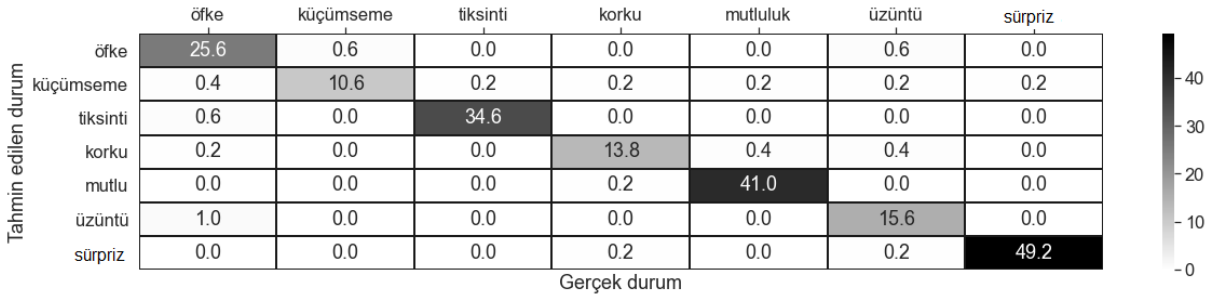
Üzüntü duygu etiketlerinden 15,6 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 1,4 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 49,2 etiket doğru tahmin edilmiştir. HOG-CNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 0,4 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 0,2 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

CK+ imge seti kullanılarak elde edilen HOG veri seti vasıtasıyla CNN sınıflayıcı modeli eğitildiğinde, bu modelin doğruluk grafiği Şekil 4.24'te görüldüğü gibi olmaktadır.

Çizelge 4.18. HOG imgeleri kullanıldığında CNN modelinin doğruluk skoru

SONUCLAR	Eğitim doğruluğu	Eğitim kaybı	Test doğruluğu	Test kaybı
fold 1 için	0,992347	0,014752	0,979695	0,048315
fold 2 için	0,987261	0,047886	0,964285	0,115006
fold 3 için	0,989808	0,039510	0,959183	0,592538
fold 4 için	0,991082	0,022112	0,959183	0,769231
fold 5 için	0,992356	0,019036	0,989795	0,061647
Ortalama	0,990571	0,02865	0,9704272	0,317347



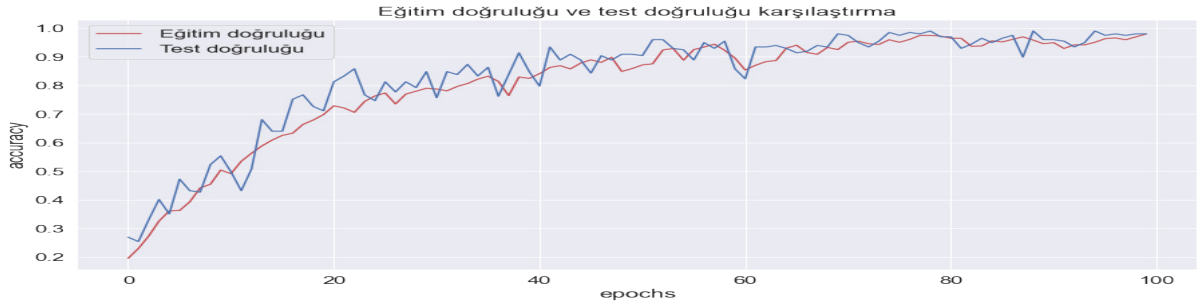
Şekil 4.23. HOG imgeleri kullanıldığında CNN modelinin karmaşıklık matrisi

Ekler bölümünde verilen Program C.3'de görüldüğü üzere asıl CK+ imge setindeki her imgenin LBP versiyonları bulunmuştur. LBP imgelerinden oluşturulan veri setinin CNN modeli uygulanması sonucunda elde edilen 5 katlama için ve ortalama olarak elde edilen başarı oranları Çizelge 4.19' da verilmiştir.

Elde edilen karmaşıklık matrisi ise Şekil 4.25'te verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Öfke duygu etiketlerinden 19 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 7,8 adet başka bir duyguya ait etiket yanlış olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 1 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 9,8 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 2,2 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey



(a) fold 1

(b) fold 2

(c) fold 3

(d) fold 4

(e) fold 5

Şekil 4.24. HOG imgeleri kullanıldığında elde edilen eğitim doğruluğu ve test doğruluğu grafiği

küçümseme sütunundan görüleceği üzere; 6,4 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 32 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 3,2 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 0,2 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 11,6 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 3,2 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 0,2 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 40,2 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan görüleceği üzere; 1 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

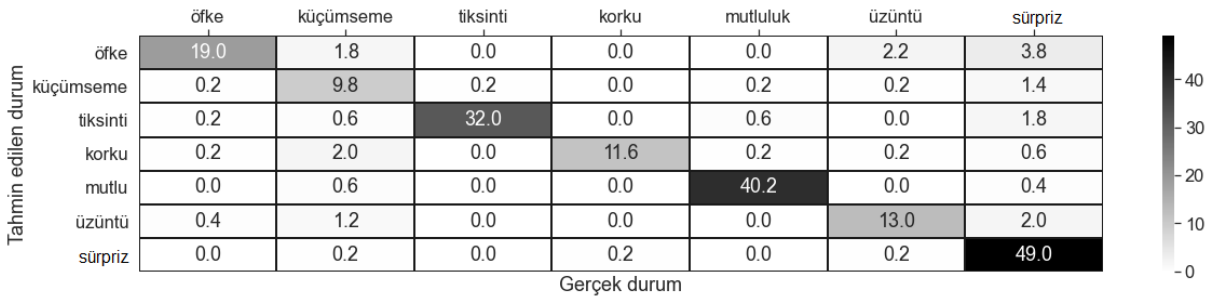
Üzüntü duygu etiketlerinden 13 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 3,6 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 2,8 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 49 etiket doğru tahmin edilmiştir. LBP-CNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 0,6 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 10 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

CK+ imge setinden elde edilen LBP veri seti kullanılarak CNN sınıflayıcı modeli eğitildiğinde, bu modelin doğruluk grafiği Şekil 4.26'da görüldüğü gibi olmaktadır.

Çizelge 4.19. LBP imgeleri kullanıldığında CNN modelinin doğruluk skoru

SONUCLAR	Eğitim doğruluğu	Eğitim kaybı	Test doğruluğu	Test kaybı
fold 1	0,876275	0,635763	0,756345	1,402834
fold 2	0,993630	0,018696	0,974489	0,191384
fold 3	0,806369	1,389928	0,765306	2,377712
fold 4	0,983439	0,043086	0,964285	0,868501
fold 5	0,993630	0,024796	0,989795	0,161123
Ortalama	0,930669	0,422454	0,890044	1,000311



Şekil 4.25. LBP imgeleri kullanıldığında CNN modelinin karmaşıklık matrisi

Ekler bölümünde verilen Program C.4'te görüldüğü üzere asıl CK+ imge setindeki her her imgenin Ayrık Dalgacık Dönüşümü (DWT) yapılmış ve DWT imgeleri bulunmuştur. DWT imgelerinden oluşturulan veri setinin CNN modeli uygulanması sonucunda elde edilen 5 katlama için ve ortalama olarak elde edilen başarı oranları Çizelge 4.20' de verilmiştir.

Elde edilen karmaşıklık matrisi ise Şekil 4.27'de verilmiştir. Uygulama kapsamında kullanılan 196 adet etiket aracılığıyla duygusal durum etiketleri tespit edilmiştir.

Çizelge 4.20. DWT imgeler kullanıldığında CNN modelinin doğruluk skoru

SONUCLAR	Eğitim doğruluğu	Eğitim kaybı	Test doğruluğu	Test kaybı
fold 1 için	0,991071	0,036572	0,984771	0,059084
fold 2 için	0,985987	0,044962	0,959183	0,169066
fold 3 için	0,982165	0,078331	0,923469	0,574603
fold 4 için	0,993630	0,024240	0,979591	0,718496
fold 5 için	0,978343	0,043638	0,959183	0,183005
Ortalama	0,986238	0,045549	0,961239	0,340851

Öfke duygu etiketlerinden 25,8 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay öfke satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış

(a) fold 1

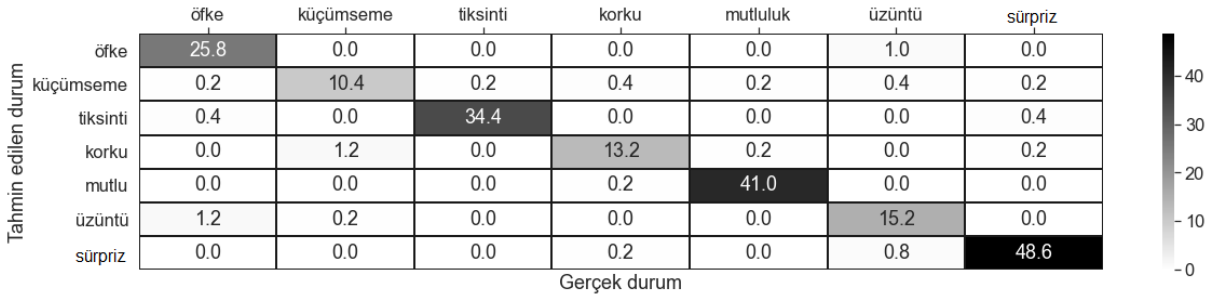
(b) fold 2

(c) fold 3

(d) fold 4

(e) fold 5

Şekil 4.26. LBP imgeleri kullanıldığında elde edilen eğitim doğruluğu ve test doğruluğu grafiği



Şekil 4.27. DWT imgeleri kullanıldığında CNN modelinin karmaşıklık matrisi

olarak öfke duygusu olarak tahmin edilmiştir. Dikey öfke sütunundan görüleceği üzere; 1,8 adet öfke duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Küçümseme duygu etiketlerinden 10,4 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay küçümseme satırından görüleceği üzere; 1,6 adet başka bir duyguya ait etiket yanlış olarak küçümseme duygusu olarak tahmin edilmiştir. Dikey küçümseme sütunundan görüleceği üzere; 1,4 adet küçümseme duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Tiksinti duygu etiketlerinden 34,4 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay tiksinti satırından görüleceği üzere; 0,8 adet başka bir duyguya ait etiket yanlış olarak tiksinti duygusu olarak tahmin edilmiştir. Dikey tiksinti sütunundan görüleceği üzere; 0,2 adet tiksinti duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Korku duygu etiketlerinden 13,2 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay korku satırından görüleceği üzere; 1,8 adet başka bir duyguya ait etiket yanlış olarak korku duygusu olarak tahmin edilmiştir. Dikey korku sütunundan görüleceği üzere; 0,8 adet korku duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Mutluluk duygu etiketlerinden 41 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay mutluluk satırından görüleceği üzere; 0,2 adet başka bir duyguya ait etiket yanlış olarak mutluluk duygusu olarak tahmin edilmiştir. Dikey mutluluk sütunundan

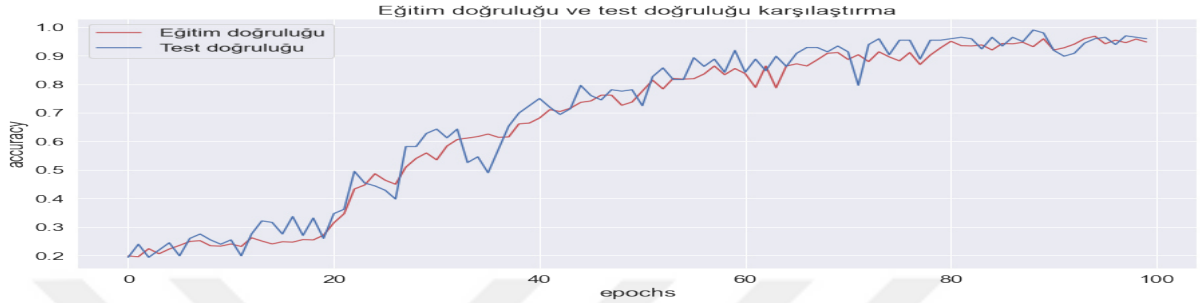
görüleceği üzere; 0,4 adet mutluluk duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Üzüntü duygu etiketlerinden 15,2 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay üzüntü satırından görüleceği üzere; 1,4 adet başka bir duyguya ait etiket yanlış olarak üzüntü duygusu olarak tahmin edilmiştir. Dikey üzüntü sütunundan görüleceği üzere; 2,2 adet üzüntü duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

Sürpriz duygu etiketlerinden 48,6 etiket doğru tahmin edilmiştir. DWT-CNN karmaşıklık matrisinde yatay sürpriz satırından görüleceği üzere; 1 adet başka bir duyguya ait etiket yanlış olarak sürpriz duygusu olarak tahmin edilmiştir. Dikey sürpriz sütunundan görüleceği üzere; 0,8 adet sürpriz duygusu olarak tahmin edilmesi gereken etiket ise başka bir duygu olarak tahmin edilmiştir.

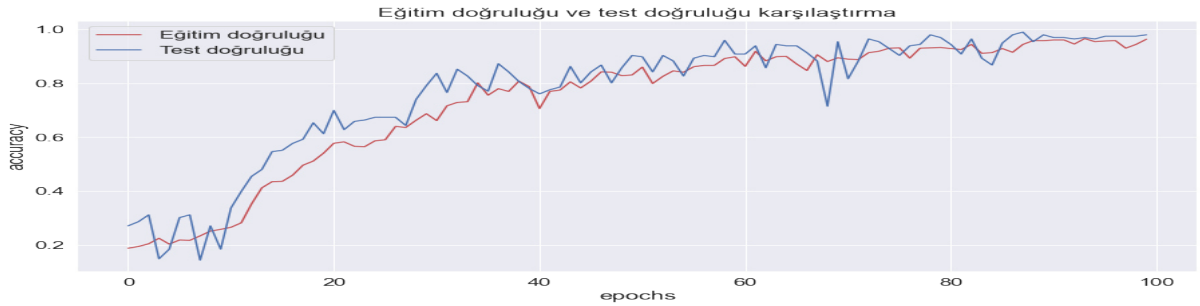
CK+ imge setininden kullanılarak elde edilen DWT veri seti aracılığıyla CNN sınıflayıcı modeli eğitildiğinde, bu modelin doğruluk grafiği Şekil 4.28'de görüldüğü gibi olmaktadır.

(a) fold 1



(b) fold 2

(c) fold 3



(d) fold 4

(e) fold 5

Şekil 4.28. DWT imgeleri kullanıldığında elde edilen eğitim doğruluğu ve test doğruluğu grafiği

5. TARTIŞMA VE SONUÇ

Bu çalışmada kullanılan özniteliklere göre sınıflayıcıların başarı durumları Çizelge 5.1'de gösterilmiştir.

Çizelge 5.1. Doğruluk skoru karşılaştırma tablosu

SONUÇLAR	HOG	LBP	Geometrik	DWT
SVM göre	% 85,5236	% 45,7728	% 81,7492	% 72,3681
KNN göre	% 58,3885	% 31,1918	% 74,9212	% 32,9224
Logistik Reg. göre	% 86,3394	% 54,0324	% 83,2803	% 76,6544
SONUÇLAR	HOG resim	LBP	Asıl	DWT
CNN göre	% 97,0427	% 89,0044	% 94,3918	% 96,1239

Duygusal ifadeleri tespit etmek için en başarılı sınıflama yönteminin CNN olduğu görülmektedir.

En başarısız sınıflayıcı yöntemin KNN sınıflayıcılar olduğu tespit edilmiştir.

Bütün sınıflayıcalar için ise LBP öznitelik verilerini kullanmanın kötü bir tercih olacağı görülmektedir.

Çizelge 5.2'de çalışmada kullanılan yüz ifadelerinden duygu tespit etme yöntemlerinin her bir duygusal ifade için ayrı ayrı başarı oranlarının özniteliklere göre sıralanmış olarak karşılaştırılması yapılmıştır.

Çizelge 5.3'te çalışmada kullanılan yüz ifadelerinden duygu tespit etme yöntemlerinin her bir duygusal ifade için ayrı ayrı başarı oranlarının sınıflayıcılara göre sıralanmış olarak karşılaştırılması yapılmıştır.

Öfke duygusunu tespit etmede en başarılı yöntemin %90,21 skor ile DWT öznitelik imgeleri ve CNN sınıflayıcı yöntemini kullanmak olduğu görülmektedir. En başarısız sonuç ise %11,20 skor ile LBP öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Çizelge 5.2. Özniteliklere göre sıralanmış, duyguları tahmin etme başarı oranları karşılaştırma tablosu

	öfke	aşagılama	tiksinti	korku	mutluluk	üzüntü	sürpriz
HOG-SVM	%59,88	%40,24	%88,70	%44,32	%90,50	%57,89	%87,36
HOG-KNN	%21,07	%29,66	%38,19	%19,44	%65,46	%11,11	%66,66
HOG-LR	%59,88	%40,24	%88,71	%44,33	%90,54	%57,89	%88,32
HOG-CNN	%88,28	%84,13	%97,74	%89,61	%98,09	%86,67	%98,80
LBP-SVM	%13,62	%16,04	%24,91	%13,33	%48,58	%6,00	%56,19
LBP-KNN	%11,20	%18,10	%12,67	%04,11	%23,90	%02,86	%37,61
LBP-LR	%14,35	%25,00	%32,64	%09,26	%59,04	%06,50	%60,12
LBP-CNN	%68,35	%53,26	%90,40	%77,33	%95,26	%67,01	%82,21
DWT-SVM	%49,45	%21,52	%63,21	%23,81	%79,64	%25,76	%81,45
DWT-KNN	%17,84	%10,74	%25,29	%06,45	%27,82	%03,52	%36,50
DWT-LR	%52,51	%26,92	%69,90	%34,83	%84,51	%26,19	%82,50
DWT-CNN	%90,21	%77,61	%97,18	%84,62	%98,56	%80,85	%96,43
Geometrik-SVM	%53,72	%28,89	%72,55	%48,86	%89,59	%42,85	%92,60
Geometrik-KNN	%53,72	%28,89	%72,55	%48,86	%89,59	%42,86	%92,61
Geometrik-LR	%55,36	%33,33	%75,24	%53,33	%89,14	%53,27	%90,11
Asıl-CNN	%86,39	%79,69	%95,56	%79,07	%96,21	%75,79	%92,09

Çizelge 5.3. Sınıflayıcılara göre sıralanmış, duyguları tahmin etme başarı oranları karşılaştırma tablosu

	öfke	aşagılama	tiksinti	korku	mutluluk	üzüntü	sürpriz
SVM-HOG	%59,88	%40,24	%88,70	%44,32	%90,50	%57,89	%87,36
SVM-LBP	%13,62	%16,04	%24,91	%13,33	%48,58	%6,00	%56,19
SVM-Geometrik	%53,72	%28,89	%72,55	%48,86	%89,59	%42,85	%92,60
SVM-DWT	%49,45	%21,52	%63,21	%23,81	%79,64	%25,76	%81,45
KNN-HOG	%21,07	%29,66	%38,19	%19,44	%65,46	%11,11	%66,66
KNN-LBP	%11,20	%18,10	%12,67	%04,11	%23,90	%02,86	%37,61
KNN-Geometrik	%53,72	%28,89	%72,55	%48,86	%89,59	%42,86	%92,61
KNN-DWT	%17,84	%10,74	%25,29	%06,45	%27,82	%03,52	%36,50
LR- HOG	%59,88	%40,24	%88,71	%44,33	%90,54	%57,89	%88,32
LR- LBP	%14,35	%25,00	%32,64	%09,26	%59,04	%06,50	%60,12
LR- Geometrik	%55,36	%33,33	%75,24	%53,33	%89,14	%53,27	%90,11
LR- DWT	%52,51	%26,92	%69,90	%34,83	%84,51	%26,19	%82,50
CNN-Asıl	%86,39	%79,69	%95,56	%79,07	%96,21	%75,79	%92,09
CNN-HOG	%88,28	%84,13	%97,74	%89,61	%98,09	%86,67	%98,80
CNN-LBP	%68,35	%53,26	%90,40	%77,33	%95,26	%67,01	%82,21
CNN-DWT	%90,21	%77,61	%97,18	%84,62	%98,56	%80,85	%96,43

Aşağılama duygusunu tespit etmede en başarılı yöntemin %84,13 skor ile HOG öznitelik imgeleri ve CNN sınıflayıcı yöntemini kullanmak olduğu görülmektedir. En başarız sonuç ise

%10,74 skor ile DWT öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Tiksinme duygusunu tespit etmede en başarılı yöntemin %97,74 skor ile HOG öznitelik imgelerinin, CNN sınıflayıcı yöntemiyle kullanılması sonucu elde edilmiştir. En başarız sonuç ise %12,67 skor ile LBP öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Korku duygusunu tespit etmede en başarılı yöntemin %89,61 skor ile HOG öznitelik imgeleri ve CNN sınıflayıcı yöntemini kullanmak olduğu görülmektedir. En başarız sonuç ise %04,11 skor ile LBP öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Mutluluk duygusunu tespit etmede en başarılı yöntemin %98,56 skor ile HOG öznitelik imgeleri ve CNN sınıflayıcı yöntemini kullanmak olduğu görülmektedir. En başarız sonuç ise %23,90 skor ile LBP öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Üzüntü duygusunu tespit etmede en başarılı yöntemin %86,67 skor ile HOG öznitelik imgelerinin, CNN sınıflayıcı yöntemiyle kullanılması sonucu elde edilmiştir. En başarız sonuç ise %02,86 skor ile LBP öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Şaşkınlık (Sürpriz) duygusunu tespit etmede en başarılı yöntemin %98,80 skor ileHOG öznitelik imgelerinin, CNN sınıflayıcı yöntemiyle kullanılması sonucu elde edilmiştir. En başarız sonuç ise %37,61 skor ile LBP öznitelik veri setinin KNN sınıflayıcı ile kullanılması sonucunda elde edilmiştir.

Genel olarak öznitelik yöntemlerinin başarısı değerlendirildiğinde CNN sınıflayıcı haricinde LBP kullanılmasının bütün duyguları tespit etmede doğru seçim olmayacağı görülmüştür.

Sınıflayıcı olarak ise CNN yönteminin bütün öznitelik yöntemlerinde her bir duygu için istikrarlı bir şekilde başarı gösterdiği görülmüştür.

KAYNAKLAR

[1] D. A. Uğur, "Görüntü İşlemeye Giriş Introduction to Image Processing İçerik," pp. 1–25, 2013.

[2] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Rio, M. Wiebe, P. Peterson, P. Gerard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[4] J. Reback, W. McKinney, jbrockmendel, J. V. den Bossche, T. Augspurger, P. Cloud, gfyoung, Sinhrks, S. Hawkins, A. Klein, M. Roeschke, J. Tratner, C. She, T. Petersen, W. Ayd, MomIsBestFriend, M. Garcia, J. Schendel, A. Hayden, V. Jancauskas, D. Saxton, P. Battiston, A. McMaster, S. Seabold, chris-b1, h-vetinari, S. Hoyer, K. Dong, W. Overmeire, and M. Winkel, *Pandas-dev/pandas: Pandas 1.1.1*, version v1.1.1, Aug. 2020. DOI: 10.5281/zenodo.3993412. [Online]. Available: https://doi.org/10.5281/zenodo.3993412.

[5] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[6] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[9] F. Chollet *et al.* (2015). "Keras," [Online]. Available: https://github.com/fchollet/keras.

[10] S. Bayrakdar, D. Akgün, and İ. Yücedağ, "Yüz ifadelerinin otomatik analizi üzerine bir literatür çalışması a survey on automatic analysis of facial expressions," pp. 383–398, 2016.

[11] M. E. Tenekeci, A. Gümüşçü, and E. Aslan, "Görüntüden opencv ile duygu analizi," pp. 861–865, 2014.

[12] A. Ait Younes, I. Truck, and H. Akdağ, "Color image profiling using fuzzy sets," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 13, no. 3, pp. 343–359, 2005, ISSN: 13000632.

[13] M. Pantic and L. L. Ü. M. Rothkrantz, "Automatic analysis of facial expressions: The state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1424–1445, 2000. DOI: 10.1109/34.895976.

[14] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003, ISSN: 01628828. DOI: 10.1109/TPAMI.2003.1195991.

[15] M. A. Turk and A. P. Pentland, "Face recognition using eigen faces," in *Advances in Intelligent Systems and Computing*, vol. 810, 1991, pp. 855–864, ISBN: 9781612848372. DOI: 10.1007/978-981-13-1513-8_87.

[16] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1064, no. 7, pp. 45–58, 1996, ISSN: 16113349. DOI: 10.1109/34.598228.

[17] Y. L. Tian, T. Kanade, and J. F. Cohn, "Recognizing lower face action units for facial expression analysis," *Proceedings - 4th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2000*, vol. 23, no. 2, pp. 484–490, 2000, ISSN: 10636919. DOI: 10.1109/AFGR.2000.840678.

[18] T. Sim, S. Baker, and M. Bsat, "The cmu pose, illumination, and expression (pie) database," *Proceedings - 5th IEEE International Conference on Automatic Face Gesture Recognition, FGR 2002*, vol. 25, no. 12, pp. 53–58, 2002, ISSN: 01628828. DOI: 10.1109/AFGR.2002.1004130.

[19] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, ISSN: 10636919. DOI: 10.1109/cvpr.2001.990517.

[20] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006, ISSN: 01628828. DOI: 10.1109/TPAMI.2006.244.

[21] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, no. July, pp. 94–101, 2010. DOI: 10.1109/CVPRW.2010.5543262.

[22] U. Ayvaz and H. Gürüler, "Bilgisayar kullanıcılarına yönelik duygusal ifade tespiti the detection of emotional expression towards computer users," pp. 231–239, 2017. DOI: 10.17671/gazibtd.309307.

[23] M. Zubair, A. Ali, S. Naeem, F. Jamal, and C. Chesneau, "Emotion Recognition from Facial Expression Using Machine Vision Approach," *Journal of Applied and Emerging Sciences*, pp. 35–40, 2020, ISSN: 1814-070X. DOI: 10.36785/buitems.jaes.350.

[24] E. Battini Sönmez, "A study on facial expression recognition," *Gazi University Journal of Science*, vol. 30, no. 3, pp. 19–27, 2017, ISSN: 13039709.

[25] C. B. Thacker and R. M. Makwana, "Human Behavior Analysis through Facial Expression Recognition in Images using Deep Learning," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 2, pp. 391–397, 2019. DOI: 10.35940/ijitee.b6379.129219.

[26] W. H. Abdulsalam, R. S. Alhamdani, and M. N. Abdullah, "Facial Emotion Recognition: A Survey," *Ijarcet*, vol. 7, no. 11, pp. 2278–1323, 2018. [Online]. Available: www.ijarcet.org.

[27] R. C. Gonzalez, R. E. Woods, and B. R. Masters, "Digital image processing, third edition," *Journal of Biomedical Optics*, vol. 14, no. 2, p. 029 901, 2009, ISSN: 10833668. DOI: 10.1117/1.3115362.

[28] A. Eldem, H. Eldem, and A. Palalı, "Görüntü işleme teknikleriyle yüüz algılama sistemi geliştirme," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 6, no. 2, pp. 44–48, 2017, ISSN: 2147-3129. DOI: 10.17798/bitlisfen.333984.

[29] C. Poynton, "Frequently-asked questions about color," 1997. [Online]. Available: https://web.archive.org/web/20110409213505/http://www.poynton.com/ColorFAQ.html.

[30] A. De, A. Saha, and M. C. Pal, "A human facial expression recognition model based on eigen face approach," *Procedia Computer Science*, vol. 45, no. C, pp. 282–289, 2015. DOI: 10.1016/j.procs.2015.03.142. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2015.03.142.

[31] R. Mygrapefruit. (). "Twenty-nine of history's most influential scientists in one photograph - now in color!" [Online]. Available: https://io9.gizmodo.com/twenty-nine-of-history-s-most-influential-scientists-in-5940299.

[32] S. Mallick. (2001). "Histogram of oriented gradients," [Online]. Available: https://www.researchgate.net/figure/a-RGB-Color-Space-7-b-YCbCr-Color-Space-8_fig1_298734907.

[33] A. Rosebrock. (2015). "Local binary patterns with python and opencv," [Online]. Available: https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/.

[34] arsho. (2017). "Local binary patterns implementation using python 3," [Online]. Available: https://github.com/arsho/local_binary_patterns.

[35] P. Ekman and W. V. Friesen, "Manual for the facial action code," *Consulting Psychologist Press*, 1978.

[36] N. Ramanathan and R. Chellappa, "Modeling shape and textural variations in aging faces," in *2008 8th IEEE International Conference on Automatic Face Gesture Recognition*, 2008, pp. 1–8. DOI: 10.1109/AFGR.2008.4813337.

[37] J. J. Milne. (2015). "Tarihsel notlarla çapraz-oran geometri üzerine temel bir inceleme," [Online]. Available: https://archive.org/details/elementarytreati00milnuoft/page/11/mode/2up.

[38] D. R. Lee, R. Gommers, F. Wasilewski, K. Wohlfahrtand, and A. O'Leary, "Pywavelets: A python package for wavelet analysis," *Journal of Open Source Software*, vol. 4(36), p. 1237, 2019. DOI: https://doi.org/10.21105/joss.01237.

[39] L. C. B. Ogiela Marek R.and Jain, *Computational intelligence paradigms in advanced pattern classification*. 2012, p. 179, ISBN: 978-3-642-24049-2.

[40] N. Kyurkchiev and S. Markov, *Sigmoid Functions Some Approximation and Modelling Aspects: Some Moduli in Programming Environment MATHEMATICA*. Aug. 2015, ISBN: 978-3-659-76045-7.

[41] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08, Helsinki, Finland: Association for Computing Machinery, 2008, pp. 160–167, ISBN: 9781605582054. DOI: 10.1145/1390156.1390177. [Online]. Available: https://doi.org/10.1145/1390156.1390177.

[42] O. Avilov, S. Rimbert, A. Popov, and L. Bougrain, "Deep learning techniques to improve intraoperative awareness detection from electroencephalographic signals," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2020, pp. 142–145. DOI: 10.1109/EMBC44109.2020.9176228.

[43] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 01, 2017, pp. 7–12. DOI: 10.1109/CBI.2017.23.

[44] R. O. K. Reddy and C. Raghavendra, "Effective facial emotion recognition using convolutional neural network algorithm," *International Journal of Recent Technology and Engineering*, vol. 8, no. 4, pp. 4351–4354, 2019. DOI: 10.35940/ijrte.d8275.118419.

[45] V. H. Phung and E. J. Rhee, "A deep learning approach for classification of cloud image patches on small dataset," *Journal of Information and Communication Convergence Engineering*, vol. 16, no. 3, pp. 173–178, Sep. 2018.

[46] D. Y. Liliana, "Emotion recognition from facial expression using deep convolutional neural network," *Journal of Physics: Conference Series*, vol. 1193, no. 1, 2019, ISSN: 17426596. DOI: 10.1088/1742-6596/1193/1/012004.

[47] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

EK-A

Program A.1: Renk Uzayı ile Yüz Tespiti Programı

```
    # Gerekli kutuphaneler yukleniyor...
    import cv2
    import numpy as np
    import matplotlib.pyplot as mplt

    # Resim okunuyor..
    imge=cv2.imread("binsanlari.jpg")

    # HSVrenk uzayi ile...
    minHSV = np.array([0, 70, 90], dtype = "uint8")
    maxHSV = np.array([20, 255, 255], dtype = "uint8")
    imgeHSV = cv2.cvtColor(imge, cv2.COLOR_BGR2HSV)
    skinRegionHSV = cv2.inRange(imgeHSV, minHSV, maxHSV)
    skinHSV = cv2.bitwise_and(imge, imge, mask=skinRegionHSV)

    # YCrCb renk uzayi ile...
    minYCrCb = np.array([0,143,97],np.uint8)
    maxYCrCb = np.array([235,173,117],np.uint8)
    imgeYCrCb = cv2.cvtColor(imge,cv2.COLOR_BGR2YCR_CB)
    skinRegionYCrCb =cv2.inRange(imgeYCrCb,minYCrCb,maxYCrCb)
    skinYCrCb =cv2.bitwise_and(imge,imge,mask=skinRegionYCrCb)

    # Resimler gosteriliyor
    mplt.figure(figsize=(12, 9))
    mplt.subplot(311)
    mplt.imshow(cv2.cvtColor(imge, cv2.COLOR_BGR2RGB))
    mplt.title("orginal resim")
    mplt.subplot(312)
    mplt.imshow(cv2.cvtColor(skinHSV, cv2.COLOR_BGR2RGB))
    mplt.title("HSV ile cikarilmis resim")
    mplt.subplot(313)
    mplt.imshow(cv2.cvtColor(skinYCrCb, cv2.COLOR_BGR2RGB))
    mplt.title("YCrCb ile cikarilmi resim")
    mplt.show()
```

Program A.2: Dlib İle Yüz Tespiti Programı

```
import dlib
import cv2
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('predictor_68facelandmarks.dat')

# giris imgesini yukle, yeniden boyutlandir, ve gri imgeye donustur
imge=cv2.imread('biliminsanlari1.jpg')
gri = cv2.cvtColor(imge, cv2.COLOR_BGR2GRAY)

# gri imgedeki yuzleri tespit et
dedekt = detector(gri, 2)

```

```
# yuz tespiti dongusu
for (i, dedekt) in enumerate(dedekts):
    landmarks = predictor(gri, dedekt)
    x = dedekt.left()
    y = dedekt.top()
    w = dedekt.right()
    h = dedekt.bottom()
    cv2.dedektangle(imge, (x, y), ( w, h), (0, 255, 0), 1)

    # tespit edilen yuzlerin numarasimi goster
    cv2.putText(imge, "Face {}".format(i + 1), (x - 10, y - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0, 255, 0), 1)
    for lm in range(0, 68):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(imge, (x, y), 1, (0, 255, 255), -1)

# Tespit edilen yuzler ve landmarklar ile birlikte
# cikis imgesini goster.
cv2.imshow("Output", imge)
cv2.waitKey(0)
```

EK-B

Program B.1: HOG özniteliklerini hesaplayan program

```
def oznitelikbul(resim):
    from skimage.feature import hog

    #HOG oznitelikler olusturulmasi
    fd, hog_imge = hog(resim, orientations=9,
                       pixels_per_cell=(8, 8),
                       cells_per_block=(2, 2),
                       visualize=True, multichannel=True)
    df = pd.DataFrame(fd).T
    return df

# Bos bir Veri Cercevesi oluturma
veri = pd.DataFrame(columns = [])

#Her bir resin icin HOG bulma ve veri setine ekleme
for p in range(0, imge_veri.shape[0]):
    resim=imge_veri[p]
    df=oznitelikbul(resim)
    veri = veri.append(df,ignore_index = True)
veri

# veri setine Her resme ait duygu etiketlerini ekleme
veri["Duygu"]=etiketler
veri

giris=veri.iloc[:,:veri.shape[1]-1]
giris
```

Program B.2: LBP özniteliklerini hesaplayan program

```
def piksel_al(imge, merkez, x, y):
    yeni_deger = 0
    try:
        # Eger yerel komsu piksel degeri,
        # merkez piksel degerlerinden
        # buyukse veya bunlara esitse, 1 olarak ayarlanir.
        if imge[x][y] >= merkez:
            yeni_deger = 1
    except:
        # Bir merkez piksel degerinin
        # komsuluk degeri bos oldugunda,
        # yani sinirlarda mevcut degerler oldugunda
        # istisna gereklidir..

        pass
    return yeni_deger

# LBP'yi hesaplama fonksiyonu:

def lbp_hsp_piksel(imge, x, y):
    merkez = imge[x][y]
```

```
    deger_ar = []
    deger_ar.append(piksel_al(imge,merkez,x-1,y-1))     # ust_sol
    deger_ar.append(piksel_al(imge,merkez,x-1,y))       # ust
    deger_ar.append(piksel_al(imge,merkez,x-1,y + 1))   # ust_sag
    deger_ar.append(piksel_al(imge,merkez,x,y + 1))     # sag
    deger_ar.append(piksel_al(imge,merkez,x + 1,y + 1)) # alt_sag
    deger_ar.append(piksel_al(imge,merkez,x + 1,y))     # alt
    deger_ar.append(piksel_al(imge,merkez,x + 1,y-1))   # alt_sol
    deger_ar.append(piksel_al(imge,merkez,x,y-1))       # sol

    # Simdi, ikili degerleri ondalik sayiya cevirmek icin;
    guc_degeri = [1, 2, 4, 8, 16, 32, 64, 128]
    val = 0
    for i in range(len(deger_ar)):
        deger += deger_ar[i] * guc_degeri[i]
    return deger

def oznitelikbul(resim):
    imge_gri = cv2.cvtColor(resim,cv2.COLOR_BGR2GRAY)
    # LBP fonksiyonunu kullarak LPB yi hesaplama:
    imge_lbp = np.zeros((resim.shape[0], resim.shape[1]),np.uint8)
    for i in range(0, resim.shape[0]):
        for j in range(0, resim.shape[1]):
            imge_lbp[i, j] = lbp_hsp_piksel(imge_gri, i, j)

    hist_imge_lbp = cv2.calcHist([imge_lbp],[0],None,[256],[0,256])
    df = pd.DataFrame(hist_imge_lbp.T[0]).T
    return df

# bos bir DataFrame  olustur
import pandas as pd
veri = pd.DataFrame(columns = [])

for p in range(0, imge_veri.shape[0]):
    resim=imge_veri[p]
    df=oznitelikbul(resim)
    veri = veri.append(df,ignore_index = True)
veri

veri["Duygu"]=etiketler
veri

giris=veri.iloc[:,:veri.shape[1]-1]
giris
```

Program B.3: Geometrik Öznitelik bulan program

```
import cv2
import numpy as np
import dlib
import math

nx=np.zeros(68)
ny=np.zeros(68)
detector = dlib.get_frontal_face_detector()
predictor=dlib.shape_predictor("predictor_68face_landmarks.dat")

```

```
def oznitelikbul(resim):
    global df

    # imgeyi yukle, yeniden boyutlandir, ve gri tona donusturme
    imge = cv2.cvtColor(resim, cv2.COLOR_BGR2GRAY)

    # gri resimdeki yuzu tespit etme
    dedekts = detector(imge, 1)
    # loop over the face detections
    for (i, dedekt) in enumerate(dedekts):
        # determine the facial landmarks for the face region, then
        # convert the facial landmark (x, y)-coordinates to a NumPy

        landmarks = predictor(imge, dedekt)

        x = dedekt.left()
        y = dedekt.top()
        w = dedekt.right()
        h = dedekt.bottom()
        for lm in range(0, 68):
            x = landmarks.part(lm).x
            y = landmarks.part(lm).y
            #text="%s" % (lm)
            #cv2.putText(imge,text,(x, y),
                    cv2.FONT_HERSHEY_COMPLEX,0.3,[0,0,255],1)
            cv2.circle(imge,(x, y), 1,(0, 255, 0),thickness=-1)

            nx[lm] = landmarks.part(lm).x
            ny[lm] = landmarks.part(lm).y

        n0_16=math.sqrt((nx[0]-nx[16])**2 + (ny[0]- ny[16])**2)
        n6_48=math.sqrt((nx[6]-nx[48])**2 + (ny[6]- ny[48])**2)
        n8_27=math.sqrt((nx[8]-nx[27])**2 + (ny[8]- ny[27])**2)
        n8_33=math.sqrt((nx[8]-nx[33])**2 + (ny[8]- ny[33])**2)
        n10_54=math.sqrt((nx[10]-nx[54])**2 + (ny[10]-ny[54])**2)
        n17_36=math.sqrt((nx[17]-nx[36])**2 + (ny[17]-ny[36])**2)
        n18_36=math.sqrt((nx[18]-nx[36])**2 + (ny[18]-ny[36])**2)
        n19_36=math.sqrt((nx[19]-nx[36])**2 + (ny[19]-ny[36])**2)
        n19_39=math.sqrt((nx[19]-nx[39])**2 + (ny[19]-ny[39])**2)
        n20_39=math.sqrt((nx[20]-nx[39])**2 + (ny[20]-ny[39])**2)
        n21_22=math.sqrt((nx[21]-nx[22])**2 + (ny[21]-ny[22])**2)
        n21_27=math.sqrt((nx[21]-nx[27])**2 + (ny[21]-ny[27])**2)
        n21_39=math.sqrt((nx[21]-nx[39])**2 + (ny[21]-ny[39])**2)
        n22_27=math.sqrt((nx[22]-nx[27])**2 + (ny[22]-ny[27])**2)
        n22_42=math.sqrt((nx[22]-nx[42])**2 + (ny[22]-ny[42])**2)
        n23_42=math.sqrt((nx[23]-nx[42])**2 + (ny[23]-ny[42])**2)
        n24_42=math.sqrt((nx[24]-nx[42])**2 + (ny[24]-ny[42])**2)
        n24_45=math.sqrt((nx[24]-nx[45])**2 + (ny[24]-ny[45])**2)
        n25_45=math.sqrt((nx[25]-nx[45])**2 + (ny[25]-ny[45])**2)
        n26_45=math.sqrt((nx[26]-nx[45])**2 + (ny[26]-ny[45])**2)
        n27_30=math.sqrt((nx[27]-nx[30])**2 + (ny[27]-ny[30])**2)
        n27_33=math.sqrt((nx[27]-nx[33])**2 + (ny[27]-ny[33])**2)
        n27_62=math.sqrt((nx[27]-nx[62])**2 + (ny[27]-ny[62])**2)
        n27_66=math.sqrt((nx[27]-nx[66])**2 + (ny[27]-ny[66])**2)
        n31_35=math.sqrt((nx[31]-nx[35])**2 + (ny[31]-ny[35])**2)
        n33_62=math.sqrt((nx[33]-nx[62])**2 + (ny[33]-ny[62])**2)
        n36_39=math.sqrt((nx[36]-nx[39])**2 + (ny[36]-ny[39])**2)
        n36_42=math.sqrt((nx[36]-nx[42])**2 + (ny[36]-ny[42])**2)
        n36_45=math.sqrt((nx[36]-nx[45])**2 + (ny[36]-ny[45])**2)
```

```
        n36_48=math.sqrt((nx[36]-nx[48])**2 + (ny[36]-ny[48])**2)
        n37_41=math.sqrt((nx[37]-nx[41])**2 + (ny[37]-ny[41])**2)
        n38_40=math.sqrt((nx[38]-nx[40])**2 + (ny[38]-ny[40])**2)
        n39_42=math.sqrt((nx[39]-nx[42])**2 + (ny[39]-ny[42])**2)
        n39_45=math.sqrt((nx[39]-nx[45])**2 + (ny[39]-ny[45])**2)
        n39_48=math.sqrt((nx[39]-nx[48])**2 + (ny[39]-ny[48])**2)
        n42_45=math.sqrt((nx[42]-nx[45])**2 + (ny[42]-ny[45])**2)
        n42_54=math.sqrt((nx[42]-nx[54])**2 + (ny[42]-ny[54])**2)
        n43_47=math.sqrt((nx[43]-nx[47])**2 + (ny[43]-ny[47])**2)
        n44_46=math.sqrt((nx[44]-nx[46])**2 + (ny[44]-ny[46])**2)
        n45_54=math.sqrt((nx[45]-nx[54])**2 + (ny[45]-ny[54])**2)
        n48_54=math.sqrt((nx[48]-nx[54])**2 + (ny[48]-ny[54])**2)
        n51_57=math.sqrt((nx[51]-nx[57])**2 + (ny[51]-ny[57])**2)
        n60_64=math.sqrt((nx[60]-nx[64])**2 + (ny[60]-ny[64])**2)
        n62_66=math.sqrt((nx[62]-nx[66])**2 + (ny[62]-ny[66])**2)

        oran0 = 1
        oran1 = (n36_42*n39_45)/ (n39_42*n36_45)
        oran2 = (n36_45*n8_33) / (n33_62*n8_27)
        oran3 = n8_27 / n0_16
        oran4 = n31_35 / n27_33
        oran5 = n39_42 / n36_45
        oran6 = ((n43_47 + n44_46) / 2)/ n42_45
        oran7 = ((n37_41 + n38_40) / 2)/ n36_39
        oran8 = ((n51_57 + n62_66) / 2)/ ((n48_54 + n60_64) / 2)
        oran9 = ((n21_27 + n22_27) / 2)/ n27_30
        oran10 = n21_22 / ((n39_42 + n36_45) / 2)
        oran11 = ((n6_48 + n10_54) / 2)/ ((n36_48 + n45_54) / 2)
        oran12 = ((n27_62 + n27_66) / 2)/ ((n39_48 + n42_54) / 2)
        oran13 = ((n25_45 + n26_45) / 2)/ n27_33
        oran14 = ((n17_36 + n18_36) / 2)/ n27_33
        oran15 = ((n22_42 + n22_42) / 2)/ n27_33
        oran16 = ((n20_39 + n21_39) / 2)/ n27_33
        oran17 = ((n24_42 + n24_45) / 2)/ n42_45
        oran18 = ((n19_36 + n19_39) / 2)/ n36_39
        oran19 = n62_66 / n8_33
        oran20 = n51_57 / n27_33
        oran21 = ((n36_48 + n39_48) / 2)/ ((n42_54 + n45_54) / 2)
        oran22= ((n36_48 + n39_48 + n42_54 + n45_54) / 4) / n27_33
        df = pd.DataFrame({"x0":[oran0],"x1":[oran1],"x2":[oran2],
            "x3":[oran3],"x4":[oran4],"x5":[oran5],"x6":[oran6],
            "x7":[oran7], "x8":[oran8],"x9":[oran9],
            "x10":[oran10],"x11":[oran11],"x12":[oran12],
            "x13":[oran13],"x14":[oran14],"x15":[oran15],
            "x16":[oran16],"x17":[oran17],"x18":[oran18],
            "x19":[oran19],"x20":[oran20],"x21":[oran21],
            "x22":[oran22]})

    return df

# Bos bir DataFrame olusturma
veri = pd.veriFrame(columns = [])

# Veri setindeki butun resimler icin oznitelik bulma
for p in range(0, imge_veri.shape[0]):
    resim=imge_veri[p]
    df=oznitelikbul(resim)
    veri = veri.append(df,ignore_index = True)
veri
```

```

# verileri 0-1 giris araligina olceklendiren fonksiyon
def olcekle(veri):
    for i in range (1,veri.shape[1]):
        veri.iloc[:,i]=(veri.iloc[:,i]-min(veri.iloc[:,i]))/
                        (max(veri.iloc[:,i])-min(veri.iloc[:,i]))

    return veri


# olcekle fonksiyonunu calistir
veri=olcekle(veri)
veri

# Duygu etiketlerini veri setine ekleme
veri["Duygu"]=etiketler
veri
```

Program B.4: DWT ile öznitelikleri hesaplayan program

```
import pywt
import pywt.data
import numpy as np

def oznitelikbul(resim):
    #giris resmini gri resme donustur.
    imge = cv2.cvtColor(resim, cv2.COLOR_BGR2GRAY)
    katsayilar2=pywt.dwt2(imge,'bior1.3')
    LL, (LH, HL, HH) = katsayilar2

    aa=LH+HL+HH
    veri = np.asarray(aa).reshape(-1)
    df=pd.DataFrame(veri).T

    return df

# create an empty DataFrame

veri = pd.DataFrame(columns = [])

for p in range(0, imge_veri.shape[0]):
    resim=imge_veri[p]
    df=oznitelikbul(resim)
    veri = veri.append(df,ignore_index = True)
veri


# Duygu etiketlerini veri setine ekleme
veri["Duygu"]=etiketler
veri

giris=veri.iloc[:,:veri.shape[1]-1]
giris
```

EK-C

Program C.1: CNN programı

```
import numpy as np # linear algebra
import pandas as pd # veri processing
import os,cv2,dlib
import matplotlib.pyplot as mplt
import matplotlib.image as mpimg
from pylab import rcParams
rcParams['figure.figsize'] = 20, 10
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense , Activation , Dropout ,Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.metrics import categorical_accuracy
from keras.models import model_from_json
from keras.callbacks import ModelCheckpoint
from keras.optimizers import *
from keras.layers.normalization import BatchNormalization
from keras import callbacks
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
import seaborn as sbn

veri_yolu = "..\\veri setleri\\CK+48"
veri_yolu_listesi = os.listdir(veri_yolu)

imge_veri_list=[]

for veriseti in veri_yolu_listesi:
    imge_list=os.listdir(veri_yolu+'/'+ veriseti)
    print ('veriseti imgeleri yuklendi-'+'{}\n'.format(veriseti))
    for imge in imge_list:
        giris_imge=cv2.imread(veri_yolu+ '/'+ veriseti + '/'+imge)
        imge_veri_list.append(giris_imge)
        ############
imge_veri = np.array(imge_veri_list)

# Duygu etiketleri
sinif_sayisi = 7
ornek_sayisi = imge_veri.shape[0]
etiketler = np.ones((ornek_sayisi,),dtype='int64')
etiketler[0:134]=0 #135
etiketler[135:188]=1 #54
etiketler[189:365]=2 #177
etiketler[366:440]=3 #75
etiketler[441:647]=4 #207
etiketler[648:731]=5 #84
etiketler[732:980]=6 #249
def getLabel(id):
    return ['ofke', 'kucumseme', 'tiksinti',
                'korku', 'mutlu', 'uzuntu', 'supriz'][id]
```

```
53
54
55 Y = np_utils.to_categorical(etiketler, sinif_sayisi)
56 # verisetini karistir
57 X,y = shuffle(imge_veri,Y, random_state=7)
58 #  veri setini boluyoruz
59 X_Egitim, X_test, y_Egitim, y_test = train_test_split(X, y,
60                                     test_size=0.3, random_state=7)
61 x_test=X_test
62
63 (imge_sayisi,yukseklik, genislik, depth)=imge_veri.shape
64 oznitelik_sayisi = 64
65 etiket_sayisi = 7
66 girisBoyut = (yukseklik, genislik, depth)
67
68 # Girisleri ve hedefleri birlestir
69 girisler = np.concatenate((X_Egitim, X_test), axis=0)
70 hedefler = np.concatenate((y_Egitim, y_test ), axis=0)
71 # K-katlamali Capraz Dogrulamayi Tanimlayin
72 kfold = KFold(n_splits=5, random_state=None, shuffle=False)
73
74
75 ## Model oluturma ##
76 katlama_no = 1
77 cms=[]
78 Egitim_acc_as=[]
79 val_acc_as=[]
80
81 # K-katlamali Capraz Dogrulama modeli degerlendirmesi
82 for train, test in kfold.split(girisler, hedefler):
83     print('------------------------------------------')
84     print(f'Katlama {katlama_no} icin egitiliyor ')
85     print('------------------------------------------')
86
87     # Model oluturma
88     cnnModel = Sequential()
89
90     cnnModel.add(Conv2D(oznitelik_sayisi, kernel_size=(3, 3),
91                  activation='relu', input_shape=girisBoyut,
92                  data_format='channels_last'))
93     cnnModel.add(Conv2D(oznitelik_sayisi, kernel_size=(3, 3),
94                 activation='relu', padding='same'))
95     cnnModel.add(BatchNormalization())
96     cnnModel.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
97     cnnModel.add(Dropout(0.5))
98
99     cnnModel.add(Conv2D(2*oznitelik_sayisi, kernel_size=(3, 3),
100                 activation='relu', padding='same'))
101    cnnModel.add(BatchNormalization())
102    cnnModel.add(Conv2D(2*oznitelik_sayisi, kernel_size=(3, 3),
103                 activation='relu', padding='same'))
104    cnnModel.add(BatchNormalization())
105    cnnModel.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
106    cnnModel.add(Dropout(0.5))
107
108    cnnModel.add(Conv2D(2*2*oznitelik_sayisi, kernel_size=(3, 3),
109                 activation='relu', padding='same'))
110    cnnModel.add(BatchNormalization())
111    cnnModel.add(Conv2D(2*2*oznitelik_sayisi, kernel_size=(3, 3),
```

```
                activation='relu', padding='same'))
    cnnModel.add(BatchNormalization())
    cnnModel.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    cnnModel.add(Dropout(0.5))

    cnnModel.add(Conv2D(2*2*2*oznitelik_sayisi, kernel_size=(3, 3),
                activation='relu', padding='same'))
    cnnModel.add(BatchNormalization())
    cnnModel.add(Conv2D(2*2*2*oznitelik_sayisi, kernel_size=(3, 3),
                activation='relu', padding='same'))
    cnnModel.add(BatchNormalization())
    cnnModel.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    cnnModel.add(Dropout(0.5))

    cnnModel.add(Flatten())

    cnnModel.add(Dense(2*2*2*oznitelik_sayisi, activation='relu'))
    cnnModel.add(Dropout(0.4))
    cnnModel.add(Dense(2*2*oznitelik_sayisi, activation='relu'))
    cnnModel.add(Dropout(0.4))
    cnnModel.add(Dense(2*oznitelik_sayisi, activation='relu'))
    cnnModel.add(Dropout(0.5))

    cnnModel.add(Dense(etiket_sayisi, activation='softmax'))
    cnnModel.summary()

    # modeli derleme
    cnnModel.compile(loss='categorical_crossentropy',
                    metrics=['accuracy'],optimizer='adam')

    dosyaAdi='cnnModel_egitim_yeni.csv'
    dosyaYolu="Best-weights-my_cnnModel-{epoch:03d}-
                        {loss:.4f}-{acc:.4f}.hdf5"

    csvKayit=callbacks.CSVLogger(dosyaAdi, separator=',',
                                    append=False)
    kontrolNoktasi = callbacks.ModelCheckpoint(dosyaYolu,
                        monitor='val_loss', verbose=1,
                        save_best_only=True, mode='min')
    geriArama_list = [csvKayit,kontrolNoktasi]
    geriArama_list = [csvKayit]

    # modeli uygula

    hist = cnnModel.fit(girisler[train], hedefler[train],
                        batch_size=7,epochs=100, verbose=1,
                        validation_data=(girisler[test],
                        hedefler[test]),callbacks=geriArama_list)

    # Genelleme olcutleri olusturun
    skorlar_test=cnnModel.evaluate(girisler[test],hedefler[test],
                                    verbose=0)
    print('Test Loss:', skorlar_test[0], '\n')
    print('Test dogruluk skoru:', skorlar_test[1], '\n')
    skorlar_Egtm = cnnModel.evaluate(girisler[train],
                                hedefler[train],verbose=0)
    print('Train Loss:', skorlar_Egtm[0], '\n')
    print('Egitim dogruluk skoru:', skorlar_Egtm[1], '\n')
```

```
    # Karmasiklik matrisi
    result = cnnModel.predict_classes(girisler[test])
    result=result.astype(int)

    y_con=np.zeros(hedefler[test].shape[0])
    for i in range (0,hedefler[test].shape[0]):
        if   hedefler[test][i][0]>=1:
            y_con[i]=0
        elif hedefler[test][i][1]>=1:
            y_con[i]=1
        elif hedefler[test][i][2]>=1:
            y_con[i]=2
        elif hedefler[test][i][3]>=1:
            y_con[i]=3
        elif hedefler[test][i][4]>=1:
            y_con[i]=4
        elif hedefler[test][i][5]>=1:
            y_con[i]=5
        else :
            y_con[i]=6
    y_con=y_con.astype(int)
    print('test etiketi sayisi:',hedefler[test].shape[0] , '\n')
    km = confusion_matrix(y_con,result)
    df_km = pd.DataFrame(km, ('ofke', 'kucumseme', 'tiksinti',
                              'korku', 'mutlu', 'uzuntu', 'supriz'),
                     ('ofke', 'kucumseme', 'tiksinti', 'korku',
                              'mutlu', 'uzuntu', 'supriz'))
    fig, ax = mplt.subplots(figsize=(20, 4))
    ax.xaxis.tick_top()
    sbn.set(font_scale=1.5) # for label size
    sbn.heatmap(df_km, annot=True,fmt=".1f", annot_kws={"size": 18},
                        linecolor="k", linewidths=1,cmap="binary",
                        cbar=True) # font size
    mplt.xlabel('Gercek durum',fontsize = 18)
    mplt.ylabel('Tahmin edilen durum',fontsize = 18)
    mplt.show()

    kms.append(km)

    # kayiplari ve dogrulugu gorsellestirme
    %matplotlib inline
    train_loss=hist.history['loss']
    val_loss=hist.history['val_loss']
    train_acc=hist.history['accuracy']
    val_acc=hist.history['val_accuracy']

    epochs = range(len(train_acc))
    fig = mplt.figure(figsize=(15,7))
    mplt.plot(epochs,train_acc,'r', label='Egitim dogrulugu')
    mplt.plot(epochs,val_acc,'b', label='Test dogrulugu')
    mplt.title('Egitim dogrulugu ve test dogrulugu karsilastirma')
    mplt.xlabel('epochs')
    mplt.ylabel('accuracy')
    mplt.legend()
    mplt.figure()
    mplt.show()

    train_acc_array = np.array(train_acc)
    train_acc_as.append(train_acc_array)
```

```
230
231     val_acc_array = np.array(val_acc)
232     val_acc_as.append(val_acc_array)
233
234     # Increase fold number
235     katlama_no = katlama_no + 1
236
237 #ortalama karmasiklik matrisi
238 kms1=np.asarray(kms)
239 kmatrix=(kms1[0]+kms1[1]+kms1[2]+kms1[3]+kms1[4])/5
240 df_km = pd.DataFrame(kmatrix, ('ofke', 'kucumseme', 'tiksinti',
241                                'korku', 'mutlu', 'uzuntu', 'supriz'),
242                      ('ofke', 'kucumseme', 'tiksinti', 'korku',
243                                'mutluluk', 'uzuntu', 'supriz'))
244 fig, ax = mplt.subplots(figsize=(20, 4))
245 ax.xaxis.tick_top()
246 sbn.set(font_scale=1.5) # for label size
247 sbn.heatmap(df_km, annot=True,fmt=".1f", annot_kws={"size": 18},
248                    linecolor="k", linewidths=1,cmap="binary",
249                    cbar=True) # font size
250 mplt.xlabel('Gercek durum',fontsize = 18)
251 mplt.ylabel('Tahmin edilen durum',fontsize = 18)
252 mplt.show()
```

Program C.2: CK+ resim setinden HOG veri seti oluşturan program

```
1  veri_yolu = "E:\\NKU\\python proje\\veri setleri\\CK+48"
2  veri_yolu_listesi = os.listdir(veri_yolu)
3  from skimage.feature import hog
4
5  imge_veri_list=[]
6
7  for veriseti in veri_yolu_listesi:
8      imge_list=os.listdir(veri_yolu+'/'+ veriseti)
9      print ('veriseti imgeleri yuklendi-'+'{}\n'.format(veriseti))
10     for imge in imge_list:
11         giris_imge=cv2.imread(veri_yolu+'/'+veriseti+'/'+imge )
12         fd, hog_imge = hog(giris_imge, orientations=9,
13                            pixels_per_cell=(4, 4), cells_per_block=(2, 2),
14                            visualize=True, multichannel=True)
15
16         imge_rsz=cv2.resize(hog_imge, (48, 48))
17         imge_reshape=imge_rsz.reshape(list(imge_rsz.shape)+[1])
18         imge_veri_list.append(imge_reshape)
19 imge_veri = np.array(imge_veri_list)
20
21 imge_veri.shape
```

Program C.3: CK+ resim setinden LBP veri seti oluşturan program

```
1  def piksel_al(imge, merkez, x, y):
2      yeni_deger = 0
3      try:
4          if imge[x][y] >= merkez:
5              yeni_deger = 1
```

```
    except:
        pass
    return yeni_deger

# LBP'yi hesaplama fonksiyonu:
def lbp_hsp_piksel(imge, x, y):
    merkez = imge[x][y]
    deger_ar = []
    deger_ar.append(piksel_al(imge,merkez,x-1, y-1))    # ust sol
    deger_ar.append(piksel_al(imge,merkez,x-1, y))      # ust
    deger_ar.append(piksel_al(imge,merkez,x-1, y+1))    # ust sag
    deger_ar.append(piksel_al(imge,merkez,x, y+1))      # sag
    deger_ar.append(piksel_al(imge,merkez,x+1, y+1))    # alt sag
    deger_ar.append(piksel_al(imge,merkez,x+1, y))      # alt
    deger_ar.append(piksel_al(imge,merkez,x+1, y-1))    # alt sol
    deger_ar.append(piksel_al(imge,merkez,x, y-1))      # sol

    # Simdi, ikili degerleri ondalik sayiya cevirmek icin;
    guc_degeri = [1, 2, 4, 8, 16, 32, 64, 128]
    deger = 0
    for i in range(len(deger_ar)):
        deger += deger_ar[i] * guc_degeri[i]
    return deger
veri_yolu = "E:\\NKU\\python proje\\veri setleri\\CK+48"
veri_yolu_listesi = os.listdir(veri_yolu)

imge_veri_list=[]
for veriseti in veri_yolu_listesi:
    imge_list=os.listdir(veri_yolu+'/'+ veriseti)
    print ('veriseti imgeleri yuklendi-'+'{}\n'.format(veriseti))
    for imge in imge_list:
        giris_imge=cv2.imread(veri_yolu+'/'+veriseti+'/'+imge )
        giris_imge = cv2.cvtColor(giris_imge, cv2.COLOR_BGR2GRAY)
        # LBP fonksiyonunu kullarak LPB yi hesaplama:
        imge_lbp = np.zeros((giris_imge.shape[0],
                                    giris_imge.shape[1]),np.uint8)
        for i in range(0, giris_imge.shape[0]):
            for j in range(0, giris_imge.shape[1]):
                imge_lbp[i, j]=lbp_hsp_piksel(giris_imge,i,j)

        giris_imge_re=imge_lbp.reshape(list(imge_lbp.shape)+[1])
        imge_veri_list.append(giris_imge_re)
        ############
imge_veri = np.array(imge_veri_list)
imge_veri.shape
```

Program C.4: CK+ resim setinden DWT veri seti oluşturan program

```
veri_yolu = "E:\\NKU\\python proje\\veri setleri\\CK+48"
veri_yolu_listesi = os.listdir(veri_yolu)

imge_veri_list=[]
import pywt
for veriseti in veri_yolu_listesi:
    imge_list=os.listdir(veri_yolu+'/'+ veriseti)
    print ('veriseti imgeleri yuklendi-'+'{}\n'.format(veriseti))
    for imge in imge_list:
```

```
        giris_imge=cv2.imread(veri_yolu+''+veriseti +''+imge )
        giris_imge = cv2.cvtColor(giris_imge,cv2.COLOR_BGR2GRAY)
        katsayilar2=pywt.dwt2(giris_imge,'bior1.3')
        LL, (LH, HL, HH) = katsayilar2
        wavelet=LH+HL+HH+LL
        giris_imge_re=wavelet.reshape(list(wavelet.shape)
                                                          +[1])
        imge_veri_list.append(giris_imge_re)
        ############
imge_veri = np.array(imge_veri_list)
```