T.C

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAPAY SİNİR AĞLARI İLE DEĞERLİ KAĞIT TANIMA SİSTEMİ

YÜKSEK LİSANS TEZİ

Tez Danışmanı

Prof. Dr. ALİ OKATAN

Hazırlayan

Serkan KABA

Şubat, 2009

İSTANBUL

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

KISALTMA LİSTESİ

ADALINE: Adaptive Linear Element

JPEG: Joint Photographic Experts Group

MADALINE: Multiple Adaptive Linear Element

TL: Türk Lirası

YSA: Yapay Sinir Ağları

YTL: Yeni Türk Lirası

ÖNSÖZ

Bana tüm yüksek lisans eğitimim boyunca yardımcı olan, desteğini esirgemeyen, bildiği bütün her şeyi paylaşan hocam Sn. Prof. Dr. Ali OKATAN'a (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği), bu tez çalışmamada yardımlarını esirgemeyen Sn. Araştırma Görevlisi Fatma DEMİREZEN YAĞMUR'a (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği), tezimi inceleyen ve görüşleriyle bana yol gösteren Yrd. Doç. Dr. Yüksel BAL (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği), Yrd. Doç. Dr. MURAT BEKEN (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği) ve Yrd. Doç. Dr. Osman ALİEFENDİOĞLU'na (Haliç Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği) teşekkür ederim.

Ayrıca hayatta her daim yanımda olan, yoğun çalışma dönemlerimde bana destek olan aileme teşekkür ederim.

Serkan KABA

Şubat, 2009

ÖZET

Resim işleme görsel verilerin sayısallaştırılması, sayısallaştırılmış resmin çeşitli işlemlerden geçirilerek üzerinde analiz yapılmaya hazır hale getirme ve ardından inceleyerek resmin belirleyici özelliklerini çıkarma işlemidir. Örüntü tanıma analiz edilmiş verilerden belirli kalıplar bulma, verileri sınıflandırma ve en nihayetinde elde edilen sınıflandırma bilgisinden yola çıkarak yeni gelen verileri de tanıyıp sınıflarını belirleme işlemidir.

Bu tez çalışmasında değerli kağıt olarak Türk Lirası banknotlarına yoğunlaşılmış, taranmış TL banknot resimleri; resim işleme, örüntü tanıma ve yapay sinir ağları kullanılarak tanınmaya çalışılmış ve sonuçları tartışılmıştır.

Anahtar Kelimeler: Resim işleme, örüntü tanıma, ayrık kosinüs dönüşümü, yapay sinir ağları, banknot

ABSTRACT

Image processing is the procedure of digitizing visual data, processing visualized image through several steps to make it ready for analysis and inspecting it to extract its identifying properties. Pattern recognition is the procedure of identifying patterns in analyzed data, classifying data and finally using this classification data to identify and classify newly acquired data.

This thesis focuses on Turkish banknotes. Scanned Turkish banknote images are identified using image processing, pattern recognition and neural network techniques and the results are discussed.

Keywords: Image processing, pattern recognition, discrete cosine transform, neural networks, banknote

1.GİRİŞ

Bir nesnenin tanınması ve diğerlerinden ayırt edilmesi pek çok otomasyon uygulamasında kritik öneme sahiptir. Güvenlik sistemleri, robotik, savaş sanayi gibi birçok alanda bilgisayarlı görme ve örüntü tanımaya dayalı sistemler geniş yer bulmaktadır..

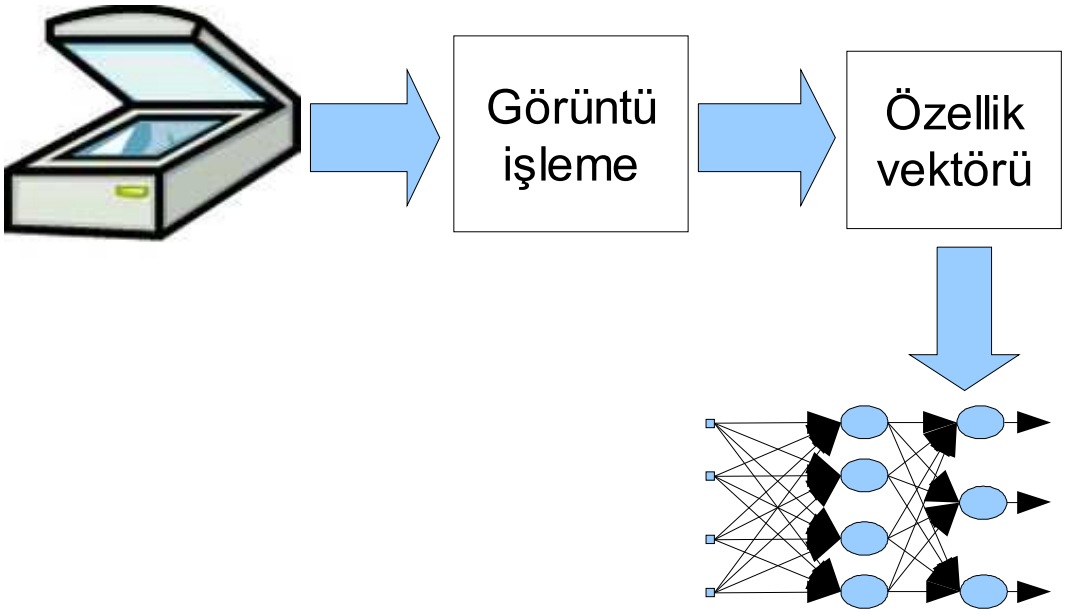
Yapay sinir ağı kullanarak bir nesneyi tanıyacak, diğerlerinden ayırt edecek sistemler geliştirmek için izlenmesi gereken temel adımlar vardır.

Görüntünün elde edilmesi ve sayısallaştırılması: Bu işlem hem donanım, hem yazılım kullanılarak gerçekleştirilir. Belli başlı donanımlar; kamera, fotoğraf makinesi, tarayıcı gibi cihazlardır.

Görüntünün işlenmesi: Görüntünün değişik işlemlerden geçirilerek örüntü tanımaya hazır hale getirilir.

Özellik vektörünün elde edilmesi: Bir görüntüye ait piksel sayısı yapay sinir ağlarının eğitimini ciddi şekilde zorlaştıracağından, o görüntüyü diğerlerinden ayırt eden ve temel özelliklerinin bulunup hesaplanması gerekmektedir. Özellik vektörü hesaplanan özelliklerin oluşturduğu sayısal bir dizidir.

Yapay sinir ağı tasarımı ve eğitilmesi: Probleme uygun yapay sinir ağının tasarımı ve eğitilmesi ve ardından test verileriyle doğruluğunun sınanması gerekmektedir. Şekil 1.1 bu adımların akış şeklini göstermektedir.



Şekil 1.1:Yapay sinir ağları ile örüntü tanımanın temel adımları

2.TÜRKİYE'DE BANKNOTUN TARİHÇESİ

2.1.Osmanlı Devleti'nde Banknotun Tarihçesi

Osmanlı Devleti'nde ilk banknot Tanzimat döneminde reformların finanse Abdülmecit tarafından edilmesi amacıyla bastırılmıştır. Kaime adı verilen bu paraların niteliği banknottan ziyada faiz getirili hisse senediydi. Daha sonra Osmanlı Bankası'nın 1856 yılında kurulmasının ardından 1863 yılında banknot basım yetkisi 30 yıllığına bu bankaya verilmiş ve Osmanlı Bankası 1863-1914 yıllarında çeşitli miktarlarda para ihraç etmiştir.

Birinci Dünya Savaşı esnasında Osmanlı Bankası para ihraç etmemiş, bu sebepten Osmanlı yönetimi altın ve Alman hazine bonolarını karşılık göstererek Evrak-ı Nakdiye adıyla 4 yıl boyunca para ihraç etmiştir.

2.2.Türkiye Cumhuriyeti'nde Banknotun Tarihçesi

Cumhuriyetin ilk yıllarında para bastırılamadığından Osmanlı Devleti'nden devrolan Evrak-ı Nakdiyeler 4 Aralık 1927'ye kadar yürürlükte kalmıştır. 30 aralıkı 1925'te kabul edilen yasanın ardından 5 aralık 1927'de Türkiye Cumhuriyeti'nin ilk banknotları piyasaya sürülmüştür. 1, 5, 10, 50, 100, 500 ve 1000 liralık olan bu banknotların metinleri Osmanlıca, değerleri Fransızca yazılmıştır.

1931 yılında Türkiye Cumhuriyet Merkez Bankası kurularak para basma yetkisi Merkez Bankası'na devredilmiştir. Merkez Bankası bu tarihten günümüze 8 emisyon grubunda (2.-9.) para basıp piyasaya ihraç etmiştir. İkinci Dünya Savaşı sırasında bastırılan ancak savaş koşulları sebebiyle İngiltere'den getirilemeyen banknotlar bu sebepten piyasaya sürülememiştir.[1]

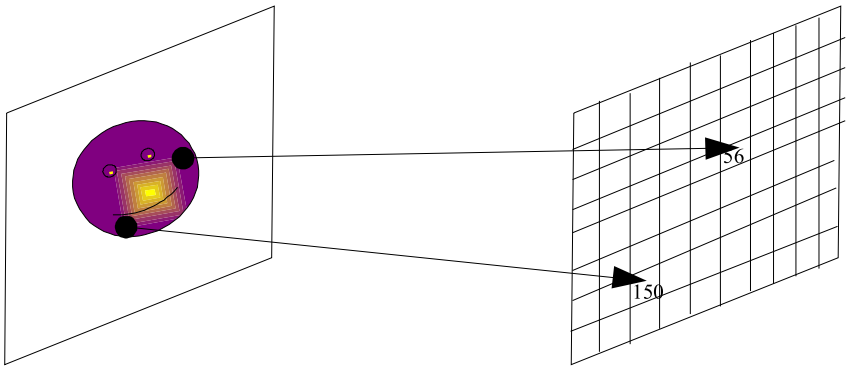
2.2.1.Türk Lirası'ndan 6 Sıfır Atılması ve TL-YTL-TL Geçiş Süreci

28 Ocak 2004'te çıkarılan kanun ile Türk Lirası'ndan 6 sıfır atılmasına karar verilmiş, bu karara dayanarak 1 Ocak 2005'te Yeni Türk Lirası banknotları 8. emisyon grubu olarak piyasaya sürülmüştür.(Ek 1). 1 Ocak 2009'da YTL geçiş sürecinin tamamlanmasının ardından TL banknotları piyasaya sürülmüştür.(Ek 2) 2009 yılı içinde YTL ve TL banknotları tedavülde kalacak, YTL banknotlar 31 Aralık 2009 tarihinde tedavülden kaldırılacaktır.

1http://www.tcmb.gov.tr/yeni/egm/b001000.html Ocak,2009

3.GÖRÜNTÜ İŞLEME

3.1.Görüntünün Elde Edilmesi



Şekil 3.1:Görüntünün sayısallaştırılması

Görüntü işleme işlemi öncelikle görüntünün bilgisayarların anlayabileceği şekilde sayısallaştırılmasıyla başlar. Sayısallaştırma Şekil 3.1'deki gibi matris şeklinde dizilmiş sensörlerin ışık seviyelerini yakalaması, quantalaması ve ışık şiddetlerini sayısal değerler ile ifade etmesiyle olur. (Gonzales ve Woods,2002,s:49)

Bu işlem her bir piksel için uygulandığında elimizde resmin sayısallaştırılmış ve matris şeklinde ifade edilmiş bir karşılığı olur.

3.2.Görüntü Üzerinde Temel İşlemler

3.2.1.Toplama ve Çıkarma:

Toplama ve çıkarma işlemleri resimlerin birbirlerine karşılık gelen piksel değerlerinin toplanması ya da çıkarılmasıyla yapılır. Bu iki işlemin yapılabilmesi, piksellerin birebir eşlenmesi gerekliğinden resimlerin aynı boyutta ve türde olması gerekmektedir. Toplama ve çıkarma işleminin tanımı X satır, Y sütun indisleri kümesi olmak üzere Denklem 2.1'deki gibidir. (Toraman,2006,s:6)

$$\forall i \in X, \forall j \in Y \; I_t(i,j) = I_1(i,j) \mp I_2(i,j) \tag{3.1}$$

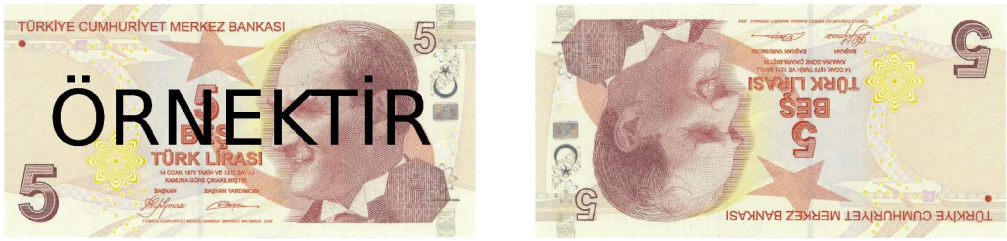
Şekil 3.2'de sıkıştırma uygulanmış bir sandalye resminin orijinaliyle olan farkı görülmektedir.

Şekil 3.2: Bir resmin sıkıştırılmış haliyle arasında olan fark

3.2.2.Resim Döndürme

Resim döndürme bir resmin bir eksen etrafında belli bir Θ açısı ile çevrilmesidir. Resmin döndürülmesi için Denklem 3.2'deki matrisle çarpılması gerekmektedir. Şekil 3.3'te 5 TL'nin normal ve ters (180º) döndürülmüş hali gözükmektedir. (Toraman,2006,s:7)

$$\begin{bmatrix} \cos(\Theta) & \sin(\Theta) \\ -\sin(\Theta) & \cos(\Theta) \end{bmatrix} \tag{3.2}$$



Şekil 3.3:5 TL ve ters çevrilmiş hali

3.2.3. Resim Dönüşümleri

Bir resmin matematiksel temsili $f(x,y)$ gibi iki değişkenli bir fonksiyon şeklindedir. Fonksiyonun çıktı değeri resmin o noktadaki yoğunluğunu verir. Resim işlemede kullanılan belli başlı dönüşümler Fourier dönüşümü, ayrık Fourier Dönüşümü, Ayrık kosinüs Dönüşümü, radon dönüşümü ve mesafe dönüşümleridir. Bir sonraki bu tez çalışmasında kullanılan ayrık kosinüs dönüşümü anlatılacaktır.

3.2.4.Ayrık Kosinüs Dönüşümü

3.2.4.1.Tanımı

Ayrık kosinüs dönüşümü bir resmi değişik genlik ve frekanslardaki sinüs dalgalarının toplamı şeklinde ifade eder. Ayrık kosinüs dönüşümünün en önemli özelliği katsayılarının çok az bir kısmının tipik bir resmin görsel olarak önemli bilgilerini tutmasıdır. Ayrık kosinüs dönüşümü bu özelliğinden ötürü görüntü ve ses sıkıştırma uygulamalarında geniş yer bulmaktadır. Örnek olarak JPEG resim sıkıştırma algoritmasının temelini oluşturmaktadır. Bir MxN matrisin ayrık kosinüs dönüşümü Denklem 3.3'teki gibi tanımlanmaktadır.[2]

$$B_{pq}=\alpha_p\alpha_q\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}A_{mn}\cos\left(\frac{\Pi(2m+1)p}{2M}\right)\cos\left(\frac{\Pi(2n+1)q}{2N}\right),\quad \begin{matrix}0\le p\le M-1\\0\le q\le N-1\end{matrix}$$

$$\alpha_p=\begin{cases}\frac{1}{\sqrt{M}}, & p=0\\ \sqrt{\frac{2}{M}}, & 1\le p\le M-1\end{cases}\qquad \alpha_q=\begin{cases}\frac{1}{\sqrt{N}}, & q=0\\ \sqrt{\frac{2}{N}}, & 1\le q\le N-1\end{cases} \tag{3.3}$$

B_{pq} değerleri ayrık kosinüs dönüşümü katsayıları olarak adlandırılır. Ayrık kosinüs dönüşümü tersinebilir bir dönüşümdür ve tersi de Denklem 3.4'teki gibi tanımlanmaktadır.

$$A_{mn}=\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\alpha_p\alpha_q B_{pq}\cos\left(\frac{\Pi(2m+1)p}{2M}\right)\cos\left(\frac{\Pi(2n+1)q}{2N}\right),\quad \begin{matrix}0\le m\le M-1\\0\le n\le N-1\end{matrix}$$

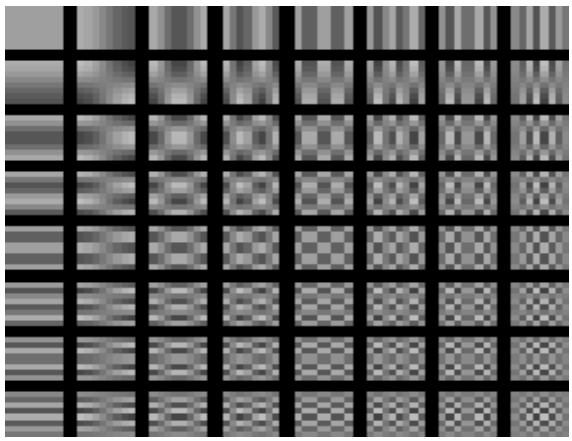
$$\alpha_p=\begin{cases}\frac{1}{\sqrt{M}}, & p=0\\ \sqrt{\frac{2}{M}}, & 1\le p\le M-1\end{cases}\qquad \alpha_q=\begin{cases}\frac{1}{\sqrt{N}}, & q=0\\ \sqrt{\frac{2}{N}}, & 1\le q\le N-1\end{cases} \tag{3.4}$$

Ters ayrık kosinüs dönüşümü fonksiyonu herhangi bir MxN matrisin aşağıdaki şekildeki MN adet fonksiyonun toplamı olarak yazılabileceğini göstermektedir. (Denklem 3.5)

$$\alpha_p\alpha_q\cos\left(\frac{\Pi(2m+1)p}{2M}\right)\cos\left(\frac{\Pi(2n+1)q}{2N}\right)\quad,\quad \begin{matrix}0\le p\le M-1\\0\le q\le N-1\end{matrix} \tag{3.5}$$

Bu fonksiyonlar ayrık kosinüs dönüşümünün baz fonksiyonları olarak adlandırılır. B_{pq} katsayı değerleri ise bu fonksiyonlara uygulanan ağırlıklardır. Şekil 3.4'te 8x8 matrisler için 64 baz fonksiyonu gösterilmiştir.

2 http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html



Şekil 3.4:8x8 matris için baz fonksiyonları

Yatay frekanslar soldan sağa, dikey frekanslar yukarıdan aşağıya gittikçe artar. Sol üst köşedeki sabit değerli baz fonksiyonu ayrık kosinüs baz fonksiyonu, B_{00} katsayısı ayrık kosinüs katsayısı olarak olarak adlandırılır.

3.2.4.2.Ayrık Kosinüs Dönüşüm Matrisi

Ayrık kosinüs dönüşümünün hesaplanmasında iki yöntem vardır. Bunlardan ilki *dct2* fonksiyonunu kullanmaktır. *dct2* büyük girdilerle hızlı hesap yapabilmek için FFT tabanlı bir algoritma kullanmaktadır. İkinci yöntem ise *dctmx* fonksiyonu ile elde edilen ayrık kosinüs dönüşüm matrisini kullanmaktır. Bu yöntem 8x8, 16x16 gibi ufak boyutlu matrislerde etkin çalışmaktadır. Bir MxM matrise ait ayrık kosinüs dönüşüm matrisi Denklem 3.6'daki gibi tanımlanmaktadır:

$$T_{pq}=\begin{cases}\dfrac{1}{\sqrt{M}}, p=0, 0\leqslant p\leqslant M-1 \\ \sqrt{\dfrac{2}{M}}\cos\left(\dfrac{\Pi(2\mathrm{m}+1)p}{2\mathrm{M}}\right), 1\leqslant p\leqslant M-1, 0\leqslant 1\leqslant M-1\end{cases} \tag{3.6}$$

3.2.4.3.Ayrık Kosinüs Dönüşümünün Örüntü Tanımada Kullanımı

Ayrık kosinüs dönüşümü katsayılarının az bir kısmının resmin temel özelliklerini ifade edebilmesi özelliğinden yararlanılarak katsayı matrisinin sol üst üçgenindeki katsayılar kullanılarak resimlere ait özellik vektörü yüksek başarımda ve etkin bir biçimde elde edilebilir.

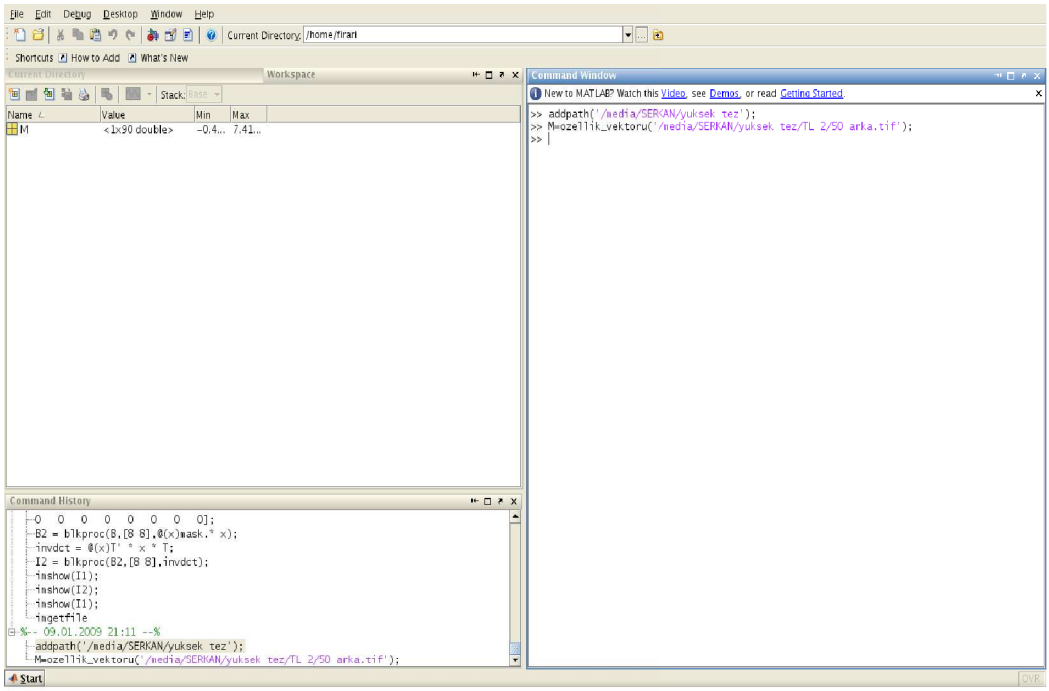
4.MATLAB GELİŞTİRME ORTAMI

MATLAB Mathworks firması tarafından geliştirilen yüksek seviye bir dil ve buna ait geliştirme ortamıdır. Yoğun hesaplama ve matematiksel işlemler gerektiren işlerin C, C++, Fortran gibi dillere göre daha hızlı geliştirilebilmesine ve çalıştırılabilmesine olanak sağlar.

MATLAB pek çok alanda, geniş bir yelpazede hazır fonksiyon kütüphanesine sahiptir. Bunlara finansal, ekonometri, yapay sinir ağları fonksiyonları örnek verilebilir. Bu sayede pek çok işlemin daha az kod yazarak yapılabilmesine olanak sağlar.

4.1.MATLAB Giriş Ekranı

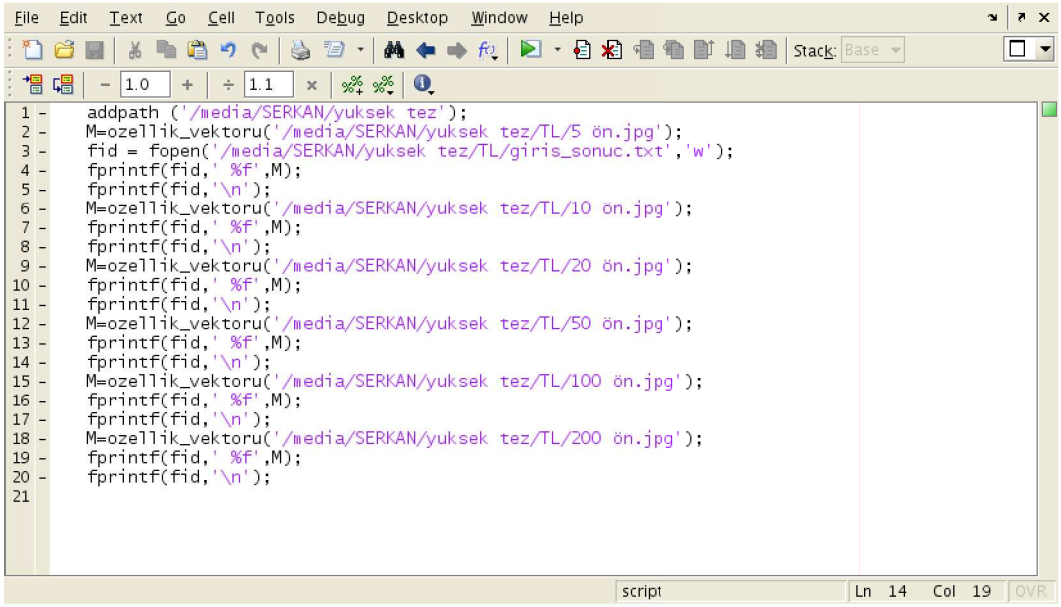
MATLAB açıldığında üç kısımdan oluşan bir ana ekran çıkar. Bu üç kısımdan kodlarımızı yazıp işleteceğimiz "Command Window" yani komut ekranıdır. Workspace" kısmında programların işletimi esnasında oluşan değişkenler yer alır. Bu değişkenlerin değerleri istenirse "Variable Editor" yani değişken düzenleyiciyle görüntülenip düzenlenebilir. Diğer kısım olan "Command History" ise daha önceden çalıştırdığımız kodları kolay erişim için gösteren ekrandır.(Şekil 4.1)



Şekil 4.1:MATLAB Ana ekranı

4.2.MATLAB Kod Düzenleyicisi

MATLAB'da kodlar komut ekranında yazılıp anında işletilebileceği gibi kod geliştirme için ayrılan düzenleyicisi yardımı ile de yapılıp işletilebilir. MATLAB Kod Düzenleyicisi geliştirme ortamlarında standart olan kod renklendirme, derleme, çalıştırma ve hata yakalama imkanlarına sahiptir.(Şekil 4.2)



Şekil 4.2:MATLAB Kod düzenleyicisi

4.3.MATLAB ile Resim İşleme İşlemleri

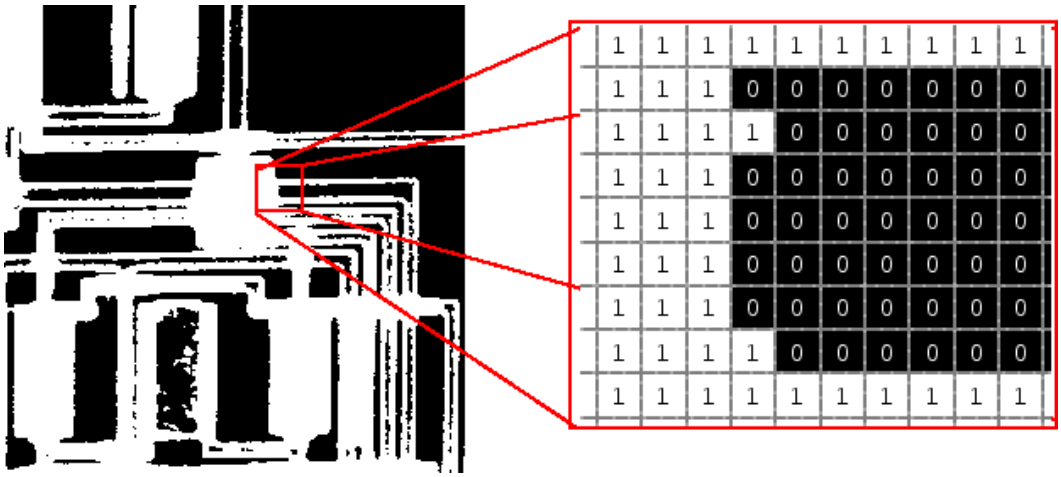
4.3.1.Temel Konular

MATLAB Resim İşleme Araç kutusu ile resimler dosyalardan okunabilir, üzerinde işlem yapılabilir ve işlemler sonucunda oluşan sonuçlar tekrar dosyaya kaydedilebilir ya da ekranda gösterilebilir. MATLAB'da temel veri tipi dizilerdir. Resimler de iki boyutlu dizilerle yani matrislerle ifade edilirler. Matrisin her bir elemanı resmin bir pikselini temsil eder. Örnek olarak 200x300 bir resim MATLAB'da buna karşılık gelen 200x300 bir matris ile ifade edilir. Renkli resimlerde ise renk kanallarını saklamak için üçüncü bir boyuta daha ihtiyaç vardır. Üçüncü boyuttaki düzlemler sırası ile kırmızı, yeşil ve mavi renk kanallarına ait bilgileri tutar.

4.3.2.Resim Tipleri

MATLAB'da resimler dört farklı türdedir. Bunlar ikili, gri tonlamalı, indekslenmiş ve gerçek renk resimlerdir. Son ikisi renkli resim türleridir.[3]

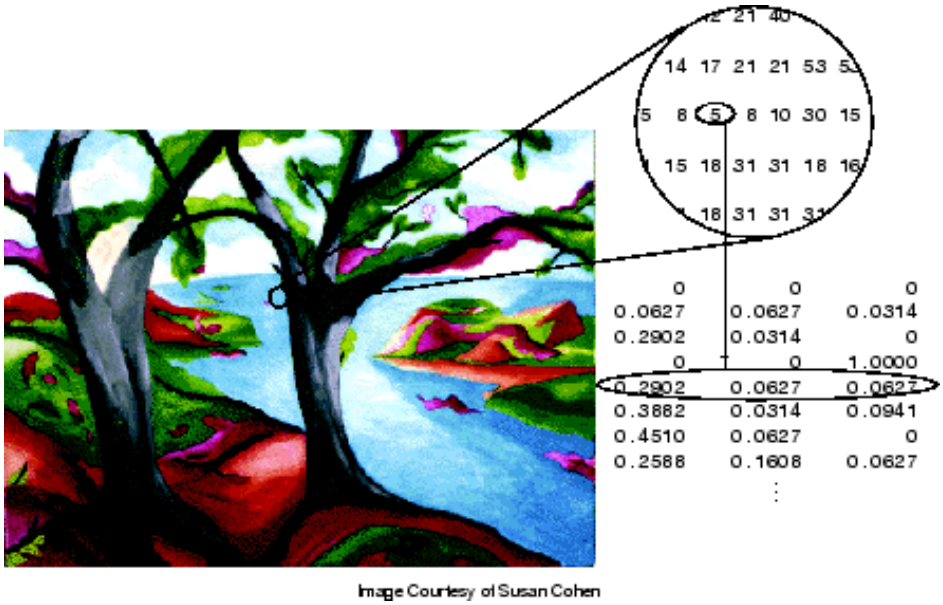
İkili resimlerde her bir piksel 1 ya da 0 değeri ile ifade edilir. 0 siyah (karanlık), 1 ise beyaz (aydınlık) pikselleri tanımlar. Şekil 4.3'te ikili bir resme ait bir alanın bit matrisinde temsil ediliş biçimi görülmektedir.



Şekil 4.3:İkili resmin MATLAB'da temsil edilişi

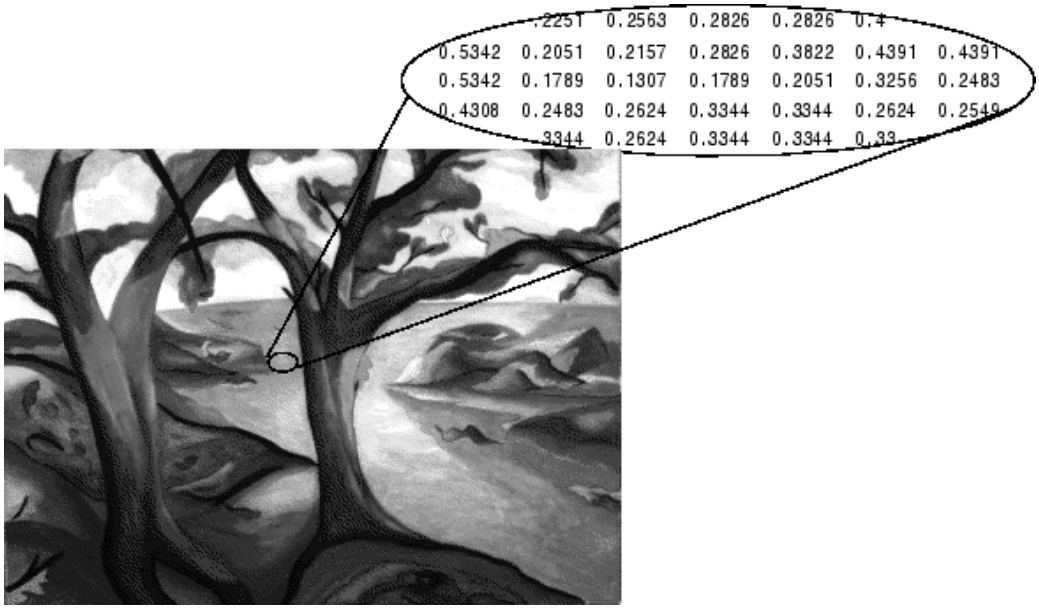
İndeksli resimler bir dizi ve renk haritası matrisinden oluşur. Dizideki piksel değerleri renk haritası matrisinin satır indisleridir. Mx3 olan bu matrisin her bir satırı bir renge ait kırmızı, yeşil ve mavi değerlerini sıfır ile bir değerleri arasında tutar. Şekil 4.4'te indeksli bir resimde bir piksele ait değerin nasıl renk haritasında tutulduğu ve ve bu renge nasıl referans verildiği görülmektedir.

3 http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html



Şekil 4.4:İndeksli resmin MATLAB'da temsil edilişi

Gri tonlamalı resimler matris halinde tutulur ve her bir piksel sıfır ile bir arasındaki bir değerle ifade edilir. Sıfır siyah, bir ise beyaz rengine karşılık gelir. Aradaki değerler ise grinin değişik tonlarını ifade etmek için kullanılır. Şekil 4.5'te gri tonlamalı bir resmin ifade ediliş biçimi görülmektedir.



Şekil 4.5:Gri tonlamalı resmin MATLAB'da temsil edilişi

Gerçek-renkli resimler ise her bir piksel için kırmızı, yeşil ve mavi olmak üzere üç farklı değer tutar. Gerçek-renkli resimlerde renk haritası yoktur. Her pikselin rengi kırmızı, yeşil ve mavi değerlerinin bileşimi ile belirlenir. Şekil dosyaları gerçek-renkli dosyaları her bir renk için 1 byte olmak üzere 24-bitlik resimler olarak tutar. Bu 16 milyon farklı rengin ifade edilebilmesine olanak sağlar. Gerçek renkleri bu hassasiyette ifade edebilmesinden dolayı gerçek-renkli isimlendirmesi kullanılmıştır. Şekil 4.6'da renk kanalları sıfır ile bir değerleri arasında gösterilen bir resim görülmektedir.

Şekil 4.6:Gerçek-renkli resmin MATLAB'da temsil edilişi

4.3.3.Resim Türleri Arasında Dönüşüm

MATLAB'da bu resim türleri arasında dönüşüm yapabilir ve amacımıza en uygun olanını kullanabiliriz. Bir piksel için 24 bit kullandığından renkli resimler örüntü tanıma ve bilgisayarlı görme alanlarında tercih edilmez. Onun yerine gri tonlamalı ya da ikili hale çevrilerek tanıma işlemlerine hazır hale getirilir. Tablo 4.1'de MATLAB'da resim türü çevrim fonksiyonları yer almaktadır. Şekil 4.7 bir paranın taranmış orijinal halini Şekil 4.8 ise *rgb2gray* fonksiyonu ile gri tonlamalı resme çevrilmiş halini göstermektedir.

Tablo 4.1: MATLAB resim türü dönüştürme fonksiyonları

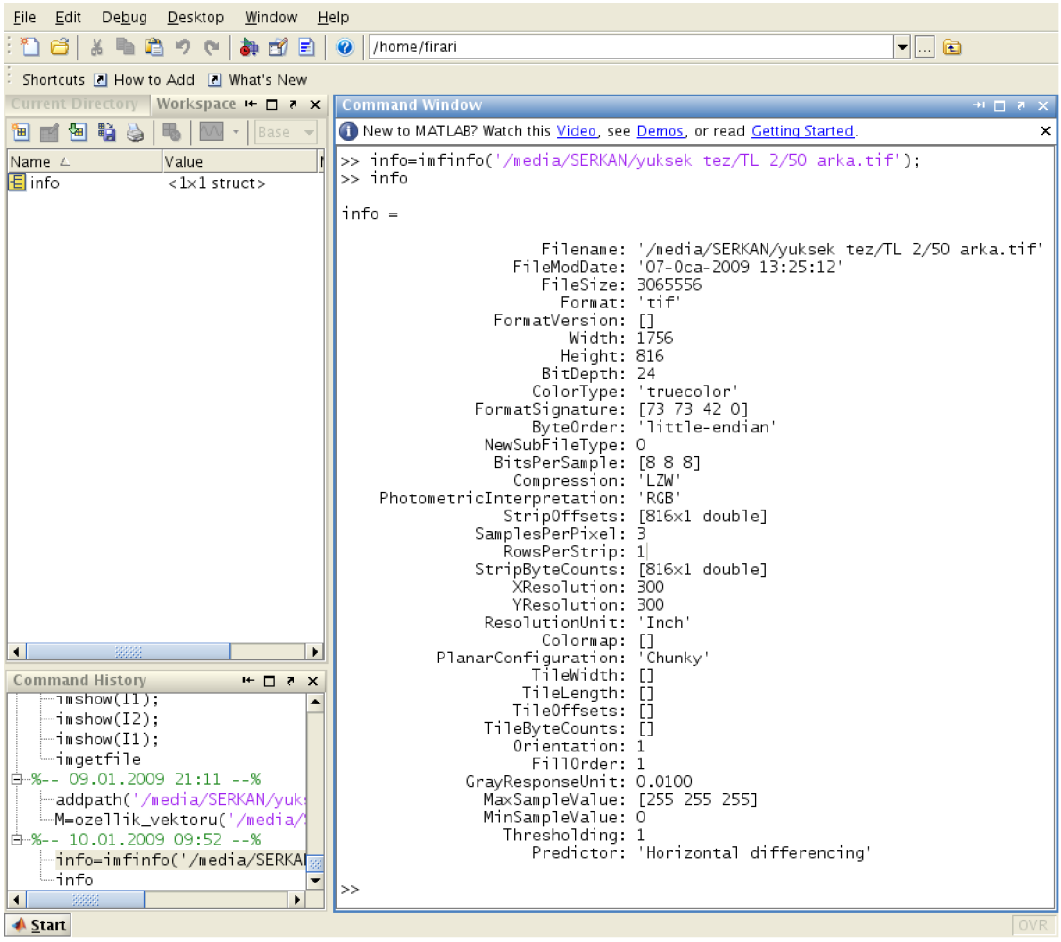
gray2ind	Gri tonlamalı resmi indeksli resme çevirir
grayslice	Gri tonlamalı resmi çok seviyeli eşiikleme ile indeksli resme çevirir
im2bw	Gri tonlamalı, indeksli ya da gerçek-renkli resimleri aydınlık eşiklemesi ile ikili resmi çevirir
ind2gray	İndeksli resmi gri tonlamalı resme çevirir
ind2rgb	İndeksli resmi gerçek-renkli resme çevirir
mat2gray	Matrisi ölçeklemenin ardından gri tonlamalı resme çevirir
rgb2gray	Gerçek-renkli resmi gri tonlamalı resme çevirir
rgb2ind	Gerçek-renkli resmi indeksli resme çevirir

Şekil 4.7:50 TL'lık banknotun orijinal hali

Şekil 4.8:50 TL'lık banknotun gri tonlamalı hali

4.3.4.Resimlerin Okunması ve Yazılması

Resim hakkındaki bilgiler *imfinfo* fonksiyonu yardımı ile gösterilebilir. Şekil 4.9 taranmış para resimlerinden birine ait bilgileri göstermektedir.



Şekil 4.9:50 TL'lık banknotun arka yüzüne ait resmin bilgileri

Resmin bir değişkene okunması *imread* fonksiyonu ile sağlanır. Bir resmin verisinin dosyaya yazılması ile *imwrite* fonksiyonu ile yapılmaktadır. Bir resmi ekranda göstermek içinse *imshow* fonksiyonu kullanılabilir. Bir para resmini okuyup gri tonlamalıya çevirdikten sonra yeniden dosyaya yazan bir program şu şekilde yazılabilir:

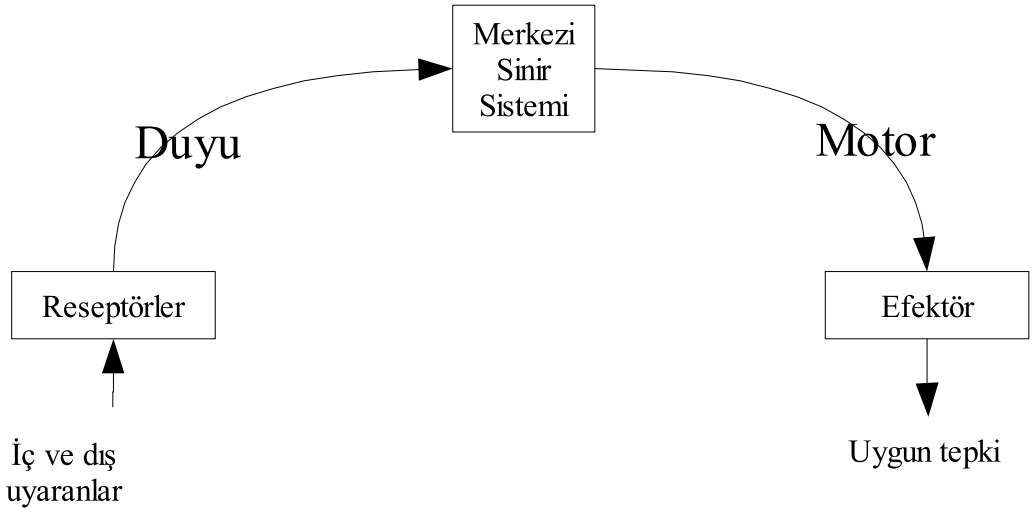
```
rgb=imread('/media/SERKAN/yuksek tez/TL 2/50 ön.tif');
I=rgb2gray(rgb);
imwrite(I,'/media/SERKAN/yuksek tez/50 ön gray.jpg');
```

5.YAPAY SİNİR AĞLARI

YSA, biyolojik sinir ağları esas alınarak geliştirilmiş bir bilgi işleme tekniğidir. Deterministtik olmayan problemlerin çözümünde etkin yöntemler sunar. Yapay sinir ağları özellikle robotik, örüntü tanıma, bilgisayarlı görme gibi alanlarda etkin çözümler sağladığı için geniş yer bulmaktadır. (Kınacı,2006,s:3)

5.1.Biyolojik Sinir Sistemi

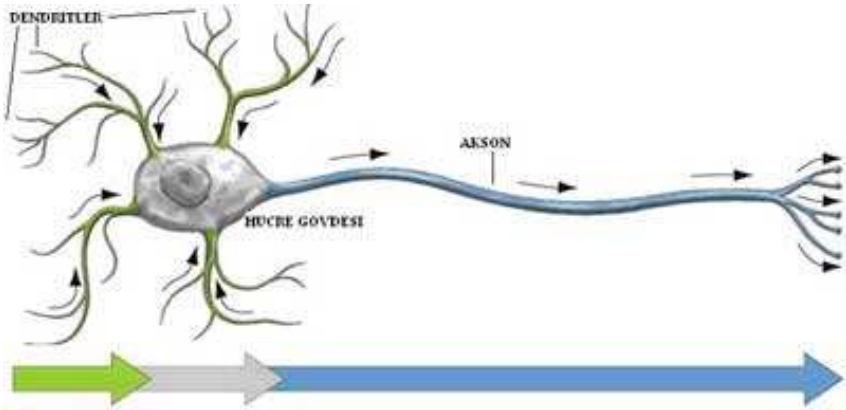
Biyolojik sinir sistemi üç ana bölümden oluşur. Merkezi sinir sistemi (beyin), bilgileri alır, yorumlar ve bir sonuç üretir. Alıcı sinirler (reseptör) organizma içindeki ya da dış dünyadaki uyarıları (soğuk, gürültü vb.) beyne iletilecek elektriksel sinyallere dönüştürür. Bu sinyaller duyu nöronları yardımıyla merkezi sinir sistemine iletilir. Tepki sinirleri (efektör) beyinden aldıkları sinyallerden çıktı olarak tepki (göz kırpma, irkilme vb.) oluştururlar. Merkezi sinir siteminde üretilen sinyalleri efektörlere motor nöronlar taşır.(Şekil 5.1)[4]



Şekil 5.1:Sinir sisteminin genel yapısı

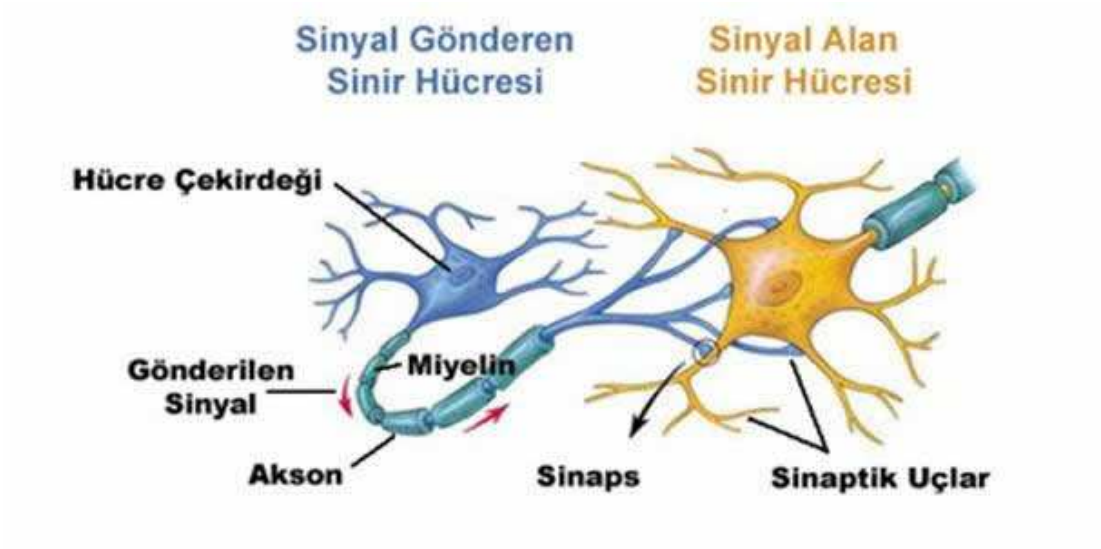
Sinir sisteminin temel yapı taşı sinir hücreleri yani nöronlardır. Nöronlar soma adı verilen hücre gövdesi, sinyalleri dışarıdan hücre gövdesine taşıyan dendritler ve sinyalleri hücre gövdesinden dışarı taşıyan aksonlardan oluşur. Şekil 5.2 bir sinir hücresinin elemanlarını ve sinyalin akış yönünü göstermektedir.

4http://www.aof.anadolu.edu.tr/kitap/EHSM//1211/unite03.pdf Aralık,2009



Şekil 5.2:Bir nöron ve nörondaki sinyal akış yönü

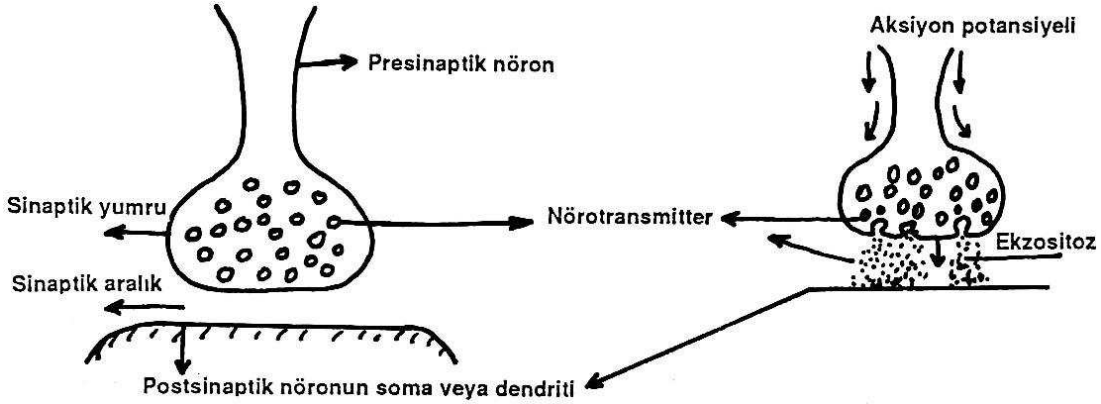
Nöronlar arası sinyal iletimi sinaps adı verilen özel bağlantı bölgelerinde olur. Bir nöronun soması ya da dentritlerinde binlerce bağlantı bulunabilir. Aynı şekilde bir akson pek çok nöronla bağlantı içinde olabilir. Sinaptik bağlantı bölgelerinde de sinyal gönderen nöronun sinaptik yumruları, sinyal alan nöron hücre zarı ile arasında 20nm'lik bir açıklık kalacak şekilde sonlanmaktadır Bu açıklığa sinaptik açıklık denilmektedir. Sinaptik iletiden sorumlu nöro-transmitterler sinyal gönderen nöronun sinaptik yumrusu içinde bulunmaktadır. (Şekil 5.3)



Şekil 5.3:Sinapsın yapısı

Sinyal gönderen nörondaki aksiyon potansiyeli, akson boyunca ilerleyip sinaptik yumrulara ulaştığı zaman, veziküller içindeki nörotransmitterler, ekzositoz ile sinaptik aralığa boşalır, bunu takiben nörotransmitterler, sinyal alan nöron zarında

bulunan kendilerine özel reseptörlere bağlanarak, sinyal alan nöronu ya uyarırlar ya da uyarmazlar (inhibisyon). Uyardıkları zaman, aksiyon potansiyeli sinyal alan nöronun aksonu boyunca taşınmaya devam eder. Eğer inhibisyon söz konusu ise sinyal alan nöron uyarılmaz ve sinirsel ileti bu noktada kesintiye uğrar. Sinyal alan nöronun uyarılması veya inhibe edilmesi sinyal gönderen nörondan salıverilen nörotransmittere bağlıdır.(Şekil 5.4)



Şekil 5.4:Sinapstaki sinyal alışverişi

5.2. Yapay Sinir Ağlarının Tarihçesi

1943'te Warren McCulloch ve Walter Pitts nöronların çalışma biçimiyle ilgili bir makale yayınladılar. Beyindeki nöronların nasıl çalıştığını göstermek için basit bir elektronik devre modeli tasarlamışlardır.

1949 yılında Donald Hebb “The Organization of Behavior” adlı kitabı yazdı. Bu kitabında sinir yollarının kullanıldıkça güçlendiğini gösterdi. Bu insanların öğrenmesinin en önemli sırrıydı. Ayrıca eğer iki nöronun aynı anda tetiklenmesinin sinyali güçlendirdiğini iddia etti.

1950'lerde bilgisayarların da gelişmesiyle birlikte varsayımsal bir yapay sinir ağının simülasyonu mümkün oldu. IBM Araştırma Laboratuvarları'nda çalışan Nathanial Rochester bu konuda ilk çalışmayı yaptı ancak başarı gösteremedi. 1959'da Bernard Widrow ve Marcian Hoff Stanford üniversitesinde "ADALINE" ve "MADALINE" adlı iki model geliştirdiler. Bunlardan ilki telefon hattından gelen bitlere göre sonraki biti tahmin etmeye yönelik bir çalışma idi. Diğeri ise uyarlamalı ağlar ile telefon hattındaki yankıları temizlemek gibi bir amaca hizmet etmekteydi.

1962'de Widow ve Hoff hatanın komşu perceptronlara dağıtılmasıyla ilgili bir

yöntem geliştirdiler. Buna göre tek bir perceptron yüksek oranda hata içerse bile ağırlık değerlerini ayarlarsa bu hatayı tüm ağa dağıtabilirdi. Bu şekilde zaman içinde kendini düzeltebilecekti.

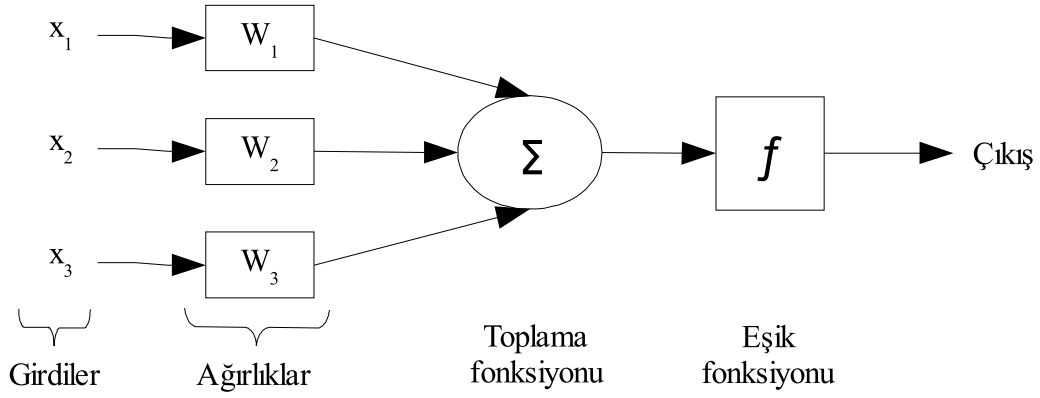
Bu gelişmelerin ardından çok katmanlı ağlar ile ilgili bir makale yayınlansa da geleneksel von Neumann mimarisi bilgisayar dünyasına hakim oldu ve yapay sinir ağlarının gerişimi bunun gerisinde kaldı.

1972'de Kohonen ve Anderson birbirinden bağımsız bir ağ üzerinde çalıştılar. Matris matematiği kullanan bu ağlar aslında ADALINE devreleri dizisinden ibaretti. Ancak 1975'te ilk olarak bir çok katmanlı öğreticisiz ağ geliştirilebildi.

1982'de John Hopfield o ana kadar geliştirilen ağların aksine çift yönlü bağlantı içeren ağlarla ilgili bir makale çıkardı. Reilly ve Cooper ise hibrit çok seviyeli ağlar ile ilgili çalışma yaptı. 1986'da David Rumelhart geriye yayılım temeline dayanan ağları buldu. Bu ağlar örüntü tanımada hatanın geri bildirimini ve düzeltilmesini sağlamaktadır. Ancak çok katmanlı oldukları için eğitilmeleri çok fazla iterasyon gerektirmektedir.[5]

5.3. Yapay Sinir Ağının Temel Elemanı: Nöron

Tıpkı biyolojik sinir ağı gibi yapay sinir ağlarının da temel yapı taşı nöronlardır. Yapay sinir ağlarında nöron kendine gelen sinyalleri toplayıp belli bir eşik değerine ulaşmış ise bağlı bulunan nöronlara ileten bir işlem elemanıdır. Girdiler toplanırken girdinin türü, tetikleyici ya da inhibitör, ve ağırlıkları dikkate alınır. YSA bu işlem elemanlarının katmanlar halinde bir araya getirilmesiyle oluşur.(Şekil 5.5)



Şekil 5.5:YSA'ndaki bir nöronun yapısı

5http://cse.stanford.edu/class/sophomore-college/projects-00/neural-networks/History/history1.html Ocak,2009

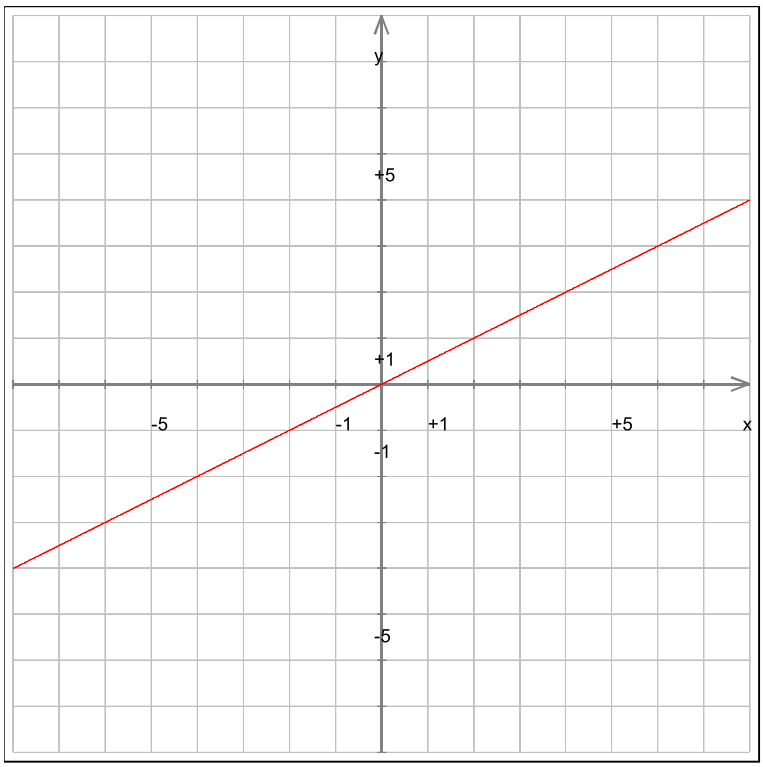
Girdiler dışarıdan olabileceği gibi çok katmanlı ağlarda diğer nöronlar olabilir. Ağırlık değerleri Çıktının düzeltilebilmesini dolayısıyla öğrenmeyi sağlar. Yani yapay sinir ağlarının hafızası ağırlık değerleridir. Toplam fonksiyonu genellikle girdilerin ağırlıklı toplamı olarak alınır. Ancak ağırlık ve girdi değerlerini farklı şekilde harmanlayarak toplama fonksiyonları da oluşturmak mümkündür. (Denklem 5.1)

$$\sum x_i \cdot y_i = x_1 \cdot w_1 + x_2 \cdot w_3 + x_1 \cdot w_3 + \ldots \tag{5.1}$$

Eşik fonksiyonları tanım kümesine karşılık belli aralıklarda değerler üreten fonksiyonlardır. Geniş ölçüde kullanılan beş adet fonksiyon vardır.

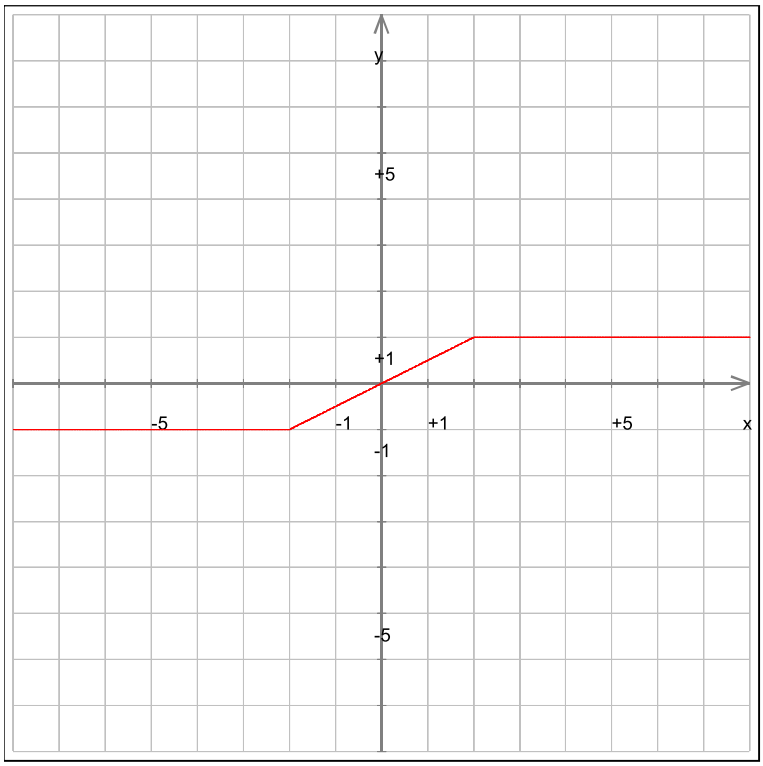
Şekil 5.6'da grafiği görülen lineer fonksiyon girdi değerini aktivasyonunu ayarlayan bir katsayıyla çarparak çıkış üretir. (Denklem 5.2)

$$f(x) = \alpha \cdot x \tag{5.2}$$



Şekil 5.6:Lineer fonksiyon grafiği

Lineer fonksiyonun bir benzeri Şekil 5.7'de grafiği görülen rampa eşik fonksiyonudur. Rampa fonksiyonu girdi değerini negatif ve pozitif belirli bir aralıkta sınırlandırır.

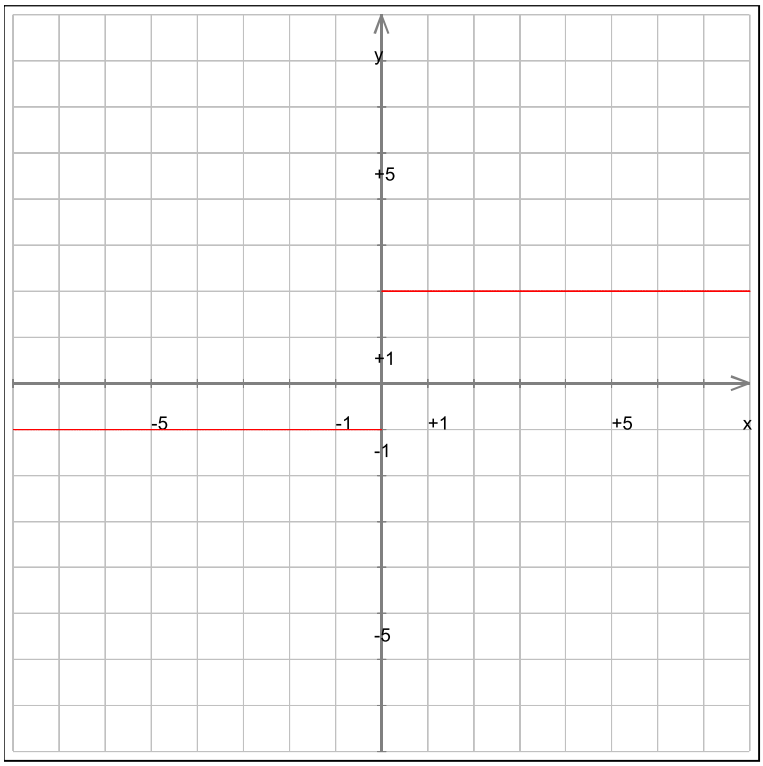


Şekil 5.7:Rampa fonksiyonu grafiği

Bu sınır değerleri doyma noktası olarak adlandırılır.(Denklem 5.3)

$$f(x)=\begin{cases}+\tau, x\geq\tau\\ x, |x|<\tau\\ -\tau, x\leq\tau\end{cases} \tag{5.3}$$

Şekil 5.8'de grafiği görülen basamak eşik fonksiyonu girdiye göre iki çıkış üretir. (Denklem 5.4)

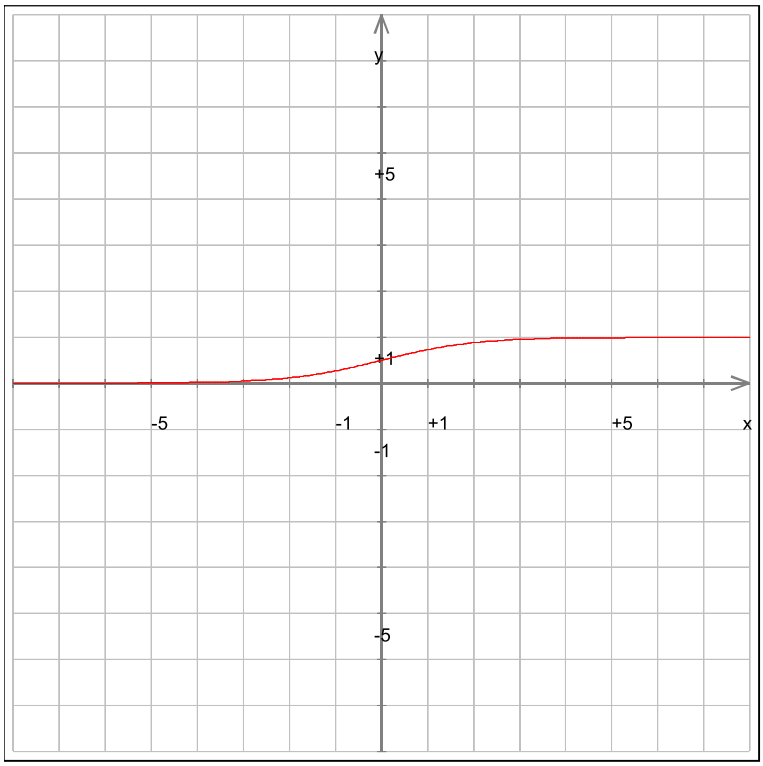


Şekil 5.8:Basamak fonksiyonu grafiği

$$f(x)=\begin{cases} +\tau, x>0 \\ x, -\delta, x\leq 0 \end{cases} \tag{5.4}$$

Şekil 5.9'da grafiği görülen sigmoid fonksiyonu sınırlı ve doğrusal olmayan bir fonksiyondur. Çıkış değerleri 0 ya da 1 değerine yakınsar. (Denklem 5.5)

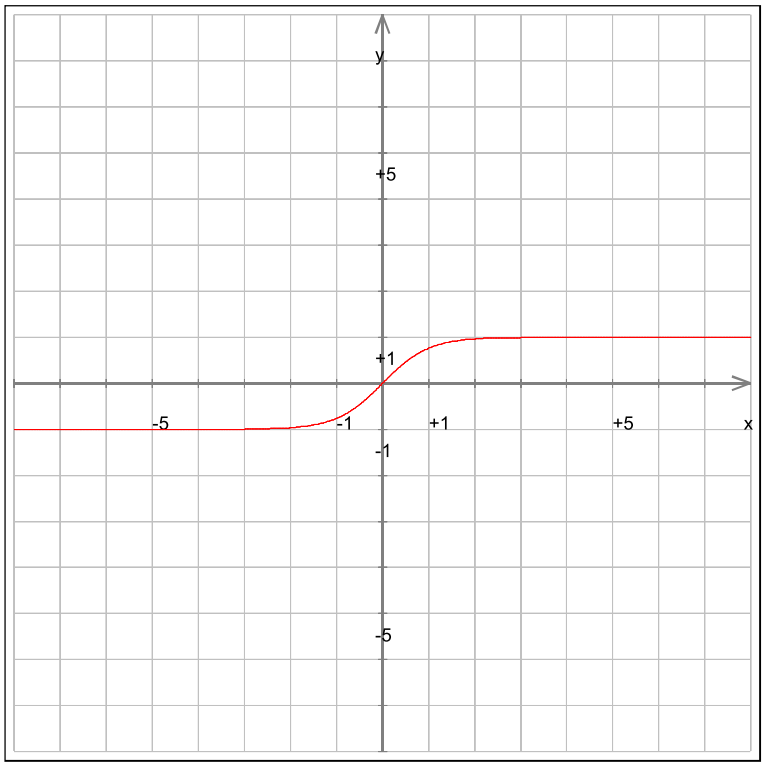
$$f(x)=\frac{1}{1+e^{-x}} \tag{5.5}$$



Şekil 5.9:Sigmoid fonksiyonu

Şekil 5.10'da grafiği görülen hiperbolik tanjant fonksiyonu da sigmoid ile benzer özellikler göstermektedir. Ancak -1 ile 1 değerleri arasında çıktı üretir. (Denklem 5.6)

$$\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}=\frac{e^{2x}-1}{e^{2x}+1} \tag{5.6}$$



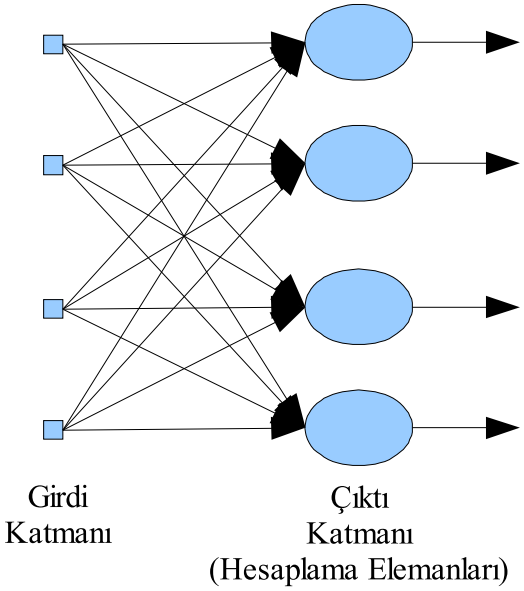
Şekil 5.10:Hiperbolik tanjant fonksiyonu

5.4. Yapay Sinir Ağı Mimarileri

Yapay sinir ağları yapısına, yani nöronların bağlantı şekillerine, göre üç temel sınıfa ayrılır. (Kınacı,2006,s:9)

5.4.1. Tek Katmanlı İleri Beslemeli Ağlar

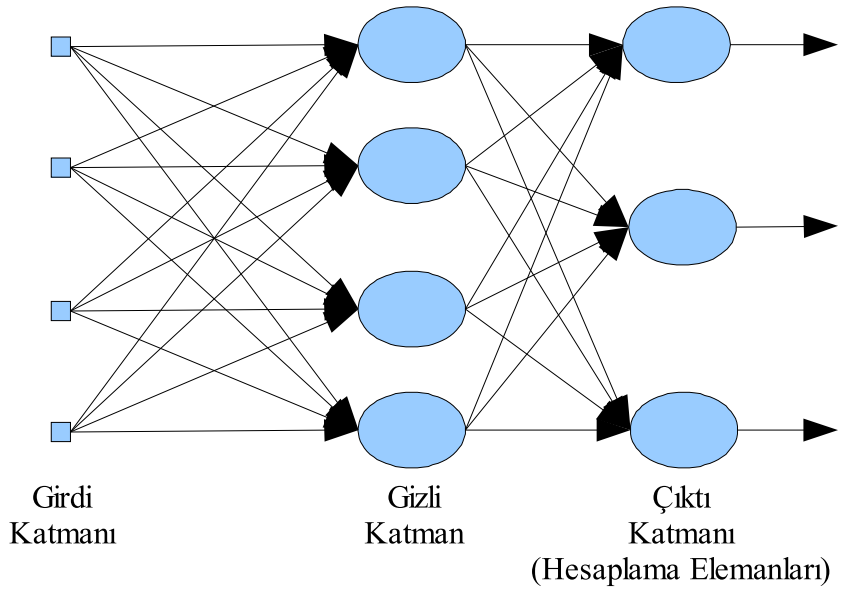
Şekil 5.11'de görülen bu ağ mimarisinde girdiler direkt olarak çıkış nöronlarına bağlıdır. Bağlantılar tek yönlü ve sadece ileri doğrudur. Girdi katmanında hesap yapılmadığı için, sadece çıkış nöronları hesaba katılarak tek katmanlı adını almışlardır. (Kınacı,2006,s:10)



Şekil 5.11:Tek katmanlı ileri beslemeli bir ağın yapısı

5.4.2.Çok Katmanlı İleri Beslemeli Ağlar

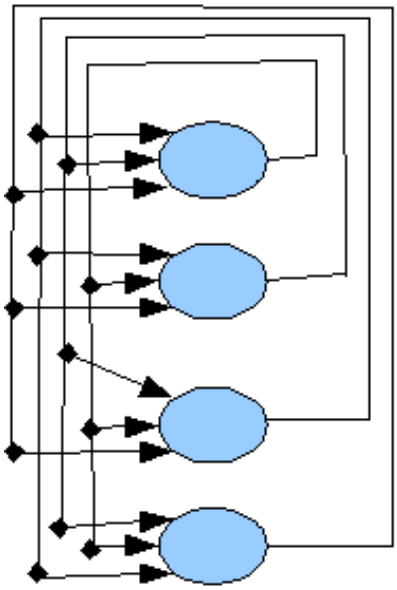
Şekil 5.12'de görülen bu ağ mimarisinde bir ya da daha çok gizli katman vardır. Gizli katmanlardaki nöronlar da gizli nöronlar olarak adlandırılır. Bu katmanlar özellikle girişlerin fazla olduğu durumda daha hassas sonuç elde etmede çok faydalıdır. Giriş katmanından gelen veriler gizli katmanlarda çıktı üretir ve bu çıktılar ilerideki katmanlara bağlanır. Çoğunlukla bir nöronun bağlantısı kendisinden önceki Ya da sonraki katmandaki nöronlarla olur. Eğer her bir düğüm sonraki katmandaki tüm düğümlerle bağlı ise ağ tam bağlı ağ olarak tanımlanır. Eğer bağlardan bir kısmı eksik ise bunlara da kısmi bağlı ağlar denir. (Kınacı,2006,s:10)



Şekil 5.12:Çok katmanlı ileri beslemeli bir ağın yapısı

5.4.3.Geri Beslemeli Ağlar

Şekil 5.13'te görülen geri beslemeli ağlarda nöronların çıktıları önceki katmanlardaki nöronlara ya da nöronun kendisine girdi olabilir. Geri beslemeli ağlar ileri beslemeli ağlarda olduğu gibi bir ya da birden çok gizli katman içerebilir. Geri besleme sayesinde hem ağın öğrenme kabiliyeti artar hem de ağ daha etkin bir biçimde işler. (Kınacı,2006,s:11)



Şekil 5.13:Geri beslemeli bir ağın yapısı

5.5.Öğrenme Kuralları

5.5.1.Hebb Kuralı

Donald Hebb 1949'da yazdığı kitabında ortaya attığı savı temel alarak Hebb kuralı oluşturulmuştur. Bu kurala göre eğer bir sinaps ile birbirine bağlı iki nöron aynı anda aktifleşirse sinapsın gücü yükselir, farklı anlarda sinapsın gücü azalır. (Stent,1973,s:997;Changeux ve Danchin,1976,s:705)

Bu kural temel olarak bir önceki nöronun çıktısı ve sonraki nöronun girdisinin bir fonksiyonu olarak Denklem 5.7'deki gibi formülize edilebilir.

$$\Delta w_{ij} = f(y_k(n), x_j(n)) \tag{5.7}$$

Basit Hebb kuralında bu fonksiyon aşağıdaki gibi lineer bir fonksiyondur ve ε katsayısı öğrenme hızı olarak adlandırılır. (Denklem 5.8)

$$\Delta w_{ij}(n) = \epsilon\, y_k(n)\, x_j(n) \tag{5.8}$$

Hebb kuralının bir türevi de kovaryans hipotezidir. Bu öğrenme kuralında Denklem 5.9'da görüldüğü gibi anlık sinyal değerlerinin yanında bu değerlerin belli bir geçmişe kadar olan ortalaması da kullanılır. (Haykin,199,s:57)

$$\Delta w_{ij}(n) = \epsilon(y_k(n) - \overline{y_k(n)})(x_j(n) - \overline{x_j(n)}) \tag{5.9}$$

5.5.2.Algılayıcılar

Algılayıcılar şekil X deki temel sinir hücresini temel alan öğrenme kurallarıdır. Toplayıcı fonksiyon ağırlıklı toplam alır. Eşik fonksiyonu ise Denklem 5.10'daki signum (işaret) fonksiyonudur. (Kınacı,2006,s:16)

$$f(x)=\begin{cases} 1, x>0 \\ 0, x=0 \\ -1, x<0 \end{cases} \tag{5.10}$$

5.5.2.1.Algılayıcı Yakınsama Teoremi ve Algılayıcı Öğrenmesi

Eğer bütün eğitim verileri için doğru sonucu üreten bir ağırlık vektörü varsa, algılayıcılı öğrenme algoritması bütün eğitim verileri için doğru sonuç üreten bir ağırlık vektörüne yakınsar. (Kınacı,2006,s:16)

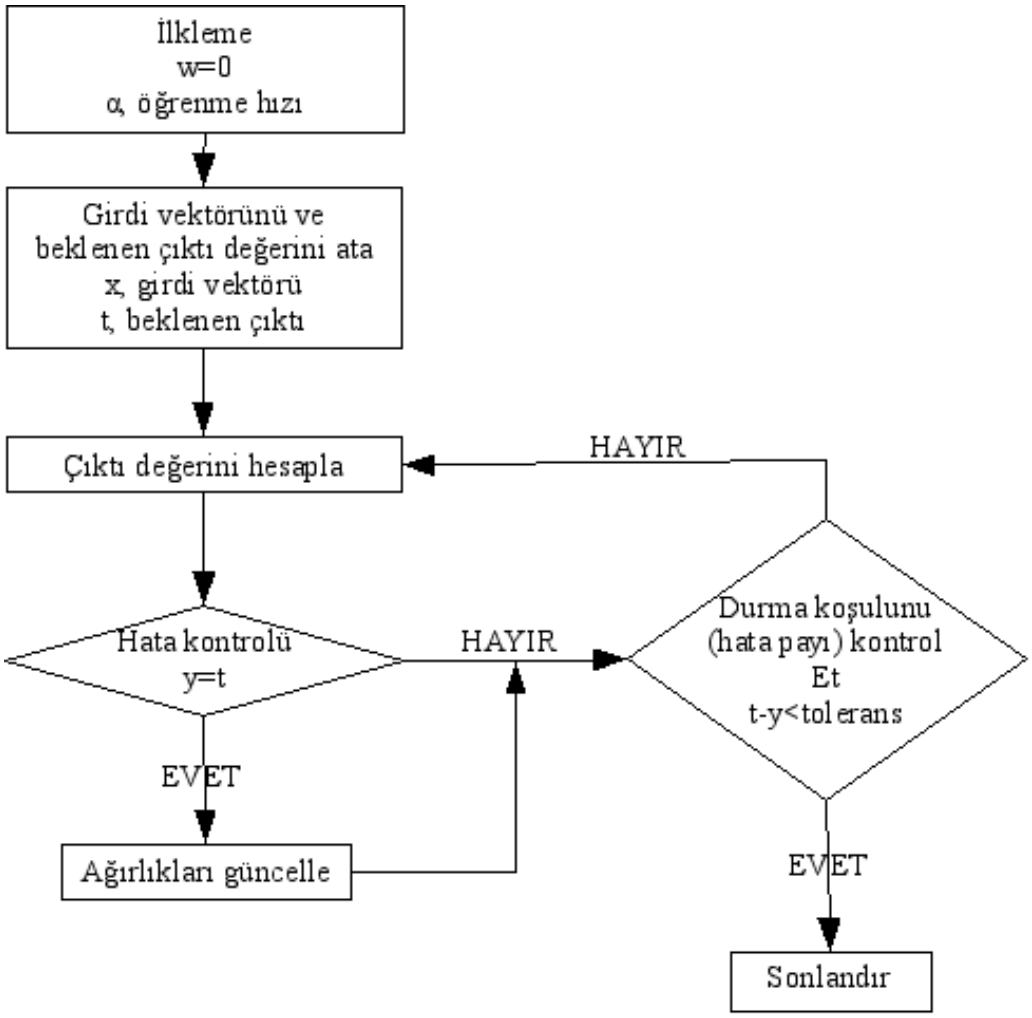
Eğitim süreci ağırlıkların ilk değerlerinin verilmesiyle başlar. Daha sonra her adımda girdi değerleri ağırlıklı olarak toplanır ve işaret fonksiyonu uygulanarak çıktı değeri elde edilir. (Denklem 5.11)

$$y=sgn(\sum_{i=1}^{N} w_i \cdot x_i) \tag{5.11}$$

Çıktı beklenen değere eşit değilse, bir diğer deyişle hata mevcut ise, bir sonraki iterasyon için yeni ağırlık değerleri Denklem 5.12'deki formülle hesaplanır.

$$w_i(n+1)=w_i(n)+\alpha \cdot (t-y) \cdot x_i \tag{5.12}$$

Daha sonra elde edilen hata tolerans değeriyle karşılaştırılır. Eğer hata tolerans değerinden düşük ise eğitim sonlandırılır. Eğer hata toleranstan büyükse yeni ağırlık değerleri ile bir sonraki iterasyona geçilir.(Şekil 5.14)



Şekil 5.14:Algılayıcı öğrenme algoritması akış şeması

5.5.3.Delta Kuralı

Delta kuralı, ya da bir diğer adıyla en küçük karesel hata algoritması aşağıda tanımlanan $E(\vec{w})$ hata fonksiyonunun (Denklem 5.13) anlık değerlerini kullanan ve bu değeri minimuma indirmeye yönelik bir algoritmadır. (Haykin, 1999,s:128)

$$E(\vec{w})=\frac{1}{2}e^2(n) \tag{5.13}$$

Bu denklemde e(n), n anındaki hata değerini tanımlamaktadır ve Denklem 5.14'teki formülle hesaplanmaktadır.

$$e(n)=d(n)-\vec{x}^T(n)\vec{w}(n) \tag{5.14}$$

Denklem 5.14'ün $\vec{w}$ ağırlık vektörüne göre kısmi türevi alındığında, Denklem 5.15 elde edilir.

$$\frac{\partial e(n)}{\partial \vec{w}} = -\vec{x}(n) \tag{5.15}$$

Denklem 5.13'ün $\vec{w}$ ağırlık vektörüne göre kısmi türevi alındığında, Denklem 5.16 elde edilir.

$$\frac{\partial E(\vec{w})}{\partial \vec{w}} = e(n)\frac{\partial e(n)}{\partial \vec{w}} \tag{5.16}$$

Bu denklemlerden yola çıkarak Denklem 5.17'deki eşitlik yazılabilir.

$$\frac{\partial E(\vec{w})}{\partial \vec{w}} = -\vec{x}(n)e(n) \tag{5.17}$$

Delta kuralı Denklem 5.17'deki eşitliği gradyan vektörünün bir kestirimi, tahminlemesi olarak kullanır. (Denklem 5.18)

$$\hat{\vec{g}} = -\vec{x}(n)e(n) \tag{5.18}$$

Denklem 5.18 kullanılarak bir sonraki iterasyona ait ağırlık vektörü Denklem 5.19'daki gibi hesaplanır.

$$\vec{w}(n+1) = \vec{w}(n) + \epsilon\vec{x}(n)e(n) \tag{5.19}$$

5.5.4.Hatanın Geriye Yayılma Algoritması

Hatanın geriye yayılması algoritması çok katmanlı ileri beslemeli ağlarda ağın ortama karesel hatasını minimuma çekecek şekilde ağırlıkları güncelleyen iteratif bir algoritmadır. (Kınacı,2006,s:24) Ağırlıkların güncellenmesi, yani eğitimde genelleştirilmiş delta kuralını kullanır. Algoritmada herhangi bir çıkış hücresi j için n'inci iterasyonda, yani ağa n'inci eğitim seti beslendiğinde, hata Denklem 5.20'deki gibi tanımlanır.

$$e_j(n) = d_j(n) - y_j(n) \tag{5.20}$$

Bu denklemde $d_j(n)$ beklenen değeri, $y_j(n)$ oluşan değeri ifade etmektedir. Çıkıştaki bir nöronun hata fonksiyonu Denklem 5.21'deki gibi, ağın hata fonksiyonu, E(n), C çıkış nöronlarının kümesini ifade etmek üzere Denklem 5.22'deki gibi tanımlanır.

$$\frac{1}{2}e_j^2(n) \tag{5.21}$$

$$E(n)=\sum_{j\in C}\frac{1}{2}e_j^2(n) \tag{5.22}$$

Ortalama karesel hata ise N eğitim seti boyutu olmak üzere Denklem 5.23'teki gibi olarak tanımlanır.

$$\frac{1}{N}\sum_{n=1}^{N}E(n) \tag{5.23}$$

Ağın eğitimi sonunda hedeflenen bu karesel hata değerini minimuma indirmektir.

Her bir çıkış nöronuna gelen girdi sinyali toplamı, bir önceki ara katmandaki çıktı toplamı, $v_j(n)$ ağırlıklı toplamlardan oluşur. (Denklem 5.24)

$$v_j(n)=\sum_{i=0}^{m}w_{ij}(n)\,y_i(n) \tag{5.24}$$

Buradan yola çıkarak j'inci çıktı katmanındaki j'inci nöronda oluşan değer, $y_j(n)$, $f(x)$ eşik fonksiyonu olmak üzere, Denklem 5.25'teki gibi hesaplanır.

$$y_j(n)=f_j(v_j(n)) \tag{5.25}$$

Genelleştirilmiş delta kuralına benzer şekilde hatanın geriye yayılması algoritması da Denklem 5.26'daki kısmi türevle doğru orantılı bir düzeltme uygular.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} \tag{5.26}$$

Denklem 2.26 zincir kuralına göre açtığımızda Denklem 5.27 elde edilir.

$$\frac{\partial E(n)}{\partial w_{ji}(n)}=\frac{\partial E(n)}{\partial e_j(n)}\cdot\frac{\partial e_j(n)}{\partial y_j(n)}\cdot\frac{\partial y_j(n)}{\partial v_j(n)}\cdot\frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{5.27}$$

Denklem 5.22'nin $e_j(n)$'e göre türevi alınırsa Denklem 5.28, Denklem 5.20'nin $e_j(n)$'e göre türevi alınırsa Denklem 5.29, Denklem 5.25'in $e_j(n)$'e göre türevi alınırsa Denklem 5.30 elde edilir.

$$\frac{\partial E(n)}{\partial w_{ji}(n)}=e_j(n) \tag{5.28}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)}=-1 \tag{5.29}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = f_j'(v_j(n)) \tag{5.30}$$

Bu elde edilen denklemler yerine konduğunda Denklem 5.31 elde edilir.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) f_j'(v_j(n)) y_i(n) \tag{5.31}$$

Delta kuralında ε öğrenme katsayısı parametresi olmak üzere, $w_{ij}(n)$ ağırlığına uygulanan düzeltme $\Delta w_{ij}(n)$ Denklem 5.32'deki gibi tanımlanmıştır.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\epsilon \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{5.32}$$

Denklem 5.32'den yola çıkarak geriye yayılma algoritmasındaki düzeltme miktarı Denklem 5.33'teki gibi hesaplanır.

$$\Delta w_{ij}(n) = \epsilon \cdot \delta_j(n) \cdot y_i(n) \tag{5.33}$$

Bu denklemdeki $\delta_j(n)$ yerel gradyan olarak adlandırılır ve Denklem 5.34 kullanılarak hesaplanır.

$$\delta_j(n) = \frac{\partial E(n)}{\partial v_j(n)}$$

$$\delta_j(n) = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)}$$

$$\delta_j(n) = e_j(n) \cdot f_j'(v_j(n)) \tag{5.34}$$

Yerel gradyan ağırlıklarda yapılması gereken düzeltmeyi gösterir.

5.5.4.1.Hatanın Geriye Yayılma Algoritmasının İşleyişi

İlk olarak ağırlık ve sapma değerleri seçilir. Daha sonra her bir eğitim turunda ağa girdi verileri beslenir. Ara katmanlar sırası ile geçilerek bir önceki ara katmandan gelen çıktıları ağırlıklandırarak toplanır. (Denklem 5.35)

$$v_j^{(l)}(n) = \sum_{i=0}^{m} w_{ij}^{(l)}(n) y_i^{(l-1)}(n) \tag{5.35}$$

Bu denklemde $y_i^{(l-1)}(n)$ n'inci iterasyonda bir önceki katmandaki i'nci nöronun oluşturduğu çıktı değeridir. Daha sonra bu değer eşik fonksiyonundan geçirilerek nöronun çıktı değeri hesaplanır. (Denklem 5.36)

$$y_j^{(l)}(n) = f_j(v_j^{(l)}(n)) \tag{5.36}$$

Çıkış katmanındaki nöronlar için hata değerini hesaplanır. (Denklem 5.37)

$$e_j(n) = d_j(n) - y_j^{(L)}(n) \tag{5.37}$$

L= katman sayısı (ağ derinliği), $d_j(n)$=j'inci beklenen değer.

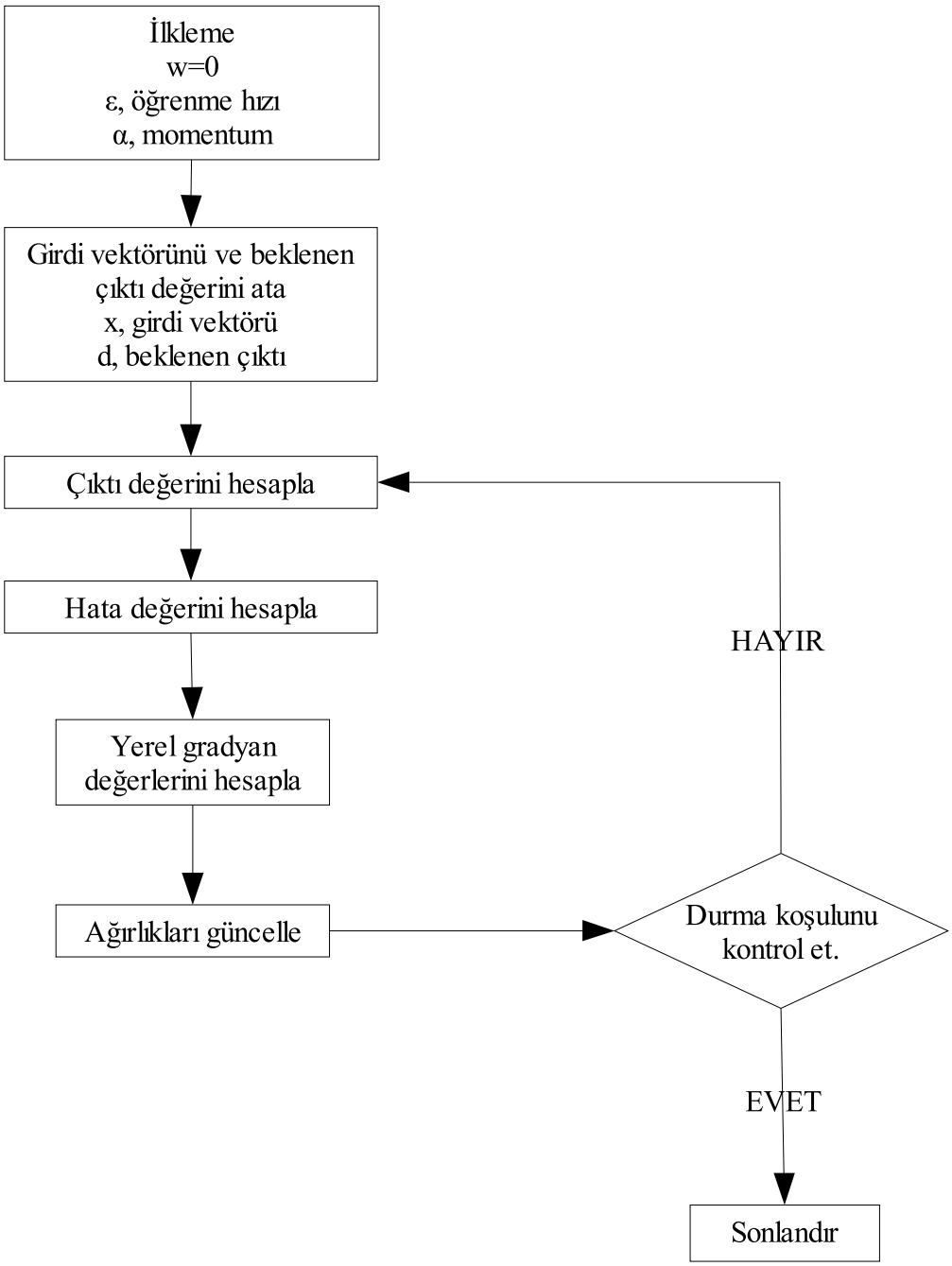
Gizli katmanlar ve çıkış katmanları için yerel gradyan değerleri hesaplanır. (Denklem 5.38)

$$\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n) \cdot f_j'(v_j^{(L)}(n)), & L \text{ çıkış katmanındaki nöron için} \\ f_j'(v_j^{(L)}(n)) \cdot \sum_k \delta_k^{(l+1)}(n) \cdot w_{kj}^{(l+1)}(n), & l\text{'inci ara katmandaki nöron için} \end{cases} \tag{5.38}$$

Gradyan değerleri kullanılarak bir sonraki iterasyonda kullanılmak üzere yeni ağırlık değerleri hesaplanır. (Denklem 5.39)

$$w_{ji}^{(L)}(n+1) = w_{ji}^{(L)}(n) + \alpha \cdot w_{ji}^{(L)}(n-1) + \epsilon \cdot \delta_j^{(l)}(n) \cdot y_i^{(l-1)}(n) \tag{5.39}$$

Algoritma durma koşulu sağlanıncaya kadar iteratif olarak işletilir.(Şekil 5.15)



Şekil 5.15:Hatanın geriye yayılması algoritması akış şeması

5.5.5.Rekabetçi Öğrenme

Rekabetçi öğrenme algoritmaları sınıflandırma işlemi gibi çıkış bilgilerinin belli olmadığı problemlerde kullanılır. Bu algoritmalarda çıkış nöronları belli sonucu üretmeyi hedef almak yerine aktif olma mücadelesi içine girerler. Nöronlardan hepsi değil sadece bir tanesi sonuç üretebilir. Bir nöronun çıktı üretebilmesi ona gelen girdilerin ağırlıklı toplamının, diğer nöronlardan büyük olması gereklidir. Bir

nöronun eşik fonksiyonu Denklem 5.39'daki formüle göre oluşmaktadır. (Haykin,1999,s:58)

$$y_k = \begin{cases} 1, \nu_k > \nu_j \; \forall \; j, \; j \neq k \\ 0, diğer\; koşullarda \end{cases} \tag{5.39}$$

Bir sonraki eğitim iterasyonu için her nöron kendine gelen aktif sinyale ait ağırlıkları günceller, diğerlerini değiştirmez. (Denklem 5.40)

$$\Delta w_{kj} = \begin{cases} \epsilon (x_j - w_{kj}), eğer\; k\; nöronu\; kazanan\; ise \\ 0, eğer\; k\; nöronu\; kaybeden\; ise \end{cases} \tag{5.40}$$

6.GELİŞTİRİLEN UYGULAMA

6.1.MATLAB ile Özellik Vektörü Çıkarma Programı

Bu tez çalışmasında geliştirilen uygulamada örüntü tanımada ayrık kosinüs dönüşümü metodu kullanılmıştır. Paralar öncelikle tarayıcı yardımıyla sayısal resim olarak bilgisayara aktarılmıştır. Daha sonra her bir para resmi gri tonlamalı resme çevrilmiştir ve işlemleri kolaylaştırmak amacıyla boyutu 24x24 kareler olarak küçültülmüştür. Bu resimlere 8x8 er bloklar halinde ayrık kosinüs dönüşümü uygulanmış ve oluşan katsayılardan her bir blok için sol üst köşedeki düşük frekanslı 10 adet katsayı seçilmiştir. Bu şekilde her bir para resmi için 90 adet katsayı oluşturulmuştur. Ancak bir paranın iki yüzünün de tanınması gerektiğinden her iki yüze ait özellik vektörleri de oluşturulmuştur.

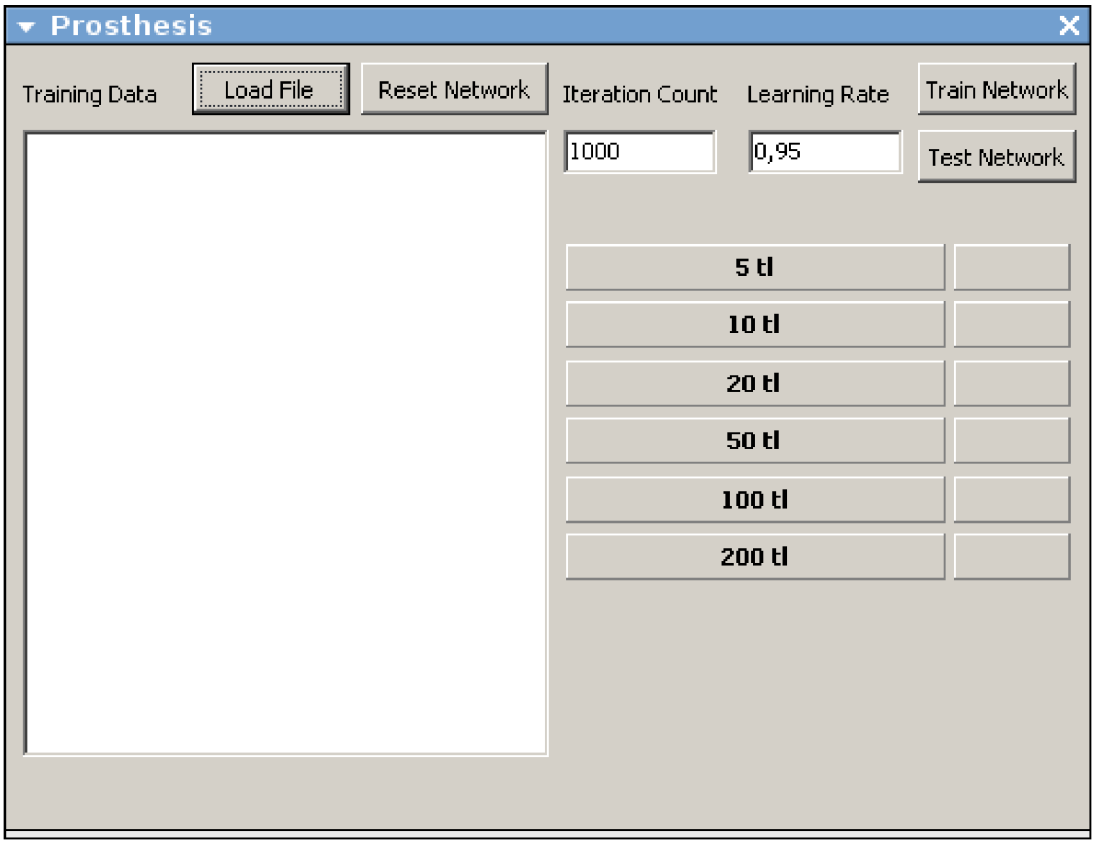
6.2.MATLAB ile Görsel Özellik Vektörü Çıkarma Programı

Bu uygulama yukarıdaki tek bir para resmi için özellik vektörü oluşturan ve ara adımlardaki gri tonlamalıya çevirme ve küçültme işlemlerini gösteren bir uygulamadır. Şekil 6.1'de 200 TL'nin ön yüzü seçildikten sonraki halidir.

Şekil 6.1:Görsel arayüz ile özellik vektörü çıkaran uygulamadan bir görünüm

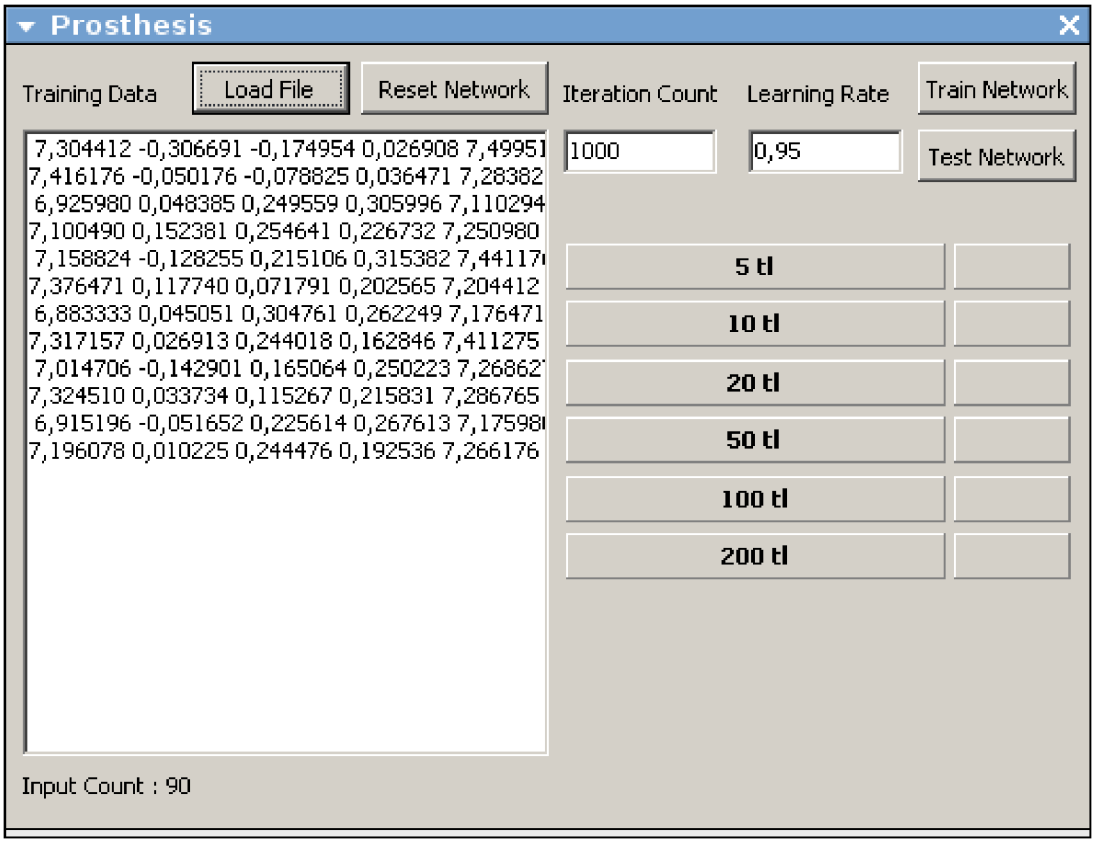
6.3.Prosthesis Programı ile Yapay Sinir Ağı Eğitimi ve Testi

1) Öncelikle ann.txt dosyasına çıkış bilgileri yazılır. Bu şekilde çıktılara anlamlı isimler üretmesi sağlanır.(Şekil 6.2)



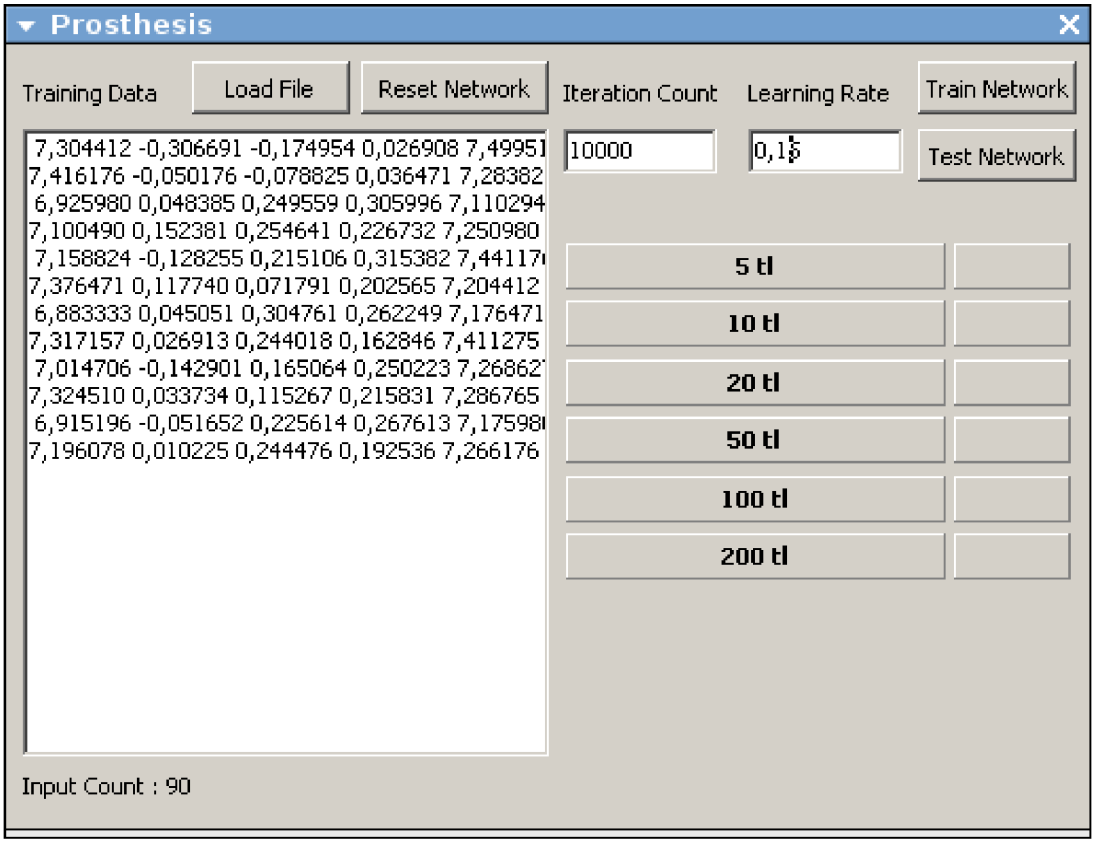
Şekil 6.2:Prothesis programının ilk açılış hali

2) Daha sonra MATLAB ile oluşturulan eğitim verileri yüklenir. (Şekil 6.3)



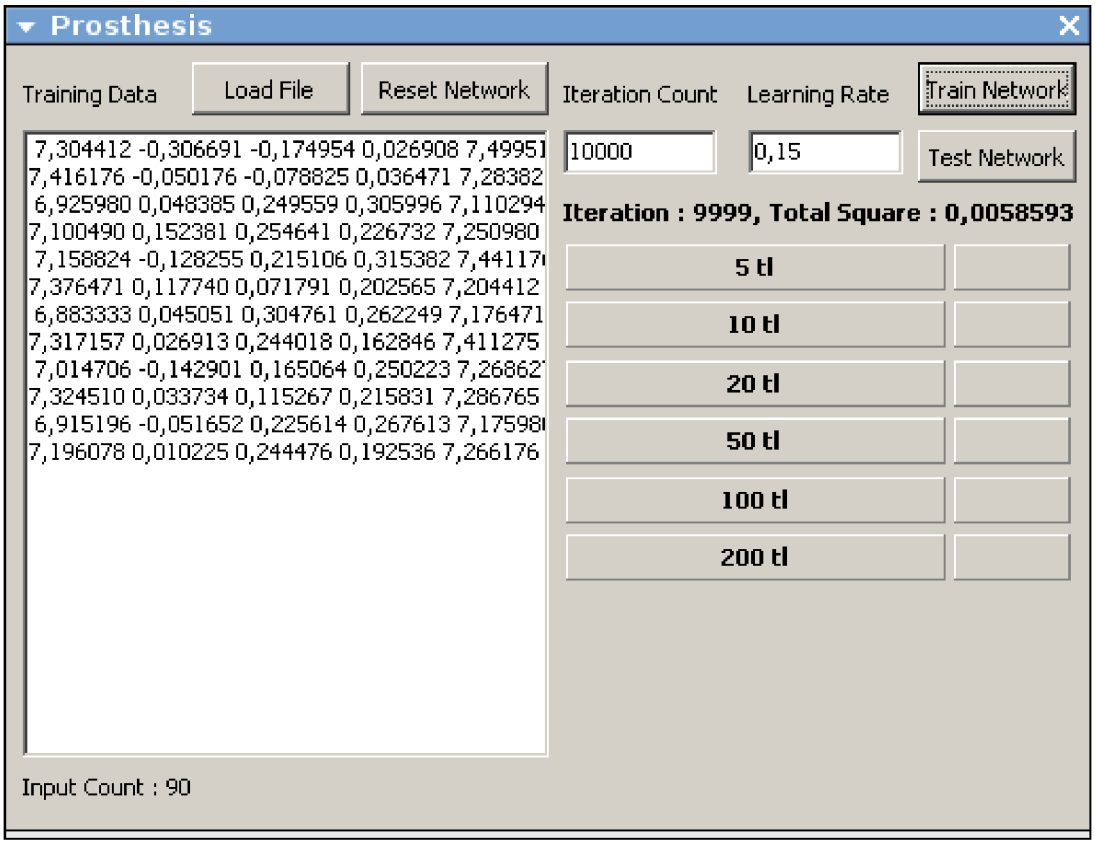
Şekil 6.3:Eğitim verileri yüklenmiş Prototthesis programı

3) İterasyon sayısı ve öğrenme oranı ayarlanır. Paralara ait verilerde iterasyon sayısı olarak 10000, öğrenme oranı olarak 0,15 makul bir karesel hataya yakınsamaktadır. (Şekil 6.4)



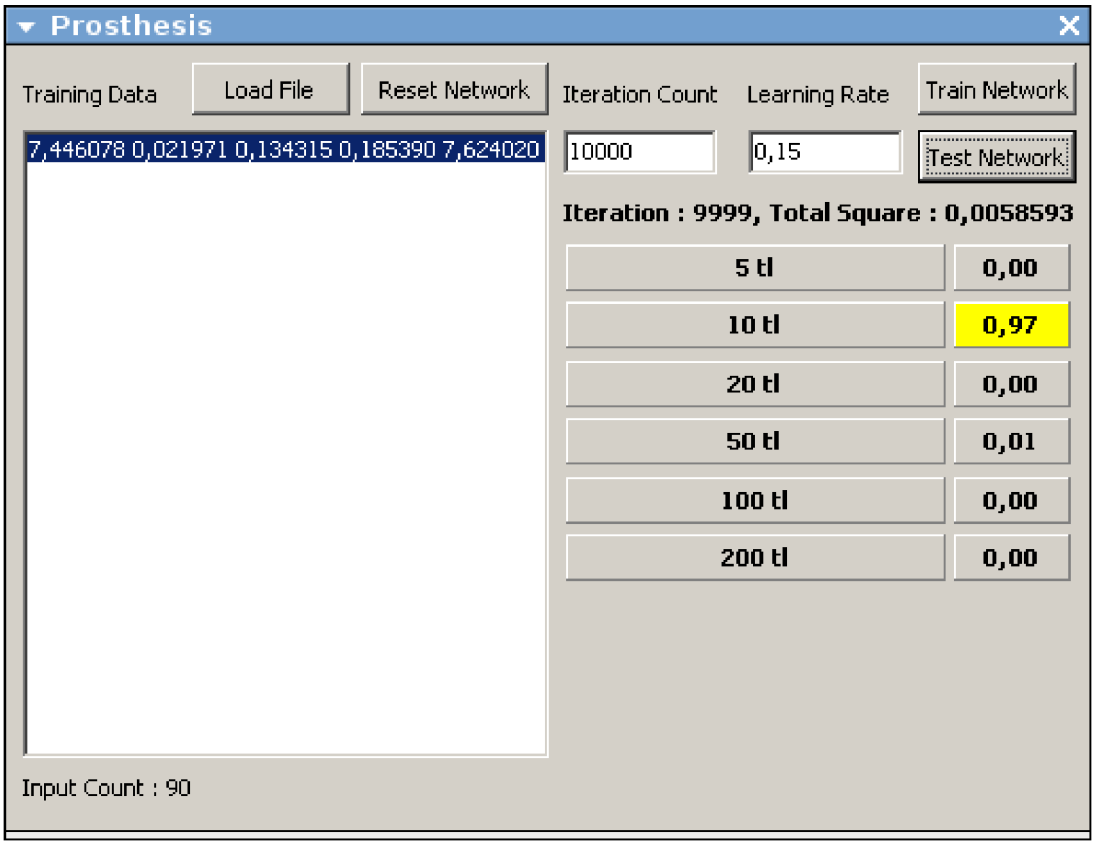
Şekil 6.4:Eğitim parametreleri ayarlanmış Prothesis programı

4) Tüm bu işlemler yapıldıktan sonra ağımız eğitime hazırdır. Eğitim tamamlandıktan sonra program o anki karesel hatayı vermektedir. (Şekil 6.5)



Şekil 6.5:Eğitim işlemi tamamlanmış Prothesis programı

5) Yine MATLAB ile başka bir taranmış para metnine ait özellik vektörüyle oluşturulan test dosyası programa yüklenir. Örnek olarak 10 TL ile test yapıldıktan sonra program bu resmin %97 oranında 10 TL'ye ait olduğunu belirlemiştir.(Şekil 6.6)



Şekil 6.6:Sınama anında Prothesis programı

SONUÇ

Bu tez çalışmasında geliştirilen uygulama ayrık kosinüs dönüşümü kullanarak basit ve etkin bir şekilde örüntü tanıma işlemleri için gereken özellik vektörünü elde edebilmiştir. Para resimleri ile sistem eğitilmiş ve ikinci bir set para setiyle sistem sınanmış ve başarı sağlanmıştır.

Bu çalışmada geliştirilen uygulamanın hassasiyeti, paranın gözle fark edilebilen ya da fark edilemeyen görsel özellikleri de incelenerek arttırılabilir. Özellikle sahte paraların tespitinde bu büyük bir fayda sağlayacaktır.

KAYNAKLAR

CHANGEUX, J.P. ve DANHCHIN A.; "Selective stabilisation of developing synapses as a mechanism for the specification of neuronal networks". Nature, vol.264. Aralık, 1976.

GONZALEZ, R. C. ve WOODS, R. E.; Digital Image Processing. Prentice Hill International,2002

HAYKIN, Simon; Neural Networks: A Comprehensive Foundation, Prentice Hill International,1999.

KINACI, Ahmet Cumhur; "Görsel Yazılım Geliştirme Ortamı ile Beraber Bir Yapay Sinir Ağı Kütüphanesi Tasarımı ve Gerçekleştirimi", Yüksek Lisans Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, İzmir, 2006.

STENT, G.S; "A Physiological Mechanism for Hebb's Postulate of Learning". Proceedings of the National Academy of Sciences, vol.70. ABD, Nisan, 1973.

TORAMAN, Suat; "Histopatolojik İmgelerin Değerlendirilmesinde Örüntü Tanıma Temelli Karar Destek Sistemleri ", Yüksek Lisans Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ, 2006.

Diğer Kaynaklar

http://www.aof.anadolu.edu.tr/kitap/EHSM//1211/unite03.pdf Aralık,2008

http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html Aralık 2008

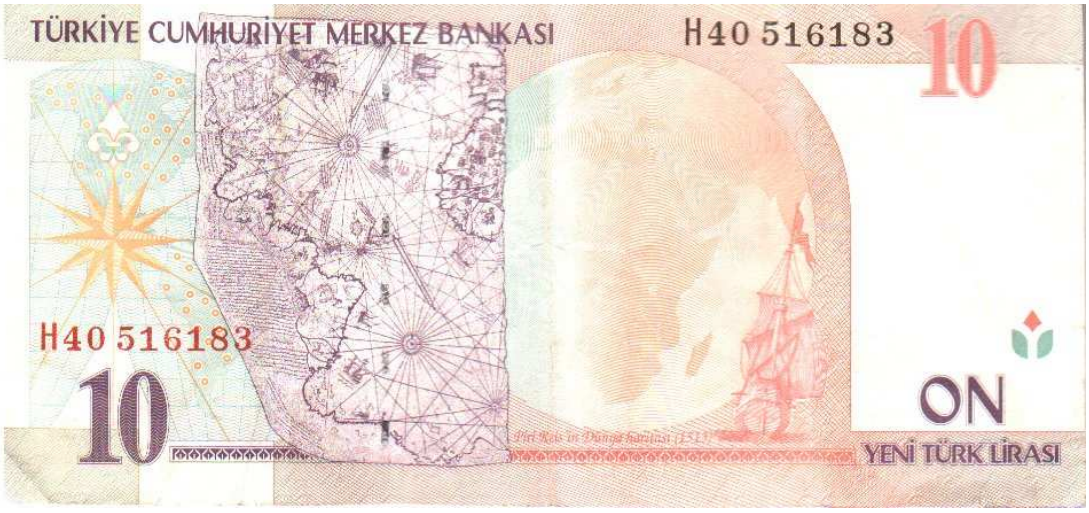
http://cse.stanford.edu/class/sophomore-college/projects-00/neural-networks/History/history1.html Ocak,2009

http://www.tcmb.gov.tr/yeni/egm/b001000.html Ocak,2009

EKLER

EK 1: YTL Banknot Resimleri


10
TÜRKİYE CUMHURİYET MERKEZ BANKASI
ON
YENİ TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211
SAYILI KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
10


TÜRKİYE CUMHURİYET MERKEZ BANKASI
H40 516183
10
H40 516183
10
ON
YENİ TÜRK LİRASI


TÜRKİYE CUMHURİYET MERKEZ BANKASI
20
YİRMİ
YENİ TÜRK LİRASI
20


TÜRKİYE CUMHURİYET MERKEZ BANKASI
I20 478447
20
20
I20 478447
YİRMİ
YENİ TÜRK LİRASI


50
TÜRKİYE CUMHURİYET
MERKEZ BANKASI
50
ELLİ
YENİ TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
2005
50
TÜRKİYE CUMHURİYET MERKEZ BANKASI BANKNOT MATBAASI


TÜRKİYE CUMHURİYET MERKEZ BANKASI
F08 956496
50
KAPADOKYA
50
F08 956496
50
ELLİ YENİ TÜRK LİRASI


100
TÜRKİYE CUMHURİYET
MERKEZ BANKASI
100
YÜZ
YENİ TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
100


TÜRKİYE CUMHURİYET MERKEZ BANKASI
C65 237932
100
C65 237932
İSHAKPAŞA SARAYI - DOĞUBAYAZIT
100
YÜZ
YENİ TÜRK LİRASI

EK 2: TL Banknot Resimleri


TÜRKİYE CUMHURİYET MERKEZ BANKASI
ÖRNEKTİR
10
ON
TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
TÜRKİYE CUMHURİYET MERKEZ BANKASI BANKNOT MATBAASI 2009


TÜRKİYE CUMHURİYET MERKEZ BANKASI
A076 740010
ÖRNEKTİR
10
Ord. Prof. Dr.
CAHİT ARF
1910 - 1997
A076 740010
ON TÜRK LİRASI


TÜRKİYE CUMHURİYET MERKEZ BANKASI
ÖRNEKTİR
YİRMİ
TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR
BAŞKAN
BAŞKAN YARDIMCISI
TÜRKİYE CUMHURİYET MERKEZ BANKASI BANKNOT MATBAASI 2009


TÜRKİYE CUMHURİYET MERKEZ BANKASI
A 103 133559
ÖRNEKTİR
MİMAR KEMALEDDİN
1870 - 1927
A 103 133559
YİRMİ TÜRK LİRASI


TÜRKİYE CUMHURİYET MERKEZ BANKASI
50
ÖRNEKTİR
ELLİ
TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
TÜRKİYE CUMHURİYET MERKEZ BANKASI BANKNOT MATBAASI 2009


TÜRKİYE CUMHURİYET MERKEZ BANKASI
A 155 041650
50
ÖRNEKTİR
FATMA ALİYE
1862 - 1936
50
A 155 041650
ELLİ TÜRK LİRASI
50


TÜRKİYE CUMHURİYET MERKEZ BANKASI
100
ÖRNEKTİR
YÜZ
TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
100
100


TÜRKİYE CUMHURİYET MERKEZ BANKASI
A067 824681
100
ÖRNEKTİR
ITRİ
(Buhurizade Mustafa Efendi)
1640 - 1712
100
A067 824681
YÜZ TÜRK LİRASI
100


TÜRKİYE CUMHURİYET MERKEZ BANKASI
200
ÖRNEKTİR
İKİYÜZ
TÜRK LİRASI
14 OCAK 1970 TARİH VE 1211 SAYILI
KANUNA GÖRE ÇIKARILMIŞTIR.
BAŞKAN
BAŞKAN YARDIMCISI
200


TÜRKİYE CUMHURİYET MERKEZ BANKASI
A030 494901
200
ÖRNEKTİR
Sevelim Sevilelim
YUNUS EMRE
1238 - 1320
200
A030 494901
İKİYÜZ TÜRK LİRASI
200

EK 3: MATLAB ile Özellik Vektörü Çıkarma Programı Kaynak Kodları

ozellik_vektoru.m

```
function [V] = ozellik_vektoru(dosya)
I = imread(dosya);
B = rgb2gray(I);
I = imresize(B, [24 24]);
I = im2double(I);
T = dctmtx(8);
dct = @(x)T * x * T';
B = blkproc(I,[8 8],dct);
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],@(x)mask.* x);
M=reshape(B2,1,24*24);
V=M(M ~= 0);
```

egitim_dosyasi.m

```
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/5 ön.jpg');
fid = fopen('/media/SERKAN/yuksek tez/TL 2/giris.txt','w');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/5 arka.jpg');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/10 ön.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/10 arka.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/20 ön.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/20 arka.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/50 ön.tif');
fprintf(fid,' %f',M);
```

```
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/50 arka.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/100 ön.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/100 arka.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/200 ön.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
M=ozellik_vektoru('/media/SERKAN/yuksek tez/TL 2/200 arka.tif');
fprintf(fid,' %f',M);
fprintf(fid,'\n');
fclose(fid);

M=ozellik_vektoru('/media/SERKAN/yuksek tez/tl/10 arka.jpg');
fid=fopen('/media/SERKAN/yuksek tez/ann/test.txt','w');
fprintf(fid,'%f ',M);
fprintf(fid,'\n');
fclose(fid);
```

EK 4: MATLAB ile Görsel Özellik Vektörü Çıkarma Programı Kaynak Kodları

paragui.m

```
function varargout = paragui(varargin)
% PARAGUI M-file for paragui.fig
%      PARAGUI, by itself, creates a new PARAGUI or raises the existing
%      singleton*.
%
%      H = PARAGUI returns the handle to a new PARAGUI or the handle to
%      the existing singleton*.
%
%      PARAGUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PARAGUI.M with the given input arguments.
%
%      PARAGUI('Property','Value',...) creates a new PARAGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before paragui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to paragui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help paragui

% Last Modified by GUIDE v2.5 12-Jan-2009 20:46:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @paragui_OpeningFcn, ...
                   'gui_OutputFcn',  @paragui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before paragui is made visible.
function paragui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to paragui (see VARARGIN)
yuklenmedi = imread('/media/SERKAN/yuksek tez/yuklenmedi.gif');
axes(handles.axes1);
imshow(yuklenmedi);
axis off
axes(handles.axes2);
imshow(yuklenmedi);
axis off
axes(handles.axes3);
axis off
handles.tablo=uitable('Position',[49,6,652,58],'Data',[]);
% Choose default command line output for paragui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes paragui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = paragui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filename=imgetfile;
backgroundImage = imread(filename);
```

```
axes(handles.axes1);
imshow(backgroundImage);
axis off

gray=rgb2gray(backgroundImage);
axes(handles.axes2);
imshow(gray);
axis off

kucuk=imresize(gray,[24 24]);
axes(handles.axes3);
image(kucuk);
axis off
handles.tablo=uitable('Position',[49,6,652,58],'Data',ozellik_vektoru(filename));
```

ÖZGEÇMİŞ

9 Haziran 1984'te Bornova, İzmir'de dünyaya geldim. İlkokulu Selçuk Yaşar Alaybey İlköğretimokulu'nda (1990-1995), ortaokul ve lise öğrenimimi Özel İzmir Amerikan Lisesi'nde (1995-2002) tamamladım. 2002-2006 yılları arasında lisans eğitimi Ege Üniversitesi Bilgisayar Mühendisliği bölümünde aldım. Yüksek lisans eğitimimi ise 2007-2009 yılları arasında Haliç Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği ana bilim dalında yaptım.

Halen bir bankada yazılım uzmanı olarak Sybase veritabanı yönetim sistemi üstünde yazılım geliştirmekteyim.