YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARIYLA YÜZ MİMİKLERİNİN TANINMASI

Elektrik ve Elektronik Müh. Melih YASAV

FBE Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Elektronik Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Prof. Dr. Tülay YILDIRIM

İSTANBUL, 2008

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ iv

KISALTMA LİSTESİ v

ŞEKİL LİSTESİ vi

ÇİZELGE LİSTESİ vii

ÖNSÖZ viii

ÖZET ix

ABSTRACT x

1 GİRİŞ 1

2 GÖRÜNTÜ İŞLEME 3

2.1 Görüntü İşleme Yöntemleri 4
2.2 Gürültü Azaltma 8
2.2.1 Alçak Geçiren, Yüksek Geçiren ve Medyan Süzgeçler 8
2.3 Kenar Bulma 10
2.4 Şablon Eşleştirme 10
2.5 Renk Uzayları 11
2.6 Morfolojik İşlemler 14
2.6.1 Aşındırma 14
2.6.2 Yayma 14
2.6.3 Açma 14
2.6.4 Kapama 15
2.7 Hough Dönüşümü 15

3 YAPAY SİNİR AĞLARI 16

3.1 Temel Nöron Modeli 18
3.2 Çok Katmanlı Algılayıcı 19
3.3 Radyal Temelli Fonksiyon Ağları 20
3.4 Konik Kesit Fonksiyonlu Sinir Ağları 21
3.5 Ağın Performans Fonksiyonu 22
3.6 Öğrenme Kuralları 23
3.6.1 Levenberg Marquardt Öğrenme Kuralı 23
3.6.2 Ölçeklenmiş Konjugate Gradyent Öğrenme Kuralı 23
3.6.3 BFGS Öğrenme Kuralı 24
3.6.4 Resilient Propagasyon Öğrenme Kuralı 24
3.7 Özellik Matrisinin Normalizasyonu 25
3.8 Temel Bileşen Analizi 26
3.9 Eğitim Sonrası İşlemler 26

4 GÖRÜNTÜ ÜZERİNDE GÖZ VE DUDAKLARIN BULUNMASI VE ÖZELLİKLERİNİN ÇIKARILMASI 27

4.1 Yüzün Bulunması 29
4.2 Hassas Yüz Belirleme 30
4.3 Yüz Görüntüsü Üzerinde Alın Bölgesinin ve Kaşların Çıkarılması 32
4.4 Gözlerin ve Dudakların Yerlerinin Bulunması 34
4.4.1 Göz Bölgelerinin Bulunması 34
4.4.2 Göz Merkezlerinin Bulunması 37
4.4.3 Dudakların Yerinin Bulunması 38
4.4.3.1 Alt Dudağın Bulunması 38
4.4.3.2 Üst Dudağın Bulunması 39
4.4.3.3 Ağzın Sol ve Sağ Sınırlarının Belirlenmesi ve Ağız Bölgesinin İşaretlenmesi 40
4.5 Göz ve Dudaklara Ait Belirleyici Özelliklerin Çıkarılması 40
4.5.1 Gözlere Ait Özelliklerin Çıkarılması 40
4.5.1.1 Renk Bilgisi Kullanılarak Göz Özelliklerinin Çıkarılması 40
4.5.1.2 Şablon Korelasyon Değeri Kullanılarak 41
4.5.2 Dudak Bölgesinin Özelliklerinin Çıkarılması 41

5 AĞ ÜZERİNDE YAPILAN DENEMELER ve SONUÇLAR 44

5.1 Özellik Matrisinden Eğitim ve Test Kümelerinin Oluşturulması 45
5.2 Çok Katmanlı Ağ ile Yapılan Denemeler 45
5.3 Radyal Temelli Fonksiyon Ağları ile Yapılan Denemeler 47
5.4 Konik Kesit Fonksiyonlu Sinir Ağları ile Yapılan Denemeler 48

6 SONUÇLAR 49

6.1 Çalışmaya Yapılabilecek Eklentiler 50

KAYNAKLAR 52

EKLER 53

Ek 1 ÇKA için eğitim ve test programı 54
Ek 2 RTFA ağları için eğitim ve test programı 60
Ek 3 KKFSA ağları için eğitim ve test programı 66
Ek 4 Orijinal özellik kümesi 72
Ek 5: Sınıflandırma işleminde kullanılan ağ konfigürasyonları tablosu 77

ÖZGEÇMİŞ 84

SİMGE LİSTESİ

b	Bias değeri
c_i	Merkez değeri
f(.)	Dijital görüntü fonksiyonu
$\varphi(x)$	Aktivasyon fonksiyonu
μ	Levenberg-Marquardt parametresi
w_{ij}	Bağlantı ağırlıkları
w	Açılma açısı
x	Giriş vektörü

KISALTMA LİSTESİ

CMY	Cyanta Magenta Yellow
ÇKA	Çok Katmanlı Algılayıcı (Multilayer Perceptron)
EKG	Elektrokardiogram
GRSA	Genel Regresyon Sinir Ağları (General Regression Neural Networks)
HSI	Hue Saturation Intensity
KKFSA	Konik Kesit Fonksiyonlu Sinir Ağları (Conic Section Function Neural Networks)
OSA	Olasılıksal Sinir Ağı (Probabilistic Neural Network)
PCA	Temel Bileşen Analizi (Principal Component Analysis)
RGB	Red Green Blue
RTFA	Radyal Temelli Fonksiyon Ağları (Radial Basis Function Networks)
SD	Standart Sapma
YSA	Yapay Sinir Ağları

ŞEKİL LİSTESİ

Şekil 2.1 Eşikleme 4
Şekil 2.2 Bileşen etiketleme 7
Şekil 2.3 Uzaklık dönüşümü uygulanması 7
Şekil 2.4 Boyut süzgeci uygulanması 8
Şekil 2.5 Alçak geçiren süzgeç kernelleri 9
Şekil 2.6 Alçak geçiren süzgeç uygulaması 9
Şekil 2.7 Yüksek geçiren süzgeç kernelleri 10
Şekil 2.8 Renk uzayları 11
Şekil 2.9 HSI Renk modelinin yapısı 12
Şekil 2.10 Hough dönüşümü ile göz bebeğinin bulunması 15
Şekil 3.1 Öğrenme seti hatası ve test seti hatasının ağ eğitimi süresince değişimi 18
Şekil 3.2 Temel Nöron Modeli 19
Şekil 3.3 Çok katmanlı algılayıcı ağın yapısı 20
Şekil 3.4 RTFA ağının yapısı 21
Şekil 3.5 Konik Kesit Fonksiyonlu Ağın yapısı 22
Şekil 4.1 Tanımlanan yüz hareketleri 27
Şekil 4.2 Çeşitli dudak ve göz durumları 28
Şekil 4.3 Yüz bölgesinin facefind.m ile elde edilmesi. 29
Şekil 4.4 Deri renginin HSV renk uzayında kapladığı alan 30
Şekil 4.5 Yüzün elde edilmesi 31
Şekil 4.6 Yüz görüntüsüne Otsu algoritmasının uygulanması 33
Şekil 4.7 Otsu görüntüsünün yatay izdüşümü 33
Şekil 4.8 Alın ve kaş bölgesi çıkarıldıktan sonra yüz bölgesi 34
Şekil 4.9 Yüz bölgesine Gradyent uygulanması 34
Şekil 4.10 Yüz görüntüsünün yatay izdüşüm grafiği 35
Şekil 4.11 Gözlerin yerlerinin işaretlenmesi 35
Şekil 4.12 Yüzün dikey ortasının bulunması 36
Şekil 4.13 Gözlerin birbirinden ayrı şekilde elde edilmesi 37
Şekil 4.14 Çalışmada kullanılan sol ve sağ göz şablonları 37
Şekil 4.15 Gözlerin yerlerinin işaretlenmesi 38
Şekil 4.16 Dudak bölgesinin bulunması için yapılan dönüşümler 39
Şekil 4.17 Açık göz görüntüsünün HSV, H, S ve V bileşenleri 40
Şekil 4.18 Kapalı göz görüntüsünün HSV, H, S ve V bileşenleri 41
Şekil 4.19 Bulunan çeşitli dudak hareketleri 42

ÇİZELGE LİSTESİ

Çizelge 4.1 Yüz hareketlerinin anlamları 29
Çizelge 4.2 Sol ve sağ göz için çıkarılmış örnek özellik kümesi 41
Çizelge 4.3 Dudaklar için çıkarılmış örnek özellik kümesi 42
Çizelge 5.1 Her yüz şekli elde edilmiş örnek özellikler 44
Çizelge 5.2 Her yüz ifadesi için hazırlanmış hedef vektörü 45
Çizelge 5.3 ÇKA ağ yapısının farklı eğitim kurallarıyla eğitilmesi 47
Çizelge 5.4 RTFA ağ yapısının farklı şekilde normalize edilmiş giriş vektörleriyle eğitilmesi 48
Çizelge 5.5 KKFSA ağ yapısının farklı şekilde normalize edilmiş giriş vektörleriyle eğitilmesi 48
Çizelge 6.1 Ağ yapılarıyla elde edilen yüz şekli sınıflandırma başarısı 49

ÖNSÖZ

Tezimin hazırlanmasında ve yüksek lisans hayatım boyunca bana büyük katkıları olan çok değerli hocam Prof. Dr. Tülay Yıldırım'a yardımlarından ve sabrından dolayı teşekkürü bir borç bilirim.

Ayrıca Sayın Burcu Erkmen ve Revna Acar Hocalarıma da tezim sırasında yardımlarını esirgemedikleri için teşekkürlerimi sunarım.

Tübitak'a yüksek lisans hayatım boyunca sağladığı destek için teşekkür ederim.

Tüm eğitim hayatımda olduğu gibi yüksek lisans çalışmam sırasında da beni yalnız bırakmayan ve her türlü desteği veren aileme de en içten duygularımla teşekkür ederim.

ÖZET

Günümüzde görüntü işleme araştırmaların yoğunlaştığı bir çalışma alanı haline gelmiştir. Bu çalışmada, iletişim yetenekleri kısıtlı, engelli veya hasta kişilerin temel ihtiyaçlarının, yüz mimiklerinin bilgisayar yardımıyla değerlendirilerek sağlanması amaçlanmıştır.

Bu tez yapı itibariyle iki ana aşamada ele alınabilir. İlk aşamasında mevcut yüz görüntülerinden göz ve dudakların durumlarını tespit etmek için belirleyici özellikler elde edilmiştir. İkinci aşamasında ise oluşturulmuş bu özellik kümesi sinir ağları kullanılarak değerlendirilmiştir.

Gözlerin ve dudakların durumlarına göre 12 yüz hareketi belirlenmiş ve bir kişiye ait her hareket için yaklaşık 27'şer olmak üzere toplam 322 görüntü ile bir veri kümesi kurulmuştur. Bu veri kümesindeki 211 görüntü sinir ağını eğitmede, 111 görüntü ise test etmede kullanılmıştır.

Ağ üzerindeki denemeler, Matlab 7.0 Neural Networks Toolbox ve Neuro Solutions 5 programları kullanılarak yapılmıştır. Yapay sinir ağı olarak Çok Katmanlı Algılayıcı Ağlar, Radyal Temelli Fonksiyon Ağları ve Konik Kesit Fonksiyonlu Sinir Ağları kullanılmış ve bu ağların başarıları karşılaştırılmıştır.

Özellik kümelerinin sinir ağlarıyla eğitimi sırasında farklı veri normalizasyon yöntemleri kullanılarak ağın eğitim ve test başarısı arttırılmıştır. ÇKA ağlarının, KKFSA ağlarına göre daha az sayıda adımda eğitilebildiği görülmüş, fakat KKFSA ağlarıyla daha yüksek sınıflama başarı oranları elde edilebildiği bulunmuştur.

Anahtar kelimeler: Yapay sinir ağları, yüz bulma yöntemleri, göz ve ağız durum tespiti, mimik tanıma

ABSTRACT

Image processing is one of the research areas that receives much attention from the researchers today. This thesis is done to evaluate human facial mimics by computer in order to help disabled people who have limited communicative abilities.

This thesis can be divided into two main sections. In the first part of the thesis, features that would help us to classify facial mimics are extracted from the facial images. Second section demonstrates evaluating this feature matrix by using Artificial Neural Networks.

12 facial gestures are designated according to the status of eyes and lips. 322 photos are taken from one person with approximately 27 photos for each set of mimics. Feature vectors extracted from these photos are kept in a database. 211 feature vectors are used to train the Neural Networks and the remaining 111 feature vectors are used for testing the Neural Networks.

Neural Network simulations are performed by using Matlab 7.0 Neural Networks toolbox and Neuro Solutions 5 software. Multilayer Perceptron, Radial Basis Function Networks and Conic Section Function Neural Networks are used to evaluate the feature matrix and the success rates for these networks are compared.

Different data normalization procedures are applied to the feature matrix in order to increase training and test success of the neural networks. It is noticed that MLP is trained in less epochs than CSFNN, however CSFNN reaches higher classification success rates.

Keywords: Artificial Neural Networks, face detection, eye and mouth state detection, mimic recognition

1. GİRİŞ

Mimik, bir duygu ve düşüncenin kaş, göz, ağız ve yüz hareketleriyle anlatılmasıdır. Mimikler günlük iletişimimizde önemli bir yer tutmaktadır. İnsanlar mimiklerini kullanarak birçok duygu ve düşüncesini karşılarındaki kişiye anlatabilir. Mimikler, ifadeleri kuvvetlendirici anlamlar da taşımaktadır. Yüz ifadelerini inceleyerek bir kişinin mutlu, üzgün, şaşırmış gibi ruh hallerini tahmin etmek mümkündür.

Bilgisayar yardımıyla insan yüzü görüntülerinin değerlendirilerek, tanımlı bir mimik kümesi içinde sınıflama işlemi yapılmasına mimik tanıma diyebiliriz. Bu çalışmada, gözler ve dudaklar ile oluşturulabilen 12 yüz hareketi incelenmiş ve her yüz hareketi belli bir istek ile eşleştirilmiştir. Bu sayede iletişim yetenekleri sınırlı, engelli veya hasta kişilerin, temel isteklerini mimiklerini kullanarak ifade etmeleri amaçlanmıştır.

Mimik tanıma işlemi için üç temel aşama vardır. İlk aşamada görüntü üzerinde yüzün yerinin bulunması ve bulunan yüz bölgesi üzerinde gözlerin ve dudakların yerlerinin tespit edilmesi gelir. İkinci aşamada elde edilmiş göz ve ağız görüntülerinin belirleyici özellikleri bulunmalıdır. Bu özelliklerin mümkün olduğunca kaliteli bir biçimde elde edilmesi çalışmanın üçüncü aşaması olan özelliklerin değerlendirilmesi için çok önemlidir. Yüz görüntüsünde gözlerin konumları, göz açıklığının genişlik ve uzunluğu, göz bölgesinde irisin bulunup bulunmaması, göz bölgesinin renk açısından değerlendirilmesi ve şablon eşleştirme yöntemiyle elde edilen korelasyon değerleri gözlerden elde edilebilecek özelliklerden bazılarıdır. Dudaklar için ise dudakların en sağ ve en sol noktaları, en üst ve en alt noktaları, dudak bölgesinin alanı, çevresinin uzunluğu gibi özellikler vardır. Bu özellikler haricinde farklı özellikler de çıkarmak mümkündür ve araştırmacıya bağlıdır.

Yüz tespiti, sayısal görüntü içinde yüzün bulunduğu ortamı ve yüzün büyüklüğünü bulan bilgisayar teknolojisidir. Bu işlem sonunda, görüntüdeki yüz dışındaki tüm öğeler yok sayılmaktadır. Yüz tespit işlemi obje tanıma işleminin bir alt sınıfı olarak kabul edilir. Görüntü içerisinde yüz bölgesi tespit edildikten sonra göz ve dudak bölgeleri bulunmalıdır.

Gözlerin yerlerinin bulunması için değişik yöntemler uygulanabilir. D'Orazio ve arkadaşları (2004) Hough dönüşümü kullanarak görüntüde irisin olup olmadığını sorgulamaktadır. Hyunwoo Kim (2004) ise gözlerin yerini belirlemek için görüntüde gözlerin köşelerini bularak gözün yerini bulmaktadır. Liang Tao gibi araştırmacılar ise gözleri yüz özelliklerini kullanarak bulmuşlardır. Zchichao Tian ve Huabiao Qin çalışmalarında göz bölgelerini

izdüşümler yardımıyla bulmuştur. Hong Liu ve arkadaşları (2005) gözlerin yerlerini tespit etmek için şablon eşleştirme kullanmışlardır.

Literatürde dudakların bulunması için çeşitli yöntemler geliştirilmiştir. Yuille ve arkadaşları (1989) çalışmalarında deforme olabilen şablonlar kullanarak ağzı bulmuşlardır. Ağzın yerini doğru olarak bulmasındaki yüksek başarısına rağmen, bu yöntem çok yüksek işlemci gücüne gereksinim duymaktadır. Bazı başka çalışmalarda ise ağzın yeri, ağız çevresindeki özellik noktalarını bularak tespit edilmiştir (Pantic, 2000). Önemli özellik noktaları üst dudağın en üst noktası, alt dudağın en alt noktası, dudakların sol ve sağ köşeleridir. Tian ve Qin (2005) çalışmalarında izdüşümler ile ağzın bulunduğu alanı belirlemişlerdir.

Yüz üzerinde gözlerin, dudakların bulunmasını ve bu bölgeler için belirleyici özelliklerin çıkarılmasını temel görüntü işleme yöntemleri kullanarak yapabiliriz. Alçak geçiren, yüksek geçiren, medyan filtre gibi filtreleme teknikleri, genişletme, aşındırma, açma, kapama, inceltme gibi morfolojik işlemler, renk uzayları arasında geçişler, görüntü izdüşümleri, şablon eşleştirme bu çalışmada kullanılan temel görüntü işleme metotlarından bazılarıdır.

Gözler ve dudaklar için belirleyici özellikler çıkarıldıktan sonra, bu özellikler yapay sinir ağları kullanılarak değerlendirilir.

Bir özellik matrisinin tanımlanması, sınıflanması, kümelenmesi birçok bilim dalı için önemli bir problemdir. Özellik matrisi, eğiticili ve eğiticisiz sınıflandırma kullanılarak değerlendirilir. Örneklerin tanımlanmış bir sınıfa ait olması durumunda eğiticili sınıflandırma, bilinmeyen bir sınıfa ait oldukları durumlarda ise eğiticisiz sınıflandırma yapılır.

Bu çalışmada kullanılan mimik kümesi için elde edilmiş özellikler, özellik matrisinin her vektörü için tanımlı bir sınıfa karşılık gelmektedir. Çalışma için 1 kişiye ait 322 görüntüden çıkarılan özellikler, yapay sinir ağının eğitilmesinde ve test edilmesinde kullanılmıştır. 211 görüntüye ait özellikler ağın eğitilmesinde, 111 görüntüye ait özellikler ise ağın test edilmesinde kullanılmıştır. Bu özellikler Çok Katmanlı Algılayıcı Ağlar, Radyal Temelli Fonksiyon Ağları ve Konik Kesit Fonksiyonlu ağ yapıları ile değerlendirilmiştir.

Tezin takip eden bölümleri, görüntü işleme, yapay sinir ağları, yüz üzerinde gözlerin ve dudakların bulunarak belirleyici özelliklerinin tespit edilmesi, bulunan özelliklerin yapay sinir ağları kullanılarak değerlendirilmesi ve sonuçlar olmak üzere 5 bölüme ayrılmıştır. Takip eden ilk bölüm, temel görüntü işleme metotları hakkında bilgi sağlamak amacıyla verilmiştir.

2. GÖRÜNTÜ İŞLEME

Görüntü işleme, biyometri, askeri keşif ve gözlem, tıbbi teşhis, endüstriyel uygulamalar, güvenlik gibi alanlarda kullanılmaktadır.

Görüntülerin bilgisayar ortamında değerlendirilmeleri için veri formatlarının sayısal ortama uygun olmaları gerekmektedir. Analog görüntülerin uygun formata dönüştürülmesi işlemine sayısallaştırma denir. Analog özellik gösteren görüntü işaretleri, sayısal ortama taşınırken ayrık işaretlere dönüştürülmektedir. Analog bir işaret sayısal işarete iki aşamada çevrilir. Öncelikle analog görüntü örneklenir. İkinci aşamada ise örneklenmiş görüntü kuantalama ile sadece belli genlik değerlerine eşleştirilir (Ertürk, 2004).

Görüntü, örnekleme sonucunda piksel cinsinden ifade edilen 256 x 256, 128 x 128 vs gibi değerler alır. Kuantalama sonucunda ise bir pikselin değerinin kaç bit ile ifade edileceği bulunur.

Sayısal görüntü denince, m x n boyutlarında piksellerden oluşan matris aklımıza gelmektedir. Görüntüleri kendi aralarında çeşitli alt gruplara bölebiliriz.

İkili (Binary) Görüntü: Siyah ve beyaz görüntülerdir. Her piksel 1 bitle ifade edilir. Pikselin aldığı değer f(x,y) görüntüsü için 0 ise siyah, 1 ise beyaz renktedir. Siyah beyaz görüntüler şeklin genel hatlarını, sınırlarını belirlemek için kullanılır.

Gri Seviye Görüntü: Bu görüntü renk bilgisi içermeden, sadece parlaklık bilgisi içerir. Genelde her piksel için 8 bit kullanılarak siyah ile beyaz arasındaki gri tonları da görüntüde görmemizi sağlar. 8 bitlik f(x,y) görüntüsü 0 ile 255 arasında parlaklık değerleri alır. Bazı özel görüntü gereksinimleri için 8'den farklı bit sayısı da kullanılmaktadır.

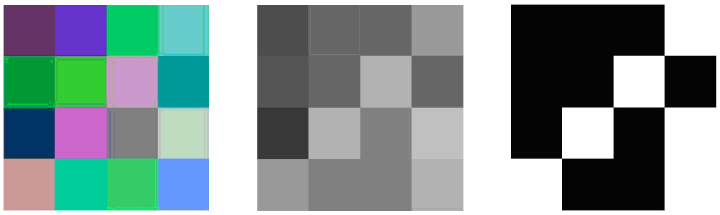
Renkli Görüntü: İnsan gözünün algılayabileceği renklerle görüntünün ifade edilmesidir. Örneğin RGB renk uzayında kırmızı, yeşil ve mavi olarak kodlanmış 3 adet renk tonu üst üste ekrana yansıtılmaktadır.

Multispektral Görüntü: İnsan gözünün algılayamayacağı görüntü bilgilerini de içerir. X-ışınları, ultraviyole ışınlar, kızılötesi ışınlar multispektral görüntüye örnektir.

Bu tez çalışması ikili görüntü, gri seviye görüntü ve renkli görüntüler kullanılarak yapılmıştır. Renkli görüntülerden gri seviye görüntü elde etmek için en basit yöntem, renkli görüntünün her bileşeninin, $f_1(x,y)$, $f_2(x,y)$, $f_3(x,y)$ toplanıp, bileşen sayısına bölünmesidir. NTSC standartlarına göre ise bir RGB görüntü 2.1 eşitliği kullanılarak gri seviye görüntüye çevrilir.

$$Gri\_Seviye\_Görüntü = 0.299R + 0.587G + 0.114B \quad (2.1)$$

Elde edilen gri seviye görüntü ise belli bir eşik değerinin üzerindeki pikseller 1'e, altındaki pikseller ise 0'a eşitlenerek ikili görüntü haline çevrilir. Şekil 2.1 a,b,c'de bu çevrimlerin yapılması görülmektedir. Matlab, renkli görüntüleri gri seviye görüntüye rgb2gray komutunu kullanarak çevirmektedir.



Şekil 2.1 Eşikleme a) Renkli Görüntü b) Gri seviye görüntü c) eşik=128 ile elde edilmiş siyah beyaz görüntü

2.1 Görüntü İşleme Yöntemleri

Görüntü işleme yöntemleri ile çeşitli nedenlerden dolayı yeterli bilgi vermeyen görüntüler işlenerek görsel anlamda kullanılabilir hale getirilir. Bu işlemleri aşağıdaki gibi sınıflayabiliriz.

- Nokta işlemleri (eşikleme, adaptif eşikleme, vs)
- Aritmetik işlemler (toplama, çıkarma, vs)
- Geometrik işlemler (ölçekleme, döndürme, vs)
- Morfolojik işlemler (açma, kapama, aşındırma, yayma)
- Filtreleme teknikleri

Görüntü işleme yöntemleri hakkında daha detaylı bilgi edinmek için Ertürk (2004) ve Gümüştekin (2005) çalışmaları faydalı kaynaklardır. Bu bölümün devamında görüntü işleme yöntemleri kısaca anlatılmaktadır.

Görüntü Ters Çevirme: Ters çevirme işleminde görüntünün en üstteki satırı ile en alttaki satırından başlamak üzere görüntünün bütün yatay satırları yer değiştirir. Görüntü yatay ortasından katlanmış olur.

Görüntü Aynalama: Bu işlem sırasında görüntünün en sol kolonu ile en sağ kolonundan başlamak üzere tüm kolonları yer değiştirir. Görüntü dikey ortasından katlanmış olur.

Görüntü Döndürme: Görüntü belirtilmiş bir açıyla döndürülür.

Görüntü Öteleme: Öteleme işleminde görüntü yatay ve düşey eksenlerde belirtilen piksel miktarı kadar kaydırılır.

Görüntü Büyütme (Interpolasyon): Bir görüntü, pikselleri kopyalanmak suretiyle büyütülür. Bu işleme interpolasyon denir. Doğrudan piksel değerlerinin kopyalanarak görüntünün büyütülmesi özellikle görüntü içindeki cisimlerin kenar bölgelerinde sert geçişlere neden olur. Bir başka interpolasyon yöntemi ise doğrudan piksel değerlerini kopyalamak yerine komşu piksellerin ortalamasını alarak görüntüyü büyütmektir. Bu yöntem ilk yönteme göre görüntüde daha düzgün geçişler sağlar.

Parlaklık Ayarı: Görüntünün parlaklığın değiştirilmesi, f(x,y) görüntüsünün piksel değerlerinin belirli bir sayıyla toplanması veya çıkartılmasıyla yapılmaktadır. Bu işlem eşitlik 2.2'de verilmiştir.

$$g(x,y)=f(x,y) + a \tag{2.2}$$

a değeri sıfırdan büyükse görüntünün parlaklığı açılır. Sıfırdan küçük olduğu durumlarda ise görüntünün parlaklığı azalır.

Karşıtlık Ayarı (Kontrast): Görüntüde karşıtlığın değiştirilmesi f(x,y) görüntüsünün piksel değerlerinin bir katsayı ile çarpılması sonucu olur. Eğer katsayı m birden küçükse karşıtlık azalır, katsayı m birden büyükse karşıtlık artar. Karşıtlık kontrast olarak da bilinir. Karşıtlık ayarı için 2.3'deki denklem kullanılmaktadır.

$$g(x,y)=m * f(x,y) \tag{2.3}$$

Görüntü Eşikleme: Gri seviye bir görüntüde belli bir eşik değerinden yukarıda olan piksel değerleri bire, eşik değerinden düşük olan piksel değerleri ise sıfıra çekilerek ikili görüntüye çevrilir. İkili görüntü siyah ve beyazdır.

Görüntü Olumsuzlama: Bir görüntünün negatifi, görüntü üzerindeki siyah değerler beyaza, beyaz değerler ise siyaha çevrilerek elde edilir. Piksel değerleri doğrusal bir biçimde ters çevrilmektedir.

Histogram: Kuantalama sonucu elde edilmiş her piksel seviyesinin görüntüdeki tekrarlama sıklığını gösterir. Gri seviye bir görüntünün histogramı 0 – 255 arası değerlerde histogramın sol tarafında yoğunlaştığı takdirde, görüntünün koyu bir görüntü olduğunu söyleyebiliriz. Eğer histogram sağ tarafta yoğunlaşmış ise bu görüntü açık tonlu bir görüntüdür. Histogram gri seviye spektrumda dar bir alanda yoğunlaşmış ise karşıtlığı düşük bir görüntü söz konusudur. Histogram tüm spektrumda yayılmış durumda ise görüntü yüksek kontrastlıdır.

Histogram Eşitleme: Görüntüdeki düşük karşıtlığı azaltmak amacıyla uygulanır. Doğrusal olmayan bir uygulamadır. Yüksek frekanslı piksel değerleri histogramda genişletilirken,

düşük frekanslı piksel değerleri ise histogram üzerinde dar bir alana yerleştirilir. Bu yöntem ile çok kullanılan piksel seviyeleri belirginleşmektedir.

Kontrast Yayma: Doğrusal bir işlem olup, görüntüdeki en düşük ve en yüksek piksel değerlerine bağlıdır. Herhangi bir f(x,y) görüntüsü için denklem 2.4 kullanılarak yapılır.

$$g(x,y) = \frac{f(x,y) - \min(f(x,y))}{maks(f(x,y)) - \min(f(x,y))} x255 \qquad (2.4)$$

Piksel Komşuluğu: Piksel temel alınarak yapılan komşuluk işlemlerinde, pikselin yeni değeri, komşu olduğu piksellerin değerleri baz alınarak hesaplanır. Pikseli, komşu piksellerle birlikte değerlendirdiğimizde daha fazla bilgi elde edebilmekteyiz.

Konvolüsyon: Konvolüsyon sonucunda bir pikselin yeni değeri çevresindeki piksellerin ağırlıklı ortalaması ile bulunur. Komşu piksellerin değerleri, konvolüsyon kernelinin katsayıları ile çarpılmaktadır. Konvolüsyon işlemi aşağıdaki denklem kullanılarak yapılır. Denklemde k konvolüsyon kernelini, f ise görüntüyü ifade etmektedir. Konvolüsyon kernelinin boyutu uygulamaya göre 3x3, 5x5 veya 7x7 olabilir.

$$g(x,y) = \sum_{j=-n}^{n} \sum_{i=-m}^{m} k(i,j) f(x-i, y-j) = k * f \qquad (2.5)$$

mxm boyutlu bir konvolüsyon kerneli ile konvolüsyon yapmak için m^2 çarpma ve m^2 toplama işlemi gereklidir. mxm boyutlarında bir konvolüsyon kerneli eğer mx1 ve 1xm boyutlarında iki kernelin çarpımı şeklinde yazılabiliyorsa bu ayrıştırılabilir bir kerneldir. Ayrıştırılabilir kernelde 2*m çarpma ve 2*m toplama işlemiyle konvolüsyon yapılır (Ertürk, 2004).

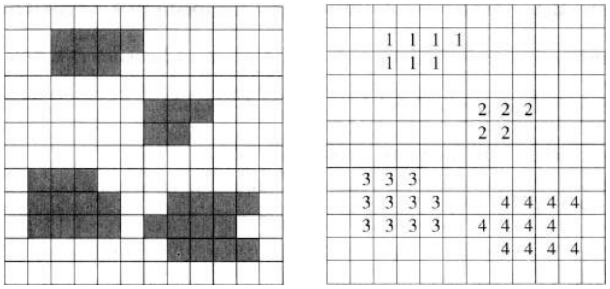
Boyut Normalizasyonu: Bir f(x,y) görüntüsünün genişliğini sabit bir değere çekebilmek için, görüntünün genişliği sabit bir değerle çarpılır. Görüntünün boyu da aynı değerle çarpılarak genişlik-yükseklik oranı değişmeden görüntünün boyutları değiştirilir. Bu şekilde görüntü üzerindeki özelliklerin oranları da sabit kalmış olurlar.

Bileşen Etiketleme (Komponent etiketleme): Siyah beyaz bir görüntü birden fazla cisim içerebilir. Bu cisimlerin etiketlenmesi bazı durumlarda faydalı olur. Bileşen etiketleme şu şekilde çalışır.

- Görüntüyü soldan sağa ve yukarıdan aşağıya doğru tara.
- Taranan piksel değerlerinden, değeri 1 olan varsa bu pikseli a olarak etiketle
- Tarama sırasında değeri 1 olan bir pikselle karşılaşılırsa, bu pikselin sol veya üstünde etiketlenmiş bir piksel varsa, bu pikselin etiketini yeni piksele kopyala.

- Tarama sırasında değeri 1 olan bir pikselle karşılaşıldığında, bu pikselin sol ve üstünde farklı etiketlenmiş pikseller varsa, herhangi birinin etiketini yeni piksele kopyala ve her iki etiketi de birbirine eşitle.
- Tarama sırasında değeri 1 olan bir pikselle karşılaşılırsa, bu pikselin sol veya üstünde etiketlenmiş bir piksel yoksa bu piksele yeni bir etiket ver.

Aşağıdaki şekil bileşen etiketlemeyi göstermektedir.



Şekil 2.2 Bileşen etiketleme a) siyah beyaz görüntü b) görüntü üzerinde etiketlenmiş cisimler

İskelet Çıkarma: Görüntüdeki şekillerin karakteristiğini bozmadan gereksiz pikselleri görüntüden çıkarmayı amaçlamaktadır. İskelet çıkarma ile şeklin karmaşıklığı azaltılmış olur. İkili görüntülerin ana hatları bulunur. İskelet çıkarma metotlarından bir tanesi uzaklık dönüşümüdür. Uzaklık dönüşümü ile asıl görüntü iskeletten geri bulunabilir. Şekil 2.3 (a)'da 1 ile gösterilen pikseller cismi göstermektedir. 0 ise cismin dışındaki pikselleri göstermektedir. Cismin üzerindeki piksellerin cismin dışındaki piksellere uzaklıkları komşuluk ilişkilerine göre bulunur. Şekil 2.3 (b), 4 komşuluk ile, Şekil 2.3 (c) ise 8 komşuluk ile yapılmış uzaklık dönüşümünü göstermektedir. 2.3 (b) ve (c)'de görüntü üzerinde 1 ile etiketlenmiş pikseller cismin iskelet halidir.

0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0
0	1	1	2	2	2	1	0
0	1	2	2	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Şekil 2.3 Uzaklık dönüşümü uygulanması (a) siyah beyaz cisim (b) 4 komşuluğuna göre uzaklık dönüşümü (c) 8 komşuluğuna göre uzaklık dönüşümü

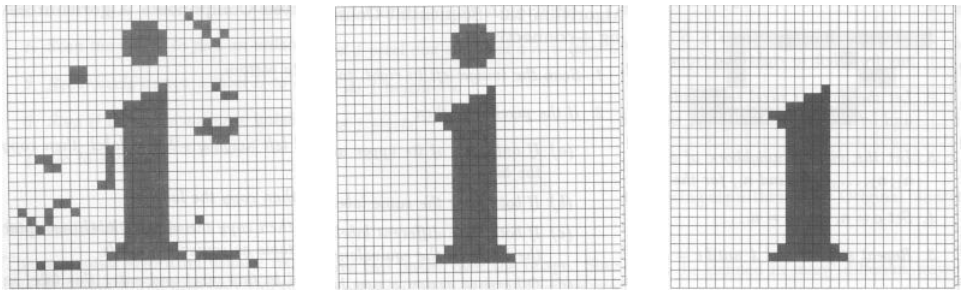
Değişken Eşikleme: Sabit bir eşik değeri kullanılarak gri seviye görüntüden ikili seviye görüntüye geçişler, farklı parlaklık ve ışık koşullarında çekilmiş görüntülerde uygun sonuçlar vermez. Dinamik olarak eşik değerinin belirlenmesi, ikili görüntünün net bir şekilde elde

edilmesini sağlar. Dinamik eşikleme Otsu algoritması kullanılarak yapılır. Otsu algoritması, görüntüdeki piksel değerlerinin dağılımlarına göre kümelendirilmesini sağlamaktadır. Bu algoritma ile her görüntü için en uygun eşik değeri hesaplanır.

2.2 Gürültü Azaltma

Görüntü üzerindeki gürültü önemli bir bilgi taşımaz ve çoğu zaman için istenmeyen bir öğedir. Çeşitli süzgeçler kullanılarak görüntülerdeki gürültü etkisini azaltmak mümkündür.

Boyut filtresi: Bazı gürültüler boyut filtresi ile kaldırılabilir. Siyah beyaz görüntü üzerinde belli bir eşik boyutundan daha küçük olan bölgelerin piksel değerleri sıfıra eşitlenerek gürültü engellenir. Eşik değeri için uygun değeri bulmak zor bir işlemdir. Eğer eşik değeri gereğinden büyük alınırsa görüntü üzerinde faydalı bilgiler kaybolabilir. Eşik değerinin uygun değerden küçük alınması durumunda ise bazı gürültüler giderilemeyecektir. Şekil 2.4'de boyut filtresinin çalışması görülmektedir. Şekil 2.4 (a) gürültü içeren görüntüyü, 2.4 (b) uygun eşik değeriyle boyut filtresinden geçirilmiş şekli, 2.4 (c) ise büyük bir eşik değeri kullanılarak faydalı bilgilerinde yok olduğu görüntüyü göstermektedir.



Şekil 2.4 Boyut süzgeci uygulanması (a) Gürültülü görüntü (b) Eşik değeri T=10 ile filtrelenmiş görüntü (c) Eşik değeri T=30 ile filtrelenmiş görüntü

2.2.1 Alçak Geçiren, Yüksek Geçiren ve Medyan Süzgeçler

Görüntüler frekans uzayında parlaklık ve renk değişimlerine göre gösterilirler. Görüntüde frekans, yatay ve düşey doğrultudaki pikseller arasındaki değişim ile ilişkilidir.

Görüntü üzerindeki pikseller arasındaki yumuşak geçiş düşük frekanslara karşılık gelmektedir. Görüntü üzerindeki pikseller arasındaki sert geçişler ise, kenarlar, nesne sınırları gibi, yüksek frekanslara karşılık gelir.

Alçak Geçiren Süzgeç: Görüntü üzerindeki alçak frekanstaki sinyalleri geçiren, yüksek frekanstaki sinyallerin ise genliklerini azaltan süzgeçtir. Eleman değerleri pozitif olan kerneller görüntü ile konvolüsyona sokulduğunda alçak geçiren süzgeç görevi görmektedir. Bu işlem sonucunda görüntüde pikseller arası daha yumuşak geçişler olur. Konvolüsyon

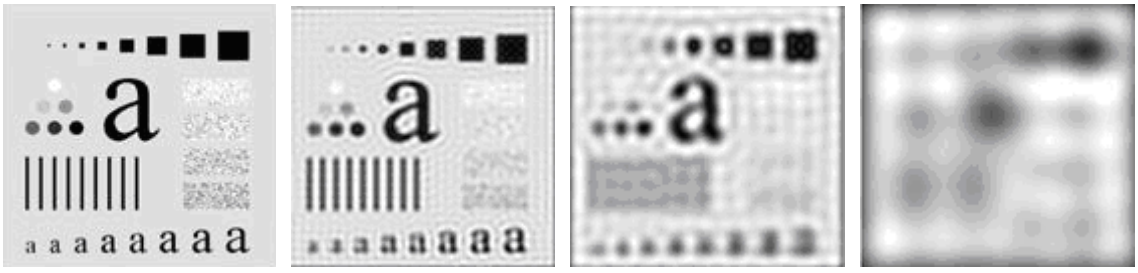
kernelinin boyutu büyüdükçe pikseller arası geçişler daha da yumuşamaktadır. Aşağıdaki şekilde doğrusal alçak geçiren süzgeç kerneli ile bir Gauss alçak geçiren süzgeç kerneli görülmektedir.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1	4	1
4	16	4
1	4	1

Şekil 2.5 Alçak geçiren süzgeç kernelleri (a) Doğrusal alçak geçiren süzgeç kerneli (b) Gauss alçak geçiren süzgeç kerneli

Düşük frekans verileri veya başka bir değişle pikselden piksele fazla değişim göstermeyen veriler alçak geçiren süzgeçten geçerler. Görüntü üzerinde değişimin az olduğu bölgeler frekans domeninde düşük frekansa karşılık gelir. Alçak geçiren süzgeçler görüntü üzerindeki gürültüyü kaldırırken, kenarlarda ve yüksek frekanslı bilgi içeren öğelerde bilgi kaybına neden olur. Alçak geçiren süzgecin kullanılmasındaki amaç gereksiz detayların görüntüden atılması ve büyük özelliklerin ortaya çıkarılmasıdır. Alçak geçiren filtrenin boyutu arttıkça görüntü üzerinde daha büyük boyutlu detaylar da kaybolacaktır. Şekil 2.6'da farklı kesim frekansları ile alçak geçiren süzgecin görüntüye uygulanması verilmiştir. Kesim frekansları arasındaki ilişki $f_1 > f_2 > f_3 > f_4$ şeklindedir.



Şekil 2.6 Alçak geçiren süzgeç uygulaması (a) kesim frekansı f_1 (b) kesim frekansı f_2 (c) kesim frekansı f_3 (d) kesim frekansı f_4

Yüksek Geçiren Süzgeç: Yüksek geçiren süzgeçlerde kernel değerleri hem pozitif, hem de negatif değerler alır. Genellikle kenar bulma ve görüntü içindeki ayrıntıları pekiştirmede kullanılır. Görüntü üzerindeki yüksek frekanstaki sinyalleri geçirir, alçak frekanstaki sinyallerin genliklerini ise azaltarak bastırır. Yüksek geçiren süzgeç, uzaktan algılama, kenarların bulunması, çizgi ve kenarların güçlendirilmesi veya imajın keskinliğini arttırmak için kullanılır. Yüksek geçiren filtrenin boyutunu, belirginleştirilmek istenilen özelliklerin boyutuna göre ayarlamak gereklidir. Yüksek geçiren süzgeç kernelinin boyutu arttıkça büyük

özellikler görülmeye başlanır. Bu süzgeç, çevresindeki piksellerin ortalamasıyla merkez pikselin farkını alır. Farkın büyük olduğu pikseller görüntü üzerinde belirginleşir. Aşağıda yüksek geçiren süzgeç kernelleri örnek olarak verilmiştir.

-1	-1	-1
-1	8	-1
-1	-1	-1

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	2	4	2	-1

Şekil 2.7 Yüksek geçiren süzgeç kernelleri a) 3 x 3 kernel b) 5 x 5 kernel

Medyan Süzgeç: Pikselin yeni değeri, mxm blok içindeki değerlerin ortanca değeri bulunarak seçilir. Medyan süzgeci tuz ve biber gürültüsünü görüntüden silmede alçak geçiren filtreye göre daha başarılıdır.

2.3 Kenar Bulma

Kenar, görüntü üzerindeki gri seviye değerlerde veya renklerde görülen ani değişimlerdir. Birçok görüntü işleme probleminin çözümü için kenar bulma kullanılır. Sık kullanılan kenar bulma kernelleri Prewitt, Sobel, Roberts, Zero Cross, Gauss of Laplacian ve Canny'dir. Kenar bulma kerneli görüntüye uygulanmadan önce görüntüde gürültü azaltıcı bir süzgeç uygulanması ve kenarların pekiştirilmesi daha iyi sonuçlar alınmasını sağlar.

2.4 Şablon Eşleştirme

İki görüntü arasındaki ilişkinin bulunmasında veya bir görüntü içerisinde önemli bir öğenin aranmasında kullanılmaktadır. Konvolüsyondan farklı olarak kerneldeki katsayılar doğrudan kendi konumlarına karşılık gelen pikseller ile çarpılmaktadır. Korelasyon denklem 2.6 kullanılarak hesaplanır.

$$g(x,y) = \sum_{k=-n}^{n} \sum_{j=-m}^{m} h(j,k) f(x+j, y+k) \tag{2.6}$$

Korelasyon sonuçları 0 ile 1 arasına normalize edilerek çekilir. Normalize edilmiş korelasyon (ilgileşim) ise denklem 2.7 kullanılarak hesaplanır. En yüksek korelasyon değeri, görüntü içinde aranmakta olan cismin olma ihtimali en yüksek olan noktayı ifade eder.

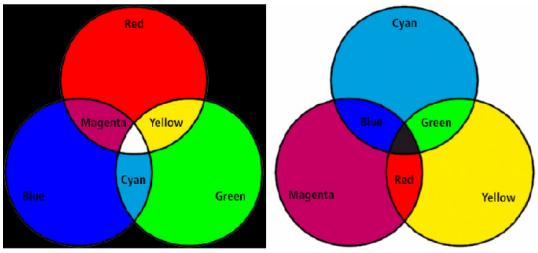
$$g'(x,y) = \frac{\sum_{k=-n}^{n}\sum_{j=-m}^{m} h(j,k) f(x+j, y+k)}{\sum_{k=-n}^{n}\sum_{j=-m}^{m} f(x+j, y+k)} \tag{2.7}$$

2.5 Renk Uzayları

Renk, 400nm ile 700nm arasında dalga boyuna sahip ışığın retinaya düşmesinin sonucu algılanmaktadır. Işık, elektromanyetik dalga spektrumunun bir parçasıdır.

İnsan gözü ışığı retina ile algılar. Retina, koni ve çubuk hücrelerinden oluşur. Koni hücreleri gündüzleri renkleri algılarken, çubuk hücreleri ise gece görüşünü sağlamak için sadece parlaklığı algılar.

Herhangi bir renk, üç ana rengin doğru oranda karışımı ile elde edilir. Aydınlatan kaynaklar için ana renkler, kırmızı, yeşil ve mavidir. Yansıtan kaynaklar için ise ana renkler camgöbeği, morumsu kırmızı ve sarıdır. Bu renkler ikincil ana renkler olarak da bilinmektedir.



Şekil 2.8 Renk uzayları (a) RGB renk modeli (b) CMY renk modeli

Bir renk modeli ana (RGB) veya ikincil (CMY) renklerle belirlenebileceği gibi, parlaklık ve renklilik bileşenleri kullanılarak da belirlenebilir. Önemli renk modelleri RGB, CMY, HSI, YCbCr ve YUV'dur.

Fazla sayıda renk gösterme gereksinimi olan durumlarda CMY veya RGB renk uzayları tercih edilmelidir. RGB renk modeli, CMY renk modeline göre insan gözünün görebildiği daha fazla sayıda rengi gösterebilir (Ertürk, 2004). RGB ve CMY renk modelleri arasında dönüşümler 2.8 ve 2.9 eşitlikleri kullanılarak yapılır. Aşağıdaki denklemlerden görüldüğü üzere CMY ve RGB renk modelleri birbirinin tümleyenidir.

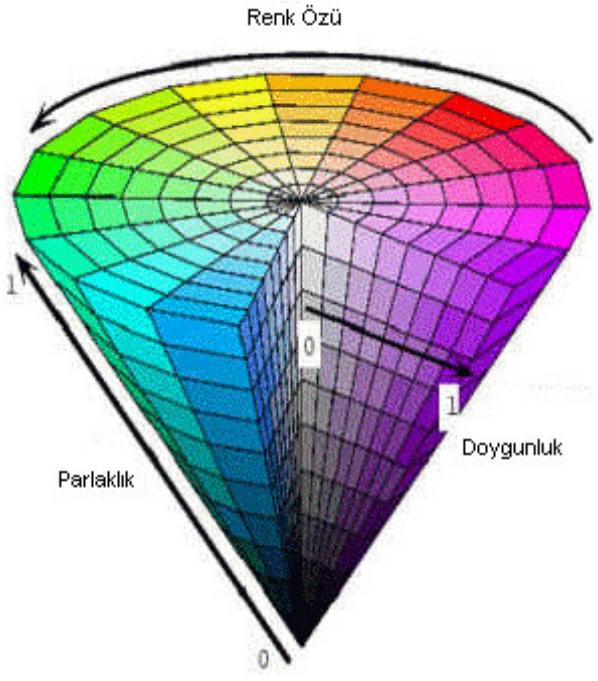
CMY ve RGB arasındaki dönüşümler:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.8}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \tag{2.9}$$

CMY renk modeli siyah renk eklenerek CMYK şeklinde geliştirilmiştir.

HSI renk modeli: Bu renk modeli, renk özü (Hue), doygunluk (Saturation) ve parlaklık (Intensity) bileşenlerinden oluşmaktadır. Renk özü bileşeni baskın rengi gösterir ve baskın dalga boyu ile ilgili bir niteliktir (Gümüştekin, 2005). Açısal olarak 0 ile 360 derece arasında değerler alır. Doygunluk bileşeni saf bir rengin beyaz ışıkla ne kadar seyreltildiğini gösterir. Rengin saflığı da denir. 0 ile 1 arasında değerler alır. Doygunluk değeri rengin canlılığını ifade eder. Rengi açmak veya koyulaştırmak için doygunluk değerini değiştirmek gerekir. Parlaklık değeri ise rengin içindeki beyaz oranını göstermektedir. Gölge etkisinden veya ışık değişimlerinden etkilenmemek için HSI renk modeli kullanılması tercih edilir. Renk farklarının analitik analizi için HSI uygun bir renk modelidir. İnsanın görsel algılamasına benzer bir yapıya sahiptir. HSI renk modeli, RGB renk uzayından doğrusal olmayan bir dönüşüm ile elde edilir.



Şekil 2.9 HSI Renk modelinin yapısı

RGB ve HSI renk modelleri arasındaki dönüşümler:

RGB modelindeki renk değerleri eşitlik 2.10'da görüldüğü üzere normalize edilir. 2.11 , 2.12 ve 2.13 denklemleriyle RGB görüntüler HSI renk modeline çevrilir. 2.14'te gösterilen üç

sütun halindeki denklemler kullanılarak HSI renk modelinden RGB renk modeline geri dönüşüm yapılır.

$$r = \frac{R}{R+G+B} \ , \ g = \frac{G}{R+G+B} \ , \ \frac{B}{R+G+B} \tag{2.10}$$

$$H = \begin{cases} \theta & if \quad b \le g \\ 360 - \theta & if \quad b \succ g \end{cases} \qquad \theta = \cos^{-1}\left\{ \frac{0.5[(\mathrm{r}-\mathrm{g}) + (\mathrm{r}-\mathrm{b})]}{[(\mathrm{r}-\mathrm{g})^2 + (\mathrm{r}-\mathrm{b})(\mathrm{g}-\mathrm{b})]^{0.5}} \right. \tag{2.11}$$

$$\mathrm{S}=1 - \frac{3}{(r+g+b)} \min(r,g,b) \tag{2.12}$$

$$\mathrm{I}= \frac{1}{3}(r+g+b) \tag{2.13}$$

$$0 \le H \prec 120 \qquad 120 \le H \prec 240 \qquad 240 \le H \prec 360 \tag{2.14}$$

b= I (1-S) r=I (1-S) g=I (1-S)

$$r = I\left[1 + \frac{S\cos H}{\cos(60-H)}\right] \quad g = I\left[1 + \frac{S\cos(H-120)}{\cos(60-(H-120))}\right] \quad b = I\left[1 + \frac{S\cos(H-240)}{\cos(60-(H-240))}\right]$$

g=1-(r + b) $b = 1-(r+g)$ r = 1-(g + b)

YUV ve YCbCr renk modeli: YUV renk modelinde Y parlaklık U ve V ise renkleri ifade eder. PAL TV yayın sisteminde kullanılan renk modelidir. Y işareti siyah beyaz, U ve V işaretleri ise mavi ve kırmızı renk bilgilerini içerir. YCbCr renk modeli, YUV renk modeline yakındır. Cb ve Cr bileşenleri U ve V bileşenlerinin ölçeklenmiş halidir. RGB renk modelinden YUV renk modeline dönüşüm aşağıdaki gibi yapılır.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.378 \\ 0.578 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.15}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1.14 & 0.0 \\ 1 & -0.394 & -0.581 \\ 1 & 0 & 2.028 \end{bmatrix} \tag{2.16}$$

YUV ve YCbCr renk uzayları iletim ve sıkıştırma hatalarını RGB renk uzayına göre daha iyi tolere etmektedir. YUV ve YCbCr renk modelleri, video aktarımı ve saklanması için diğer renk modellerine göre daha başarılıdır.

2.6 Morfolojik İşlemler

Matematiksel morfoloji geometrik şekilleri analiz etmeye ve işlemeye yarayan bir teoridir. Genelde ikili görüntülere uygulanmaktadır.

Morfolojik görüntü işlemede iki temel işlem vardır. Bunlar aşındırma (erosion) ve yayma (dilation) işlemleridir. Açma ve kapama işlemleri bu işlemler temel alınarak yapılır.

2.6.1 Aşındırma

A bir ikili görüntüyü ifade ederken, B yapı elemanını temsil eder. Aşındırma işlemi aşağıdaki denklem kullanılarak bir küme işlemi şeklinde hesaplanır.

$$A\Theta B = \{z \mid (B)_z \subset A\} \tag{2.17}$$

Yapı elemanı B'nin merkezi A görüntüsündeki hedef pikselin üstündeyken, eğer B tümüyle A görüntüsündeki cisim tarafından kapsanıyorsa, A üzerindeki merkez piksele karşılık gelen piksel saklanır. Aksi durumda bu merkez piksel silinir. Başka bir deyişle, B tamamen A görüntüsündeki cismin içindeyse merkez piksel değeri sabit kalmakta, eğer B tamamen A görüntüsündeki cisim tarafından kapsanmamaktaysa A görüntüsü üzerinde yapısal elemanın merkez pikseline karşılık gelen piksel silinmektedir.

2.6.2 Yayma

İkili görüntüdeki cismi büyütmeye veya kalınlaştırmaya yarayan morfolojik işlemdir. Kalınlaştırma işleminin nasıl yapılacağını yapı elemanının şekli belirler. A bir ikili görüntüyü ifade ederken, B yapı elemanını temsil etmektedir. Yayma aşağıdaki formülle yapılır.

$$A \oplus B = \bigcup_{b \in B} Ab \tag{2.18}$$

Yapı elemanı B'nin merkezi A görüntüsündeki hedef pikselin üzerindeyken, A üzerinde yapı elemanının dokunduğu sıfıra karşılık gelen pikseller bire dönüşür. Böylelikle cisim kalınlaşır.

2.6.3 Açma İşlemi

Aşındırma ve yayma işlemlerinin ardışık bir şekilde görüntüye uygulanmasıyla elde edilir.

$$A \circ B = (A\Theta B) \oplus B \tag{2.19}$$

2.6.4 Kapama İşlemi

Yayma ve aşındırma işlemlerinin ardışık bir şekilde görüntüye uygulanmasıyla elde edilir.

$$A \bullet B = (A \oplus B)\Theta B \tag{2.20}$$

2.7 Hough Dönüşümü

Bu dönüşüm kullanılarak görüntü içerisinde çizgi, daire, elips gibi şekilleri bulabiliriz. Yüz ile ilgili yapılan işlemlerde özellikle irisin yerinin bulunmasında faydalıdır. Şekil 2.10 Hough dönüşümünün uygulanışını göstermektedir.

Şekil 2.10 Hough dönüşümü ile göz bebeğinin bulunması (a) Siyah beyaz göz görüntüsü (b) r=8 piksel alınarak bulunan göz bebeği

3. YAPAY SİNİR AĞLARI

Yapay Sinir Ağları, insanların sinir sisteminden yola çıkarak sistemlere, öğrenme, hatırlama, genelleme yapma gibi yetenekler kazandırmayı amaçlayan bilgi işleme mekanizmasıdır. Yapay Sinir Ağları, dışarıdan gelen girdilere dinamik bir şekilde yanıt oluşturur. Bir kuralı veya algoritması olmayan problemlerin çözümünde kullanılır. Yapay Sinir Ağları işlem elemanları ve bu elemanlar arası bağlantılardan oluşmaktadır. Giriş değerlerine karşılık çıkış değerleri üretir. En büyük avantajı öğrenme yoluyla ilk defa karşılaşılan problemlere çözüm bulabilmesidir.

Yapay Sinir Ağları insan beyninin çalışma prensibi üzerine kurulmuştur. Sinir sistemimizin temel işlem elemanı sinir hücresidir. İnsan beyninin 100 milyar sinir hücresinden oluştuğu düşünülmektedir. Bir sinir hücresi 3 bileşenden meydana gelir. Bu bileşenler, hücre gövdesi, dendritler ve aksondur. Dendritler diğer hücrelerden aldığı bilgileri hücre gövdesine iletir. Aksonlar ise bilgiyi elektriksel darbeler şeklinde hücreden dışarı taşımaktadır. Sinir hücreleri arasında sinaptik bağlantılar vardır ve bu bağlantıların sayısı 100 trilyon civarındadır.

İnsan beyninde nöronlar (sinir hücreleri) aralarındaki sinaptik bağlantıları ayarlayarak öğrenmeyi sağlar. İnsanlar doğdukları andan itibaren sürekli öğrenmektedirler. Yeni bilgiler alındıkça, nöronlar arasındaki sinaptik bağlantılar yeniden ayarlanır, hatta nöronlar arasında yeni bağlantılar oluşur.

Yapay Sinir Ağları (YSA) biyolojik nöron modelinden esinlenerek ortaya çıkmıştır. YSA ile insan beyninin temel özellikleri olan öğrenme, ilişkilendirme, sınıflandırma, genelleme, optimize etme, kümeleme yapılır. Matematiksel olarak modellenmesi mümkün olmayan, karmaşık problemler YSA ile modellenir. YSA olayları ve arkasındaki ilişkileri otomatikman öğrenir. Değişen koşullara adapte olur, ayrıca yeni durumların ortaya çıkması halinde yeniden eğitilebilir.

YSA'lar ağırlıklandırılmış şekilde birbirine bağlanmış birçok işlem biriminden oluşan sistemlerdir. Nöral hesaplamanın temelinde dağıtılmış, doğrusal olmayan işlem kavramı vardır. Normal işlemcilerden farklı olarak çalışırlar. Normal işlemcilerde, tek bir işlemci her işlemi sırasıyla seri bir şekilde yapmaktadır. YSA ise paralel çalışan çok sayıda basit işlem biriminden oluşmaktadır. Paralel çalışan bir sistemde, yavaş çalışan bir işlem biriminin tüm sisteme etkisi çok azdır. Herhangi bir işlem biriminin hasar görmesi, sonuca çok az etki yapar. Nöral hesaplamanın gücü, toplam işlem yükünü paylaşan işlem birimlerinin birbiri arasındaki yoğun bağlantı yapısından gelmektedir.

Yapay Sinir Ağlarının en yoğun uygulama alanı örnek tanımadır. Görüntü işleme ve işaret işlemede yoğun olarak kullanılırlar. Tıpta çeşitli hastalıkların teşhisinde, EKG, EEG gibi işaretlerin incelenmesinde uygulama alanına sahiptir. Askeri olarak radar sinyallerinin analizi, uzaktan algılama gibi uygulamaları vardır. Ayrıca ekonomi, meteoroloji gibi çeşitli konularda kullanım alanları da bulunmaktadır.

YSA'nın dezavantajları ise, ağ oluşturmada, ağ topolojisi belirlemede belli kurallarının olmamasıdır. Ağın davranışlarının açıklanması mümkün değildir. Bazı ağlar hariç kararlılık analizleri yapılamaz (Elmas, 2003).

Yapay Sinir Ağları, eğiticili öğrenme ve eğiticisiz öğrenme olarak iki şekilde öğrenir. Eğiticili öğrenmede sinir ağının eğitimi için eğitici veriler kullanılmaktadır. Eğitim seti, giriş bilgileri ve istenen (hedef) bilgiler olmak üzere iki ayrı matris şeklindedir. Eğitimin başlangıcında bağlantı ağırlıklarına rasgele değerler atanmaktadır. Bir öğrenme kuralı kullanılarak ağın ağırlıkları güncellenir. Hedef değerler ile ağın çıkışı arasındaki hata azalıncaya kadar eğitim sürdürülür. Ağ çıkışındaki hatanın azalması, ağın kararlılık kazanması anlamına gelir.

Eğiticisiz öğrenmede eğitim seti kullanılmaz. Eğiticisiz öğrenmede sinir ağı, birbirine benzer giriş bilgilerini gruplamaktadır. Ağın eğitimi için sadece giriş bilgileri yeterli olmakta, hedef vektörlerine ihtiyaç duyulmamaktadır. Bu tez çalışmasında ağın öğrenmesinde kullanılan yöntem, eğiticili öğrenmedir.

Yapay Sinir Ağlarının problemleri öğrenme başarısı testlerle sınanmalıdır. Bir bölüm örnek, ağı eğitmek için kullanılmalı, başka bir bölüm örnekle ise ağ test edilmelidir. Test işlemi için, eğitim kümesinde olmayan veriler kullanılmalıdır. Test işlemindeki amaç, yapay sinir ağının gerekli genellemeleri yapıp yapamadığını kontrol etmektir.

Başarılı sonuçlar alabilmek için, eğitim kümesindeki örneklerin problemi iyi bir şekilde tanımlıyor olması gereklidir.

Ağın eğitilmesinde öğrenme kuralının seçimi önemlidir. Öğrenme kuralları işlem elemanları arasındaki ağırlıkların güncellenmesini sağlar. Geri yayılım, Levenberg-Marquardt, DeltaBarDelta, Quickprob, Konjugate Gradyent, Resilient Propagasyon öğrenme kurallarından bazılarıdır.

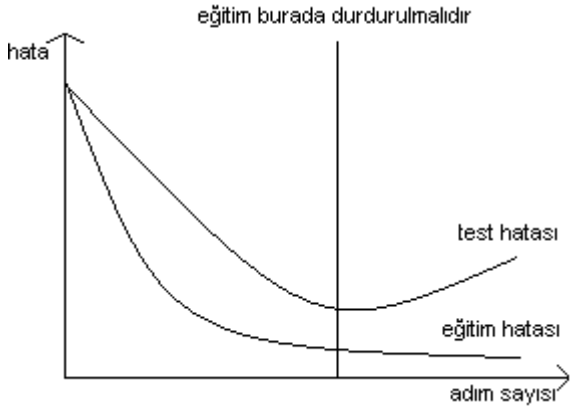
Geri yayılım algoritması çok katmanlı ağların eğitiminde sık kullanılan öğrenme kurallarından biridir. Hatanın gradyent düşümünü bulmaya dayanır. Kavramsal olarak basittir ve birçok örnek tanıma probleminde öğrenim kuralı olarak kullanılır. Geri yayılım kuralı

momentum ve öğrenme oranı değerlerini kullanarak ağın ağırlıklarını ayarlar. Bölüm 3.6'da öğrenme kuralları daha detaylı anlatılacaktır.

Öğrenme oranı, ağın ağırlıklarındaki değişim miktarını belirler. Eğer öğrenme oranı uygun orandan büyük seçilirse, problem uzayında rasgele salınım olur. Başka bir ifadeyle ağırlıklar çıkış değerleri ile hedef vektörü arasındaki hatayı küçültmez. Öğrenme oranı çok küçük seçildiği durumda ise çözüme ulaşılması uzun zaman alır.

Ağın öğrenmesine etki eden bir başka katsayı ise momentum değeridir. Momentum katsayısı, ağın yerel minimumlara takılmasını önler.

Eğer test kümesi hatası ile öğrenme kümesi hatası arasındaki fark her adımda artıyorsa, eğitim durdurulmalıdır. Ağ, öğrenme kümesi için çok fazla adımda eğitilirse, öğrenme kümesini ezberler ve test kümesi için genelleme yapma yeteneği zayıflar.

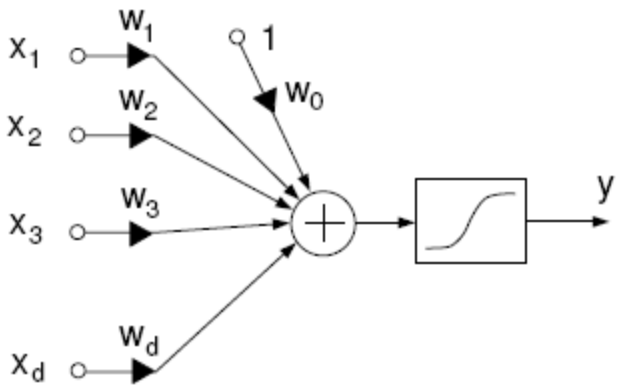


Şekil 3.1 Öğrenme seti hatası ve test seti hatasının ağ eğitimi süresince değişimi

Takip eden kısımlarda, Temel Nöron modeli, Çok Katmanlı Algılayıcılar, Radyal Fonksiyon Temelli Ağlar ve Konik Kesit Fonksiyonlu Ağlar hakkında bilgi verilecektir.

3.1 Temel Nöron Modeli

YSA nöronu, biyolojik sinir hücresinin matematiksel olarak modellenmesiyle elde edilmiştir. Aşağıdaki şekil, temel nöron modelini göstermektedir. x vektörü hücrenin giriş değeridir. Hücrenin bağlantı ağırlıkları w, giriş vektörüyle çarpılır ve her çarpım toplanarak hücrenin aktivasyon değeri elde edilir. Aktivasyon değeri, aktivasyon fonksiyonuna uygulanır. Temel Nöron Modeliyle doğrusal olarak sınıflama yapılır. Bu model doğrusal olması nedeniyle XOR problemini çözmeye uygun değildir.



Şekil 3.2 Temel Nöron Modeli

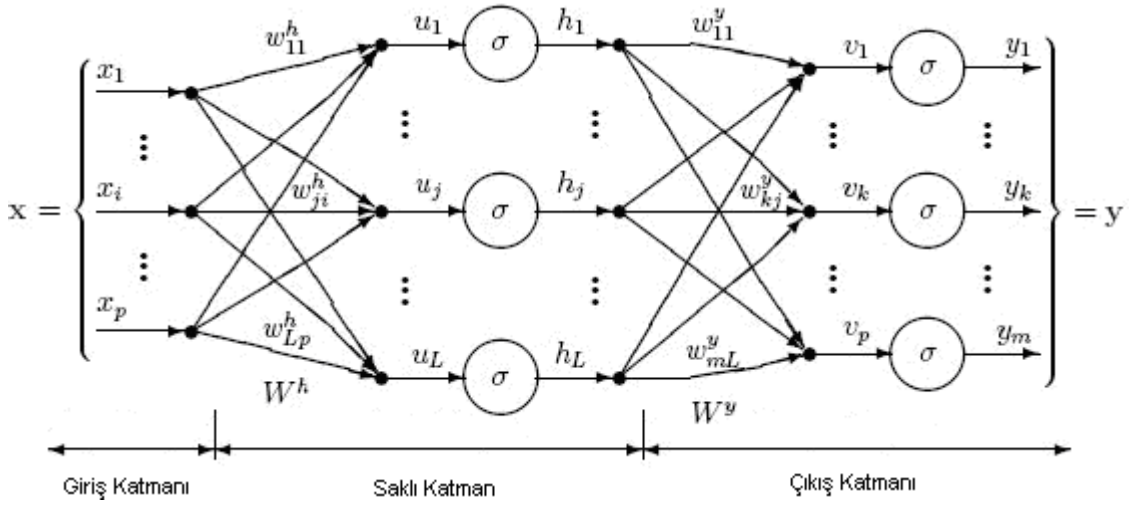
Yapay Sinir Ağlarında aktivasyon fonksiyonunun seçimi, girişlerin ve sınıfların sayısal değerleriyle ilişkilidir. Aktivasyon fonksiyonu olarak doğrusal fonksiyon, logaritmik sigmoid, tanjant sigmoid vs. kullanılabilir. Denklem 3.1 tanjant sigmoid fonksiyonu, denklem 3.2 ise logaritmik sigmoid fonksiyonunu vermektedir.

$$f(x) = \frac{2}{1+e^{-2n}} - 1 \tag{3.1}$$

$$f(x) = \frac{1}{1+e^{-n}} \tag{3.2}$$

3.2 Çok Katmanlı Algılayıcı (ÇKA)

Çok Katmanlı Algılayıcı bir giriş, bir veya daha fazla saklı katman ve bir çıkış katmanından oluşur. Bir katmandaki tüm işlem elemanları, bir üst katmandaki işlem elemanlarına bağlıdır. Bilgi akışı ileri yöndedir. İleri beslemeli sinir ağı modeli olarak da adlandırılır. Giriş katmanında herhangi bir bilgi işleme yapılmayıp, buradaki işlem elemanı sayısı uygulanan problemin giriş elemanı sayısına bağlıdır. Saklı katman sayısı ve saklı katmanlardaki sinir hücresi sayısı deneme yanılma yoluyla bulunur. Çıkış katmanındaki işlem elemanı sayısı ise uygulanan problemin çıktı sayısına bağlıdır (Demuth ve Beale 2000). Şekil 3.3 bir gizli katmanlı, p giriş elemanlı ve m çıkış elemanlı ÇKA ağını göstermektedir.



Şekil 3.3 Çok katmanlı algılayıcı ağın yapısı

ÇKA ile XOR probleminin çözülmesi mümkündür. XOR problemi, AND, OR ve NOT fonksiyonları kullanılarak oluşturulabilir. AND, OR ve NOT problemleri ise basit algılayıcılar (perceptronlar) ile çözülebilir.

Stone-Weierstrass teoremi kullanılarak saklı katmanında yeterli sayıda işlem elemanı bulunan tek saklı katmanlı bir ÇKA ile herhangi bir problemin çözülebildiği kanıtlanmıştır (Seung, 2002).

Saklı katmanlar ve çıkış katmanındaki aktivasyon fonksiyonları genelde birbirinden farklı seçilmektedir. Bir katmandaki işlem elemanlarının aktivasyon fonksiyonları ise aynı seçilmelidir.

3.3 Radyal Temelli Fonksiyon Ağları (RTFA)

RTFA ağları, ÇKA ağlarına benzer şekilde giriş, saklı katman ve çıkış katmanından oluşur. Giriş katmanından saklı katmana radyal tabanlı bir aktivasyon fonksiyonu kullanılarak geçilir. Radyal Temelli Fonksiyon Ağları aşağıdaki denklem ile tanımlanır.

$$f(x) = \sum_{k=1}^{K} \mathrm{w}_{\mathrm{k}}\ \varphi(\| x - c_k \|) + b \tag{3.3}$$

Yukarıdaki eşitlikte K saklı nöron sayısını ifade eder. c_k k. nöronun merkezini, w_k ise k. nöronun ağırlığını göstermektedir. b biasdır. Merkezler, rasgele veya sabit olarak seçilebilmektedir.

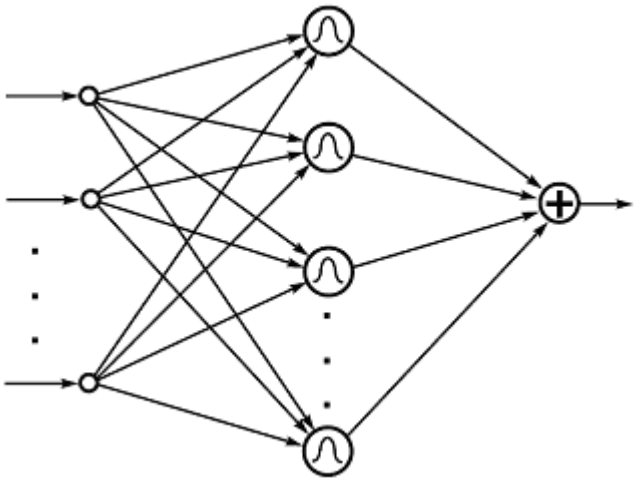
Doğrusal olmayan RTFA aktivasyon fonksiyonu olarak multikuadratik, ters multikuadratik veya Gauss fonksiyonları kullanılır. Bu fonksiyonlar aşağıda verilmiştir (Demuth ve Beale 2000).

$$\varphi(r) = \sqrt{r^2 + c^2} \tag{3.4}$$

$$\varphi(r) = \frac{1}{\sqrt{r^2 + c^2}} \tag{3.5}$$

$$\varphi(r) = e^{\frac{-r^2}{2c^2}} \tag{3.6}$$

RTFA ağı kullanılarak doğrusal olmayan herhangi bir fonksiyon istenilen dereceye kadar uydurulabilir. RTFA ağlarının genişlik parametresi, fonksiyon yaklaştırmanın ne kadar düzgün yapılacağını belirler. Ağın genişliği küçüldükçe genelleme yapma kapasitesi azalır. Genişlik her sinir hücresi için farklıdır. Aşağıdaki şekilde bir RTFA ağının yapısı görülmektedir.



Şekil 3.4 RTFA ağının yapısı

RTFA ağları ÇKA ağlarına göre daha hızlı öğrenme avantajına sahiptir. ÇKA ağlarına göre daha fazla sayıda sinir hücresine gereksinim duymaktadır. Eğitimde kullanılmak üzere çok sayıda özellik vektörünün olduğu durumlarda oldukça başarılı sonuçlar verir.

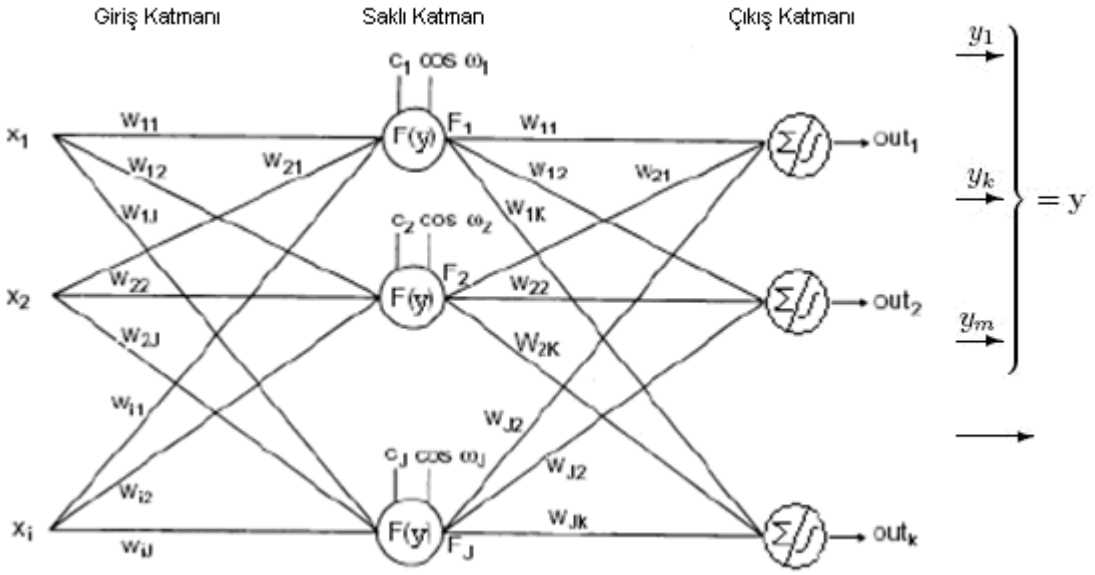
GRSA ve OSA, RTFA ağlarının farklı varyasyonlarıdır.

3.4 Konik Kesit Fonksiyonlu Sinir Ağları (KKFSA)

ÇKA ağlarının karar sınırları parabol, doğrular gibi açık eğrilerdir. RTFA ağlarının karar sınırları ise daire, elips gibi kapalı eğrilerdir. Konik Kesit Fonksiyonlu Sinir Ağları ise karar sınırları olarak ÇKA ve RTFA ağlarını bir arada kullanmaktadır. Matematiksel olarak konik kesit, koni ile düzlemin kesişiminden oluşmaktadır (Şenol ve Yıldırım, 2004).

$$y_j = \sum_{i=1}^{m} (x_i - c_{ij}) w_{ij} - \cos w_j \sqrt{\sum_{i=1}^{m} (x_i - c_{ij})^2} \tag{3.7}$$

KKFSA ağ yapısının yayılım kuralı yukarıdaki gibidir. Bu yayılım kuralı ÇKA ve RTFA ağların özelliklerini bir arada bulundurmaktadır. Bu denklemdeki w_{ij} değeri ÇKA için giriş ve saklı katman ağırlıklarını, c_{ij} ise RTFA ağındaki merkezlere karşılık gelmektedir. w_j değeri ÇKA ile RTFA arasındaki geçişi sağlayan açıdır. Görüldüğü üzere eşitliğin sol kısmı ÇKA özelliği taşımaktadır ve w_j açısı $\pi/2$ olduğu durumda fonksiyon ÇKA özelliği göstermektedir. Eşitliğin ikinci kısmı ise RTFA'nın girişleri ile merkezleri arasındaki Öklid uzaklığıdır. y Konik Kesit Fonksiyonlu Ağların aktivasyon değeridir. Aşağıdaki şekil Konik Kesit Fonksiyonlu Ağların yapısını göstermektedir.



Şekil 3.5 Konik Kesit Fonksiyonlu Ağın yapısı (Erkmen veYıldırım, 2007)

Konik Kesit Fonksiyonlu ağlarda eğitim sürecinde ağırlıklar, merkez değerleri ve açılma açısı güncellenmektedir.

3.5 Ağın Performans Fonksiyonu

Sinir ağları, giriş vektörüne karşılık bir çıktı vektörü üretmektedirler. Bu çıktı vektörünün hedef vektöründen çıkarılması ile ağın hatası bulunur. Ortalama karesel hata (MSE), 3.8'deki denklem ile hesaplanır. Q ağın çıkış elemanı sayısıdır.

$$MSE = \frac{1}{Q} \sum_{k=1}^{Q} (h(k) - a(k))^2 \tag{3.8}$$

Ortalama karesel hata, h(k) hedef vektöründen a(k) ağın çıkış değerinin çıkarılması ve karelerinin toplanmasıyla elde edilir. Ortalama karesel hata değeri ile ağın eğitim başarısı değerlendirilmekte ve eğitimin durdurulması sağlanmaktadır (Demuth ve Beale 2000). Sinir ağları performans fonksiyonu olarak toplam karesel hata (SSE) ve RMS'de kullanabilir.

3.6 Öğrenme Kuralları

3.6.1 Levenberg Marquardt Öğrenme Kuralı

Levenberg Marquardt günümüzde kullanılan en popüler öğrenme kurallarından birisidir. Bu öğrenme kuralı quasi Newton yönteminde olduğu gibi ikinci derece eğitim hızını Hessian matrisini hesaplamadan yakınsar. Bu öğrenme kuralı önemli bir sorun olan yavaş yakınsama probleminden etkilenmez. Levenberg Marquardt kuralı ağırlık vektörünü hatanın en küçük olduğu durum için bulur. Hessian matrisi ve gradyent 3.9 ve 3.10'daki denklemlerle hesaplanır (Demuth ve Beale 2000).

$$H = J^T J \tag{3.9}$$

$$g = J^T e \tag{3.10}$$

Aşağıda J, ağ hatasının birince derece türevini içeren Jakobyen matrisi, **e** ise ağ hatalarının vektörüdür. μ Marquardt parametresi, I ise birim matrisdir. Ağın ağırlıklarındaki değişim aşağıdaki şekilde hesaplanır.

$$w_{k+1} - w_k = -[J^T J + \mu I]^{-1} J^T e \tag{3.11}$$

Levenberg Marquardt öğrenme kuralı μ değeri 0 olduğunda Newton öğrenme kuralına dönüşür. μ değeri büyük bir sayı aldığında ise Gradyent Düşümü öğrenme kuralı gibi davranır. μ her başarılı adımda azaltılır. Ağın hata oranı yükselirse μ değeri arttırılır. Levenberg Marquardt öğrenme kuralının Matlab uygulamasında (trainlm) mu_dec, mu_inc ve mu_max değerleri μ'deki azalmayı, artmayı ve alabileceği maksimum değeri göstermektedir.

Levenberg Marquardt öğrenme kuralı çok hızlı bir şekilde sonuca ulaştığı halde, hesaplama için çok fazla belleğe ihtiyaç duymaktadır.

3.6.2 Ölçeklenmiş Konjugate Gradyent Öğrenme Kuralı

Ölçeklenmiş Konjugate Gradyent öğrenme kuralı Moller (1993) tarafından ortaya atılmış olup, Konjugate Gradyent öğrenme kuralındaki arama algoritmalarını kullanmaz. Bu kural Levenberg Marquardt ve Konjugate Gradyent metotlarını birleştiren bir yöntemdir. Matlab uygulamasında (trainscg) sigma (σ) ve lamda (λ) olmak üzere 2 değer almaktadır. Sigma

değeri ile ağırlıklardaki değişim miktarı bulunmaktadır. Lambda değeri ise Hessian matrisinin belirsizliğini ayarlamaktadır. Ölçeklenmiş Konjugate Gradyent kuralı ile eğitilen ağ, diğer Konjugate Gradyent yöntemlerine göre daha fazla adımda eğitildiği halde, her adımdaki işlem miktarı oldukça azdır (Demuth ve Beale 2000).

3.6.3 BFGS Öğrenme Kuralı

Quasi Newton yöntemleri arasında BFGS yöntemi en başarılı yöntemdir. Broyden Fletcher Goldfarb Shanno isimlerinin baş harfleriyle adlandırılmıştır. BFGS öğrenme kuralı Konjugate Gradyent metotlarına alternatif bir yöntemdir. BFGS öğrenme kuralı ağın ağırlıklarını

$$w_{k+1} = w_k - A_k^{-1} g_k \tag{3.12}$$

şeklinde hesaplar. A_k Hessian matrisidir. Hessian matrisinin hesaplanması ileri beslemeli ağlarda işlemsel açıdan ağır yük getirmektedir. Konjugate Gradyent öğrenme kuralına göre daha fazla sayıda işlem gereksinimi vardır. Büyük boyutlu ağlarda Konjugate Gradyent yönteminin kullanılması tercih edilir (Demuth ve Beale 2000). Matlab uygulaması trainbfg komutu kullanılarak yapılır.

3.6.4 Resilient Propagasyon Öğrenme Kuralı

Çok Katmanlı Ağlar gizli katmanlarında genelde sigmoid aktivasyon fonksiyonları kullanırlar. Sigmoid fonksiyonlarının girdileri büyüdükçe eğimleri 0'a yaklaşmaktadır. Aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanan Çok Katmanlı Ağlar Gradyent değerinin çok küçülmesi sonucunda ağırlıkları ve biaslarını yeterince güncelleyememe problemiyle karşı karşıya kalmaktadır.

Resilient Propagasyon algoritmasının amacı türevin boyutu yerine yönünü kullanarak ağırlıkları ve biasları güncellemektir. Türevin işareti kullanılarak ağırlıklar belirli bir değişim miktarı kadar değiştirilir. Bu değişim miktarı sabit olmayıp, her eğitim adımında güncellenmektedir.

Resilient Propagasyon Öğrenme kuralının Matlab uygulaması trainrp komutu ile yapılır.

Bu tezde ele alınan öğrenme kuralları dışında kalan kurallar hakkında bilgiye Demuth ve Beale (2000) kaynağından ulaşabilirsiniz.

3.7 Özellik Matrisinin Normalizasyonu

Sinir ağları, ağın girişleri ve hedef değerleri üzerinde ön işlemler yapılarak daha verimli bir hale getirilir. Eğitim öncesi giriş ve hedef değerlerinin belli bir aralıkta ölçeklenmesi genellikle ağın öğrenmesinde faydalıdır.

Bir özellik matrisinin ölçeklenmesi, sabit bir sayı ekleme/çıkarma ve daha sonra sabit bir sayıyla çarpma/bölme yoluyla yapılır. Celcius'dan Fahrenhait cinsine çevrim ölçeklemeye örnektir.

Ağ giriş ve hedef değerleri [-1,1] veya [0,1] arasına ölçeklenebilir. Ölçekleme için sık kullanılan bir başka yöntem ise giriş ve hedef değerlerini ortalama ve standart sapmasına göre normalize etmektir. Bu yöntem giriş ve hedef değerlerini ortalaması 0, standart sapması ise 1 olacak şekilde normalize eder.

[0,1] arası normalizasyon denklem 3.13 ile yapılır. [-1,1] arası normalizasyon denklem 3.14'deki gibi yapılır. x_n normalize edilmiş özellik vektörüdür. min(x) ve maks(x) değerleri ise x vektörünün en küçük ve en büyük aldığı değerlerdir. Ortalaması 0, standart sapması 1 olacak şekilde normalizasyon, x değerinden, girdilerin ortalamasının çıkarılması ve elde edilen sonucun girdilerin standart sapmasına bölünmesiyle elde edilir. x_{ort} giriş vektörünün ortalamasıdır, x_{std} ise giriş değerlerinin standart sapmasıdır. Denklem 3.15 özellik matrisinin ortalaması 0, standart sapması 1 olacak şekilde normalizasyonunu göstermektedir.

$$x_n = \frac{x - \min(x)}{maks(x) - \min(x)} \tag{3.13}$$

$$x_n = \frac{2(x - \min(x))}{maks(x) - \min(x)} - 1 \tag{3.14}$$

$$x_n = \frac{x - x_{ort}}{x_{std}} \tag{3.15}$$

Ölçekleme işleminin Matlab uygulaması, girişlerin [-1,1] aralığına getirilmesi için prenmx komutu ile yapılır. Ortalamanın 0, standart sapmanın ise 1 olması için ise prestd komutu kullanılır.

Ağın giriş değerlerinin ölçeklenmesi, Öklid uzaklığı gibi uzaklık fonksiyonları kullanılarak bir sonraki katmanın giriş değeri hesaplandığı RTFA ağlarında önemlidir. Örneğin bir giriş değeri 0 ile 1 arasında değişirken, diğer giriş değerinin 0 ile 1000000 arasında değişmesi, küçük olan giriş değerinin hesaplama sırasında önemsizleşmesine neden olacaktır. Hangi giriş

değerinin daha önemli olduğu bilinmeyen durumlarda tüm giriş değerlerini bir aralıkta normalize etmek bu sorunun çözülmesinde faydalıdır.

Çok Katmanlı Algılayıcı Ağlarda giriş değerleri doğrusal olarak birleştirilerek bir sonraki katmanın giriş değerleri elde edildiğinden giriş değerlerinin normalize edilmesi her durumda gerekli değildir. Çok Katmanlı Algılayıcı Ağ girdileri herhangi bir sayı aralığında olabilir. Bazı durumlarda bu girdilerin [0,1] aralığında ölçeklenmesi faydalıdır. Çok Katmanlı Ağlarda giriş değerlerinin normalize edilmesi farklı eğitim algoritmalarında değişik sonuçlar verir. Örneğin, bazı durumlarda normalize edilmiş giriş vektörü eğitimi hızlandırır ve yerel optima noktalarında takılma şansını azaltır.

Kolonların normalize edilmesi genelde ağın eğitilmesi ve test edilmesine faydalı sonuçlar verdiği halde, satır vektörlerinin (ağ giriş vektörlerinin) normalize edilmesi sadece özel veriler için faydalı sonuçlar verir. Kohonen Ağları gibi ağlarda satır vektörlerinin sabit bir Öklid uzaklığına normalize edilmesi ağın başarısını arttırabilir. Satır vektörlerinin normalize edilmesi verilerde bilgi kaybına neden olduğundan dikkatle yapılmalıdır. Kaybolan bilgiler önemli özellikler içerdiği takdirde ağın başarısı önemli ölçüde düşecektir.

3.8 Temel Bileşen Analizi (PCA)

Temel bileşen analizi ağın giriş sayısını azaltmak amacıyla kullanılır. Ağ giriş vektörünün bazı elemanları arasında korelasyon yüksektir. Bu durumda korelasyonu yüksek olan giriş değerleri elenerek giriş vektörü küçültülür. Bu şekilde ağın boyutu küçülür. (Şenol ve Yıldırım, 2004)

PCA tekniğinin 3 faydası vardır. Giriş değerlerini birbirine dik yaparak, aralarındaki korelasyonu engeller. Dik bileşenleri (temel bileşenleri), en büyük varyasyona sahip olan ilk gelecek şekilde sıraya sokar. Veri kümesindeki en düşük varyasyona sahip bileşenleri ise eler.

PCA'nın Matlab uygulaması prepca komutu ile yapılır. Prepca komutu uygulanmadan önce giriş vektörlerinin prestd komutu ile ortalaması sıfır ve standart sapması bir olacak şekilde normalize edilmesi gereklidir.

3.9 Eğitim Sonrası İşlemler

Eğitilmiş bir ağın performansı, hata fonksiyonunun eğitim, test ve doğrulama verilerindeki davranışına göre ölçülür. Ayrıca ağın yanıtının detaylı bir şekilde incelenmesi faydalıdır. Ağın yanıtının incelenmesi için kullanılan bir yöntem ağın çıkışları ile hedef değerler arasında regresyon analizi yapmaktır. Matlab'da regresyon analizi postreg komutuyla yapılır.

4. GÖRÜNTÜ ÜZERİNDE GÖZ VE DUDAKLARIN BULUNMASI VE ÖZELLİKLERİNİN ÇIKARILMASI

Bu tez çalışmasının amacı, yüz görüntüleri kullanılarak göz ve dudakların durumlarının değerlendirilmesi ve her yüz durumunun bir istek ile ilişkilendirilmesidir. Bu sayede özellikle iletişim olanakları sınırlı felçli veya konuşamayan hastaların temel isteklerini, basit yüz mimikleri aracılığıyla sisteme iletmesi ve isteklerin otomatik olarak işlenerek ilgili kişiye bildirilmesi hedeflenmiştir.

Dudak ve göz durumlarına göre toplam 12 yüz hareketi sisteme tanıtılmıştır. Aşağıdaki şekilde bu yüz hareketleri görülmektedir.

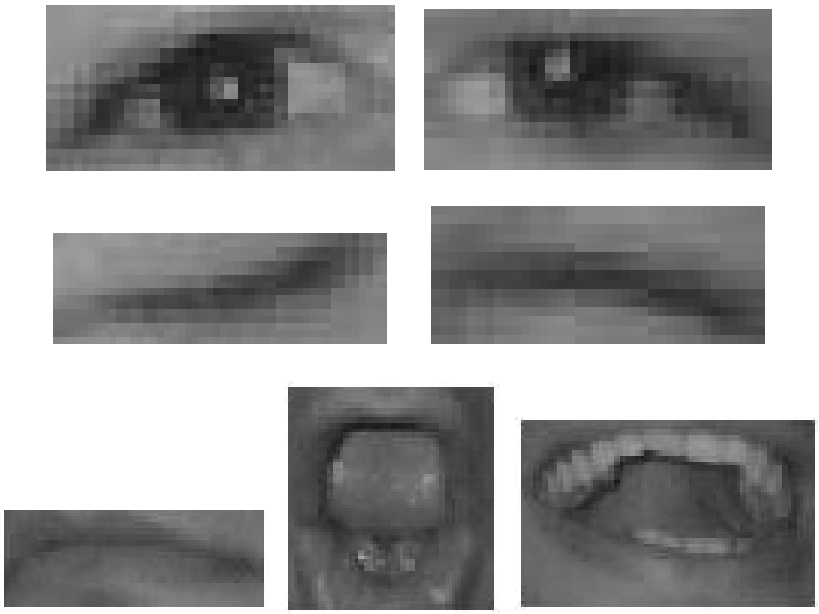
Şekil 4.1 Tanımlanan yüz hareketleri

Çalışmada 322 adet 640 * 480 çözünürlükte renkli görüntü kullanılmıştır. Kullanılan görüntüler bir kişiye aittir. Yukarıdaki şekilde görülen her yüz şekli için 24 ile 28 arasında görüntü mevcuttur. Görüntü kümesi, değişik zamanlarda çekilen fotoğraflardan oluşturulmuş, görüntüler gün ışığı, sarı ışık, beyaz ışık ve flaş gibi farklı ışık koşulları altında çekilmiştir. Görüntü kümesinde, fotoğrafı çekilen kişinin sakalsız, kirli sakallı ve sakallı durumlarda çekilmiş pozları bulunmaktadır. Görüntüler, yüz bölgesi tüm görüntünün en az % 20 sini kaplayacak şekilde 1-1.5 metre uzaklıktan çekilmiştir. Görüntüler içerisindeki kişi, karşıya veya +/-15 derece ile sağa, sola, yukarı, aşağı bakmaktadır. Kullanılan görüntülerin arka planı sabit olmayıp, farklı arka planı olan görüntüler kullanılmıştır. Çalışmanın amacı dolayısıyla, görüntülerde sadece tek yüz bulunmaktadır.

Mimik tanıma için tasarlanan bir sistem aşağıdaki özelliklere sahip olmalıdır.

1) Tanımlı bir mimik kümesi
2) Görüntü içerisinde yüzün mevcut olması ve yerinin tespit edilebilmesi
3) Yüz görüntüsünün işlenerek ağız ve gözlerin yerlerinin tespit edilmesi
4) Dudak ve göz görüntülerinden durumlarını belirleyici özelliklerin çıkarılması
5) Bulunan özelliklerin sinir ağı yardımıyla değerlendirilmesi

Bu çalışmada gözler için iki temel durum, açık ve kapalı durumları tespit edilmiş, ağız için ise kapalı, "o" şeklinde ve gülme şeklinde açık olmak üzere 3 temel durum bulunmuştur. Aşağıdaki şekilde verilen, ağız ve göz durumları kullanılarak toplam 12 yüz hareketi tespit edilebilmektedir.



Şekil 4.2 Çeşitli dudak ve göz durumları

İnsanlar gözleri çok hafif aralık şekilde dışarıyı görebildikleri halde, bilgisayarın gözlerin açık olduğunu anlayabilmesi için irisin ve göz akının görüntü üzerinde görülebilir durumda olması gerekmektedir. İrisin ve gözlerin beyaz kısımlarının görülemediği görüntülerde gözler kapalı olarak değerlendirilecektir.

Yüz hareketlerini oluşturan dudak ve göz durumları, takip eden sayfada çizelge 4.1'de gösterilmektedir. Bu çizelgedeki her yüz hareketi hasta bir kişinin ihtiyaçlarını karşılayacak bir anlam ile ilişkilendirilmiştir. Çizelge 4.1'deki anlamlar kişilerin gereksinimlerine göre özelleştirilebilir.

Çizelge 4.1 Yüz hareketlerinin anlamları

Yüz hareketi	Anlamı	Sol göz durum	Sağ göz durum	Dudaklar durum
1	Herhangi bir isteğim yok	Açık	Açık	Kapalı
2	Televizyonu aç	Açık	Kapalı	Kapalı
3	Doktoru çağır	Kapalı	Açık	Kapalı
4	Uykum var	Kapalı	Kapalı	Kapalı
5	Hafif ağrım var	Açık	Açık	O şeklinde
6	Sağ tarafım ağrıyor	Açık	Kapalı	O şeklinde
7	Sol tarafım ağrıyor	Kapalı	Açık	O şeklinde
8	Çok ağrım var	Kapalı	Kapalı	O şeklinde
9	Acıktım	Açık	Açık	Gülme şeklinde
10	Dinlenmek istiyorum	Açık	Kapalı	Gülme şeklinde
11	WC	Kapalı	Açık	Gülme şeklinde
12	Susadım	Kapalı	Kapalı	Gülme şeklinde

Mimik tanıma sisteminin ilk aşaması, görüntü üzerinde yüz bölgesinin tespit edilmesidir. Bölüm 4.1'de yüz bölgesinin bulunması anlatılmaktadır.

4.1 Yüzün Bulunması

Dudak ve göz mimiklerinin belirlenmesi için öncelikle görüntüde yüzün mevcut olması ve çerçevesinin belirlenmesi gerekir.

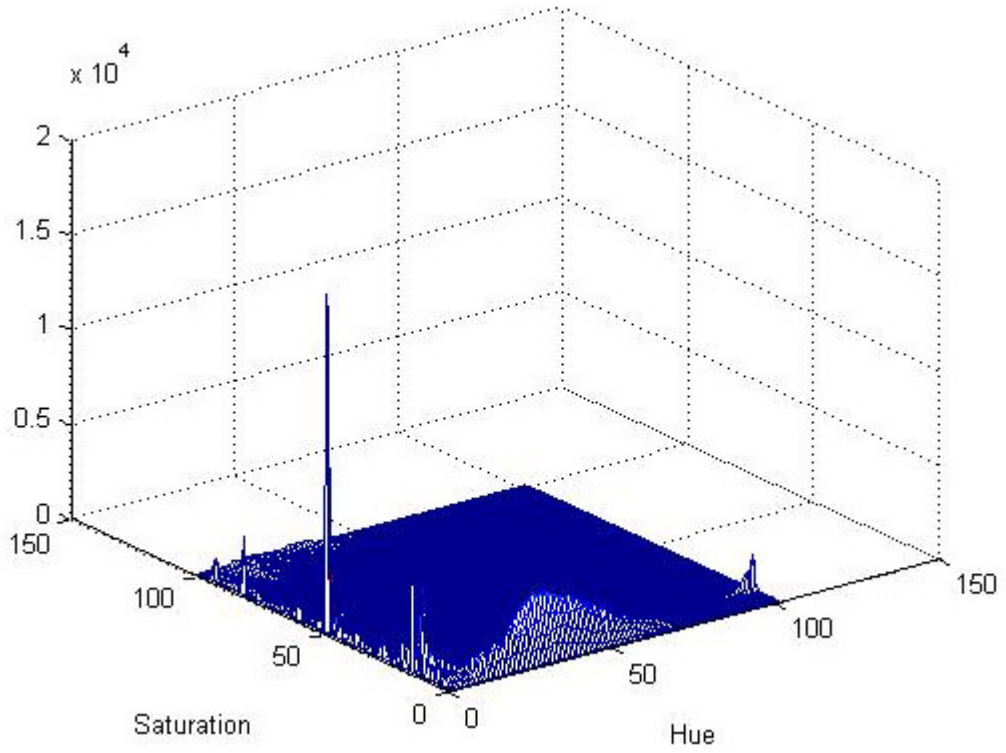
Yüz bölgesinin bulunması için çeşitli yöntemler geliştirilmiştir. Bu çalışmada yüz bölgesinin görüntü içerisinde bulunması için Mikael Nilsson (2005) tarafından yazılmış ve Lokal Successive Mean Quantization Transform (SMQT) özelliklerini ve split up Sparse Network of Winnows (SNoW) sınıflandırıcıyı temel alan findface.m isimli bir MATLAB programı çalışmaya bütünleştirilmiştir.

Şekil 4.3 Yüz bölgesinin facefind.m ile elde edilmesi (a) Orijinal görüntü (b) Yüz bölgesi

Bu program görüntü içerisindeki yüz bölgesini bulup, bir portre boyutlarında görüntü haline getirmektedir. Yukarıda programdan elde edilmiş görüntü görülmektedir. Findface.m programının işlevini, görüntünün büyük kısmını (en az %60) yüzün oluşturduğu fotoğraflar kullanarak da yapabiliriz. Şekil 4.3'de görülen yüz bölgesi sağ, sol, alt ve üst kısımlarında boşluklar içermektedir. Ayrıca, saç ve omuz gibi göz ve dudakları bulmamızda faydası olmayan öğelerde görüntüde bulunmaktadır. Görüntüyü boşluklardan ve bu öğelerden arındırmamız göz ve dudak yerlerini tespit etmekteki başarı oranını arttırmaktadır. Bir sonraki bölümde şekil 4.3'teki görüntü kullanılarak sadece yüz bölgesinin bulunmasını sağlayan yöntem anlatılmıştır.

4.2 Hassas Yüz Belirleme

Bu çalışmada yüz görüntüsü ağız, burun ve gözleri içeren görüntü olarak kabul edilmektedir. İnsan deri renginin, renk uzayında küçük bir alanda toplandığı bilgisi kullanılarak bir görüntüdeki piksellerin deri veya deri pikseli değil şeklinde sınıflandırılması mümkündür. Geçmişte yapılmış çalışmalarda deri bölgelerini deri rengine göre bulma yönteminde HSI , normalize edilmiş RGB , YCbCr ve CIELAB renk uzayları kullanılmıştır.



Şekil 4.4 Deri renginin HSI renk uzayında kapladığı alan

Şekil 4.4, 730.000 deri rengi pikselden elde edilmiş bir renk histogramında, deri renginin tüm HSI renk modeli içerisinde kapladığı yeri göstermektedir.

Zarit ve arkadaşları (1999) yaptıkları çalışmalarda HSI renk uzayının RGB ve YCbCr renk uzaylarına göre deri piksellerini belirlemede daha başarılı sonuçlar verdiğini bulmuşlardır.

Şekil 4.3'de görülen RGB görüntü, HSI renk uzayına dönüştürülür. Çalışmada bu işlem Matlab'daki rgb2hsv komutu ile yapılmıştır. HSI renk uzayına dönüştürülmüş görüntüde deri piksellerinin belirlenmesi için görüntünün renk özü bileşenini 120 ile 180 arası, doygunluk bileşeni ise 10 ile 80 arası alınarak bu aralıktaki pikseller deri pikseli olarak işaretlenmiştir.

Yukarıda verilen aralıktaki değerler deri rengini yansıtmaktadır ve denemeler sonucu bulunmuştur. Bu yöntemle elde edilen ikili görüntüde deri pikselleri beyaz renk olarak kodlanmıştır. Portre görüntüden elde edilen deri bölgelerini içeren ikili görüntü, gürültü nedeniyle tam olarak yüz bölgesini ifade edememektedir. Yüz bölgesinin sınırlarını örnekleme yoluyla daha kesin bir şekilde elde edebiliriz. İkili deri görüntüsünün her bir kenarı 5 eşit parçaya bölünür ve her parçadan görüntünün iç kısmına doğru ilerlenerek deri pikseliyle karşılaşılıp karşılaşılmadığı sorgulanır. Deri pikseliyle karşılaşılması durumunda, 8 komşuluğunda başka deri pikseli olup olmadığına bakılır. Eğer başka deri pikselleri de mevcutsa bu pikselin koordinatları kaydedilir. Başka deri pikselleri yoksa görüntünün içine doğru örneklemeye devam edilir. Görüntünün eninin veya boyunun yarısına ulaşılmış ve uygun bir deri pikseliyle karşılaşılmamışsa bu örnekleme iptal edilir. Görüntünün bir kenarı boyunca yapılan 4'er örnekleme sonucunda elde edilen koordinatların ortalaması alınır. Bu işlem tüm kenarlar için tekrarlanır. Elde edilen aralık şekil 4.5 (a)'deki yüz görüntüsünden çıkartılır. Aşağıdaki şekilde örneklemenin yapılışı ve elde edilen yüz görüntüsü görülmektedir.

Şekil 4.5 Yüzün elde edilmesi (a) Yüzü içeren görüntü (b) Deri haritasının bulunması (c) Bulunan yüz

Şekil 4.5 (c) görüntüsü elde edilen yüz bölgesini göstermektedir. Bulunan bu bölgenin yüze ait olduğunu kontrol etmek amacıyla yükseklik/genişlik oranına bakılır. Bu oran, altın oran

denilen (1+√5)/2'ye belli bir tolerans civarında yakın ise bu bölge yüz bölgesidir. Söz konusu tolerans deneme yanılma yöntemiyle seçilmiştir. Bu çalışmada tolerans 0.5 olarak alınmıştır.

Elde edilen yüz görüntüsünün boyutu orijinal görüntüdeki yüz bölgesinin kapladığı alana göre değişmektedir. Yüzün standart bir boyuta getirilmesi bundan sonraki işlemler için önemlidir. Yüzün en ve boy olarak belli bir boyuta getirilmesi görüntüde şekil kaybına neden olabilir. Bu nedenle sadece yüzün eni 200 piksel olarak genişletilmiş/daraltılmış, yüzün boyu ise (200/yüzün ilk genişliği) katsayısıyla çarpılarak boyu uzatılıp/kısaltılmıştır. Normalize edilmiş yüz görüntüsü üzerinde göz ve dudakların bulunması daha kolay olacaktır.

4.3 Yüz Görüntüsü Üzerinde Alın Bölgesinin ve Kaşların Çıkarılması

Alın bölgesi yüz görüntüsünün yaklaşık üçte birlik kısmını kaplamaktadır. Alın kısmının görüntüden kesilmesi göz ve dudakların bulunması için yapılacak işlemlerde önemli bir azalma sağlar ve güvenilirliği arttırır.

Kaşlar, şekil itibariyle kapalı göz durumuna çok benzer bir yapıya sahip olduklarından, göz merkezlerinin bulunması sırasında hatalı sonuçlara neden olma ihtimalleri yüksektir.

Alın ve kaş bölgelerinin yüz görüntüsü üzerinde tespiti için Otsu algoritması kullanılmıştır. Otsu algoritması görüntüleri dinamik eşikleme ile ikili görüntüye çevirir. Çoğu durumda, görüntü farklı parlaklık ve ışık nedeniyle sabit eşikleme ile tüm özellikleri korunarak siyah beyaz görüntüye döndürülemez. Dinamik olarak seçilen bir eşik değeri görüntülerin ikili görüntüye çevrilmesi sırasında daha iyi sonuç vermektedir (Yavuz, 2003).

Otsu algoritması, görüntüdeki piksel değerlerinin dağılımlarına göre bu piksellerin kümelendirilmesini sağlamaktadır. Görüntüdeki piksel değerleri M gri seviye ile temsil edilirlerse, i. seviyedeki piksel sayısı n_i olduğunda, toplam piksel sayısı aşağıdaki gibidir.

$$N = n_1 + n_2 + ... + n_i + ... + n_M \tag{4.1}$$

t değerinde sabit bir eşik değeri ile pikseller iki seviyeye ayrılırsa [1,2,...,t] ve [t+1,...M] gri seviye değerlerini içeren C_1 ve C_2 gibi iki sınıfa ayrılırlar.

$$p_i = n_i/N \tag{4.2}$$

olarak ifade edildiğinde, C_i kümeleri aşağıdaki bağıntılarla ifade edilir.

$$C_1 : p_1/w_1(t), p_2/w_1(t) .... p_M/w_1(t) \quad , \quad C_2 : p_{t+1}/w_2(t), p_{t+2}/w_2(t) .... p_M/w_2(t) \tag{4.3}$$

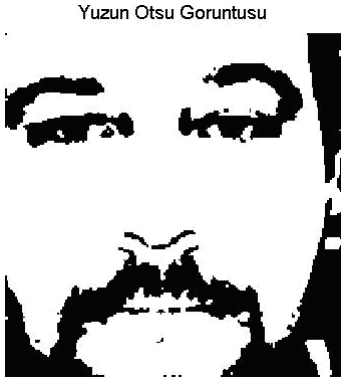
$$w_1(t) = \sum_{i=1}^{t}(pi) \quad , \quad w_2(t) = \sum_{i=t+1}^{M}(pi) \tag{4.4}$$

Otsu algoritmasının kullanım amacı, C_1 ve C_2 sınıfları arası dağılımı maksimum yapacak eşik değerini belirlemektir.

$$\sigma_B^2 = w_1 * w_2 * (\mu_2 - \mu_1)^2 \quad (4.5)$$

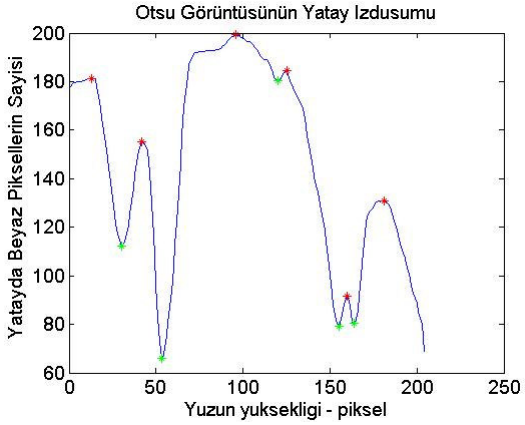
$$\mu_1 = \sum_{i=1}^{t} ip_i / w_1(t) \quad , \quad \mu_2 = \sum_{i=t+1}^{M} ip / w_2(t) \quad (4.6)$$

σ değeri ile sınıflar arası dağılımı maksimum yapacak eşik değeri belirlenir. σ 'yı maksimum yapan t değeri eşik değeri olarak alınır. Aşağıdaki şekil Otsu algoritmasının yüz görüntüsüne uygulanması sonucu elde edilmiştir. Bu şekil için bulunan dinamik eşik değeri 68'dır.



Şekil 4.6 Yüz görüntüsüne Otsu algoritmasının uygulanması

Yukarıdaki şekil incelendiğinde, alın bölgesi genel olarak beyaz renkte izlenmekte ve sadece bazı saç bölgelerinin alın üzerinde siyah olarak yer aldığı görülmektedir. Kaşlar görüntünün yukarısından aşağı doğru inildiğinde yatay doğrultuda en uzun bölgedir. Yatay olarak görüntünün izdüşümü alındığında, ilk global minimum kaşların bulunduğu hattın üzerindedir.



Şekil 4.7 Otsu görüntüsünün yatay izdüşümü

Şekil 4.7'de ilk global minimum kaşların bulunduğu noktayı vermektedir. Bu nokta görüntünün yukarıdan 35. pikseline karşılık gelmektedir. Kaş bölgesi yukarıdaki grafikteki

ikinci global maksimum noktasında son bulmaktadır. Bu nokta görüntünün yukarıdan 45. pikselidir. Yüz görüntüsünde 45. pikselden yukarısı kesildiğinde aşağıdaki şekil elde edilir. Aşağıdaki şekilde görüldüğü üzere kaş bölgesi ve alın bu işlem sonucunda büyük oranda görüntüden çıkmıştır.

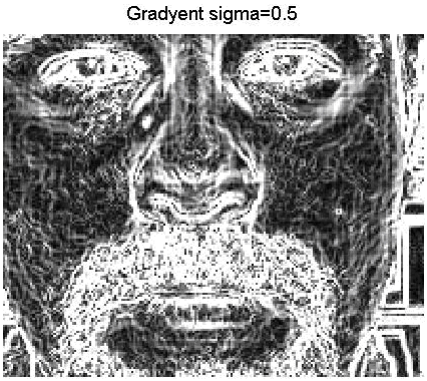
Şekil 4.8 Alın ve kaş bölgesi çıkarıldıktan sonra yüz bölgesi

Şekil 4.7'deki izdüşüm grafiğinden göz bölgesinin 55. pikseldeki global minimum noktasında olduğu ve 45. piksel ile 100. piksellerdeki global maksimum noktaları arasında kaldığını tahmin edebiliriz. Bu bilgi gözlerin yerlerinin tespit edilmesinde kaşların bulunması kadar güvenilir değildir. Bu nedenle takip eden bölümde anlatılan yöntem ile gözlerin bulunması daha sağlıklı olur.

4.4 Gözlerin ve Dudakların Yerlerinin Bulunması

4.4.1 Göz Bölgelerinin Bulunması

Yüz bölgesindeki değişimlerin bulunması için görüntünün gradyenti alınır. σ=0.5 değeri ile alınan yüz görüntüsünün gradyenti şekil 4.9'da görülmektedir.

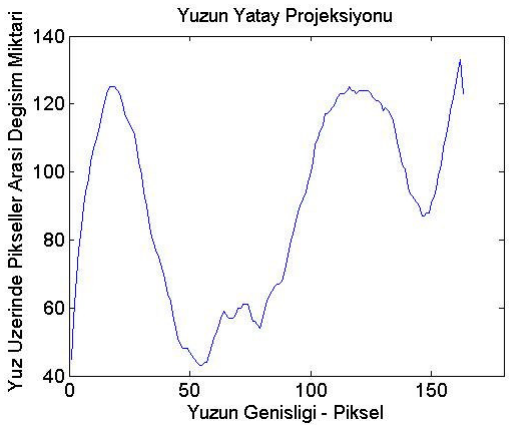


Şekil 4.9 Yüz bölgesine Gradyent uygulanması σ=0.5

Değişim, yüzün sınırlarının, gözlerin, burun, kaş ve ağız gibi yüz özelliklerinin olduğu yerlerde yoğundur. Bu bilgi kullanılarak yüz üzerinde gözlerin yeri tespit edilir. Gözlerin bulunduğu bölge yatay doğrultuda en yüksek değişimin olduğu yerdir (Tian ve Qin, 2005). Yatay doğrultudaki değişimin grafiğini Gradyent görüntüsüne yatay izdüşüm uygulayarak bulabiliriz. Yatay izdüşüm aşağıdaki denklem ile uygulanır.

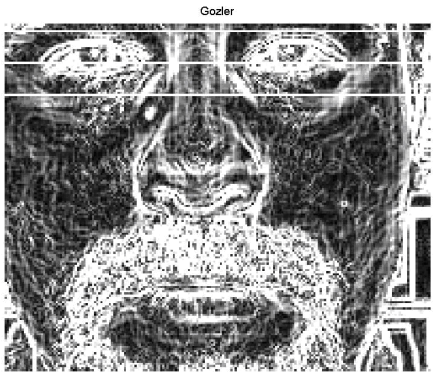
$$\Pr ojeksiyon(y) = \sum_{x=1}^{n} I(x, y) \tag{4.7}$$

y düzleminde pikseller sabit kalırken, her x pikseli için I(x,y) Gradyent görüntüsünün değeri toplanır. Oluşturulan grafik, düzleştirme işlemi ile yerel tepe ve minimum noktalarından ayrıştırılır. Aşağıdaki şekilde gradyent görüntünün yatay izdüşümü görülmektedir. Gözler, ilk global tepe noktasından önceki ve sonraki global minimum noktaları arasında kalan bölgededir.



Şekil 4.10 Yüz görüntüsünün yatay izdüşüm grafiği

Yukarıda bulunan göz aralığı şekil 4.11'de Gradyent görüntü üzerinde çizilmiştir. İlk beyaz çizgi ve üçüncü beyaz çizgi yukarıdaki grafikteki 1. ve 2. minimum noktalarını temsil eder. 2. beyaz çizgi ise ilk global tepe noktasıdır.



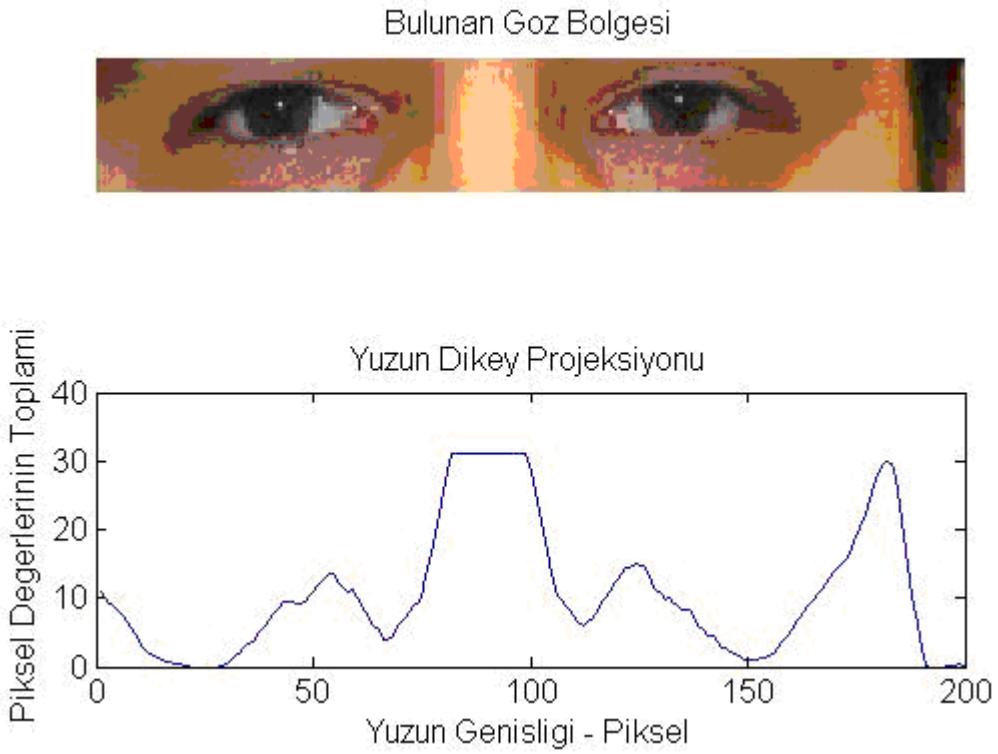
Şekil 4.11 Gözlerin yerlerinin işaretlenmesi

Gözlerin bulunduğu bölge yatay eksende bulunduktan sonra, iki gözü birbirinden ayırarak sağ ve sol göz görüntülerini elde etmek gereklidir. Yatay aralıktaki göz görüntüsünü tam ortadan bölmek, yüzün sağa veya sola dönük olduğu durumlarda başarılı sonuçlar vermeyebilir. Bu nedenle, yüzün dikey ortasını bulmak ve gözleri bu dikey ortadan ayırmak gerekir.

Yüzü dikey merkezden kesen doğruyu bulmak için şekil 4.11'deki gözlerin bulunduğu yatay aralığı kullanmak yeterlidir. Bu yatay aralık, renkli görüntüden kesilerek, elde edilen görüntü gri seviye görüntüye dönüştürülür ve bu görüntünün dikey izdüşümü alınır. Dikey izdüşümü aşağıdaki denklem kullanılarak yapılmaktadır.

$$\Pr ojeksiyon(x) = \sum_{y=1}^{n} I(x, y) \tag{4.8}$$

Dikey izdüşümü sonucu elde edilen grafikteki en yüksek tepe noktası keskinlik değerinin en yüksek olduğu yeri vermektedir. En yüksek keskinlik değeri, alın ve burnun üstünden geçen ve yüzü dik olarak ikiye bölen çizgi üzerindedir. Şekil 4.12 (a) göz bölgesini ve 4.12 (b) bu bölgenin dikey izdüşümünü göstermektedir.



Şekil 4.12 Yüzün dikey ortasının bulunması (a) Bulunan göz bölgesi (b) Göz bölgesinin dikey izdüşümü

Yukarıdaki şekilden elde edilmiş değerler kullanılarak sağ ve sol gözü içeren alanlar bulunur.

Şekil 4.13 Gözlerin birbirinden ayrı şekilde elde edilmesi

Bir sonraki bölümde, yukarıda bulunan sol ve sağ göz bölgelerinde şablon eşleştirme uygulaması yapılarak gözlerin merkezlerinin koordinatları bulunacaktır.

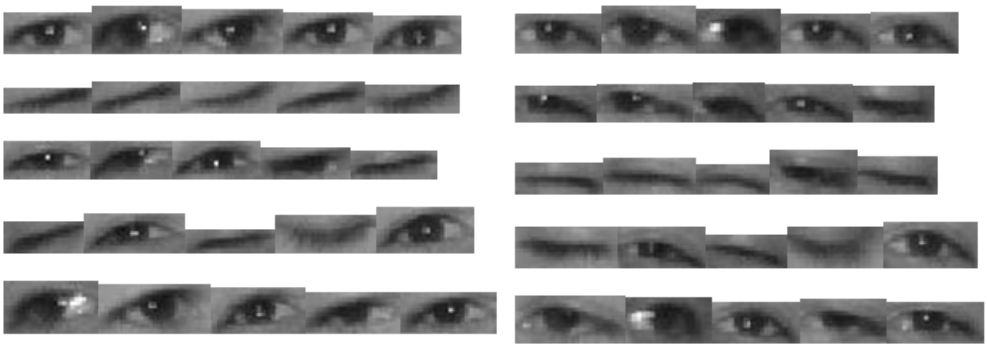
4.4.2 Göz Merkezlerinin Bulunması

Şekil 4.13'deki göz bölgeleri, göz dışında deri bölgeleri de içermektedir. Göz kısmını bölgenin geri kalanından ayırt edebilmek için şablon eşleştirme metodu kullanılır.

Bu çalışmada bir şablon kümesi oluşturulmuştur. Bu şablonlar açık ve kapalı göz görüntülerinden oluşmaktadır. Şablon eşleştirme yapılması için göz şablonu da, gözü içeren bölgede gri seviye görüntüye dönüştürülür. Göz bölgesiyle şablonlar arasındaki normalize edilmiş korelasyon denklem 4.9 ile hesaplanır.

$$g'(x,y)=\frac{\sum_{k=-n}^{n}\sum_{j=-m}^{m}h(j,k)f(x+j,y+k)}{\sum_{k=-n}^{n}\sum_{j=-m}^{m}f(x+j,y+k)} \tag{4.9}$$

Çalışmada sağ ve sol göz için 25'er şablon hazırlanmıştır. Oluşturulan şablon kümesi aşağıda görülmektedir.



Şekil 4.14 Çalışmada kullanılan sağ ve sol göz şablonları

Şablon görüntü üzerinde bir alçak geçiren süzgeç gibi davranmaktadır. Eşleşmenin en yüksek olduğu nokta en yüksek korelasyon değerini alır. Şablonlar, bir döngü içerisinde göz bölgelerine uygulandıktan sonra, her şablonun maksimum korelasyon değerleri karşılaştırılır. En yüksek ilk 3 korelasyon değerine sahip şablonların maksimum korelasyona sahip oldukları noktaların görüntü üzerindeki koordinatları bulunur.

Eğer maksimum normalize korelasyon değerlerinden bir tanesi 0.9'un üzerindeyse, şablonun görüntüye oldukça uygun eşleştiği görülür. Bu değere ait koordinatlar göz koordinatı olarak işaretlenir.

Maksimum normalize korelasyon değerlerinin hepsi 0.9'dan düşük ise, en yüksek korelasyon değeri ile takip eden en yüksek 2. korelasyon değeri arasındaki farkın 0.1'de büyük olup olmaması durumu kontrol edilir. Eğer 0.1'den fazla bir fark var ise, en yüksek korelasyon değerinin koordinatları göz koordinatı olarak işaretlenir. Halen göz koordinatı bulunmamışsa, her 3 korelasyon değerinin koordinatları birbiriyle karşılaştırılır. Birbirine +/-2 yakınlıktaki koordinatların ortalama değeri alınır. Bu koordinatlar göz koordinatı olarak işaretlenir.

Yapılan denemeler sonucunda, yüz genişliği 200 piksel olarak normalize edilmiş görüntülerde göz boyutunun 20-40 piksel arası değişen uzunlukta ve 5-15 piksel arası değişen genişlikte olduğu görülmüştür. Bu nedenle göz boyutu maksimum 20 x 50 piksel olarak kabul edilmiş ve elde edilmiş göz merkezini merkez kabul eden bir dikdörtgen çizilerek göz işaretlenmiştir.

Şekil 4.15 Gözlerin yerlerinin işaretlenmesi

4.4.3 Dudakların Yerinin Bulunması

4.4.3.1 Alt Dudağın Bulunması

Ağız, iki gözün konum itibariyle arasındadır. Gözleri birleştiren doğruyu dik kesen doğru, burun ve ağız üzerinden geçmektedir. Dudakların bulunması için yüz üzerindeki simetri

özellikleri kullanılır. Yüz görüntüsünü yukarı ve aşağı olarak ikiye böldüğümüzde ağız alt bölgede kalmaktadır. Dudaklar renk olarak yüzün geri kalanından farklı bir renge sahiptir (Eveno vd., 2001). Dudak bölgesini renk bilgisi ile bulabilmek için RGB renk uzayından YCbCr renk uzayına çevirmek gereklidir. Dudakları YCbCr renk uzayı kullanarak bulurken alt dudak baskın bir biçimde görülür.

Görüntüdeki yüksek Cr değerleri dudaklarda bulunur. Krominans değeri dudaklarda Cb değerinden yüksektir. Ayrıca dudaklar Cr/Cb değerinde düşük yanıta, Cr^2 değerinde ise yüksek yanıta sahiptir (Hsu vd., 2002). Bu özellikteki pikseller ile bir ağız(x,y) görüntüsü oluşturulur.

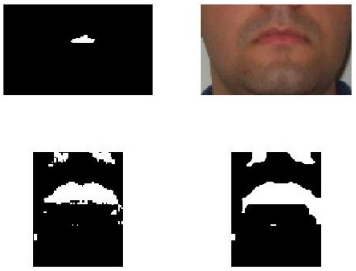
Oluşturulan ikili ağız(x,y) görüntüsü alt dudağı kapsamaktadır. İkili ağız görüntüsünde medyan filtreleme ve kapatma (close) morfolojik işlemi uygulanır. Kapatma işleminde dudağın şekil bozulmasını en az seviyede bırakmak için disk şeklinde yapılama elemanı kullanılır. Elde edilen yeni ağız(x,y) görüntüsüne yatay izdüşüm uygulanır ve dudağın alt sınırı bulunur (Şekil 4.16).

4.4.3.2 Üst Dudağın Bulunması

Üst dudağın sınırlarının bulunması için tek başına önceki bölümde anlatılan yöntem yeterli değildir. Bu nedenle ağız bölgesinin daha kapsamlı görüntüsünü elde etmek gerekir. RGB renk uzayındaki görüntüde, deri rengi olan pikseller arasında R>G>B bağıntısı vardır. Dudak rengi piksellerde ise R>G~B şeklindedir ve 4.10'deki bağıntı mevcuttur (Yavuz, 2003).

$$\text{esik} = R - (B + G) \tag{4.10}$$

Eşik değeri 0'dan küçük ise deri pikseli, 0'dan büyük ise dudak pikseli olarak yorumlanır. Bu işlem sonucunda Şekil 4.16 (c) elde edilir. Bu şekil gürültü içermektedir. Medyan filtreleme ve kapatma işlemi uygulanarak aradaki boşluklar kapatılır ve ağzın sadece dış görüntüsü bulunur [9]. Üst dudak sınırı Şekil 4.16 (d)'ye yatay izdüşümü uygulanmasıyla elde edilir.



Şekil 4.16 Dudak bölgesinin bulunması için yapılan dönüşümler (a) YCbCr renk modeli ile dudakların bulunması (b) ağız bölgesi (c) RGB renk uzayı ile dudakların bulunması (d) medyan süzgeç ve kapatma işlemi uygulanması

4.4.3.3 Ağzın sol ve sağ sınırlarının belirlenmesi ve ağız bölgesinin işaretlenmesi

Ağzın alt ve üst sınırları hesaplandıktan sonra, bu sınırlar arasında kalan aralıkta şekil 4.16 (d)'ye dikey izdüşümü kullanılarak sol ve sağ sınırlar elde edilir. İzdüşümlerden elde edilen koordinatlar kullanılarak ağız bölgesi dikdörtgen içerisine alınır.

4.5 Göz ve dudaklara ait belirleyici özelliklerin çıkarılması

4.5.1 Gözlere ait özelliklerin çıkarılması

Görüntülerde bulunan gözler açık veya kapalı durumdadır. Gözlerin özelliklerinin belirlenmesi için bu çalışmada aşağıda anlatılan 2 çözüm önerisi birlikte uygulanarak özellikler elde edilmiştir.

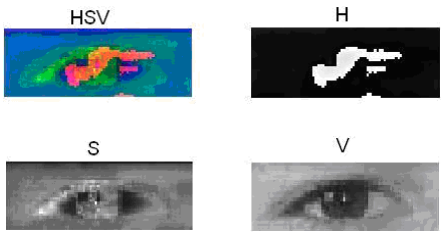
4.5.1.1 Renk Bilgisi Kullanılarak Göz Özelliklerinin Çıkarılması

Renk bilgisi insanların görüntü algılamasında önemli bir yer tutmaktadır. RGB renk uzayı gözlerin durumunu belirlemek için yeterli özellik sağlayamadığından dolayı görüntü HSI (bazı kaynaklarda HSV) renk uzayına dönüştürülür.

Deri pikselleri gözün geri kalan kısmından daha yüksek bir doygunluk (S) değerine sahiptir. Doygunluk bileşeninde belirlenen eşiğin altında kalan piksellere siyah, üstünde olan piksellere beyaz kodlanır. Bu çalışmada eşik değeri olarak 40 alınmıştır.

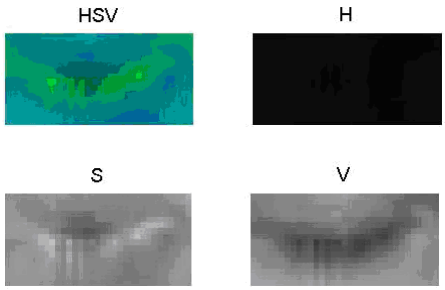
Farklı göz durumları için eşik değerinden küçük olan piksel sayısı değişiklik gösterir. Göz daha geniş açıldığı durumlarda daha fazla sayıda piksel eşik değerinden küçük doygunluk değerleri alacaktır.

Deri pikselleri gözün geri kalan kısmından daha düşük bir renk özü (H) değerine sahiptir. Aşağıdaki şekil incelendiğinde, HSV görüntünün renk özü (H) bileşeninde, göz bölgesinin deri bölgesine göre daha yüksek bir renk özü değerine sahip olduğu görülmektedir. Aynı şekilde doygunluk (S) bileşeninde göz bölgesi deri bölgesine göre daha düşük bir doygunluğa sahiptir. Özellikle gözün beyaz kısımları siyah olarak görülmektedir (Chau ve Betke, 2005).



Şekil 4.17 Açık göz görüntüsünün HSV, H, S ve V bileşenleri

Yukarıdaki şekil parlaklık (V) bileşeninden belli olduğu üzere açık göze ait bir görüntüdür. Aynı yöntemin kapalı göze uygulanarak H S V bileşenlerine ayrılması aşağıdaki şekilde görülmektedir.



Şekil 4.18 Kapalı göz görüntüsünün HSV, H, S ve V bileşenleri

4.5.1.2 Şablon Korelasyon Değeri Kullanılarak

Gözlerin durumlarını bulmaya yardımcı olacak bir başka özellik ise bulunan göz bölgesine açık göz şablonu uygulanması sonucu elde edilen korelasyon değerini kullanmaktır. Bölüm 4.4.2'de gözlerin merkezlerinin bulunması sırasında, 13 açık göz şablonu sırasıyla göz bölgesine uygulanmıştır. Açık şablonlardan göz ile en yüksek benzerliği sağlayan şablonun korelasyon değeri hesaplanır. Kapalı olan göz görüntüsünün açık göz şablonu ile benzerliği azdır. Açık olan göz görüntüsünün ise benzerliği daha fazladır.

Kapalı göz görüntüsüyle açık göz şablonu eşleştirildiğinde genelde daha küçük bir korelasyon değeri elde edilmektedir (Chau ve Betke, 2005).

Örnek göz görüntülerinden elde edilen özellikler aşağıdaki çizelgede görülmektedir. Çizelge 4.2'nin 1 ve 2. kolonları gözlerin durumlarını, 3, 4 ve 5. kolonları sol göze ait özellikleri, 6, 7 ve 8. kolonları ise sağ göze ait özellikleri vermektedir.

Çizelge 4.2 Sol ve sağ göz için çıkarılmış örnek özellik kümesi

Göz durumları		Sol göz özellikleri			Sağ göz özellikleri		
Sol Göz	Sağ Göz	S	H	Korelasyon	S	H	Korelasyon
Kapalı	Kapalı	0	0	0.64	1	0	0.71
Kapalı	Açık	7	0	0.69	63	0	0.80
Açık	Kapalı	12	2	0.79	0	0	0.72
Açık	Açık	178	57	0.80	54	6	0.75

4.5.2 Dudak Bölgesinin Özelliklerinin Çıkarılması

Aşağıdaki özellikler kullanılarak dudak şekli tanımlanır (Pong vd., 2004).

• Dudağın dış çerçevesi

• Alt dudağın ve üst dudağın bulundukları koordinatlar

• Dudak bölgesinin eni

• Dudak bölgesinin yüksekliği

• Dişlerin görüntüde görülüp görülmemesi

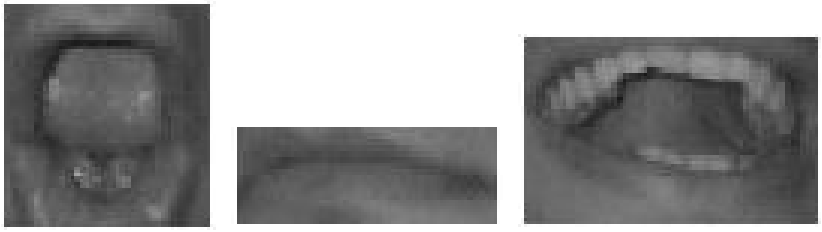
• Ağız açıklığının toplam alanı

• Dudak şeklinin merkezi

Üst ve alt dudağın en üst ve en alt kısımlarının birbirine olan uzaklığının, ağzın sağ ve sol köşelerinin birbirine olan uzaklığına bölümünü kullanarak çeşitli ağız şekillerini tanımlayabiliriz. Açık bir ağız yatay izdüşümde iki tane minimum değeri verir. Kapalı bir ağız ise bir tane minimum değeri verir. Aynı şekilde dikey izdüşümde açık ağız iki tane minimum değeri verirken, kapalı ağız bir tane minimum değeri vermektedir. Yatayda ve dikeyde minimum değer sayısı ağzın açık veya kapalı durumda olmasını belirlememize yardımcı olmaktadır.

Dudak genişliği sağ ağız koordinatından sol ağız koordinatının çıkarılmasıyla elde edilir. Dudak yüksekliği ise alt dudağın bulunduğu yerden üst dudağın bulunduğu koordinatın çıkarılmasıyla bulunur.

Dudakların yükseklik ve genişlik oranı, ağzın o anki şekliyle ilgili olarak tahminde bulunmamıza yardımcı olur.



A B C

Şekil 4.19 Bulunan çeşitli dudak hareketleri

Çizelge 4.3 Dudaklar için çıkarılmış örnek özellik kümesi

Dudak Şekli	Yükseklik	En	Yükseklik/En	Alan
A	74	67	1.1045	4958
B	26	85	0.3059	2210
C	56	97	0.5773	5432

Çizelge 4.3'de üç değişik şekilde açılmış ağız görüntülerinin yükseklik/genişlik oranları, yükseklik ve genişlikleri, kapladıkları alanlar piksel cinsinden verilmiştir.

Göz ve dudaklara ait bulunan özelliklerden bir özellik matrisi oluşturulmuştur. Bu özellik matrisi kullanılarak farklı sinir ağlarının eğitilmesi ve test edilmesi bir sonraki bölümde ele alınacaktır.

5. AĞ ÜZERİNDE YAPILAN DENEMELER

Bir önceki bölümde elde edilen göz ve dudaklara ait özellik matrisleri birleştirilerek göz ve dudak özelliklerini içeren tek bir matris oluşturulmuştur. Bu matris yapı itibariyle, 1'den 6'ya kadar olan kolonları göz özelliklerini, 7'den 10'a kadar olan kolonları ise dudak özelliklerini içerecek biçimde yapılandırılmıştır. Bu çalışmada oluşturulan özellik matrisinin elemanları 322 görüntüden elde edilmiş olup, 322 x 10 elemanlıdır. Oluşturulan matrisin bir bölümü aşağıda verilmiştir. Çizelge 5.1'deki örnek matristeki her özellik vektörü bir yüz hareketine karşılık gelmektedir. Özellik matrisinin tümüne çalışmanın ekler bölümünde ulaşabilirsiniz.

Çizelge 5.1 Her yüz şekli için elde edilmiş örnek özellikler. En üst özelik satırı çizelge 4.1'deki 1. yüz hareketine, en alt özellik satırı ise 12. yüz hareketine karşılık gelmektedir.

Sol Göz Özellikleri			Sağ Göz Özellikleri			Dudak Özellikleri			
H	S	Kor.	H	S	Kor.	Dudak yüksekliği	Dudak eni	Yükseklik/en	Dudak alanı
2	0	0,65813	0	0	0,66138	27	102	0.26471	2754
134	26	0,76899	5	0	0,63376	27	105	0,25714	2835
4	0	0,69972	136	0	0,77842	29	121	0,23967	3509
111	51	0,77725	81	21	0,72716	29	89	0,32584	2581
210	40	0,79528	212	23	0,8249	74	80	0,925	5920
79	13	0,79437	0	0	0,70696	83	62	1,3387	5146
0	0	0,70757	49	17	0,75251	80	69	1,1594	5520
0	0	0,70948	0	0	0,70505	78	82	0,95122	6396
116	29	0,7914	138	11	0,77334	56	93	0,60215	5208
69	12	0,70286	0	0	0,63706	74	107	0,69159	7918
4	0	0,63227	137	6	0,79327	61	99	0,61616	6039
0	2	0,67125	2	2	0,60594	57	102	0,55882	5814

Her yüz ifadesi bir hedef vektörüyle eşleştirilmiştir. Hedef vektörü 4 elemanlı olup, ilk iki elemanı gözlerin durumunu, kalan iki elemanı ise dudakların durumunu gösterir. Hedef vektöründe gözlere durumlarına göre 0,1 veya 0,9 değeri verilir. 0,1 kapalı gözü, 0,9 ise açık gözü ifade etmektedir. Sol ve sağ göz için birer olmak üzere toplam iki değer hedef vektöründe gözlerin durumunu gösterir.

Dudakların durumu, kapalı ve açık olmak üzere öncelikle ikiye ayrılır. Hedef vektörünün 3. değeri 0,1 olduğu zaman dudakların kapalı olduğunu, 0,9 olduğu zaman ise dudakların açık olduğunu belirtmektedir. Hedef vektörünün 4. elemanı ise dudak şeklini belirtir. Dudakların kapalı olduğu zaman 4. eleman önemsizdir. Dudakların açık olduğu zamanda ise 0,1 değeri

dudakların gülme şeklinde açık olduğunu, 0,9 değeri ise dudakların "o" şeklinde açık olduğunu gösterir. Aşağıdaki çizelge yüz ifadelerine karşılık gelen hedef vektörlerini göstermektedir. Örneğin 1. yüz ifadesi, kişinin herhangi bir isteğinin olmadığını, 2. yüz ifadesi ise "televizyonu aç" isteğini belirtmektedir. Diğer ifadelerin karşılıkları çizelge 4.1'de bulunmaktadır.

Çizelge 5.2 Her yüz ifadesi için hazırlanmış hedef vektörü. Her hedef vektörü çizelge 4.1'deki yüz hareketleri ile eşleştirilmiştir.

	Hedef Vektörü			
Yüz ifadesi karşılıkları	Sol göz	Sağ göz	Dudak Açık/Kapalı	Dudak Açıklık Şekli
1	0,9	0,9	0,1	0,1
2	0,9	0,1	0,1	0,1
3	0,1	0,9	0,1	0,1
4	0,1	0,1	0,1	0,1
5	0,9	0,9	0,9	0,9
6	0,9	0,1	0,9	0,9
7	0,1	0,9	0,9	0,9
8	0,1	0,1	0,9	0,9
9	0,9	0,9	0,9	0,1
10	0,9	0,1	0,9	0,1
11	0,1	0,9	0,9	0,1
12	0,1	0,1	0,9	0,1

Ağa uygulanan özellik vektörü 10 elemanlı olduğundan, özellik vektörü üzerinde temel bileşen analizi uygulayarak özellik sayısını azaltmaya gerek yoktur.

5.1 Özellik Matrisinden Eğitim ve Test Kümelerinin Oluşturulması

322 x 10 elemanlı özellik matrisi 2/3 eğitim kümesi, 1/3 test kümesi olma esasına göre eğitim ve test kümelerine ayrılmıştır. Eğitim kümesi 211 x 10 boyutunda, test kümesi ise 111 x 10 boyutundadır. Eğitim kümesini oluştururken, her yüz ifadesinden birbirine yaklaşık sayıda olmasına dikkat edilmiş, her yüz ifadesi için 16-19 arasında özellik vektörünün eğitim kümesine dahil edilmesi sağlanmıştır.

5.2 Çok Katmanlı Ağ ile Yapılan Denemeler

Bir sinir ağı probleminin çözümünde ağın tasarımı hesapsal karmaşıklığı ve genelleme kapasitesini direk olarak etkilemesinden dolayı önemlidir. Birçok teorik ve deneysel çalışma, ağ boyutlarının gereğinden büyük olması durumunda, eğitim kümesinin ezberlendiğini ve

genelleme yapamadığını göstermektedir. Ağ boyutları gereğinden küçük seçildiği durumlarda ise ağın eğitim verilerini öğrenmede zorluklarla karşılaştığı görülmektedir. Bu çalışmada tek gizli katmanlı, 10 x M x 4 boyutlarında, 10 giriş ve 4 çıkış elemanlı, 1<M<16 olmak üzere çeşitli ÇKA yapıları eğitilmiştir. ÇKA ağı için en ideal gizli katman işlem elemanı sayısı 7 olarak bulunmuştur.

ÇKA ağ yapısının Matlab uygulaması newff komutuyla yapılmış olup, nümerik optimizasyon tekniklerini kullanan Levenberg Marquardt (trainlm), Ölçeklenmiş Konjugate Gradyent (trainscg), BFGS (trainbfg) ve Resilient Propagasyon (trainrp) eğitim kurallarıyla ağ eğitilmiştir. Ağın performans fonksiyonu olarak MSE kullanılmış ve hata değeri 10^{-6} alınmıştır.

İşlem elemanlarının aktivasyon fonksiyonu gizli katmanda tansig, çıkış katmanında ise logsig olarak alınmıştır. Çıkış katmanında logsig aktivasyon fonksiyonu kullanılarak, çıkış değerlerinin hedef değerleri gibi 0-1 arasında elde edilmesi sağlanmıştır.

ÇKA ağının yüz şeklini doğru sınıflama başarısı Levenberg Marquardt eğitim kuralıyla eğitilmiş ağda, ağın eğitim başarısı %100 olmuş, test başarısı ise %98'e kadar ulaşmıştır. Levenberg – Marquardt μ parametresi 10^{-5} ile 0.9 arasında farklı değerler alınmıştır.

Ölçeklenmiş Konjugate Gradyent öğrenme kuralıyla eğitilmiş ağda eğitim başarısı %100, test başarısı ise en yüksek %98 olmuştur. Ağ parametreleri $\lambda=5*10^{-7}$ ve $\sigma=5*10^{-5}$ olarak alınmıştır.

BFGS öğrenme kuralıyla eğitilmiş ÇKA sınıflayıcının ise eğitim başarısı %100 olmuş, test başarısı ise %95 doğrulukla yüz ifadelerini sınıflamıştır. Ağ parametresi 0 ile 1 arası değerler alınmıştır.

Resilient Propagasyon öğrenme kuralı ile eğitilmiş ÇKA sınıflayıcı, eğitim başarısı olarak %100, test başarısı olarak ise %96'a ulaşmıştır. Öğrenme oranı 0 ile 1 arası değerler alınmıştır.

Farklı öğrenme kuralları ve farklı normalizasyon işlemleri arasındaki ilişki çizelge 5.3'de görülmektedir. ÇKA ağın bu çalışmada elde edilmiş veri kümesi ile eğitilmesinde en başarılı sonuçlar özellik kümesinin ortalaması 0, standart sapması (σ) 1 olacak şekilde normalize edilmesiyle bulunmuştur. Çizelgede görülen sonuçlar ağın test edilmesinde elde edilen en yüksek başarı oranlarıdır.

Çizelge 5.3 ÇKA ağ yapısının farklı eğitim kurallarıyla eğitilmesi. Çizelgedeki 1. ve 2. satırlar normalize edilmiş verinin ÇKA ağ yapısına uygulanmasını, 3. satır ise orijinal veri kümesinden elde edilmiş sonuçları göstermektedir.

	Levenberg Marquardt Eğitim Kuralı		Ölçeklenmiş Konjugate Gradyent Eğitim Kuralı		BFGS eğitim kuralı		Resilient Propagasyon Eğitim Kuralı	
Norm. Tipi	Eğitim	Test	Eğitim	Test	Eğitim	Test	Eğitim	Test
Ort.=0 $\sigma = 1$	%100	%98,19	%100	%98,13	%100	%95.49	%100	%96.39
[-1,1]	%100	%77.47	%100	%53.15	%100	%60.36	%100	%68.46
Orijinal veri	%100	%97.30	%98.1	%97.29	%99.05	%92.79	%11.37	%10.81

5.3 Radyal Temelli Fonksiyon Ağları ile Yapılan Denemeler

Radyal Temelli Fonksiyon ağları Matlab'da iki farklı metot kullanılarak uygulanır. Bu metotlardan newrbe, gizli katmanda özellik kümesindeki özellik vektörü sayısı kadar sinir hücresi oluşturur. İkinci metot olan newrb ise ağın hata oranını belli bir hata oranına kadar yaklaştırmaya çalışır. İstenilen hata oranına erişilinceye kadar her adımda gizli katmana belli sayıda sinir hücresi ilave eder.

Newrb iki katmanlı bir ağ oluşturmaktadır. Birinci katmanda radyal tabanlı transfer fonksiyonu kullanılır ve ağırlıklar Öklid uzaklıklarına göre hesaplanır. Ağın ikinci katmanı ise lineer transfer fonksiyonu kullanmaktadır. Ağ eğitiminin başlangıcında ilk katmanda sinir hücresi bulunmamaktadır. Her adımda ilk katmana bir sinir hücresi eklenerek ağın ortalama karesel hatası hesaplanır. Ortalama karesel hata istenilen seviyenin altına düşünceye kadar ilk katmana sinir hücresi eklenmeye devam eder.

Newrb ile yaratılmış Radyal Temelli Fonksiyon ağlarının gizli katmanlarında daha az sayıda sinir hücresi bulunmaktadır. Buna rağmen, Radyal Tabanlı Sinir ağlarının boyutları diğer ağlardan büyüktür.

Çalışmada newrb ve newrbe yöntemlerinin ikisi de kullanılmış olup, hata değeri 10^{-6} alınmıştır. Newrb ile her adımda gizli katmana 1 işlem elemanı eklenmiştir. Her iki yöntemde de saklı katmanda 211 işlem elemanı kullanıldığı görülmüş, ağın performansı hata değerinin altına düşmemiştir.

RTFA ağı için denemeler 0.1 ile 10^5 arası farklı genişlik değerleri kullanılarak yapılmıştır. Ağın başarısı eğitimde %100'e ulaşırken, test kümesinde %93 olmuştur. Çizelge 5.4 RTFA

ağ yapısı kullanılarak çalışmada elde edilen başarı oranlarını göstermektedir. RTFA ağ yapısı en başarılı sonuçlara orijinal özellik vektörü kullanıldığında ulaşmıştır.

Çizelge 5.4 RTFA ağ yapısının farklı şekilde normalize edilmiş giriş vektörleriyle eğitilmesi

Normalizasyon Tipi	Eğitim Başarısı %	Test Başarısı %
Verinin ortalama=0, σ = 1 olacak şekilde normalizasyonu	% 100	% 57.65
Verinin [-1,1] arası normalizasyonu	% 100	% 28.82
Orijinal veri kümesi	% 100	% 93.69

5.4 Konik Kesit Fonksiyonlu Sinir Ağları ile Yapılan Denemeler

Konik Kesit Fonksiyonlu Sinir ağları ÇKA ve RTFA ağlarının özelliklerini taşımaktadır. Bu çalışmada Konik Kesit Fonksiyonlu Sinir ağlarının başarısı eğitimde %100, yüz şeklini doğru sınıflamada ise %99 olmuştur. KKFSA ağı öğrenme oranı ve momentum 0.1-0.9 arası değerler verilerek eğitilmiş ve test edilmiştir. Çizelge 5.5 KKFSA ağ yapısı ile elde edilen sonuçları göstermektedir.

Çizelge 5.5 KKFSA ağ yapısının farklı şekilde normalize edilmiş giriş vektörleriyle eğitilmesi

Normalizasyon Tipi	Eğitim Başarısı %	Test Başarısı %
Verinin ortalama=0, σ = 1 olacak şekilde normalizasyonu	% 100	% 99.09
Verinin [-1,1] arası normalizasyonu	% 99.56	% 67.56
Orijinal veri kümesi	% 85.78	% 12.613

Konik Kesit Fonksiyonlu Sinir ağları ile yapılan denemelerde, gizli katmanda 7 sinir hücresi olduğu durumda, normalize edilmiş veriler için ağ başarı ile eğitilebilmektedir. Özellikle veri ortalaması 0, standart sapması (σ) 1 olacak şekilde normalize edildiğinde KKFSA ağının test başarısı %99'a ulaşmaktadır. Orijinal veri kümesi gizli katmanda 7 sinir hücresi ile eğitilememekte olup, gizli katmandaki sinir hücresi sayısı 12'e çıkarıldığında, bu durumdaki test başarısı %87'ye kadar çıkmaktadır.

5.2, 5.3 ve 5.4 bölümlerinde ilgili ağlarla yapılmış denemelerin bazılarını ve kullanılan ağ parametrelerini ekler bölümünde detaylı bir şekilde bulabilirsiniz.

6. SONUÇLAR

Bu tez çalışmasında temel yüz ifadelerini otomatik olarak sınıflayan bir program geliştirilmiştir. Çalışmada ele alınan yüz ifadeleri gözler ve dudakların durumlarına göre değişmektedir. Her yüz ifadesi bir kullanıcı tanımlı istek ile eşleştirilmiş ve bu isteklerin bilgisayar ile görüntüden otomatik olarak algılanması sağlanmıştır. Bilgisayar göz ve dudak durumlarına göre 12 farklı yüz ifadesini ayırt edebilmiştir. Görüntüden algılanan her yüz ifadesi için bilgisayar sesli ve yazılı bir uyarı sinyali üretmektedir. Yapay Sinir ağına öğretilen istekler çizelge 4.1'de bulunmaktadır.

Çalışmada yüz ifadelerini tanımlayabilmek için gözlerin ve dudakların durumları belirleyici öğe olarak kullanılmıştır. Gözlerin ve dudakların durumunu belirleyebilmek için 10 adet ayırt edici özellik çıkarılmıştır. Yüz ifadelerinin belirlenmesinde sol ve sağ göz için 3'er özellik, dudaklar için ise 4 özellik kullanılmıştır. Bu özellikler ile çeşitli sinir ağı yapılarıyla denemeler yapılmış, ağın büyüklüğü ve parametrelerinin ağın öğrenmesine etkisi incelenmiştir.

Özellik matrisinin hazırlanması sırasında gözlerin ve dudakların konumlarının doğru olarak tespit edilmesinin önemi anlaşılmıştır. Dudakların konumlarının bulunması sırasında alt dudak kolayca bulunabildiği halde, üst dudağın inceliğinden dolayı bulunmasının zor olduğu görülmüştür. Gözlerin yerlerinin doğru olarak bulunabilmesi için deforme olabilen şablon uygulamasının gözlerin yerinin tespitinde başarıyı arttıracağı düşünülmektedir.

Özellik matrisi ile yazılımsal olarak ÇKA, RTFA ve KKFSA ağları eğitilmiş ve test edilmiştir. Bu ağ yapılarının oluşturulması için Matlab 7.0 ve Neuro Solutions 5 yazılımları kullanılmıştır.

Elde edilen deneysel sonuçlar çizelge 6.1'de irdelenmektedir. ÇKA için verilen sınıflandırma başarısı Levenberg Marquardt, Ölçeklenmiş Konjugate Gradyent, BFGS ve Resilient Propagasyon eğitim kurallarının başarı ortalaması alınarak çizelgeye yazılmıştır.

Çizelge 6.1 Farklı ağ yapılarıyla elde edilen yüz şekli sınıflandırma başarısı

AgYapisi / *NormalizasyonTipi*	Ortalama=0,σ = 1 olacak şekilde normalize edilen veri kümesi	[-1,1] arası normalize edilen veri kümesi	Orijinal veri kümesi
ÇKA	% 97.05	% 64.86	% 74.54
RTFA	% 57.65	% 28.82	% 93.69
KKFSA	% 99.09	% 67.56	% 12.61

ÇKA ağları ve KKFSA ağları için en iyi sonuçların, ortalaması 0, standart sapması (σ) 1 olacak şekilde normalize edilen özellik kümesiyle elde edildiği görülmüştür. RTFA ağları ise orijinal özellik kümesiyle en iyi şekilde eğitilebilmektedir.

Genel olarak ağın eğitiminin ÇKA ağlarında, KKFSA ağlarına göre bir miktar daha hızlı tamamlandığı görülmüştür.

YSA'nın yüz ifadelerini tanıma başarısının birçok etkene bağlı olduğu anlaşılmıştır. YSA yapısının seçimi, saklı katmanda bulunan sinir hücresi sayısı, saklı katman sayısı, öğrenme algoritmasının ve ilgili parametrelerin belirlenmesi, ağda kullanılan transfer ve performans fonksiyonları, verilerin farklı şekillerde normalize edilmesi, veri üzerinde yapılan ön işleme ve son işleme teknikleri ağın sınıflama başarısını doğrudan etkileyen etkenlerdir. Bu etkenlerin uygun olarak seçilmesi ağ başarısını olumlu olarak değiştirmektedir.

Başarısı % 57.65 ile % 67.56 arasında kalan ağ yapılarında ağın başarısındaki düşüşün sol göz sınıflama hatasından kaynaklandığı tespit edilmiş, sağ göz ve ağzın durumlarının genelde doğru tespit edildiği bulunmuştur. Sol göz özellik verileri dikkatli incelendiğinde, sol gözden elde edilen renk özü ve doygunluk verilerinin sağ göze göre daha düşük olduğu anlaşılmıştır. Sağ gözün kapalı olduğu durumlarda sol gözün daha güç açılması, renk özü ve doygunluk özelliklerinin sağ göze göre az olmasına neden olmaktadır. Çalışmada, özellik kümesinin doğru bir şekilde normalizasyonunun sınıflama başarısını arttırıcı bir etken olduğu görülmüştür.

Çalışma için yazılan programın yüz üzerinde gözlerin ve ağzın yerini tespit etmesi, ilgili özellikleri çıkarması ve bu özellikleri sinir ağıyla değerlendirmesi yaklaşık 1 saniye sürmektedir. İşlem yükü göz önüne alındığında bu başarılı bir süre olarak değerlendirilmiştir.

Sonuç olarak en az sayıda özellik kullanılarak 12 yüz ifadesini ayırt edebilen bir sistem geliştirilmiştir. Yüz ifadelerini ayırt etmek için kullanılan özelliklerin azlığı ve başarısı dikkat çekicidir.

Yapay Sinir Ağlarının farklı veri türlerini birleştirerek sınıflandırma yapabilen etkin bir araç olduğu görülmüş ve görüntü işleme problemlerinin çözümünde Yapay Sinir Ağlarının etkin ve yüksek başarı oranlarına sahip bir teknoloji olduğu ortaya konmuştur.

6.1 Çalışmaya Yapılabilecek Eklentiler

Yazılımsal denemelerde ÇKA ve KKFSA ağlarının eğitimi için optimal ağ boyutu olarak 10 x 7 x 4 elde edilmiştir. 10 giriş, 4 çıkış ve 7 gizli katman işlem elemanıyla oluşturulan bu ağ yapısı birçok Sinir Ağı Tümdevresine uygulanabilecek bir yapıdadır.

Yazılımsal olarak elde edilen yüksek başarı oranları, özelliklerin sinir ağı tümdevresine uygulanması için ümit verici niteliktedir.

Kullanılan özellikler dışında başka özellikler çıkarılarak sinir ağının sınıflama başarısı arttırılabilir.

Bu tezde kullanılan sinir ağları bir kişiye ait görüntülerden elde edilen özelliklerle eğitilmiş olduğundan farklı kişilerde yeterli başarıyı gösterememektedir. Özellik kümesinin farklı kişilere ait görüntüler ile zenginleştirilmesi önemli bir eklenti olacaktır.

Bu çalışmada tanımlanan yüz ifadeleri, göz ve ağız durumlarının birleştirilmesinden elde edilmiştir. Çalışma yanak, dil ve kaş hareketlerini tanıyacak şekilde geliştirilerek daha fazla sayıda kişisel isteğin yüz ifadeleri ile anlatılabilir hale getirilmesi mümkündür.

Çalışmanın kodları Matlab ortamından daha alt seviye yazılım dillerine taşınarak yüz ifadelerinin tespit edilmesi daha kısa sürede yapılabilir.

İleri bir aşama olarak, değerlendirilen yüz ifadelerinin karşılığı olan istekler web veya cep telefonu üzerinden ilgili kişilere yönlendirilebilir.

KAYNAKLAR

Chau, M. ve Bekte, M., (2005), "Real Time Eye Tracking and Blink Detection with USB Cameras", Boston University Computer Science Technical Report.

D'Orazio, T., Leo, M. ve Distante, A., (2004), "Eye Detection in Face Images Vigilance System". IEEE Intelligent Vehicles Symposium University of Parma, Italy.

Demuth, H. ve Beale, M. (2000), Neural Network Toolbox, Mathworks, Natick.

Elmas, Ç., (2003), Yapay Sinir Ağları, Seçkin Yayınları, İstanbul.

Erkmen, B., Yıldırım, T., (2007), "VLSI Implementation of General Purposed Conic Section Function Neural Network" 5th International Conference on Electrical and Electronics Engineering (ELECO'2007), Bursa, Turkey, December.

Ertürk, S.,(2004), İmge İşleme Dersi Notları, KOÜ.

Eveno, N., Caplier, A. ve Coulon., P., (2001), "New Color Transformation for Lips Segmentation." Multimedia Signal Processing, IEEE Fourth Workshop, p.3-8.

Gümüştekin, Ş., (2005), Görüntü İşleme Dersi Notları, İYTE.

Hsu, R., Abdel-Mottaleb, M. ve Jain, A.K., (2002) "Face Detection in Color Images." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, no.5.

Hyunwoo K., Jong L. ve Kee, S.C., (2004), "A Fast Eye Localization Method for Face Recognition". Proceedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication, Japan, p.20-22.

Moller, M. F. (1993), "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", Neural Networks, Vol. 6, pp. 525-533.

Pantic, M. ve Rothkrantz, L.J.M., (2000), "Automatic Analysis of Facial Expressions", State of the art, IEEE Trans. PAMI, Vol.22, no.12, p.1425-2556.

Pong, C., Lai, J. H. ve Huang, Q. Y., (2004), "Mouth State Estimation in Mobile Computing Environment." Automatic Face and Gesture Recognition, Sixth IEEE International Conference, p.705-710.

Seung, S., (2002), Neural Networks Dersi Notları, MIT.

Şenol, C. (2004), "Standart ve Hibrid Yapılar Kullanarak Yapay Sinir Ağları ile İmza Tanıma", Yıldız Teknik Üniversitesi Yüksek Lisans tezi.

Tavşanoğlu, V., (2005), "Görüntü İşleme Ders Notları", Yıldız Teknik Üniversitesi.

Tian, Z. ve Qin, H., (2005), "Real-time Driver's Eye State Detection", IEEE International Conference on Vehicular Electronics and Safety, p. 285-289

Yıldırım, T., (2005), Yapay Sinir Ağları ve Uygulamaları Dersi Notları, Yıldız Teknik Üniversitesi.

Yavuz, Z., (2003), "Bilgisayarlı Dudak Okuma." Karadeniz Teknik Üniversitesi Yüksek Lisans tezi.

Yuille, A.L., Cohen, D.S. ve Hallinan, P.W., (1989), "Feature Extraction from Faces Using Deformable Templates", Proc. IEEE CVPR, p.104-109

Zarit, B. D., Super, B. J. ve Quek, F. K. H., (1999) "Comparison of Five Color Models in Skin Pixel Classification". In ICCV'99 Int'lWorkshop on recognition, analysis and tracking of faces and gestures in Real-Time systems

EKLER

EK 1: ÇKA ağ için yazılmış eğitim ve test programı
EK 2: RTFA ağları için yazılmış eğitim ve test programı
EK 3: KKFSA ağları için yazılmış eğitim ve test programı
EK 4: Orijinal özellik kümesi (eğitim için)
EK 5: Sınıflama işleminde kullanılan ağ konfigürasyonları tablosu

EK 1: ÇKA ağ için yazılmış eğitim ve test programı

```
clear all;
close all;

load sayaclar sayac1 sayac2;
normalize=0;
saklikatman=7;
cikiskatman=4;

if normalize==0
        load traintarget;
        load trainolcek;
else
        load traintargetn;
        load trainolcekn;
end

net=newff(minmax(trainolcek),[saklikatman cikiskatman],{'tansig','logsig'},'trainlm');

%net.trainParam.delta=1
%net.trainParam.min_grad= 0.00000000000000000000000000001;
%net.trainParam.lr_inc=1.01;
% net.trainParam.delta_inc=0.01;
% net.trainParam.delta_dec=0.01;
% net.trainParam.delta0=0.1;
% net.trainParam.deltamax=100;
%net.trainParam.minstep=10^-300;
net.trainParam.show = 100;
% net.trainParam.lr = 0.01;
% net.trainParam.lr_inc=1.05;
% net.trainParam.lr_dec=0.7;
% net.trainParam.max_perf_inc=1.04;
lnr=0:0.1:1;
net.trainParam.lr=0.5;
%net.trainParam.lr;
%net.trainParam.sigma=10^-6;
%net.trainParam.lambda=10^-28;
%net.trainParam.alpha=0.01;
net.trainParam.epochs = 5000;
net.trainParam.goal = 0.000001;
net.trainParam.mc=0.5;
%net.trainParam.mu=0.0001;

net = train(net,trainolcek,traintarget);

result=sim(net,trainolcek);
for i=1:size(result,2)
   for j=1:4
      if result(j,i)>0.5
         result(j,i)=0.9;
```

```
        else if result(j,i)<=0.5
            result(j,i)=0.1;
            end
        end
    end
end

solgozdogru=0;
solgozyanlis=0;
saggozdogru=0;
saggozyanlis=0;
agizacikkapalidogru=0;
agizacikkapaliyanlis=0;
solgozflag=0;
saggozflag=0;
gozlerdogru=0;
gozleryanlis=0;
agizsekildogru=0;
agizsekilyanlis=0;
agizakflag=0;
agizsekilflag=0;
agizdogru=0;
agizyanlis=0;
gozflag=0;
agizflag=0;
yuzseklidogru=0;
yuzsekliyanlis=0;

for i=1:size(result,2)
    if result(1,i)==traintarget(1,i)
        solgozdogru=solgozdogru+1;
        solgozflag=1;
    else
        solgozyanlis=solgozyanlis+1;
    end

    if result(2,i)==traintarget(2,i)
        saggozdogru=saggozdogru+1;
        saggozflag=1;
    else
        saggozyanlis=saggozyanlis+1;
    end

    if solgozflag==1 & saggozflag==1
        gozlerdogru=gozlerdogru+1;
        gozflag=1;
    else
        gozleryanlis=gozleryanlis+1;
    end

    if result(3,i)==traintarget(3,i)
```

```
        agizacikkapalidogru=agizacikkapalidogru+1;
        agizakflag=1;
    else
        agizacikkapaliyanlis=agizacikkapaliyanlis+1;
    end

    if result(4,i)==traintarget(4,i)
        agizsekildogru=agizsekildogru+1;
        agizsekilflag=1;
    else
        agizsekilyanlis=agizsekilyanlis+1;
    end

    if agizakflag==1 & agizsekilflag==1
        agizdogru=agizdogru+1;
        agizflag=1;
    else
        agizyanlis=agizyanlis+1;
    end

    if gozflag==1 & agizflag==1
        yuzseklidogru=yuzseklidogru+1;
    else
        yuzsekliyanlis=yuzsekliyanlis+1;
    end

    solgozflag=0;
    saggozflag=0;
    agizakflag=0;
    agizsekilflag=0;
    gozflag=0;
    agizflag=0;
end

display('Egitim Sonuclari');
solgozbasari=(solgozdogru / (solgozdogru+solgozyanlis))*100 ;
saggozbasari=(saggozdogru / (saggozdogru+saggozyanlis))*100 ;
gozlerbasari=(gozlerdogru / (gozlerdogru+gozleryanlis))*100;
agizacikkapalibasari=(agizacikkapalidogru /
(agizacikkapalidogru+agizacikkapaliyanlis))*100;
agizsekilbasari=(agizsekildogru / (agizsekildogru+agizsekilyanlis))*100;
agizbasari=(agizdogru / (agizdogru+agizyanlis))*100;
yuzseklibasari=(yuzseklidogru / (yuzseklidogru+yuzsekliyanlis))*100;

display('lnr  solgozbasari saggozbasari gozlerbasari agizacikkapalibasari agizsekilbasari
agizbasari  yuzseklibasari');
eg=[lnr(sayac1) solgozbasari saggozbasari gozlerbasari agizacikkapalibasari agizsekilbasari
agizbasari yuzseklibasari]

% TEST ASAMASI
```

```
if normalize==0
        load testolcek;
        load testtarget;
else
        load testolcekn;
        load testtargetn;
end

result=sim(net,testolcek);

for i=1:size(result,2)
   for j=1:4
      if result(j,i)>0.5
         result(j,i)=0.9;
      else if result(j,i)<=0.5
         result(j,i)=0.1;
         end
      end
   end
end

solgozdogru=0;
solgozyanlis=0;
saggozdogru=0;
saggozyanlis=0;
agizacikkapalidogru=0;
agizacikkapaliyanlis=0;
solgozflag=0;
saggozflag=0;
gozlerdogru=0;
gozleryanlis=0;
agizsekildogru=0;
agizsekilyanlis=0;
agizakflag=0;
agizsekilflag=0;
agizdogru=0;
agizyanlis=0;
gozflag=0;
agizflag=0;
yuzseklidogru=0;
yuzsekliyanlis=0;

for i=1:size(result,2)
   if result(1,i)==testtarget(1,i)
      solgozdogru=solgozdogru+1;
      solgozflag=1;
   else
      solgozyanlis=solgozyanlis+1;
   end

   if result(2,i)==testtarget(2,i)
```

```
        saggozdogru=saggozdogru+1;
        saggozflag=1;
    else
        saggozyanlis=saggozyanlis+1;
    end

    if solgozflag==1 & saggozflag==1
        gozlerdogru=gozlerdogru+1;
        gozflag=1;
    else
        gozleryanlis=gozleryanlis+1;
    end

    if result(3,i)==testtarget(3,i)
        agizakflag=1;
        agizacikkapalidogru=agizacikkapalidogru+1;
    else
        agizacikkapaliyanlis=agizacikkapaliyanlis+1;
    end

    if result(4,i)==testtarget(4,i)
        agizsekilflag=1;
        agizsekildogru=agizsekildogru+1;
    else
        agizsekilyanlis=agizsekilyanlis+1;
    end

    if agizakflag==1 & agizsekilflag==1
        agizdogru=agizdogru+1;
        agizflag=1;
    else
        agizyanlis=agizyanlis+1;
    end

    if gozflag==1 & agizflag==1
        yuzseklidogru=yuzseklidogru+1;
    else
        yuzsekliyanlis=yuzsekliyanlis+1;
    end

    solgozflag=0;
    saggozflag=0;
    agizakflag=0;
    agizsekilflag=0;
    gozflag=0;
    agizflag=0;
end

display('Test Sonuclari');
solgozbasari=(solgozdogru / (solgozdogru+solgozyanlis))*100 ;
saggozbasari=(saggozdogru / (saggozdogru+saggozyanlis))*100 ;
```

```
gozlerbasari=(gozlerdogru / (gozlerdogru+gozleryanlis))*100;
agizacikkapalibasari=(agizacikkapalidogru /
(agizacikkapalidogru+agizacikkapaliyanlis))*100;
agizsekilbasari=(agizsekildogru / (agizsekildogru+agizsekilyanlis))*100;
agizbasari=(agizdogru / (agizdogru+agizyanlis))*100;
yuzseklibasari=(yuzseklidogru / (yuzseklidogru+yuzsekliyanlis))*100;

display('lnr  solgozbasari saggozbasari gozlerbasari agizacikkapalibasari agizsekilbasari
agizbasari  yuzseklibasari');
test=[lnr(sayac1) solgozbasari saggozbasari gozlerbasari agizacikkapalibasari agizsekilbasari
agizbasari yuzseklibasari]

load mlpeg.txt;
load mlptest.txt;

sizeeg=size(mlpeg,1);
sizetest=size(mlptest,1);

mlpeg(sizeeg+1,:)=eg;
mlptest(sizetest+1,:)=test;

save mlpeg.txt mlpeg -ASCII;
save mlptest.txt mlptest -ASCII;
```

EK 2: RTFA ağları için yazılmış eğitim ve test programı

```
for u=0:0.002:2
   save u u;
clear all;
close all;
load u u;
normalize=1;
saklikatman=7;
cikiskatman=4;

if normalize==0
        load traintarget;
        load trainolcek;
else
        load traintargetn;
        load trainolcekn;
end

GOAL=0.000001;
SPREAD=u;

net=newrbe(trainolcek,traintarget,SPREAD);
%net=newrb(trainolcek,traintarget,GOAL,SPREAD,211,1);
%net=newgrnn(trainolcek,traintarget,SPREAD);

result=sim(net,trainolcek);
for i=1:size(result,2)
   for j=1:4
      if result(j,i)>0.5
         result(j,i)=0.9;
      else if result(j,i)<=0.5
         result(j,i)=0.1;
         end
      end
   end
end

solgozdogru=0;
solgozyanlis=0;
saggozdogru=0;
saggozyanlis=0;
agizacikkapalidogru=0;
agizacikkapaliyanlis=0;
solgozflag=0;
saggozflag=0;
gozlerdogru=0;
gozleryanlis=0;
agizsekildogru=0;
agizsekilyanlis=0;
agizakflag=0;
```

```
agizsekilflag=0;
agizdogru=0;
agizyanlis=0;
gozflag=0;
agizflag=0;
yuzseklidogru=0;
yuzsekliyanlis=0;

for i=1:size(result,2)
   if result(1,i)==traintarget(1,i)
      solgozdogru=solgozdogru+1;
      solgozflag=1;
   else
      solgozyanlis=solgozyanlis+1;
   end

   if result(2,i)==traintarget(2,i)
      saggozdogru=saggozdogru+1;
      saggozflag=1;
   else
      saggozyanlis=saggozyanlis+1;
   end

   if solgozflag==1 & saggozflag==1
      gozlerdogru=gozlerdogru+1;
      gozflag=1;
   else
      gozleryanlis=gozleryanlis+1;
   end

   if result(3,i)==traintarget(3,i)
      agizacikkapalidogru=agizacikkapalidogru+1;
      agizakflag=1;
   else
      agizacikkapaliyanlis=agizacikkapaliyanlis+1;
   end

   if result(4,i)==traintarget(4,i)
      agizsekildogru=agizsekildogru+1;
      agizsekilflag=1;
   else
      agizsekilyanlis=agizsekilyanlis+1;
   end

   if agizakflag==1 & agizsekilflag==1
      agizdogru=agizdogru+1;
      agizflag=1;
   else
      agizyanlis=agizyanlis+1;
   end
```

```
   if gozflag==1 & agizflag==1
      yuzseklidogru=yuzseklidogru+1;
   else
      yuzsekliyanlis=yuzsekliyanlis+1;
   end

   solgozflag=0;
   saggozflag=0;
   agizakflag=0;
   agizsekilflag=0;
   gozflag=0;
   agizflag=0;
end

display('Egitim Sonuclari');
solgozbasari=(solgozdogru / (solgozdogru+solgozyanlis))*100 ;
saggozbasari=(saggozdogru / (saggozdogru+saggozyanlis))*100 ;
gozlerbasari=(gozlerdogru / (gozlerdogru+gozleryanlis))*100;
agizacikkapalibasari=(agizacikkapalidogru /
(agizacikkapalidogru+agizacikkapaliyanlis))*100;
agizsekilbasari=(agizsekildogru / (agizsekildogru+agizsekilyanlis))*100;
agizbasari=(agizdogru / (agizdogru+agizyanlis))*100;
yuzseklibasari=(yuzseklidogru / (yuzseklidogru+yuzsekliyanlis))*100;

display('Spread  solgozbasari saggozbasari gozlerbasari agizacikkapalibasari
agizsekilbasari  agizbasari  yuzseklibasari');
eg=[SPREAD solgozbasari saggozbasari gozlerbasari agizacikkapalibasari agizsekilbasari
agizbasari yuzseklibasari]

% TEST ASAMASI

if normalize==0
         load testolcek;
         load testtarget;
else
         load testolcekn;
         load testtargetn;
end

result=sim(net,testolcek);

for i=1:size(result,2)
   for j=1:4
      if result(j,i)>0.5
         result(j,i)=0.9;
      else if result(j,i)<=0.5
         result(j,i)=0.1;
         end
      end
   end
end
```

```
solgozdogru=0;
solgozyanlis=0;
saggozdogru=0;
saggozyanlis=0;
agizacikkapalidogru=0;
agizacikkapaliyanlis=0;
solgozflag=0;
saggozflag=0;
gozlerdogru=0;
gozleryanlis=0;
agizsekildogru=0;
agizsekilyanlis=0;
agizakflag=0;
agizsekilflag=0;
agizdogru=0;
agizyanlis=0;
gozflag=0;
agizflag=0;
yuzseklidogru=0;
yuzsekliyanlis=0;

for i=1:size(result,2)
   if result(1,i)==testtarget(1,i)
      solgozdogru=solgozdogru+1;
      solgozflag=1;
   else
      solgozyanlis=solgozyanlis+1;
   end

   if result(2,i)==testtarget(2,i)
      saggozdogru=saggozdogru+1;
      saggozflag=1;
   else
      saggozyanlis=saggozyanlis+1;
   end

   if solgozflag==1 & saggozflag==1
      gozlerdogru=gozlerdogru+1;
      gozflag=1;
   else
      gozleryanlis=gozleryanlis+1;
   end

   if result(3,i)==testtarget(3,i)
      agizakflag=1;
      agizacikkapalidogru=agizacikkapalidogru+1;
   else
      agizacikkapaliyanlis=agizacikkapaliyanlis+1;
   end
```

```
    if result(4,i)==testtarget(4,i)
       agizsekilflag=1;
       agizsekildogru=agizsekildogru+1;
    else
       agizsekilyanlis=agizsekilyanlis+1;
    end

    if agizakflag==1 & agizsekilflag==1
       agizdogru=agizdogru+1;
       agizflag=1;
    else
       agizyanlis=agizyanlis+1;
    end

    if gozflag==1 & agizflag==1
       yuzseklidogru=yuzseklidogru+1;
    else
       yuzsekliyanlis=yuzsekliyanlis+1;
    end

    solgozflag=0;
    saggozflag=0;
    agizakflag=0;
    agizsekilflag=0;
    gozflag=0;
    agizflag=0;
end

display('Test Sonuclari');
solgozbasari=(solgozdogru / (solgozdogru+solgozyanlis))*100 ;
saggozbasari=(saggozdogru / (saggozdogru+saggozyanlis))*100 ;
gozlerbasari=(gozlerdogru / (gozlerdogru+gozleryanlis))*100;
agizacikkapalibasari=(agizacikkapalidogru /
(agizacikkapalidogru+agizacikkapaliyanlis))*100;
agizsekilbasari=(agizsekildogru / (agizsekildogru+agizsekilyanlis))*100;
agizbasari=(agizdogru / (agizdogru+agizyanlis))*100;
yuzseklibasari=(yuzseklidogru / (yuzseklidogru+yuzsekliyanlis))*100;

display('Spread  solgozbasari saggozbasari gozlerbasari  agizacikkapalibasari
agizsekilbasari  agizbasari  yuzseklibasari');
test=[SPREAD solgozbasari saggozbasari gozlerbasari agizacikkapalibasari agizsekilbasari
agizbasari yuzseklibasari]

load rbfeg.txt;
load rbftest.txt;

sizeeg=size(rbfeg,1);
sizetest=size(rbftest,1);

rbfeg(sizeeg+1,:)=eg;
rbftest(sizetest+1,:)=test;
```

```
save rbfeg.txt rbfeg -ASCII;
save rbftest.txt rbftest -ASCII;
end
```

EK 3: KKFSA ağları için yazılmış eğitim ve test programı

```
for init=1:2;
        for Alfa=0.1:0.1:1;
                for lr=0.1:0.1:1;

save agdata init Alfa lr;
clear all
clc
load agdata;

%Agin tanimlanmasi
in_num=10;
out_num=4;
hid_neuron_num=7;
act_hid = 3;
act_out = 2;
train_set_size=211;
test_set_size=111;
Epoch=1000;

load traintarget.txt;
train_targets=traintarget;
load trainolcekn;
train_inputs=trainolcek';
load testtarget.txt;
test_targets=testtarget;
load testolcekn;
test_inputs=testolcek';

%initial values for weights
if (init==1)
%MLP olarak baslangiç
   w_hid=rand(hid_neuron_num,in_num);
   w_hid=(w_hid-0.5);
   w_out=rand(out_num,hid_neuron_num);
   w_out=(w_out-0.5);
   om_hid=(pi/2)*ones(hid_neuron_num,1);
else
   %RBF olarak Baslangiç
   w_hid=zeros(hid_neuron_num,in_num);
   w_out=rand(out_num,hid_neuron_num);
   w_out=(w_out-0.5)*0.1;
   om_hid=(pi/4)*ones(hid_neuron_num,1);
end

%initial values for centers
load centers_hid.txt
c_hid=centers_hid;

%Training Process
```

```
for j = 1:Epoch
        for i = 1:train_set_size
        x = train_inputs(i,:);
        x_matris=ones(hid_neuron_num,1)*x;
        %%feed forward propagation for train
        u_hid=consec(x_matris,c_hid,w_hid,om_hid);
        a_hid=act_func(act_hid,u_hid);
        a_hid_matris=ones(out_num,1)*a_hid';
        u_out=w_out*a_hid;
        a_out=act_func(act_out,u_out);
        Y_train(:,i)=a_out;

        d_train=train_targets(i,:);
        %Backpropagation Phase
Delta_weights_out=w_out_delta(lr,d_train,a_out,a_hid_matris,hid_neuron_num,act_out);
Delta_weights_hid=w_hid_delta(lr,x_matris,d_train,a_out,a_hid,c_hid,w_out,in_num,act_out,
act_hid);
Delta_centers_hid=c_hid_delta(lr,x_matris,d_train,a_out,w_out,w_hid,a_hid,om_hid,c_hid,in
_num,act_out,act_hid);
Delta_angels_hid=om_hid_delta(lr,x_matris,d_train,a_out,a_hid,om_hid,c_hid,w_out,act_out,
act_hid);

%Updating Phase
if i==1 & j==1
    w_out_old=w_out;
    w_hid_old=w_hid;
    c_hid_old=c_hid;
    om_hid_old=om_hid;
    %Updating Phase
    w_out=w_out+Delta_weights_out;
    w_hid=w_hid+Delta_weights_hid;
    c_hid=c_hid+Delta_centers_hid;
    om_hid=om_hid+Delta_angels_hid;
 else
    w_out_new =w_out+Delta_weights_out+Alfa*(w_out-w_out_old);
    w_out_old=w_out;
    w_out=w_out_new;
    w_hid_new =w_hid+Delta_weights_hid+Alfa*(w_hid-w_hid_old);
    w_hid_old=w_hid;
    w_hid=w_hid_new;
    c_hid_new =c_hid+Delta_centers_hid+Alfa*(c_hid-c_hid_old);
    c_hid_old=c_hid;
    c_hid=c_hid_new;
     om_hid_new =om_hid+Delta_angels_hid+Alfa*(om_hid-om_hid_old);
    om_hid_old=om_hid;
    om_hid=om_hid_new;
 end
%Limitation of angels between -pi/2 : pi/2
om_hid=angels_control(om_hid,hid_neuron_num);
        end
end
```

```
%Test Process
for i = 1:test_set_size
        test_in = test_inputs(i,:);
        test_x_matris=ones(hid_neuron_num,1)*test_in;
        %%feed forward propagation for test
        u_hid=consec(test_x_matris,c_hid,w_hid,om_hid);
        a_hid=act_func(act_hid,u_hid);
        a_hid_matris=ones(out_num,1)*a_hid';
        u_out=w_out*a_hid;
        a_out=act_func(act_out,u_out);
        Y_test(:,i)=a_out;
end

% Train basarim orani
train_basarimi_tum=0;
train_basarimi_A=0;
train_basarimi_B=0;
train_basarimi_C=0;
train_basarimi_D=0;
train_basarimi_E=0;
train_basarimi_F=0;
train_basarimi_G=0;
train_basarimi_H=0;
train_basarimi_I=0;
train_basarimi_J=0;
train_basarimi_K=0;
train_basarimi_L=0;

for i = 1:211
  if Y_train(1,i)<=0.5 & Y_train(2,i)<=0.5 & Y_train(3,i)<=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.1 & train_targets(i,2)==0.1 & train_targets(i,3)==0.1 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_A=train_basarimi_A+1;

  elseif Y_train(1,i)<=0.5 & Y_train(2,i)<=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.1 & train_targets(i,2)==0.1 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_B=train_basarimi_B+1;

  elseif Y_train(1,i)<=0.5 & Y_train(2,i)<=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)>=0.5
train_targets(i,1)==0.1 & train_targets(i,2)==0.1 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.9
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_C=train_basarimi_C+1;

  elseif Y_train(1,i)<=0.5 & Y_train(2,i)>=0.5 & Y_train(3,i)<=0.5 & Y_train(4,i)<=0.5
train_targets(i,1)==0.1 & train_targets(i,2)==0.9 & train_targets(i,3)==0.1 &
train_targets(i,4)==0.1
```

```
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_D=train_basarimi_D+1;

  elseif Y_train(1,i)<=0.5 & Y_train(2,i)>=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.1 & train_targets(i,2)==0.9 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_E=train_basarimi_E+1;

  elseif Y_train(1,i)<=0.5 & Y_train(2,i)>=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)>=0.5 &
train_targets(i,1)==0.1 & train_targets(i,2)==0.9 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.9
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_F=train_basarimi_F+1;

  elseif Y_train(1,i)>=0.5 & Y_train(2,i)<=0.5 & Y_train(3,i)<=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.9 & train_targets(i,2)==0.1 & train_targets(i,3)==0.1 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_G=train_basarimi_G+1;

  elseif Y_train(1,i)>=0.5 & Y_train(2,i)<=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.9 & train_targets(i,2)==0.1 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_H=train_basarimi_H+1;

  elseif Y_train(1,i)>=0.5 & Y_train(2,i)<=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)>=0.5 &
train_targets(i,1)==0.9 & train_targets(i,2)==0.1 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.9
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;  train_basarimi_I=train_basarimi_I+1;

  elseif Y_train(1,i)>=0.5 & Y_train(2,i)>=0.5 & Y_train(3,i)<=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.9 & train_targets(i,2)==0.9 & train_targets(i,3)==0.1 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;  train_basarimi_J=train_basarimi_J+1;

  elseif Y_train(1,i)>=0.5 & Y_train(2,i)>=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)<=0.5 &
train_targets(i,1)==0.9 & train_targets(i,2)==0.9 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.1
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_K=train_basarimi_K+1;

  elseif Y_train(1,i)>=0.5 & Y_train(2,i)>=0.5 & Y_train(3,i)>=0.5 & Y_train(4,i)>=0.5 &
train_targets(i,1)==0.9 & train_targets(i,2)==0.9 & train_targets(i,3)==0.9 &
train_targets(i,4)==0.9
    i=i+1; train_basarimi_tum=train_basarimi_tum+1;
train_basarimi_L=train_basarimi_L+1;

  else
     i=i+1; train_basarimi_tum=train_basarimi_tum;
```

```
  end
end

train_basarim_orani=100*(train_basarimi_tum/211)

% TEST basarim orani
test_basarimi_tum=0;
test_basarimi_A=0;
test_basarimi_B=0;
test_basarimi_C=0;
test_basarimi_D=0;
test_basarimi_E=0;
test_basarimi_F=0;
test_basarimi_G=0;
test_basarimi_H=0;
test_basarimi_I=0;
test_basarimi_J=0;
test_basarimi_K=0;
test_basarimi_L=0;


for i = 1:1:111;
  if Y_test(1,i)<=0.5 & Y_test(2,i)<=0.5 & Y_test(3,i)<=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.1 & test_targets(i,2)==0.1 & test_targets(i,3)==0.1 &
test_targets(i,4)==0.1
    i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_A=test_basarimi_A+1;

  elseif Y_test(1,i)<=0.5 & Y_test(2,i)<=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.1 & test_targets(i,2)==0.1 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.1
    i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_B=test_basarimi_B+1;

  elseif Y_test(1,i)<=0.5 & Y_test(2,i)<=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)>=0.5
test_targets(i,1)==0.1 & test_targets(i,2)==0.1 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.9
    i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_C=test_basarimi_C+1;

  elseif Y_test(1,i)<=0.5 & Y_test(2,i)>=0.5 & Y_test(3,i)<=0.5 & Y_test(4,i)<=0.5
test_targets(i,1)==0.1 & test_targets(i,2)==0.9 & test_targets(i,3)==0.1 &
test_targets(i,4)==0.1
    i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_D=test_basarimi_D+1;

  elseif Y_test(1,i)<=0.5 & Y_test(2,i)>=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.1 & test_targets(i,2)==0.9 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.1
    i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_E=test_basarimi_E+1;

  elseif Y_test(1,i)<=0.5 & Y_test(2,i)>=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)>=0.5 &
test_targets(i,1)==0.1 & test_targets(i,2)==0.9 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.9
    i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_F=test_basarimi_F+1;
```

```
  elseif Y_test(1,i)>=0.5 & Y_test(2,i)<=0.5 & Y_test(3,i)<=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.9 & test_targets(i,2)==0.1 & test_targets(i,3)==0.1 &
test_targets(i,4)==0.1
     i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_G=test_basarimi_G+1;

  elseif Y_test(1,i)>=0.5 & Y_test(2,i)<=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.9 & test_targets(i,2)==0.1 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.1
     i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_H=test_basarimi_H+1;

  elseif Y_test(1,i)>=0.5 & Y_test(2,i)<=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)>=0.5 &
test_targets(i,1)==0.9 & test_targets(i,2)==0.1 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.9
     i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_I=test_basarimi_I+1;

  elseif Y_test(1,i)>=0.5 & Y_test(2,i)>=0.5 & Y_test(3,i)<=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.9 & test_targets(i,2)==0.9 & test_targets(i,3)==0.1 &
test_targets(i,4)==0.1
     i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_J=test_basarimi_J+1;

  elseif Y_test(1,i)>=0.5 & Y_test(2,i)>=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)<=0.5 &
test_targets(i,1)==0.9 & test_targets(i,2)==0.9 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.1
     i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_K=test_basarimi_K+1;

  elseif Y_test(1,i)>=0.5 & Y_test(2,i)>=0.5 & Y_test(3,i)>=0.5 & Y_test(4,i)>=0.5 &
test_targets(i,1)==0.9 & test_targets(i,2)==0.9 & test_targets(i,3)==0.9 &
test_targets(i,4)==0.9
     i=i+1; test_basarimi_tum=test_basarimi_tum+1;  test_basarimi_L=test_basarimi_L+1;

  else
     i=i+1; test_basarimi_tum=test_basarimi_tum;
  end
end
test_basarim_orani=100*(test_basarimi_tum/111)

load sonuckonik.txt;
sizesonuc=size(sonuckonik,1);
sonuckonik(sizesonuc+1,:)=[Alfa lr train_basarim_orani test_basarim_orani init];
save sonuckonik.txt sonuckonik -ascii;
                    end
          end
end
```

EK 4: Orijinal özellik kümesi (eğitim için)

H	S	Kor.	H	S	Kor.	Dudak yüksekliği	Dudak eni	Yükseklik / en	Dudak alanı
2	0	0,65813	0	0	0,66138	27	102	0,26471	2754
0	0	0,66012	0	0	0,60673	27	86	0,31395	2322
1	0	0,6827	1	0	0,63201	31	95	0,32632	2945
1	0	0,68075	4	0	0,65633	27	86	0,31395	2322
9	0	0,56096	1	0	0,65915	26	72	0,36111	1872
8	0	0,68258	0	0	0,62476	31	91	0,34066	2821
0	0	0,65252	2	0	0,66753	24	81	0,2963	1944
0	0	0,65385	0	0	0,64146	27	88	0,30682	2376
0	0	0,68071	0	0	0,70172	28	80	0,35	2240
0	0	0,70378	0	0	0,70741	31	82	0,37805	2542
0	0	0,65697	0	0	0,64989	35	81	0,4321	2835
0	0	0,68052	0	0	0,62645	29	74	0,39189	2146
0	0	0,67302	0	0	0,62957	38	81	0,46913	3078
0	0	0,67106	0	0	0,62023	25	81	0,30864	2025
0	0	0,69845	0	0	0,63225	26	80	0,325	2080
0	0	0,65019	0	0	0,62929	26	83	0,31325	2158
0	0	0,68472	0	0	0,70101	29	77	0,37662	2233
0	0	0,67602	0	0	0,62021	29	82	0,35366	2378
0	0	0,70165	0	0	0,64586	35	95	0,36842	3325
0	0	0,68456	0	0	0,62706	44	106	0,41509	4664
0	0	0,60862	0	1	0,6709	65	104	0,625	6760
0	0	0,6933	0	0	0,63617	53	90	0,58889	4770
0	0	0,69688	0	0	0,70915	64	98	0,65306	6272
0	1	0,685	0	0	0,70209	66	107	0,61682	7062
0	2	0,70668	0	0	0,70108	70	108	0,64815	7560
1	0	0,6771	3	0	0,66847	62	101	0,61386	6262
0	0	0,66828	0	0	0,6675	59	98	0,60204	5782
4	0	0,67652	1	0	0,66051	51	95	0,53684	4845
0	2	0,67125	2	2	0,60594	57	102	0,55882	5814
0	0	0,67353	0	0	0,65567	62	105	0,5904	6510
3	0	0,69775	0	0	0,61638	66	112	0,58929	7392
0	0	0,68942	0	0	0,67624	52	105	0,49524	5460
2	0	0,67353	0	0	0,61592	58	107	0,54206	6206
1	2	0,69267	0	0	0,66075	53	97	0,54639	5141
2	0	0,67777	0	0	0,65767	51	95	0,53684	4845
3	1	0,67852	0	0	0,65398	51	102	0,5	5202
5	0	0,60455	3	0	0,65754	87	74	1,1757	6438
0	0	0,66978	0	0	0,70024	81	66	1,2273	5346
0	0	0,70312	0	0	0,70377	84	81	1,037	6804
0	0	0,70948	0	0	0,70505	78	82	0,95122	6396
3	0	0,67171	2	0	0,6525	63	64	0,98438	4032
4	0	0,66545	3	0	0,6466	63	56	1,125	3528
2	2	0,68321	2	0	0,70658	74	61	1,2131	4514
3	1	0,66548	4	0	0,66306	64	59	1,0847	3776
3	1	0,67197	2	0	0,65497	69	66	1,0455	4554
0	0	0,676	2	0	0,6647	69	57	1,2105	3933

0	0	0,66487	4	0	0,66145	80	58	1,3793	4640
1	0	0,64722	2	0	0,66122	73	56	1,3036	4088
4	0	0,69972	136	0	0,77842	29	121	0,23967	3509
1	0	0,61299	91	0	0,77519	33	82	0,40244	2706
1	0	0,69552	83	0	0,80314	27	80	0,3375	2160
0	0	0,68739	126	0	0,75938	34	89	0,38202	3026
0	0	0,69208	134	0	0,7706	38	90	0,42222	3420
1	0	0,69201	61	6	0,89116	33	82	0,40244	2706
0	0	0,6462	20	0	0,81035	33	80	0,4125	2640
0	0	0,65621	20	0	0,77161	33	78	0,42308	2574
0	0	0,71473	20	0	0,81578	26	78	0,33333	2028
0	0	0,64639	20	0	0,77706	29	75	0,38667	2175
1	0	0,71299	91	0	0,77519	32	86	0,37209	2752
7	0	0,69552	63	0	0,80314	27	80	0,3375	2160
0	0	0,68739	126	0	0,74938	36	93	0,3871	3348
0	0	0,69208	134	0	0,7706	30	90	0,33333	2700
7	0	0,63657	82	15	0,81957	63	96	0,65625	6048
4	0	0,63227	137	6	0,79327	61	99	0,61616	6039
0	0	0,69266	0	0	0,81527	68	116	0,58621	7888
0	0	0,68962	10	0	0,75023	62	100	0,62	6200
0	0	0,67523	10	2	0,77356	66	96	0,6875	6336
0	0	0,64901	72	27	0,80593	61	87	0,70115	5307
0	0	0,70886	72	24	0,80496	52	101	0,51485	5252
0	0	0,69055	89	23	0,846	62	87	0,71264	5394
0	0	0,69109	50	10	0,81242	62	90	0,68889	5580
0	0	0,69017	133	29	0,79415	63	96	0,65625	6048
0	0	0,65553	86	40	0,80018	61	92	0,66304	5612
0	0	0,68765	74	63	0,79165	62	89	0,69663	5518
0	0	0,65095	56	33	0,82703	56	100	0,56	5600
0	0	0,6562	90	34	0,79888	51	100	0,51	5100
0	0	0,70132	104	38	0,84651	46	91	0,50549	4186
0	0	0,70603	73	35	0,815	60	95	0,63158	5700
0	0	0,71136	116	51	0,76506	56	88	0,63636	4928
0	0	0,66749	104	15	0,79838	73	107	0,68224	7811
1	0	0,68574	108	6	0,75618	109	70	1,5571	7630
0	0	0,70757	49	17	0,75251	80	69	1,1594	5520
0	0	0,64465	123	43	0,79257	83	88	0,93258	7304
0	0	0,64675	189	83	0,75984	67	75	0,89333	5025
3	0	0,65981	57	31	0,75584	69	58	1,1897	4002
5	0	0,68069	166	65	0,73314	69	62	1,1129	4278
0	0	0,63637	196	62	0,74233	65	62	1,0484	4030
0	0	0,63836	160	82	0,73787	80	72	1,1111	5760
0	0	0,64915	18	29	0,73879	75	67	1,1194	5025
0	0	0,64633	37	12	0,74704	71	73	0,9726	5183
134	26	0,76899	5	0	0,63376	27	105	0,25714	2835
75	4	0,82655	0	0	0,6488	31	71	0,43662	2201
78	16	0,96646	0	0	0,70467	30	81	0,37037	2430
78	4	0,91412	8	0	0,64806	31	76	0,40789	2356
96	9	0,7704	1	3	0,62704	26	70	0,37143	1820

20	2	0,81949	0	0	0,70926	22	83	0,26506	1826
20	0	0,77936	0	0	0,70022	32	82	0,39024	2624
20	2	0,79475	0	2	0,70751	26	79	0,32911	2054
20	0	0,77646	0	0	0,63892	31	88	0,35227	2728
20	2	0,79022	0	0	0,70064	25	83	0,3012	2075
20	0	0,77862	0	0	0,70941	29	71	0,40845	2059
0	0	0,75111	0	0	0,70565	27	74	0,36486	1998
78	16	0,96646	0	0	0,70467	30	82	0,36585	2460
78	4	0,91412	4	0	0,70806	31	76	0,40789	2356
91	44	0,84631	0	1	0,70968	31	97	0,31959	3007
59	32	0,79673	0	0	0,70739	34	92	0,36957	3128
95	44	0,80504	0	0	0,70859	39	93	0,41935	3627
32	4	0,77377	0	0	0,70742	37	91	0,40659	3367
19	1	0,7825	0	0	0,64114	75	107	0,70093	8025
10	0	0,7554	0	0	0,70206	59	91	0,64835	5369
69	12	0,70287	0	0	0,63706	74	107	0,69159	7918
56	43	0,76614	0	0	0,70407	68	95	0,71579	6460
78	37	0,69004	0	0	0,63052	72	97	0,74227	6984
188	31	0,82087	0	0	0,70684	59	106	0,5566	6254
79	74	0,71234	0	0	0,70849	66	103	0,64078	6798
39	24	0,70852	0	0	0,70923	64	103	0,62136	6592
55	42	0,7797	0	0	0,70477	75	101	0,74257	7575
218	47	0,7147	2	0	0,70496	56	96	0,58333	5376
160	52	0,8161	0	0	0,63512	63	104	0,60577	6552
154	68	0,82903	0	0	0,70719	66	107	0,61682	7062
117	68	0,73196	0	0	0,70332	67	90	0,74444	6030
156	80	0,80194	0	0	0,70424	75	93	0,80645	6975
113	70	0,7408	0	0	0,70314	66	93	0,70968	6138
127	54	0,7831	0	0	0,72787	79	98	0,80612	7742
120	76	0,83882	0	0	0,71974	73	92	0,79348	6716
58	72	0,85457	0	0	0,70364	75	93	0,80645	6975
135	51	0,81712	6	0	0,70989	65	95	0,68421	6175
73	40	0,76858	0	0	0,70314	69	100	0,69	6900
48	30	0,69422	0	0	0,70516	62	109	0,56881	6758
98	72	0,81126	0	0	0,7044	75	91	0,82418	6825
79	13	0,79437	0	0	0,70696	83	62	1,3387	5146
188	56	0,75601	2	0	0,70057	86	66	1,303	5676
220	67	0,68956	0	0	0,70554	86	67	1,2836	5762
22	0	0,74109	0	0	0,70344	115	70	1,6429	8050
91	59	0,72966	0	7	0,70213	80	64	1,25	5120
70	20	0,75185	1	0	0,66077	66	71	0,92958	4686
20	0	0,85579	0	0	0,64347	89	70	1,2714	6230
37	2	0,72033	3	3	0,64613	68	65	1,0462	4420
99	28	0,73233	0	0	0,70973	90	62	1,4516	5580
84	34	0,73736	0	0	0,7091	81	80	1,0125	6480
142	60	0,73336	3	0	0,64254	79	70	1,1286	5530
116	78	0,79656	0	0	0,70481	89	74	1,2027	6586
166	44	0,81417	0	0	0,70449	78	60	1,3	4680
79	64	0,73638	0	1	0,65031	70	71	0,98592	4970

152	47	0,81809	9	0	0,64327	88	63	1,3968	5544
210	92	0,78074	90	10	0,7816	30	86	0,34884	2580
249	60	0,83745	160	0	0,80456	36	115	0,31304	4140
184	4	0,86711	144	8	0,80955	28	68	0,41176	1904
90	88	0,73373	122	16	0,72642	34	83	0,40964	2822
71	10	0,75714	132	18	0,78101	31	82	0,37805	2542
148	21	0,78092	123	36	0,74098	22	95	0,23158	2090
268	48	0,69971	156	0	0,79228	36	92	0,3913	3312
20	21	0,8942	20	15	0,84992	32	92	0,34783	2944
27	30	0,79333	27	14	0,74045	27	85	0,31765	2295
81	28	0,90033	72	21	0,76788	34	85	0,4	2890
111	51	0,77725	81	21	0,72716	29	89	0,32584	2581
12	51	0,88315	20	41	0,74869	30	74	0,40541	2220
14	41	0,80905	22	24	0,79832	29	76	0,38158	2204
12	41	0,85688	28	33	0,83663	33	78	0,42308	2574
78	36	0,79164	30	16	0,75885	30	88	0,34091	2640
79	31	0,81341	112	4	0,81039	70	110	0,63636	7700
116	29	0,7914	138	11	0,77334	56	93	0,60215	5208
178	57	0,80499	154	0	0,75022	77	97	0,79381	7469
98	10	0,83337	133	4	0,81519	76	90	0,84444	6840
101	12	0,74835	102	0	0,83076	57	100	0,57	5700
129	0	0,86123	133	0	0,79677	55	112	0,49107	6160
42	40	0,79477	42	13	0,78734	54	108	0,5	5832
40	26	0,80201	54	3	0,74789	57	106	0,53774	6042
33	28	0,85617	19	27	0,81109	60	83	0,72289	4980
22	50	0,827	12	12	0,77401	65	84	0,77381	5460
39	3	0,79788	39	1	0,77511	69	112	0,61607	7728
29	0	0,86123	33	0	0,79677	56	102	0,54902	5712
101	12	0,74835	102	0	0,83076	64	97	0,65979	6208
178	57	0,80499	54	0	0,75022	77	98	0,78571	7546
98	10	0,83337	133	4	0,81519	74	91	0,81319	6734
73	4	0,87988	46	10	0,774	58	96	0,60417	5568
20	40	0,78349	28	0	0,88074	55	86	0,63953	4730
10	1	0,86684	16	2	0,86182	64	95	0,67368	6080
20	5	0,84903	20	0	0,78676	45	90	0,5	4050
13	2	0,82278	20	0	0,79781	57	109	0,52294	6213
88	1	0,75605	114	0	0,73318	51	84	0,60714	4284
20	0	0,70493	20	1	0,73474	70	88	0,79545	6160
10	0	0,77935	10	0	0,79838	56	90	0,62222	5040
104	0	0,76203	132	0	0,71247	66	110	0,6	7260
152	47	0,80644	183	30	0,75458	81	60	1,35	4860
210	40	0,79528	212	23	0,8249	74	80	0,925	5920
120	0	0,91118	156	0	0,8378	71	60	1,1833	4260
136	17	0,82316	143	4	0,7767	99	58	1,7069	5742
204	24	0,75392	225	52	0,84265	93	61	1,5246	5673
256	72	0,81004	176	0	0,76973	83	74	1,1216	6142
141	21	0,79249	230	0	0,80072	87	69	1,2609	6003
36	37	0,8393	40	19	0,7827	75	62	1,2097	4650
4	43	0,83419	20	8	0,75997	82	55	1,4909	4510

80	46	0,84098	63	24	0,80137	85	65	1,3077	5525
22	40	0,78681	20	31	0,76235	84	70	1,2	5880
120	0	0,91118	56	0	0,8378	73	67	1,0896	4891
136	17	0,82316	143	4	0,72767	89	64	1,3906	5696
204	24	0,70392	225	52	0,84265	98	64	1,5313	6272
256	72	0,71004	176	0	0,76973	86	69	1,2464	5934
56	4	0,86013	58	0	0,71936	79	61	1,2951	4819
17	5	0,8114	22	0	0,76782	82	63	1,3016	5166
20	7	0,81655	26	0	0,78289	72	65	1,1077	4680
31	3	0,70497	21	2	0,74955	80	74	1,0811	5920
17	17	0,78665	18	5	0,73131	75	60	1,25	4500
14	11	0,82713	20	0	0,79084	72	68	1,0588	4896
17	9	0,81921	15	6	0,78482	70	62	1,129	4340
20	7	0,82186	30	6	0,7421	68	75	0,90667	5100
10	10	0,80268	20	5	0,78098	85	61	1,3934	5185
28	32	0,81805	28	14	0,74056	74	64	1,1563	4736
13	29	0,80092	13	14	0,81112	78	69	1,1304	5382
29	42	0,82783	34	25	0,79131	85	82	1,0366	6970

EK 5: Sınıflandırma işleminde kullanılan ağ konfigürasyonları tablosu

ORT=0 SD=1	Levenberg Marquardt
Mu	Yüz Şekli Sınıflama Başarısı %
0,001	95,495
0,005	98,198
0,01	98,198
0,05	97,297
0,1	96,396
0,25	96,396
0,5	97,297
0,75	97,297
0,9	95,495

[-1,1]	Levenberg Marquardt
Mu	Yüz Şekli Sınıflama Başarısı %
1,00E-05	72,072
0,0001	71,171
0,0005	63,063
0,001	58,559
0,005	58,559
0,01	63,964
0,1	61,261
0,25	71,171
0,5	52,252
0,75	53,153
0,9	71,171

Orijinal Veri	Levenberg - Marquardt
Mu	Yüz Şekli Sınıflama Başarısı %
0,0001	90,09
0,001	93,694
0,005	91,892
0,01	94,595
0,05	89,189
0,1	97,297
0,25	92,793
0,5	93,694
0,75	97,297
0,9	95,495
1	90,991

ORT=0 SD=1	Ölçeklenmiş Konjugate Gradyent
sigma=5*10^-5	Yüz Şekli Sınıflama Başarısı %
lambda=5*10^-7	86,486
	97,297
	97,297
	92,793
	97,297
	94,595
	84,685
	85,586
	95,495
	96,396
	94,595
	97,297
	97,297
	98,198
	96,396

[-1,1]	Ölçeklenmiş Konjugate Gradyent
sigma=5*10^-5	Yüz Şekli Sınıflama Başarısı %
lambda=5*10^-7	48,649
	48,649
	48,649
	47,748
	47,748
	48,649
	46,847
	48,649
	47,748
	45,946
	46,847
	53,153
	46,847
	48,649
	47,748

ORT=0 SD=1	Resilient Propagasyon
Öğrenme Oranı	Yüz Şekli Sınıflama Başarısı %
0	90,09
0,1	89,189
0,2	82,883
0,3	82,883
0,4	88,288
0,5	83,784
0,6	92,793
0,7	91,892
0,8	85,586
0,9	83,784
1	90,09

[-1,1]	Resilient Propagasyon
Öğrenme Oranı	Yüz Şekli Sınıflama Başarısı %
0	49,55
0,1	46,847
0,2	47,748
0,3	55,856
0,4	51,351
0,5	46,847
0,6	68,468
0,7	45,045
0,8	47,748
0,9	47,748
1	44,144

Orijinal Veri	Resilient Propagasyon
Öğrenme Oranı	Yüz Şekli Sınıflama Başarısı %
0	10,811
0,1	10,811
0,2	10,811
0,3	10,811
0,4	10,811
0,5	10,811
0,6	10,811
0,7	11,712
0,8	10,811
0,9	10,811
1	10,811

ORT=0 SD=1	BFGS
Delta	Yüz Şekli Sınıflama Başarısı %
0	94,595
0,1	91,892
0,2	95,495
0,3	90,991
0,4	90,991
0,5	86,486
0,6	92,793
0,7	90,991
0,8	91,892
0,9	88,288
1	93,694

[-1,1]	BFGS
Delta	Yüz Şekli Sınıflama Başarısı %
0	55,856
0,1	49,55
0,2	55,856
0,3	48,649
0,4	52,252
0,5	58,559
0,6	53,153
0,7	52,252
0,8	49,55
0,9	60,36
1	53,153

Orijinal Veri	BFGS
Delta	Yüz Şekli Sınıflama Başarısı %
0	88,288
0,1	80,633
0,2	70,715
0,3	10,811
0,4	73,874
0,5	81,982
0,6	92,793
0,7	80,18
0,8	10,811
0,9	19,82
1	10,811

Orijinal Veri	RTFA
Ağ genişliği	Yüz Şekli Sınıflama Başarısı %
0,1	6,3063
1	9,9099
10	10,811
20	13,514
30	25,225
40	33,333
50	33,333
60	32,432
70	34,234
80	35,135
90	36,937
100	39,64
1000	32,432
5000	61,261
10000	74,775
15000	79,279
25000	81,982
30000	83,784
35000	84,685
40000	84,685
50000	85,586
60000	88,288
70000	89,189
80000	91,892
90000	90,09
1,00E+05	92,793

ORT=0 SD=1	RTFA
Ağ genişliği	Yüz Şekli Sınıflama Başarısı %
1	49,55
1,05	52,252
1,1	54,054
1,15	54,054
1,2	54,955
1,25	57,65
1,3	56,757
1,35	55,856
1,4	50,45
1,45	50,45
1,5	47,748
1,55	48,649

ORT=0 SD=1	KKFSA	
Momentum	Öğrenme Oranı	Yüz Şekli Sınıflama Başarısı %
0,1	0,1	96,396
0,1	0,1	93,694
0,1	0,2	97,297
0,1	0,3	96,396
0,1	0,4	96,396
0,1	0,5	95,495
0,1	0,6	92,793
0,1	0,7	93,694
0,1	0,8	93,694
0,1	0,9	93,694
0,1	1	90,09
0,2	0,1	99,099
0,2	0,2	99,099
0,2	0,3	99,099
0,2	0,4	97,297
0,2	0,5	99,099
0,2	0,6	98,198
0,2	0,7	95,495
0,2	0,8	97,297
0,2	0,9	93,694
0,2	1	86,486
0,3	0,1	98,198
0,3	0,2	96,396
0,3	0,3	97,297
0,3	0,4	98,198
0,3	0,5	96,396
0,3	0,6	82,883
0,3	0,7	90,991
0,3	0,8	93,694
0,3	0,9	96,396
0,3	1	90,991
0,4	0,1	96,396
0,4	0,2	99,099
0,4	0,3	78,378
0,4	0,4	95,495
0,4	0,5	95,495
0,4	0,6	94,595
0,4	0,7	98,198
0,4	0,8	67,568
0,4	0,9	54,955
0,4	1	58,559
0,5	0,1	98,198
0,5	0,2	97,297
0,5	0,3	100
0,5	0,4	96,396
0,5	0,5	95,495
0,5	0,6	96,396
0,5	0,7	68,468
0,5	0,8	50,45
0,5	0,9	53,153
0,5	1	50,45

0,6	0,1	98,198
0,6	0,2	99,099
0,6	0,3	67,568
0,6	0,4	78,378
0,6	0,5	93,694
0,6	0,6	82,883

ÖZGEÇMİŞ

Doğum tarihi	14.07.1982	
Doğum yeri	İzmir	
Lise	1997-2000	MEV Özel İzmir Fen Lisesi
Lisans	2000-2005	Doğu Akdeniz Üniversitesi Mühendislik Fak. Elektrik-Elektronik Mühendisliği Bölümü
Yüksek Lisans	2005-...	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Müh. Anabilim Dalı, Elektronik Yüksek Lisans Programı

Çalıştığı kurumlar

2006-2007	Kentkart Ltd.Şti.
2007-Devam ediyor	Nortel Netaş Telekomünikasyon A.Ş