OBJELERİN BÖLÜTLENMİŞ GÖRÜNTÜLERİ KULLANILARAK YAPAY SİNİR AĞLARIYLA TANINMASI

ATİLLA DEMİRAY

Yüksel Lisans Tezi

Elektrik-Elektronik Mühendisliği Anabilim Dalı

TEMMUZ - 1998

T.C.
ANADOLU ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

Atilla Demiray tarafından hazırlanan 'Objelerin Bölütlenmiş Görüntüleri Kullanılarak Yapay Sinir Ağlarıyla Tanınması' başlıklı yüksek lisans tezi, 13./ 7 / 1998 tarihinde aşağıdaki jüri tarafından Lisansüstü Öğretim Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Jüri (Tez Danışmanı) :Prof.Dr. Atila Barkana.....

Jüri :Doç.Dr. Osman Parlaktuna...

Jüri :Dr. Hakan Şenel.....

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..27/08/1998 tarih ve .....13/1..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Orhan ÖZER
Fen Bilimleri Enstitüsü
Müdürü

ÖZET

Yüksek Lisans Tezi

OBJELERİN BÖLÜTLENMİŞ GÖRÜNTÜLERİ KULLANILARAK YAPAY SİNİR AĞLARIYLA TANINMASI

ATİLLA DEMİRAY

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği
Anabilim Dalı

Danışman: Prof.Dr. Atila BARKANA
1998

Şablon tanıma sistemleri, 1990'ların temel teknolojilerinden biri olarak yapay zeka alanında önemli bir yere sahiptir. Çünkü şablon verilerini anlamak, tanımlamak ve sınıflandırmak için birçok yapay zeka algoritmaları kullanılır. Çok sayıda uygulaması bulunan bu sistemlerin, özellikle askeri teknolojide önemli bir yeri vardır ve sürekli yeni algoritmalar ve sistemler geliştirilmektedir. Halen NASA (Ames Research Center) gibi birçok araştırma merkezinde, yapay zeka ve şablon tanıma sistemleri üzerine yoğun çalışmalar yapılmaktadır.

Şablonları anlamak, tanımak veya sınıflandırmak amacıyla kullanılan şablon tanıma sistemlerine örnek olarak, son yıllarda çok sık uygulamaları geliştirilen radar hedefi tanıma sistemlerini verebiliriz. Bu sistemler radar sinyallerinden hedef bilgilerini elde ederek sınıflandırma yaparlar.

Bu tezde de gerçek dünya objelerinin bölütlenmiş görüntülerinden tanınması amacıyla bir yapay sinir ağı yaklaşımı sunulmaktadır ve bu sistemi simule etmek amacıyla C++ dilinde programlar yazılmış ve ekte sunulmuştur. Bu sistem kullanılarak, obje tanıma uygulamaları yapılmıştır böylelikle sistemin performansı gözlenmiş ve sistemin dezavantajları belirlenmeye çalışılmıştır.

ABSTRACT

Master of Science Thesis

RECOGNITION OF SEGMENTED OBJECTS USING ARTIFICAL NEURAL NETWORKS

ATİLLA DEMİRAY

Anadolu University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

Supervisor: Prof. Atila BARKANA
1998

As one's of 1990's basis technology, pattern recognition systems take an important part in the field of artificial intelligence. Many artificial intelligence algorithms is used in these systems for understanding, description and classification of pattern data. Many existing applications of this systems hold an important place in advanced military technology and these systems are enhanced continuously. In many research centers such as NASA-Ames Research Center, artificial and pattern recognition systems have been studied extensively.

Radar target recognition systems can be given as an example on pattern recognition systems. These systems classify extracted target information from radar signals.

In this thesis, as neural network approach to be recognition of real world objects using their segmented images is presented. A C++ program which is used in this research is included in the appendix 2. Object recognition applications was implemented using this software, the system performance was observed and disadvantages of the system was determined.

Bu çalışmanın hazırlanmasındaki yardımlarından dolayı Sayın Prof.Dr. Atila Barkana ve Sayın Doç.Dr. Osman Parlaktuna'ya teşekkür ederim.

İÇİNDEKİLER

İÇİNDEKİLER (devam)

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1. GİRİŞ

1.1 Objelerin Bölütlenmiş Görüntüleri Kullanılarak Yapay Sinir Ağlarıyla Tanınması

Üç boyutlu gerçek objelerin kamera görüntülerinden tanınması, teknolojik ve bilimsel gelişim süreciyle birlikte çoğu sistemlerin temel ihtiyacı olmuştur. Özellikle, sistemlerde kullanılan insan komuta ve yönlendirme gereksinimini azaltmak amacıyla ve kendi kendine karar veren sistemlerin gerçeklenmesi amacıyla obje tanıma üzerine yapılan çalışmalar günümüzde oldukça popülerdir.

Temelde, objelerin görüntülerinden, objeleri tanımlayıcı nitelikleri ayrıştırma ve bunları sınıflandırma problemlerine dayanan bir obje tanıma sistemi bir çok farklı algoritmalar kullanılarak gerçeklenebilir. Ancak objenin tanımlayıcı niteliklerini iki boyutlu görüntülerinden ayrıştırmada kullanılan birçok algoritma, çoğu zaman bazı eksiklikler ve dezavantajlara sahip olduğu için sistemin performansını kötü yönde etkileyebilir. Bu eksikliklerin temel nedeni, üç boyutlu obje bilgisinin iki boyutlu bir görüntüde tam olarak ifade edilememesinden ve bu görüntülerin optimal verimlilikle temsil edilememesinden kaynaklanır. Bir objenin herhangi bir perspektif görüntüsünü, birçok obje farklı açılardan verebileceği için tek bir görüntüyle objeyi tanımlamak yetersiz kalır. Bu nedenle bir objeyi, onun perspektif görüntüleriyle tanımlamak için olası tüm perspektif görüntülerini kullanmak gerekir.

Obje tanıma sisteminin, tüm bu perspektif görüntülerin tek bir objeye ait olarak sınıflandırması için, bu görüntüler veya görüntüleri temsil eden tanımlayıcılar arasında bazı ortak nitelikleri keşfetmesi gerekir. Özellikle simetrik olmayan kompleks objelerin perspektif görüntüleri birbirinden farklı ve zor ilişkilendirilebilen görüntülerdir.

Bir obje sınıfına ait perspektif görüntüler arasındaki ortak özelliklerin seçimi birçok metodla yapılabilir. Örneğin yapısal nitelik ayrıştırma yaklaşımına göre, bir düzleme oturtulmuş silindirin yatay ve 90 derece dikey hariç bütün perspektif görüntülerinde, paralel iki kenar (silindir yüzeyi) ve bir elips (üst yüzey) mevcuttur. Ancak kompleks objeler için böyle bir yaklaşımda bulunmak zor ve eksik bir tanımlamaya neden olur. Bu çalışmada, sınıflandırıcı olarak bir yapay sinir ağın kullanılmasının temel nedeni budur.

Bir yapay sinir ağının en önemli özelliği, veri guruplarını belli bölgelere sınıflandırabilmesi (classification) veya haritalayabilmesi (mapping) dir. Sinir ağı, eğitimi esnasında, kendisine giriş olarak uygulanan kompleks veri gurupları arasında ortak bazı nitelikler saptayabilir ve bu verileri guruplandırabilir. Bu nedenle, bu çalışmada, şablonların sınıflandırılma işlemi için, yapay sinir ağı kullanılmaktadır.

Bir objenin perspektif görüntüsüne ait moment tanımlayıcıları, o görüntünün şekil bilgisini içeren ve dönme (rotation), yer değiştirme (translation) ve büyüklük değişimlerine (scaling) karşı sabit kalan bir tanımlayıcı vektör olarak kullanılabilir. Objenin tüm perspektif görüntülerinden elde edilen vektörler, o objeyi tanımlayan bir set olarak kabul edilirse, bir yapay sinir ağı bu setteki vektörler arasında doğrusal ve doğrusal olmayan ilişkilendirmeler yapabilir ve bunları sınıflandırabilir.

Bu çalışmada tanımlayıcı olarak kullanılan moment sabitlerinin en önemli dezavantajı, kesikleştirme (quantization) hatalarının yüksek dereceli momentlerde belirginleşmesi ve objenin global gri seviye değişimlerine yani ışıklandırma, kontrast veya parlaklık değişimlerine karşı yüksek sabitlik gösterememesidir.

Kesikleştirme hatalarının azaltılması birçok algoritmayla gerçekleştirilebileceği gibi görüntülerin boyutu arttırılarak ta momentlerde oluşan hatalar azaltılabilir. İkinci probleme karşı türetilen standart momentler ise bu hataları belirli ölçüde bastırmaktadır.

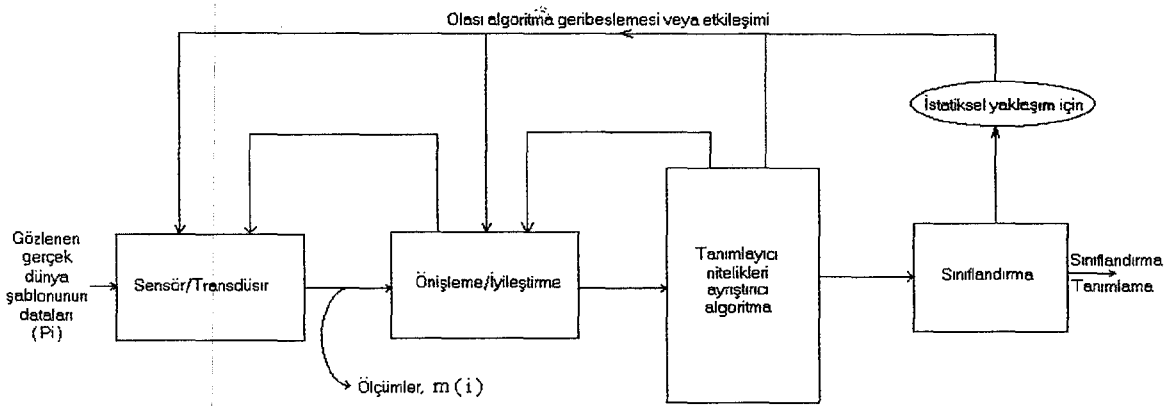
1.2 Şablon Tanıma Problemine Genel Bir Bakış

Şablon tanıma, 1990'ların temel teknolojisi olmaya aday bir alandır. Çünkü, şablon veya model tanıma teknikleri yapay zeka sistemlerinin önemli bir parçasıdır ve genel olarak ölçümleri veya verileri sınıflandırmak ve onlar hakkında karar vermek için kullanılır.

Şablon tanıma, belirli bir yaklaşımdan oluşmaz, fakat diğer bilgi ve tekniklerle gevşek bir bağı vardır. Önceleri şablon tanıma problemlerine iki temel yaklaşım vardı, bunlar istatiksel ve dizimsel (syntactic) di. İstatiksel şablon tanıma, adından da anlaşılacağı gibi önce şablonların istatiksel özelliklerini ayırıp, bu özelliklerle tanıma işlemini yapacak olan sistemi eğitir. Dizimsel şablon tanıma ise şablonların yapısal özelliklerini kullanır. Örneğin, bir objeye ait görüntüden objedeki köşe sayısı, kenar sayısı, simetrikliği gibi özellikleri bulunarak bu objenin mevcut sınıflardan birine ait olup olmadığına karar verir. 1980'lerin sonundan itibaren, yapay sinir ağlarının gelişen teknolojisi, yapay sinir ağlarıyla şablon tanıma yaklaşımının doğmasına neden olmuştur. Bu çalışmada da objeleri tanıma için bir sinirsel (neural) yaklaşım sunulmaktadır. Maalesef verilen bir şablon tanıma problemi için daima optimum çözümü verecek olan bir yaklaşım yoktur, fakat uygulamalarda hesapsal konular ve gerçek zamanlı çalışma amacıyla bir seçim yapılabilir.

Şablon tanıma sistemlerinin kullanıldığı başlıca uygulamalar; görüntü ayrıştırma ve analiz, sismik analiz, radar sinyali sınıflandırma ve analiz, yüz resmi tanıma, ses tanıma ve anlama, parmak izi tanıma, karakter tanıma (harf veya sayı), el yazısı tanıma, elektrokardiyografik sinyal anlama ve analiz etme, tıbbi teşhis yapma olarak söylenebilir. Başlıca sinirsel şablon tanıma (neural pattern recognition) uygulamaları ise, yüz resmi tanıma, el yazısı posta kodu tanıma, protein yapısı tanıma, tıbbi teşhis yapma, sonar hedef tanıma, araba yardımcı sistemi ve ses tanıma olarak söylenebilir.

1.3 Tipik Bir Şablon Tanıma Sisteminin Yapısı



Şekil 1.1. Şablon tanıma sistemi

Yukarıdaki sistemde sensör olarak bir kamera veya görüntü sensörü olabilir. Bu görüntüdeki obje veya resim parçası, tanımlayıcı nitelikleri ayrıştırılmadan önce görüntüden bölütlenmeli ve iyileştirilmelidir, yani bölütleme işleminden kaynaklanan hata ve gürültüler giderilmeli ve diğer işlemler için uygun hale getirilmelidir. Daha sonra yapılacak olan tanımlayıcı nitelikleri ayrıştırma işlemi şablonun tanımlayıcı niteliklerinin keşfedildiği bir algoritmadır. Sınıflandırma ise bu nitelikleri ilgili şablon sınıfına sınıflandıran bir algoritmadır ve şablon tanıma yaklaşımına bağlı olarak istatiksel, yapısal veya sinirsel olabilir.

2. ŞABLON TANIMA SİSTEMİ VE SİNİRSEL ŞABLON TANIMA

2.1 Şablon Tanıma Sisteminin tasarımı

Bir şablon tanıma sisteminin tasarımı genellikle aşağıda belirtilen beş temel adımda yapılır:

- *Objenin yerini bulmak ve ayırmak:* Bir görüntüden belirli bir cismi algılayıp ayırt eden bu algoritma genellikle bir tespit etme (detection) ve bölütleme (segmentation) algoritmasıdır.
- *Belirleyici niteliklerin seçimi:* Objeyi tanımlayıcı özellikleri hesaplayan algoritmadır.
- *Sınıflandırıcı tasarımı:* Hesaplanan niteliklerin ne tür bir objede olabileceğini belirleyen bir sınıflandırma algoritmasıdır.
- *Sınıflandırıcının eğitimi:* Sınıflandırıcının değişik parametrelerinin belirlenmesi amacıyla kullanılan algoritmadır. Bu algoritma sayesinde sınıflandırıcı, obje sınıflarına ait karar verme sınırlarını belirler.
- *Performansın hesaplanması:* Sınıflandırıcının yapabileceği olası hatalarının tahmin edilerek belirli bir yanlış tanıma oranının hesaplandığı kısımdır.

2.2 Şablonlar Ve Tanımlayıcı Nitelikleri

2.2.1 Şablonlar (Patterns)

Şablonlar ses, görüntü veya başka formlardaki sinyaller veya onların matris veya vektör notasyonundaki temsilî ifadeleridir.

Şablon tanıma sistemi, ölçüm veya gözlemlerden elde edilen şablonları sınıflandırır. Bazen bu şablon setleri başka bir şablon setine çevrildikten sonra kullanılacağı gibi, bazen de şablon tanıma sisteminin karakteristiğine uygun hale getirilmesi gerekir.

2.2.2 Şablonların tanımlayıcı nitelikleri (Features)

Bunlar, şablonlardan elde edilen herhangi ayrıştırılabilir değerlerdir. Örneğin düşük seviyeli bir sinyal özelliği olarak sinyal yoğunluğunu gösterebiliriz. Yüksek seviyeli bir özellik olarak ise, bir görüntüdeki bir objenin veya bir bölgenin geometrik tanımlayıcılarını gösterebiliriz.

Bu özellikler sembolik veya sayısal olabilirler, sembolik bir özellik olarak rengi gösterebiliriz, sayısal bir özellik olarak da ölçülen bir uzunluğu gösterebiliriz.

Bu özellikler bulunurken iki önemli durumla karşılaşılabilinir:

1. İyi bir hesaplama performansı gerekebilir.
2. Hesaplanan özellikler hataları veya gürültüyü içerebilir.

Bir şablondan çok sayıda belirleyici özellik çıkarmak mümkündür. Önemli olan şey, şablonu en iyi temsil edecek, nadir bulunabilen belirleyici özellikleri seçmektir. Bunlar

genelde yüksek seviyeli özelliklerdir. Ayrıca bunlar hesaplamada kolaylık sağlamalı, şablon tanıma sistemini başarılı yapabilmeli ve veri sayısının indirgenmiş (önemli bilgiler ihmal edilmemek şartıyla) olması gerekir.

2.3 Şablon Tanıma Problemine Sinir Ağı (Neural) Yaklaşımı

Şablon tanıma olgusuna son on yılda en etkili yaklaşım sinirsel yaklaşım olmuştur. Çünkü yapay sinir ağlarının yüksek sınıflandırma performansı ve yapılan birçok şablon tanıma uygulamasında alınan yüksek verim bu yaklaşımı popüler kılmıştır. Yapay sinir ağları temelde biyolojik model örnek alınarak ortaya çıkmıştır, yani biyolojik sinir sisteminin bir nevî matematiksel modellenmiş halidir. Yapay sinir ağları, üç temel yapısal özelliğe sahiptir.

1. Ağ topolojisi veya sinirsel birimlerin karşılıklı bağlantı yapıları.
2. Yapay sinirlerin veya diğer elemanların öz nitelikleri.
3. Sinir ağının eğitim stratejisi.

Biyolojik sistemler, birbirine bağlantılı fiziksel hücreler veya sinirler vasıtasıyla şablon tanıma yaparlar. Bu sinirsel şablon tanıma için temel oluşturmuştur. Bu amaçla bilgisayar, mühendislik, psikoloji, matematik, nöroloji gibi farklı bilim dallarında şablon tanıma ile ilgili hesapsal, yapısal ve biyolojik çalışmalar yapılmaktadır. Son on yılda, özellikle insan (veya hayvan) beyninin yaratıcılık olgusu ve hesaplayıcı yapısının yeniden yaratılması için yapay zeka sistemleri çalışmaları yapılmaktadır. Bu çalışmalar yapay sinir ağları, bağlamsal modelleme, nöromorfik modelleme, paralel dağılmış işleme isimleri altında toplanmaktadır.

Zeka olgusu, küçük işlemsel birimlerden oluşan büyük bir topluluğun karşılıklı etkileşimi vasıtasıyla ortaya çıkar. İnsan beyninin bilgi işlemede kullandığı temel elemanların hızı, gerçekte mili saniyeler düzeyinde ve yavaştır. Fakat bütün toplam işlem süresi birkaç yüz mili saniyeyi bulmaktadır. Bu eylem bize, büyük paralel yapıların, birkaç küçük seri adımdan sonra sonuca ulaştığı fikrini vermektedir. Ayrıca bu doğal paralel mimarideki her işlem elemanı yerel olarak birbirine bağlanmış ve basit bir yapıdadır. Bu nedenle, bağlamsal veya sinirsel hesaplama yaklaşımı yeni veya devrimci bir fikir olmamıştır.

2.4 Temel Yapay Sinir Ağı Kavramı

- Tüm modelin temel elemanı, değişebilen bağlantılara sahip basit birimlerdir.
- Bu elemanları, eğitim verilerini kullanarak eğitmek, ana öğrenme stratejisini oluşturur. Diğer bir ifadeyle, eğitim vasıtasıyla, sistemin bilgi ve deneyimi, bu elemanlar arasındaki bağlantılarda saklanır.
- Sinirsel sistemlerin kullanışlı olabilmesi için bilgiyi depolama kapasitesinin olması gerekir. Yani bir şablonu sınıflandırmak veya tanımak için sinir ağına girdiğimizde, tüm yapısal elemanların ortak bir davranış göstererek bir sonuç üretmesi beklenir.
- Sinir ağları dinamik sistemler oldukları için harici bir girişe veya anlık bir duruma karşılık iç bağlantı ve dış çıkış birimlerinde değişim gösterir.

2.5 'Sinir Ağlarıyla Şablon Tanıma' Problemlerinin Genel Karakteristiği

Sinirsel problemleri üç ana sınıfa ayırabiliriz, zor problem sınıfında; tanımlama, araştırma, ve diğer kesin sonuç alınması gereken problemler gösterilebilir. Şablon veya obje tanıma sınıfında; ses ve görüntü tanıma vardır. Ve son olarak kusurlu sonuçlar alınabilen sınıfta ise; bulanık (fuzzy) ve olasılıksal verilerin kullanıldığı problemler vardır.

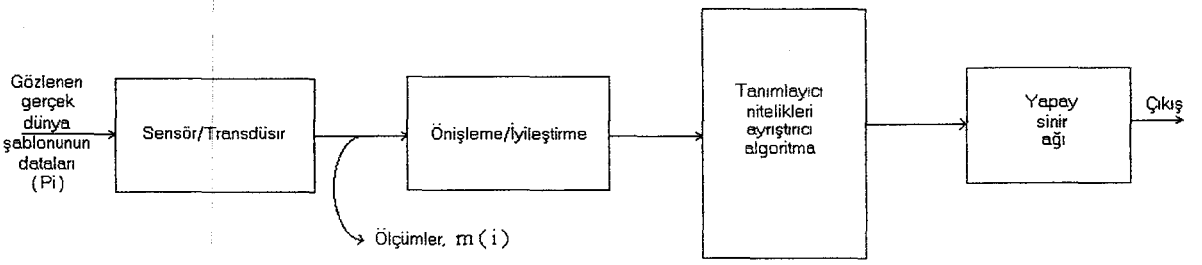
Sinirsel problemlerin hepsinin genel karakteristiği şu şekildedir.

- Problem uzayı yüksek boyutludur.
- Problem değişkenleri arasında karmaşık bir etkileşim vardır.
- Çözüm uzayı boş olabilir, tek bir çözüm içerebilir veya genellikle yararlı çözümlerden oluşan bir gurup içerebilir.

Sinir ağlarıyla şablon tanıma uygulamalarının birçoğu aşağıdaki üç ana tip problemden oluşur.

- Kompleks veri gruplarından özel niteliklerin ayrıştırılması (Feature extraction)
- Karakter tanıma ve görüntü işleme
- Araştırma ve eşleştirme (matching) algoritmalarının doğrudan veya paralel uygulaması.

2.6 Yapay Sinir Ağları İle Şablon Tanıma Sistemlerinin Genel Yapısı



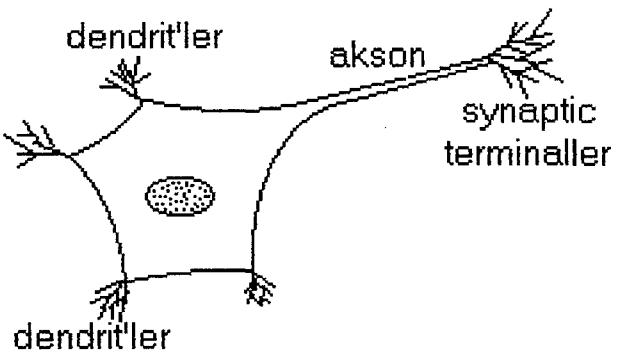
Şekil 2.1. Sinirsel şablon tanıma sistemi

Yukarıdaki sistemde, performansı etkileyecek en önemli kısımlar nitelik ayrıştırma ve sınıflandırma kısımlarıdır. Gerçek dünya görüntüsünün bir kamera veya görüntü algılayıcı tarafından algılanıp, görüntüyü iyileştirici ön işlemlerden geçirdikten sonra genellikle görüntüdeki objeyi veya ilgilenilen resim parçasını görüntüden ayrıştırmak gerekir, bu işlem için tarayıcı veya algılayıcı bir algoritma kullanılarak resmin ilgili bölgesi saptanır ve bu bölgeye ayrıştırıcı (segmentation) bir algoritma uygulanarak ilgilenilen obje veya cisim resimden ayrıştırılır. Artık bu resim parçasını tanımlayıcı (descriptor) veya temsil edici (representive) özelliklerin ayrıştırılması (feature extraction) gerekir. Bu ayrıştırılmış özellikler bir matris veya vektör notasyonunda, eğitilmiş yapay sinir ağına girilir. Yapay sinir ağı kendisine gelen bu verilere, önceden eğitildiği şekilde bir tepki gösterir, ve çıkışındaki

nöronlardan yine belirli bir vektör veya matris notasyonunda bir grup üretir. Bu veriler bize objeyi veya resim parçasını tanımlar.

2.7 Sinir Ağı Yapısı Ve Nöron Modeli

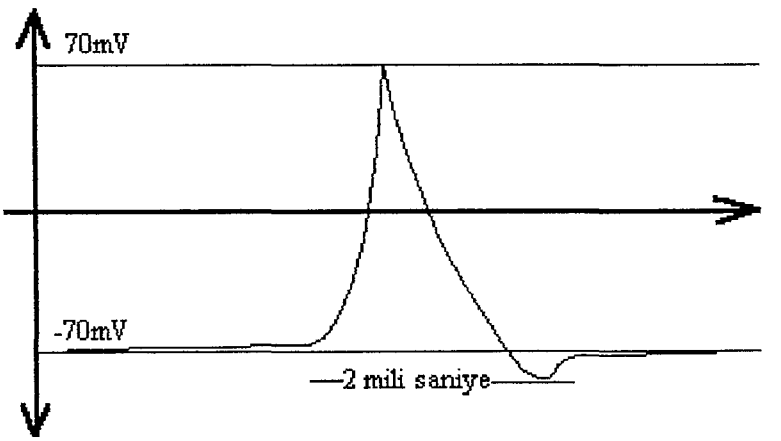
2.7.1 Biyolojik yapı



Şekil 2.2. Basit bir sinir hücresi

Milyonlarca biyolojik sinir hücresinin kendi aralarında tamamen düzensiz bir şekilde akson denilen uzantılarla birbirine bağlanarak oluşturduğu yapıya sinir ağı denir. Sinir hücreleri, yüzey zarlarında dolaşan elektriksel akımların neden olduğu potansiyel değişimlerin içerdiği bilgiyi birbirlerine iletirler ve her hücre kendine ulaşan potansiyel bilgiye göre bir tepki olarak yine bir potansiyel gerilim üretir.

İletişimi sağlayan aksonlarda potansiyel bilgiyi oluşturan özel bir dalga formuna sahip bir sinyal vardır, bu tıpkı bir dijital bilgi gibidir. Bu dalga formu aşağıdaki gibidir.



Şekil 2.3. Sinir hücresinin aksonlarındaki potansiyel sinyal

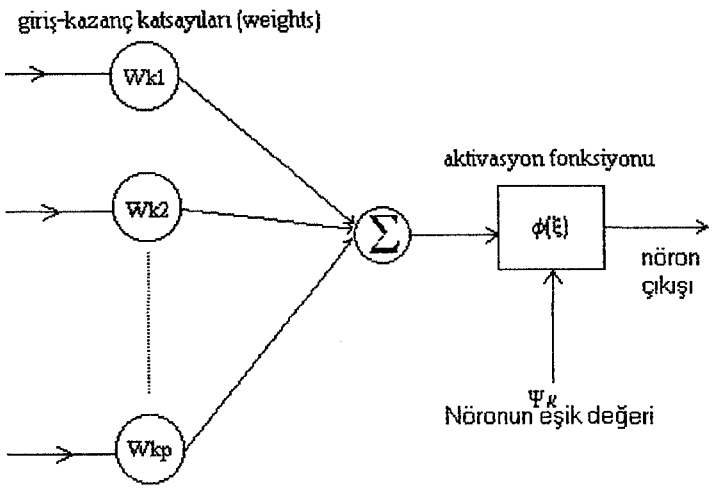
Bu sinyal nöronun 'synapse' denilen sinyal algılayıcı koluna ulaştıktan sonra buradaki iyonları faaliyete geçirir ve bu iyonlar dendirt denilen sinir uçları ile 'synapse' arasında yer değiştirerek bir potansiyel oluştururlar bu potansiyelin değeri yaklaşık 1mV'luk bir tepe gerilimidir. Dendirt denilen bu sinir uçlarından sinir hücrelerine gelen bu potansiyeller, sinir hücresinin eşik değerini geçerse sinir hücresi bir çıkış gerilimi üretir, sinir hücresinin bu eşik gerilimi onlarca mili volt seviyesinde zamana bağlı bir değerdir. Sinir hücresinin ürettiği bu gerilim tekrar aksonlar vasıtasıyla başka hücrelere ulaşır ve oralarda işlem görür.

Bu biyolojik model örnek alınarak aşağıdaki gibi bir matematiksel nöron modeli geliştirilmiştir.

2.7.2 Matematiksel nöron modeli

Yukarıdaki biyolojik model göz önüne alınarak geliştirilen matematiksel nöron modelinin üç temel bileşeni vardır.

1. 'Synapses' veya bağlantı hatları (Ağırlıklar veya bağlantı katsayıları)
2. Giriş sinyallerinin toplanması için bir toplayıcı
3. Nöron çıkışının büyüklüğünü sınırlandırmak için bir aktivasyon fonksiyonu
4. Nöronun eşik değeri



Şekil 2.4. Matematiksel nöron modeli

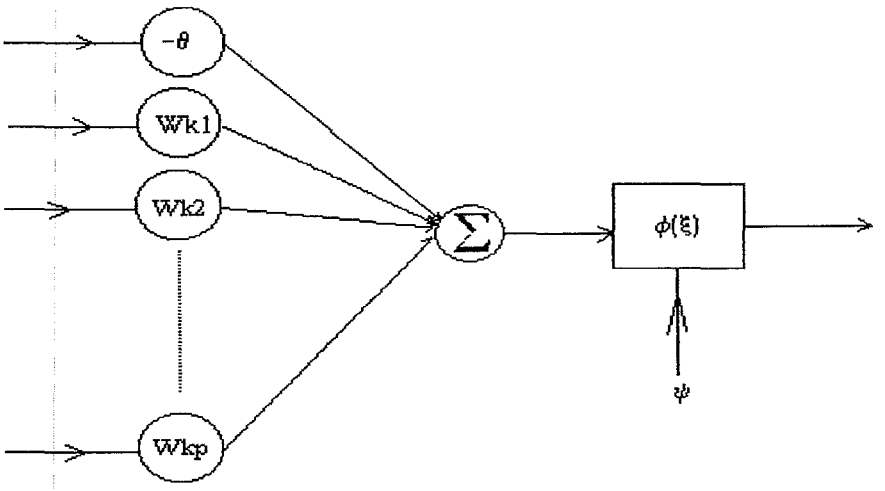
Aktivasyon fonksiyonu sinir hücresinin çıkışını kompanse etmek için diğer bir ifadeyle sınırlandırmak için kullanılır. Örneğin [0,1] aralığında değer alan

$$\varphi(x_i) = \frac{1}{1 + e^{-\psi x_i}} \qquad (2\text{-}1)$$

şeklindeki sigmoid fonksiyonu aktivasyon fonksiyonu olarak kullanılır.

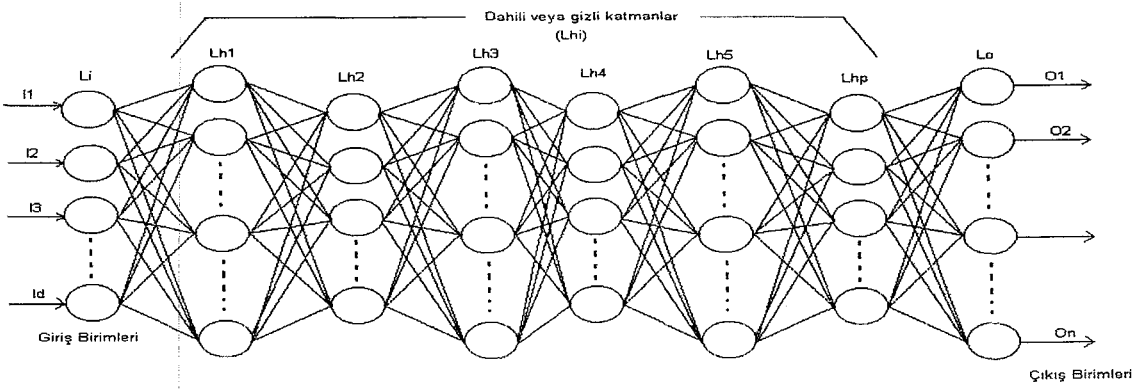
$$x_j = \sum_i W_{ji} O_i - \theta \qquad (2\text{-}2)$$

Diğer nöronların çıkışından gelen sinyaller ağırlıklarla çarpılıp, çarpımlar toplanır, bu toplamdan bir eşik (θ) değeri çıkarılır, ve sonuç nöronun aktivasyon fonksiyonuna (Denk. 2-1) girer, bu fonksiyonun çıkışı da diğer hücrelere giriş olarak dağılır. Bu eşik değerini nöronun girişine alırsak aşağıdaki nöron modeli ortaya çıkar, böylelikle yapay sinir ağı eğitilirken, tıpkı ağırlık değerleri gibi eşik değeri de ayarlanarak, her nöronda farklı eşik değeri elde edilir, bu da yapay sinir ağının algoritmik olarak daha basitleştirilmesini sağlar.



Şekil 2.5. Eşikli sinir hücresi modeli

2.7.3 İleri beslemeli (feedforward) sinir ağı yapısı



Şekil 2.6. İleri besleme sinir ağı

Bu yapıdaki sinir ağını, sinirsel hücrelerden oluşan katmanlar oluşturur. İlk katmana giriş katmanı, son katmana çıkış katmanı ve aradaki katmanlara da gizli katmanlar denir. Giriş katmanında, giriş vektörünün boyutu kadar hücre bulunmalıdır. Çıkış katmanındaki hücre sayısı ise sistemin tasarımına göre değişir. Fakat genellikle sinir ağı tarafından tanınması planlanan toplam sınıf sayısına eşittir. Gizli katmanlardaki nöron sayısını belirleyen bir kural maalesef yoktur. Çıkış katmanı hariç diğer tüm katmanlardaki her hücre, bir sonraki katmandaki tüm hücrelere bağlıdır. Yani bu katmanlardaki her hücrenin çıkışı bir sonraki katmandaki tüm hücrelere dağılır. Hücreler arasındaki bu bağlar çeşitli ağırlıklara sahiptir yani bu katmanlardaki herhangi bir hücrenin çıkışı bir sonraki katmandaki herhangi bir hücreye, belirli bir sabitle çarpıldıktan sonra girilir. Bu katsayılar sinir ağının eğitimi sonucunda belirlenir ve giriş-çıkış değerleri arasındaki fonksiyonel ilişkinin bilgilerini ihtiva ederler. Giriş katmanının rolü diğerlerinden farklıdır. Bu katmandaki hücreler giriş değerlerini alıp bir sonraki katmandaki hücrelere dağıtırlar ve bu hücrelerin ağırlıkları yoktur. İleri besleme sinir ağı eğitilerek verileri ortak niteliklerine göre guruplandırabilmelidir. Eğitim, sinir ağının ağırlıklarını ayarlar. Eğitici şablonlar sinir ağına girilir, sinir ağının verdiği çıkışla, olması gereken çıkış karşılaştırılır ve bir hata ölçüsü bulunur. Bu hata sinir ağı içinden geriye doğru bildirilir ve hücrelerin ağırlık değerleri buna göre değişir. Bu süreç sinir ağı istenen tepkiyi verinceye kadar tekrarlanır.

2.7.4 Gizli katmanların rolü nedir ?

- Veri ifadelerini daha fazla sınıflandırılabilir veya ayrıştırılabilir kılmak için kendinden önceki katmanların sonuçlarını tekrar haritalar (remap).
- Katmanların belirli giriş kombinasyonlarını anlamsal olarak bağdaştırır.

2.7.5 Ne kadar gizli katman gerekir ?

Maalesef bu sorunun belirli bir cevabı yoktur. 1957'de Kolmogorov'un yeterli olmayan bir teoremi vardır ve bu soruya belirli bir cevap verememektedir. Sinir ağının sınıflandırma esnasında yapabileceği hata oranını düşürmek için bu katmanların sayısı yeterince fazla olmalıdır bu miktar sınıflandırılacak şablon sayısı ile ilişkilidir.

Çok fazla veya çok az sayıda gizli katman kullanılması sinir ağının eğitiminin yakınsama hızını ve eğitim seviyesini düşürebilir. Sınıflandırma hatasını azaltmak ve eğitimin hızını artırmak için optimal gizli katman sayısı denemeyle bulunur.

2.8 Yapay Sinir Ağının Şablon Tanımada Kullanımı

2.8.1 Yapay sinir ağının eğitilmesi ve eğitilmiş sinir ağının kullanılması

Bu çalışmada, objelerin eğitici modellerine ait görüntülerden, yapay sinir ağının girişine uygulanacak eğitici nitelik vektörleri (feature vector) nitelik ayrıştırma (feature

extraction) algoritmasıyla elde edilir. Yapay sinir ağını eğitmek için kullanılan örnek modeller özenle seçilmelidir, yani o objenin herhangi bir durumda alınmış görüntüsüne ait nitelik-vektörü, eğitim setinin uzayı içinde olmalıdır. Böylelikle sinir ağının eğitim sonunda şekillenen fonksiyonel yapısı, objeye ait rasgele bir görüntüye ait nitelik vektörünü sınıflandırabilir.

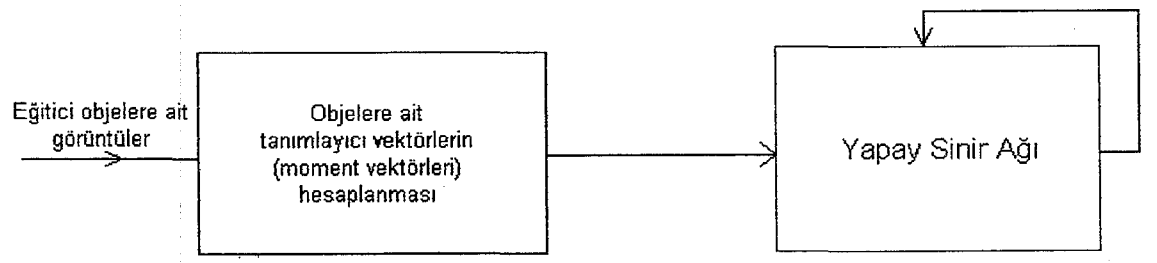
Birçok yapay sinir ağı modeli ve eğitim algoritması mevcuttur. Bu çalışmada ileri besleme (feed-forward) yapay sinir ağı modeli kullanılmıştır, ve bu model, geri yayılım (backpropagation) algoritmasıyla eğitilmiştir.

Eğitim algoritmasında sinir ağına uygulanan giriş sonucunda sinir ağının verdiği sonuç ile olması gereken sonuç karşılaştırılır ve elde edilen hataya bağlı olarak sinir ağının parametreleri değiştirilir. Bu işlem, sinir ağına girilen eğitici verilerin, sinir ağının çıkışında istenilen sonuçları vermesiyle veya belirli bir hata toleransıyla sonlandırılır.

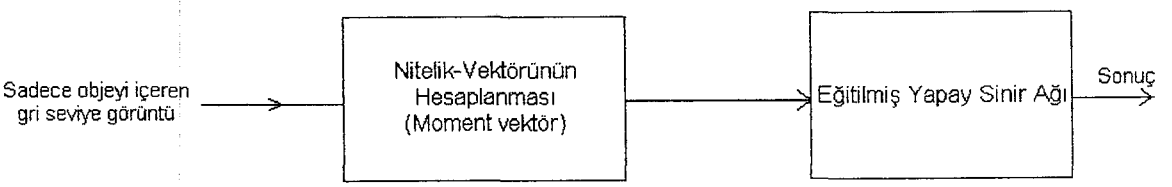
Eğitim algoritmasına ait parametreler ve hata toleransları sayısal girişlere ve problemin önemine göre değişir.

Eğitilmiş sinir ağının kullanımı esnasında da, yine objeye ait görüntüden elde edilen nitelik vektörü sinir ağına girilir ve sinir ağının verdiği sonuç tanıtılan objelerden birine ait olduğu şeklindedir.

2.8.2 Sistemin yapısı



Şekil 2.7. Sinirsel obje-tanıma sisteminin eğitimi



Şekil 2.8. Sinirsel obje-tanıma sisteminin kullanılması

İlk önce objeye ait özel olarak seçilen eğitici görüntülerle yapay sinir ağı eğitilir. Bu görüntüler özel seçilmelidir, çünkü yapay sinir ağının tanıma performansı büyük ölçüde buna bağlıdır. Bu nedenle objenin olası bütün görüntülerini hatasız sınıflandırabilmek için, eğitimde kullanılan görüntüler, bu olası görüntüleri kapsayan bir uzay oluşturmalıdır, yani herhangi bir görüntü geldiğinde, sinir ağı bunu eğitici görüntülerden birine benzetebilmelidir.

Bu sistemde, tanımlayıcı vektörü elde etmek için kullanılan tüm görüntüler, kameradan veya başka bir görüntü algılayıcı aygıttan alınan orijinal görüntüden, bölütleme ile elde edilir. Bu görüntülerde, objeden başka bir cisim veya resim parçası bulunmamalıdır. Bununla birlikte bölütleme hatalarından veya diğer hatalardan oluşabilecek gürültüler olabilir ve bunlar sistemin performansını etkileyebilir.

Objeye ait bu şablon görüntülerden, tanımlayıcı vektörler, momentlerle elde edilir. Momentler, iki boyutlu bir fonksiyonun şekil bilgisini içeren sayısal değerlerdir. Bu sayısal değerler çeşitli matematiksel denklemlerle elde edilen bir sabit (invariant) setine dönüştürülür. Bu sayısal değerler, bir görüntünün uğrayabileceği ölçekleme, kayma, yansıma, ve dönme etkilerinden etkilenmez. Dolayısıyla bir şeklin tüm olası görüntüleri, tek bir vektörle ifade edilebilir. Bu özelliği objelere uygulamak için, bu objenin olası perspektif görüntülerinin moment sabitlerinin sinir ağının eğitiminde kullanılması gerekir. O zaman bu objeye ait herhangi bir perspektif görüntü, eğitimde kullanılan şablonlardan birine benzetilebilir. Eğitim için kullanılacak perspektif görüntülerin seçimi yine objenin yapısal özelliğine bağlıdır, çünkü çok kompleks ve simetrik olmayan bir objenin perspektif görüntüleri de birbirinden bağımsız olur ama örneğin bir silindirin perspektif görüntüleri sadece silindirin düşey ekseni boyunca değişir. Dolayısıyla diğer açılardan alınan görüntüler aynı olacaktır. Ayrıca, bu moment sabitlerinin yukarıda belirtilen yansıma ve dönme etkilerine olan değişmezliği nedeniyle, simetrik olan objelerin belirli bir açıdan sonra alınan görüntüleri eski görüntüleri ile aynı olur yani periyodik olarak değişir. Örneğin bir küpün simetrik olması nedeniyle, bu küpün perspektif görüntüleri her 45 derecelik periyotlarda aynıdır veya bir silindir objesinin perspektif görüntüleri sadece düşey eksen boyunca değişir.

Moment sabitlerinin en önemli dezavantajlarından biri, obje üzerindeki aydınlatma, parlaklık veya ışığın geliş açısı gibi ortamın ışıksal değişimlerine karşı iyi bir değişmezlik gösterememesidir. Bu nedenle, bu çalışmada, bu sorunu aşmak için, moment sabitlerinin önerilen bir türü olan standart momentler kullanılmıştır. Standart momentler bu sorunu belirli bir ölçüde bastırırlar.

Moment sabitlerinin bir başka dezavantajı, yüksek dereceli sabitlerin gayet dinamik bir değişim gösterebilmesidir, bunu biraz daha açacak olursak, örneklenmiş bir sayısal resmin kesikleştirmeden kaynaklanan hataları, yüksek dereceden sayısal işlemler nedeniyle daha da belirgin hale gelmektedir.

3. OBJELERİN BÖLÜTLENMİŞ PERSPEKTİF GÖRÜNTÜLERİNDEN TANIMLAYICI NİTELİKLERİN ELDE EDİLMESİ

Örneklenmiş bir sinyal veya bir görüntüden nitelik ayrıştırma işlemi için öncelikle o sinyal veya görüntüdeki şablon çıkarılmalıdır, örneğin bir objenin de içinde bulunduğu görüntüde önce obje bölütleme işlemi ile resimden ayrılmalıdır. Daha sonra objeye ait veriler kullanılarak o şablonun belirleyici nitelikleri ayrıştırılır. Örneğin bir objeye ait görüntüde, objenin rengi, büyüklüğü, veya şekil bilgisi onun düşük seviyeli tanımlayıcı nitelikleridir.

Bu çalışmada objelerin görüntülerinden objeleri tanıma ve sınıflandırma işlemi yapmak için objenin perspektif görüntülerine ait şekil bilgisi (moment tanımlayıcıları) çıkarılmakta ve bu bilgiler tanımlayıcı nitelik olarak kullanılarak yapay sinir ağı eğitilmektedir.

Bu şekil bilgisi 'invariant moment' tekniği ile bulunmaktadır. Bu teknikte, resme ait şekil bilgisi altı boyutlu bir vektöre dönüştürülmektedir, bu vektör, görüntünün dönme, kayma, ölçekleme, yansıma gibi uğrayabileceği etkilerden etkilenmez. Bu özelliğinden dolayı birçok şablon tanıma uygulamasında bu nitelik ayrıştırma tekniği kullanılmıştır.

Bir objenin perspektif görüntüleri, objeye bakış açısına göre değişebileceğinden aynı objeye ait birçok görüntü farklı moment değerlerine sahiptir, fakat sinir ağı bu görüntülerin moment verileri arasında ortak bilgiler bulmakta veya fonksiyonel bir ilişki keşfetmektedir.

Moment tanımlayıcılarının kullanıldığı birçok şablon tanıma uygulaması mevcuttur. Bunlar arasında daktilo karakteri tanıma, el yazısı karakteri tanıma, uçak tanıma, tank tanıma, gemi tanıma, büyük bir radar görüntüsünün bir parçasını saptamak, üretim bandındaki endüstriyel parçaların tanınması, kan hücrelerinin sınıflandırılması gibi örnekleri sayabiliriz.

3.1 Moment Tanımlayıcıları

Bölütlenmiş görüntüleri, değişik bir formda ifade etmenin bir yolu da, bu görüntülerin şekil bilgisini dönüştürmektir (transformation). Yapılan dönüşümün, resmin uğrayabileceği dönme, kayma, ve ölçekleme gibi değişimlere karşı sabit olması (invariant) gerekir.

Moment metodu bunu mümkün kılar. Bu metodun daha önceden belirtilen dezavantajlarına ilaveten, bu sistemde problem olmayan bir dezavantajı da, belirli bir obje sınıfına ait moment setlerini, aralarındaki ilişkiyi keşfedip sınıflandırmanın zor olduğudur. Bu problem, bu çalışmada kullanılan sinir ağı sınıflandırıcısı ile aşılmaktadır. Daha önceden de belirtilen bir problem olarak, yüksek dereceli momentler, geniş-dinamik bir aralıkta değişebilmektedir. Detaylı şekillerin, detay bilgisini elde edebilmek için yüksek dereceli momentleri bulunmalıdır, fakat yukarıda belirtilen özelliğinden dolayı yüksek dereceli momentler, görüntünün kesikleştirme hatalarını da geniş-dinamik bir aralığa taşır. Bunlara rağmen, moment teoremi, birçok şablon tanıma uygulamasında kullanılan önemli tanımlayıcılar türetir.

3.2 Momentlerin Matematiksel Tanımı

İki boyutlu, sürekli bir f(x,y) fonksiyonunun karakteristik fonksiyonu, onun eşlenik Fourier dönüşümüyle tanımlanır.

$$F^*(\xi_1, \xi_2) = \iint_R f(x, y) e^{2\pi j(x\xi_1 + y\xi_2)} dx dy \qquad (3\text{-}1)$$

f(x,y) nin moment türetme fonksiyonu ise aşağıdaki gibi tanımlanır.

$$M(\xi_1, \xi_2) = \iint_R f(x, y) e^{(x\xi_1 + y\xi_2)} dx dy \qquad (3\text{-}2)$$

Bu denklemden, momentler aşağıdaki denklemle elde edilir.

$$m_{pq} = \left. \frac{\partial^{p+q} M(\xi_1, \xi_2)}{\partial \xi_1^{\,p} \partial \xi_2^{\,q}} \right|_{\xi_1 = \xi_2 = 0} \qquad (3\text{-}3)$$

Bu denklem bize f(x,y) fonksiyonunun p+q'uncu dereceden momentini verir ve çözümü aşağıdaki denklemdir.

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \qquad p, q = 0,1,2,3,..... \qquad (3\text{-}4)$$

Bu denklemle bulabileceğimiz sonsuz sayıdaki momentlerle, f(x,y) fonksiyonunu tekrar elde edebiliriz, yani bu momentler f(x,y) fonksiyonunun yegane tanımlayıcılarıdır. Bunu matematiksel olarak aşağıdaki gibi gösterebiliriz.

Yukarıdaki sürekli fonksiyon f(x,y) nin moment denkleminin kesikli fonksiyon şekli aşağıdaki gibi olmak üzere;

$$m_{pq} = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} x^p y^q f(x, y) \qquad p, q = 0,1,2,3,..... \qquad (3\text{-}5)$$

f(x,y)'nin eşlenik Fourier dönüşüm denklemindeki eksponansiyel terimini, güç serisiyle açarsak ve, yukarıdaki kesikli moment fonksiyonunu bu denklemde yerine koyup, entegrali toplam işaretiyle değiştirirsek aşağıdaki ifadeyi elde ederiz.

$$F^*(\xi_1, \xi_2) = \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} m_{pq} \frac{(j2\pi)^{p+q}}{p!q!} \xi_1{}^p \xi_2{}^q \qquad (3\text{-}6)$$

Bu denklemin her iki tarafının ters Fourier dönüşümünü alırsak,

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-j2\pi(x\xi_1 + y\xi_2)} \left[\sum_{p=0}^{\infty} \sum_{q=0}^{\infty} m_{pq} \frac{(j2\pi)^{p+q}}{p!q!} \xi_1{}^p \xi_2{}^q \right] d\xi_1 d\xi \qquad (3\text{-}7)$$

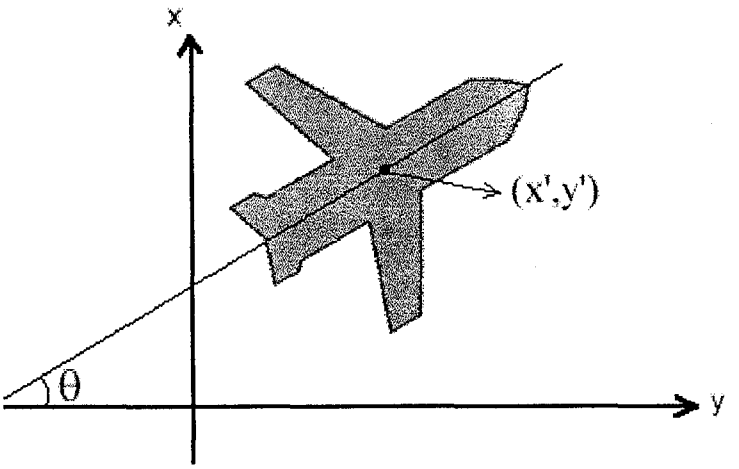
olur. Sonuç olarak bir f(x,y) fonksiyonunu onun momentleriyle ifade edebildiğimize göre bu momentler f(x,y)'nin bir tanımlayıcısıdır.

3.3 Moment Sabitlerinin Türetilmesi

Kesikli bir f(x,y) fonksiyonu için momentler;

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \qquad p, q = 0,1,2,3,..... \qquad (3\text{-}8)$$

olur. Bu momentlerin, f(x,y) kesikli fonksiyonunun uğrayabileceği çeşitli etkilere karşı değişmemesi için bu etkilere karşı sabitlenmeleri gerekir.



Şekil 3.1. Bir uçak objesine ait perspektif görüntü f(x,y)

3.3.1 Kayma etkisinin sabitlenmesi

f(x,y) fonksiyonunun ağırlık merkezi olan (x',y') noktası;

$$x' = \frac{m_{10}}{m_{00}},\ y' = \frac{m_{01}}{m_{00}} \qquad (3\text{-}9)$$

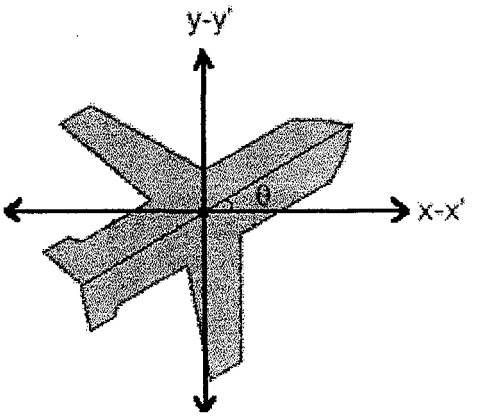
denklemleriyle bulunur.

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-x')^p(y-y')^q f(x,y)dxdy \qquad p,q=0,1,2,3,..... \qquad (3\text{-}10)$$

Eğer f(x,y)'yi, (x-x') ve (y-y') noktaları ile ifade edersek, f(x,y) fonksiyonunun temsil ettiği şekli, x-y koordinat sisteminin merkezine taşımış oluruz. Dolayısıyla f(x,y) şekli, bulunduğu görüntü içinde nerede olursa, bu şekli koordinat merkezine sabitleyebiliriz. Böylelikle momentleri de, f(x,y) şeklinin uğrayabileceği kayma veya yer değiştirmelerden etkilenmeden hesaplayabiliriz. Bu momentlere merkezi momentler denir.

Bu denklem, iki boyutlu kesikli f(x,y) fonksiyonuna uyarlanırsa;

$$\mu_{pq} = \sum_x \sum_y (x-x')^p (y-y')^q f(x,y) \qquad p,q=0,1,2,3,..... \qquad (3\text{-}11)$$



Şekil 3.2. Kayma etkisine karşı sabitlenmiş f(x,y)

Eğer f(x,y) şeklinin bulunduğu görüntü siyah-beyaz (binary) ise m_{00} değeri şeklin alanını ifade eder.

Bir siyah-beyaz görüntüde şeklin x' ve y' değerlerini hesaplayarak, şeklin görüntüdeki yaklaşık yerini bulabiliriz. Örneğin, bir uygulamada, beyaz kan hücrelerinin hareketleri bu şekilde takip edilerek, dış faktörlerin bu hareketlere etkisi incelenmiştir. Ayrıca bu metotla, bir cismin hareketi, görüntüsünden tespit edilerek, başka bir hareketle karşılaştırılabilmektedir.

Merkezi momentler, x' ve y' değerleriyle, aşağıdaki denklemle de tanımlanabilir.

$$\mu_{pq} = \sum_{r=0}^{p}\sum_{s=0}^{q} C_r^p C_s^q (-x')^r (-y')^s m_{p-r,q-s} \qquad p,q = 0,1,2,3,..... \qquad (3\text{-}12)$$

$$C_r^p = \frac{p!}{r!(p-r)!} \qquad (3\text{-}13)$$

Böylelikle, yukarıdaki denklemden elde edilen sıfırıncı, birinci, ikinci ve üçüncü derece merkezi momentler aşağıdaki gibidir.

$$\mu_{00} = m_{00} = \mu \qquad (3\text{-}14)$$

$$\mu_{10} = \mu_{01} = 0 \qquad (3\text{-}15)$$

$$\mu_{20} = m_{20} - \mu x'^2 \qquad (3\text{-}16)$$

$$\mu_{11} = m_{11} - \mu x' y' \qquad (3\text{-}17)$$

$$\mu_{02} = m_{02} - \mu y'^2 \qquad (3\text{-}18)$$

$$\mu_{30} = m_{30} - 3m_{20}x' + 2\mu x'^3 \qquad (3\text{-}19)$$

$$\mu_{21} = m_{21} - m_{20}y' - 2m_{11}x' + 2\mu x'^2 y' \qquad (3\text{-}20)$$

$$\mu_{12} = m_{12} - m_{02}x' - 2m_{11}y' + 2\mu x' y'^2 \qquad (3\text{-}21)$$

$$\mu_{03} = m_{03} - 3m_{02}y' + 2\mu y'^3 \qquad (3\text{-}22)$$

Birçok uygulamada, objenin perspektif görüntüsünün silueti, yani siyah-beyaz (binary) görüntüsü kullanılarak, tanıma yapılmıştır. Bu uygulamalarda kullanılmak üzere, yedi adet moment tanımlayıcı denklem türetilmiştir. Bu denklemler kayma, dönme, ölçekleme gibi etkilere karşı değişim göstermezler.

Tablo 3.1. Hu'nun merkezî momentlerden türettiği sabit seti

$\phi_1=\mu_{20}+\mu_{02}$
$\phi_2=(\mu_{20}-\mu_{02})^2+4\mu_{11}^2$
$\phi_3=(\mu_{30}-3\mu_{12})^2+(3\mu_{21}-\mu_{03})^2$
$\phi_4=(\mu_{30}+\mu_{12})^2+(\mu_{21}+\mu_{03})^2$
$\phi_5=(\mu_{30}-3\mu_{12})(\mu_{30}+\mu_{12})[(\mu_{30}+\mu_{12})^2-3(\mu_{21}+\mu_{03})^2]+(3\mu_{21}-\mu_{03})(\mu_{21}+\mu_{03}))[3(\mu_{30}+\mu_{12})^2-(\mu_{21}+\mu_{03})^2]$
$\phi_6=(\mu_{20}-\mu_{02})[(\mu_{30}+\mu_{12})^2-(\mu_{21}+\mu_{03})^2]+4\mu_{11}(\mu_{30}+\mu_{12})(\mu_{21}+\mu_{03})$
$\phi_7=(3\mu_{21}-\mu_{03})(\mu_{30}+\mu_{12})[(\mu_{30}+\mu_{12})^2-3(\mu_{21}+\mu_{03})^2]-(\mu_{30}-3\mu_{12})(\mu_{21}+\mu_{03})[3(\mu_{30}+\mu_{12})^2-(\mu_{21}+\mu_{03})^2])$

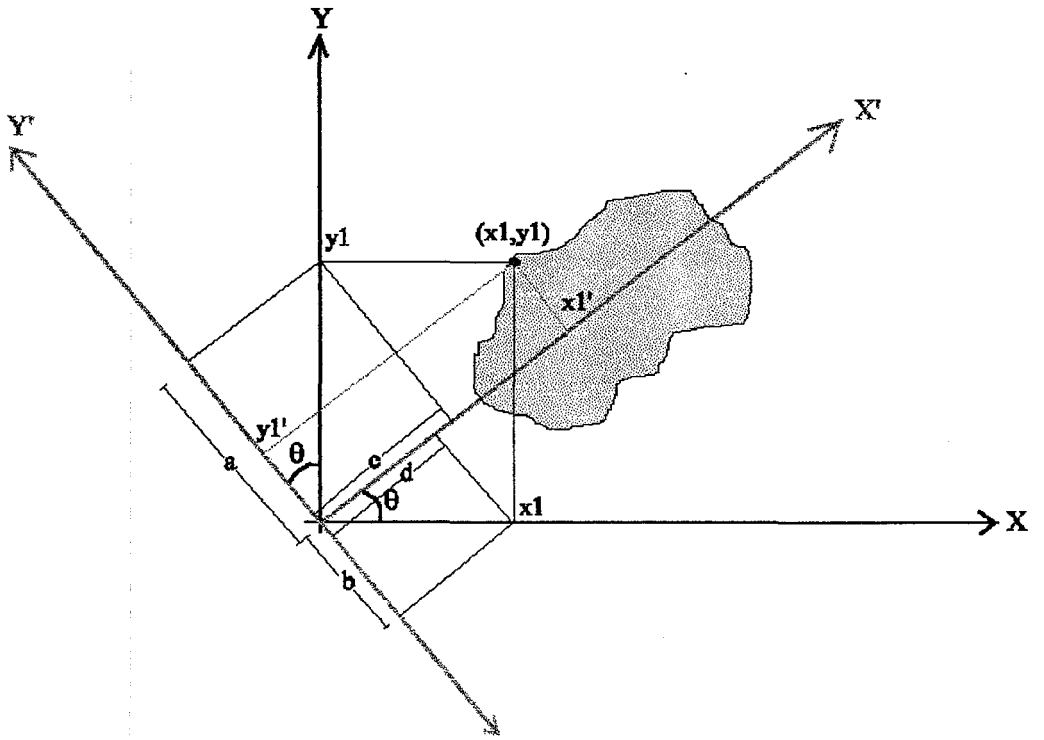
Bu moment setinin ilk altı değeri, kayma, dönme, ölçekleme ve yansıma özelliklerine karşı sabittirler. Yedinci değer ise yansımaya karşı işaret değiştirir, dolayısıyla, yansıma yüzünden M_7'nin sadece büyüklüğüne güvenilebilir. Bu moment sabitleri maalesef görüntünün kontrastına karşı sabit olmadığı için binary görüntülerde kullanılır, fakat diğer değişimlere yüksek derecede sabitlik gösterir.

Bu çalışmada gri-seviye görüntüler kullanıldığı için aşağıdaki moment seti kullanılmamıştır. Merkezi momentlerle türetilen ve gri-seviye görüntülere de uygulanabilen bu moment setine standart momentler denir ve aşağıda gösterildiği gibi dönme ve ölçekleme etkilerine karşı sabitlenirler.

3.3.2 Dönme açısının bulunması ve dönme etkisinin sabitlenmesi

İki boyutlu bir X_Y uzayında bulunan bir objenin ana ekseninin, yatay eksenle yaptığı açıyı sıfırlarsak, bu objenin ana ekseni X_Y ekseni olur.

Farz edelim ki, iki boyutlu bir X_Y uzayındaki herhangi bir objenin ana eksenleri X'_Y' olsun ve bu eksenler X_Y eksenleriyle θ açısını yapsın. (x_1,y_1) noktası bu objeye ait herhangi bir nokta ve (x_1',y_1') noktası da bu noktanın X'_Y' uzayına göre konumu olsun.



Şekil 3.3. İki boyutlu bir objenin açısal konumu

dönüştürülmüş koordinatlar:

$$x_1' = c + d, y_1' = a - b \qquad (3\text{-}23)$$

$$x_1' = y_1 Sin\,\theta + x_1 Cos\,\theta \qquad (3\text{-}24)$$

$$y_1' = y_1 Cos\,\theta - x_1 Sin\,\theta; \qquad (3\text{-}25)$$

olur.

Peki bir objenin ana ekseni nedir ?

Bir objenin ana ekseni, o objenin bütün noktalarının bu eksene olan dik uzaklıklarının kareleri toplamını minimum yapan eksendir. O zaman aşağıdaki denklemi minimum yapan θ açısı, X'_Y' uzayının X_Y ile yaptığı dar açıdır.

$$X(\theta) = \sum_{x \in R} \sum_{y \in R} (y_1 Cos\theta - x_1 Sin\theta)^2 f(x, y) \qquad (3\text{-}26)$$

Bu denklemin minimumunu bulmak için türevini sıfıra eşitlersek;

$$\frac{d[\bar{X}(\theta)]}{d\theta} = \sum 2(y_1 Cos\theta - x_1 Sin\theta)(y_1 Sin\theta + x_1 Cos\theta) f(x,y) = 0 \qquad (3\text{-}27)$$

$$\sum (y_1^2 Cos\theta Sin\theta + x_1 y_1 (Cos^2\theta - Sin^2\theta) - x_1^2 Cos\theta Sin\theta) f(x,y) = 0 \qquad (3\text{-}28)$$

$$\sum (0.5 y_1^2 Sin2\theta + x_1 y_1 Cos2\theta - 0.5 x_1^2 Sin2\theta) f(x,y) = 0 \qquad (3\text{-}29)$$

$$\tan 2\theta = \frac{2\sum x_1 y_1 f(x,y)}{\sum (x_1^2 - y_1^2) f(x,y)} \qquad (3\text{-}30)$$

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\sum x_1 y_1 f(x,y)}{\sum (x_1^2 - y_1^2) f(x,y)}\right] \qquad (3\text{-}31)$$

Bu ifadeyi, merkezi momentler cinsinden ifade etmek için aşağıdaki üç moment ifadesini kullanırız.

$$\mu_{11} = \sum_x \sum_y (x - x')^1 (y - y')^1 f(x,y) \qquad (3\text{-}32)$$

$$\mu_{20} = \sum_x \sum_y (x - x')^2 f(x,y) \qquad (3\text{-}33)$$

$$\mu_{02} = \sum_x \sum_y (y - y')^2 f(x,y) \qquad (3\text{-}34)$$

$$\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} = 2\frac{\sum_x \sum_y (x - x')(y - y') f(x,y)}{\sum_x \sum_y ([(x - x')^2 - (y - y')^2] f(x,y))} \qquad (3\text{-}35)$$

$$\theta = \frac{1}{2}\arctan\left[\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right] \qquad (3\text{-}36)$$

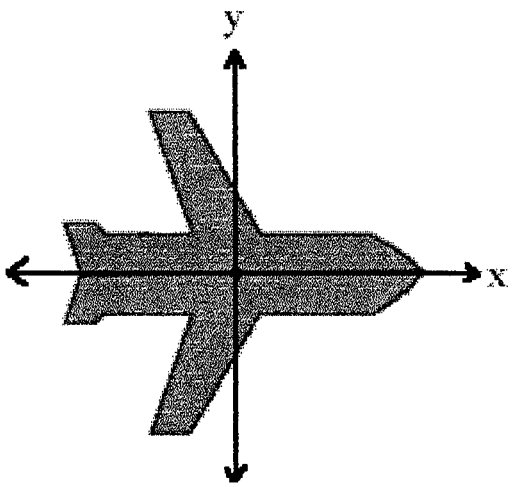
Bu durumda, bir görüntüdeki obje ne kadar döndürülürse döndürülsün, şeklin yatay eksenle yaptığı bu açıyı hesaplayarak, dönme nedeniyle değişen momentleri sabitleyebiliriz. Moment denkleminde x,y yerine '-θ' ile döndürülmüş koordinatlar konulursa moment sabitleri dönme etkisine karşı sabit kalırlar.

Merkezi momentleri dönme etkisine karşı sabitleyen denklem aşağıdaki gibi verilir.

$$\varphi_{pq} = \sum_{r=0}^{p}\sum_{s=0}^{q}(-1)^{q-s}C_r^p C_s^q (Cos\theta)^{p-r+s}(Sin\theta)^{q-s+r}\mu_{p-r+q-s,r+s} \qquad (3\text{-}37)$$

$p, q = 0,1,2,3,.....$

$$C_r^p = \frac{p!}{r!(p-r)!} \qquad (3\text{-}38)$$



Şekil 3.4. Kayma ve dönme etkilerine karşı sabitlenmiş f(x,y)

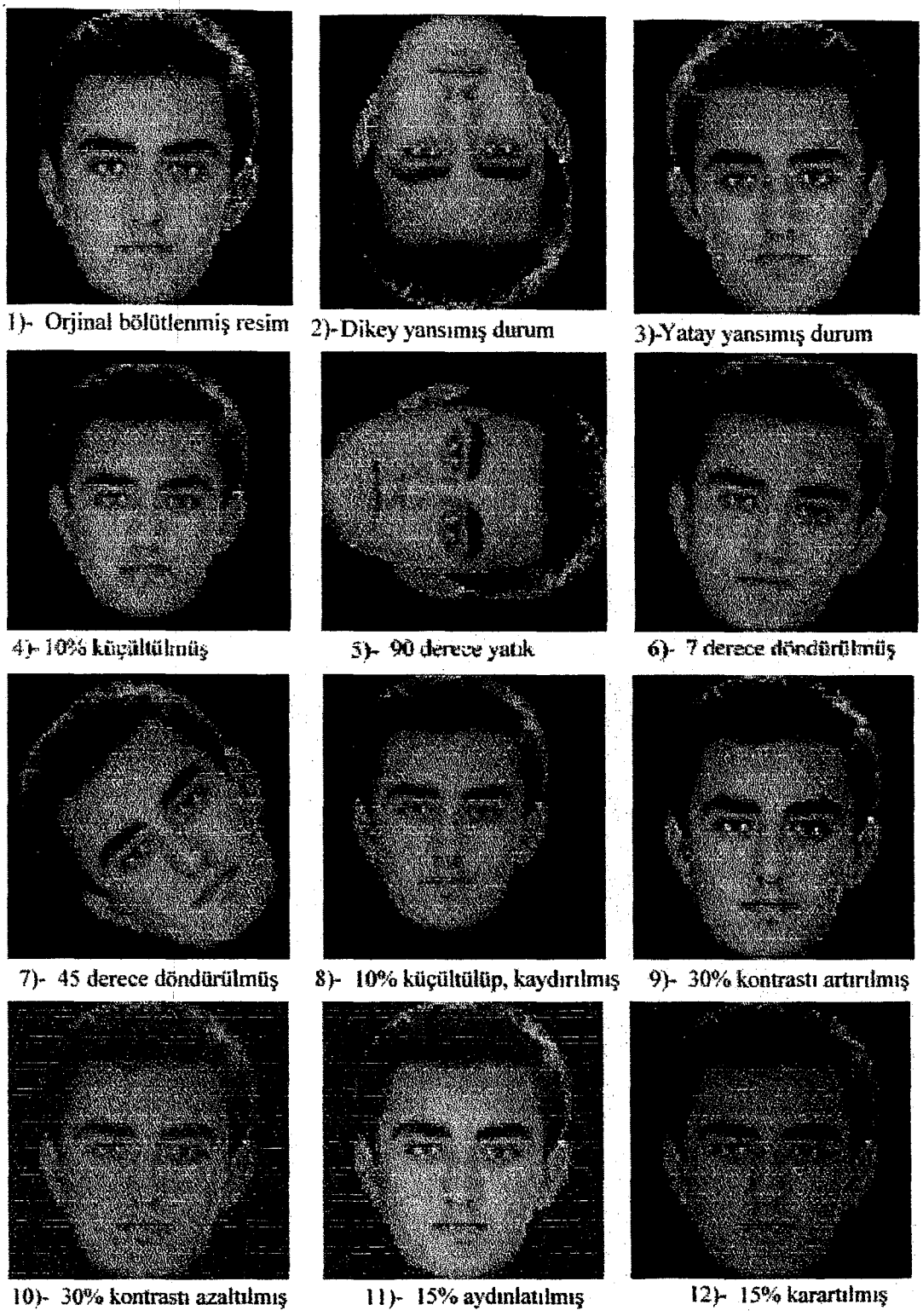
3.3.3 Ölçekleme

Görüntüdeki f(x,y) şeklinin büyüme veya küçülmesinin, yukarıdaki momentlere (ϕ_{pq}) olan etkisini gidermek için bu değeri şeklin sıfırıncı dereceden momentiyle bölerek normalize ederiz. Böylelikle tüm değişimlere karşı sabit olan standart momentleri buluruz.

$$N_{pq} = \frac{\varphi_{pq}}{\varphi_{00}^{\ \gamma}} \qquad (3\text{-}39)$$

$$\gamma = \frac{1}{2}(p+q)+1 \qquad (3\text{-}40)$$

Sırasıyla, kayma, dönme ve ölçekleme değişimlerine karşı sabitlenen standart momentlerin birinci dereceden sabitleri sıfırdır ($N_{01}=0, N_{10}=0$) çünkü birinci derece merkezi momentler de sıfırdır. Boyut sabitleme işlemi nedeniyle N_{00}'da daima '1' olur. ($N_{00}=1$) Dönme etkisini sabitleme işlemi nedeniyle N_{11} değeri de daima sıfırdır. ($N_{11}=0$) Geriye kalan standart moment sabitleri ise gri seviye görüntülere uygulanabilen görüntü tanımlayıcıları olarak kullanılabilir.



Şekil 3.5. Yüz resmi ve deforme edilmiş görüntüleri

Tablo 3.2. Şekil 3.5'deki görüntülere ait standart moment vektörleri

	N_{20}	N_{02}	N_{30}	N_{21}	N_{12}	N_{03}
1)-	9,00E-04	5,05E-04	8,22E-06	1,18E-06	1,87E-06	-5,58E-07
2)-	9,00E-04	5,05E-04	8,22E-06	-1,18E-06	1,87E-06	5,58E-07
3)-	9,00E-04	5,05E-04	-8,22E-06	1,18E-06	-1,87E-06	-5,58E-07
4)-	9,02E-04	5,08E-04	8,20E-06	1,23E-06	1,88E-06	-5,79E-07
5)-	9,00E-04	5,05E-04	8,22E-06	1,18E-06	1,87E-06	-5,58E-07
6)-	9,02E-04	5,10E-04	8,40E-06	1,13E-06	1,84E-06	-5,42E-07
7)-	9,02E-04	5,09E-04	-8,39E-06	-1,15E-06	-1,84E-06	5,47E-07
8)-	9,02E-04	5,08E-04	8,20E-06	1,23E-06	1,88E-06	-5,79E-07
9)-	8,71E-04	4,94E-04	8,76E-06	1,27E-06	1,70E-06	-2,21E-07
10)-	8,52E-04	6,87E-04	4,60E-06	2,75E-07	1,78E-06	-8,70E-07
11)-	6,75E-04	5,16E-04	3,74E-06	2,64E-07	1,37E-06	-6,57E-07
12)-	1,22E-03	6,95E-04	1,45E-05	2,11E-06	2,84E-06	-3,94E-07

Yukarıdaki görüntülere ait standart moment vektörleri incelendiğinde, dikey yansımış 2'inci görüntüde N_{21} ve N_{03} işaret değiştirmekte yani bu momentlerin sadece büyüklükleri sabit kalmakta, aynı şekilde yatay yansımış üçüncü görüntüde de N_{12} ve N_{30} değerleri işaret değiştirmektedir. 45 derece döndürülmüş yedinci görüntüde ise son dört değerin hepsi işaret değiştirmektedir. Bu işaret değişimleri sinir ağının eğitilememesine veya yanlış sınıflandırma yapmasına neden olabileceği için bu değerlerin sadece büyüklükleri ile ilgilenilmelidir.

Bu durumda bu değerlerin büyüklüklerini alarak değiştirilmiş görüntülerin moment vektörlerinin orijinal görüntünün moment vektörüne göre hatalarını hesaplarsak;

Tablo 3.3. Şekil 3.5'deki görüntülere ait standart moment vektörleri ve hata değerleri

	N_{20}	N_{02}	N_{30}	N_{21}	N_{12}	N_{03}	Hata
1)-	9,00E-04	5,05E-04	8,22E-06	1,18E-06	1,87E-06	5,58E-07	
2)-	9,00E-04	5,05E-04	8,22E-06	1,18E-06	1,87E-06	5,58E-07	5,88E-22
3)-	9,00E-04	5,05E-04	8,22E-06	1,18E-06	1,87E-06	5,58E-07	6,23E-23
4)-	9,02E-04	5,08E-04	8,20E-06	1,23E-06	1,88E-06	5,79E-07	2,86E-06
5)-	9,00E-04	5,05E-04	8,22E-06	1,18E-06	1,87E-06	5,58E-07	1,35E-20
6)-	9,02E-04	5,10E-04	8,40E-06	1,13E-06	1,84E-06	5,42E-07	5,01E-06
7)-	9,02E-04	5,09E-04	8,39E-06	1,15E-06	1,84E-06	5,47E-07	4,55E-06
8)-	9,02E-04	5,08E-04	8,20E-06	1,23E-06	1,88E-06	5,79E-07	2,86E-06
9)-	8,71E-04	4,94E-04	8,76E-06	1,27E-06	1,70E-06	2,21E-07	3,16E-05
10)-	8,52E-04	6,87E-04	4,60E-06	2,75E-07	1,78E-06	8,70E-07	1,88E-04
11)-	6,75E-04	5,16E-04	3,74E-06	2,64E-07	1,37E-06	6,57E-07	2,26E-04
12)-	1,22E-03	6,95E-04	1,45E-05	2,11E-06	2,84E-06	3,94E-07	3,75E-04

Değiştirilmiş görüntülerin hata değerleri incelendiğinde, dikey, yatay yansıma ve 90 derecelik döndürme durumunda meydana gelen hata neredeyse sıfır, fakat diğer değişimlerde

biraz daha fazladır, özellikle görüntüye ışık değişimleri uygulandığında (9,10,11,12) hata oranlarının biraz daha fazla olduğu görülmektedir.

Bu sonuçlardan şu çıkarılabilir: Hatanın iki önemli nedeni vardır. İlki, görüntünün kesikleştirme hatalarından kaynaklanan kayıpların, özellikle yüksek dereceli moment sabitlerinde (N_{03},N_{30},N_{21},N_{12}) belirginleşmesidir. İkincisi, görüntüdeki objenin aydınlatılma şekli ve şiddeti ve görüntü algılayıcı sistemin kontrast veya parlaklık ayarıdır.

Yukarıdaki tablolardan görüleceği gibi, moment sabitleri normalizasyon yani ölçekleme değişimlerine karşı yapılan sabitleme işlemi nedeniyle gayet küçük değerler alırlar. Bu sabitlerin, bu şekilde sinir ağına girilmesi, önemli eğitim ve sınıflandırma hatalarına yol açabilir, çünkü sistemin eğitimi esnasında milyarlarca aritmetik işlem yapılmaktadır ve bu işlemlerde hassasiyet mutlaka olmalıdır. Moment sabitleri, genellikle sınıflandırılması çok zor vektörler oluşturmaktadır, çoğu zaman iki farklı şablon gurubunun farklılık bilgisi, çok küçük ondalık hanelerde saklıdır. Bu hanelerin ihmali, büyük hatalara yol açabilir. Bu nedenle moment sabitlerinin bu değerleri, yapay sinir ağında kullanılabilecek dinamik bir aralığa aktarılmalıdır. Bu işlemi bir logaritmik fonksiyonla yapabiliriz.

4. ÇOK KATMANLI İLERİ-BESLEMELİ YAPAY SİNİR AĞI MODELİ VE EĞİTİLMESİ

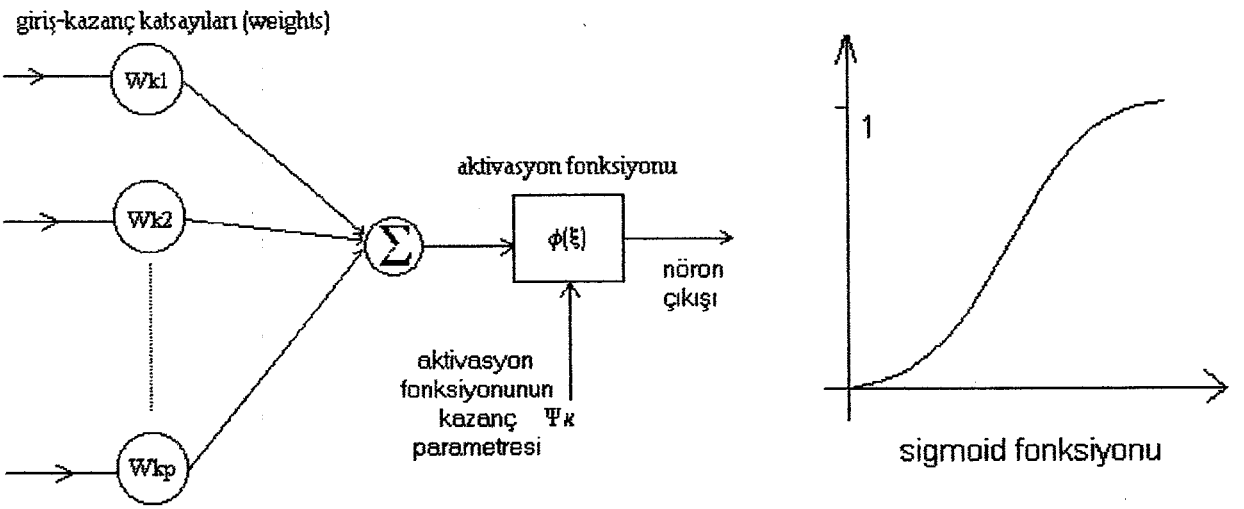
İleri beslemeli yapay sinir ağlarının (Multilayer Feedforward Neural Networks) basit modellerinin ilk çalışmaları, doğrusal fark denklemlerinin kullanımı ve eğitimi amacıyla yapılmıştır. İlk olarak Rosenblatt tarafından 1959 yılında yapılan basit çalışmaların ardından, İleri-beslemeli yapay sinir ağlarının genel modelleri üzerindeki çalışmalar Widrow/Hoff tarafından 1960'da ve Nilsson tarafından 1965'de yapılmıştır. Bu çalışmalar Minsky/Papert tarafından 1969 'da daha da ilerletilmiştir.

İleri beslemeli yapay sinir ağlarının, sınıflandırma ve tanıma amacıyla kullanımı için diğer yaklaşımlarla karşılaştırmasını, 1987'de Huang/Lippmann tarafından yapılmıştır. Daha sonra yine 1987'de Wieland/Leighton tarafından yapılan bir geometrik analizin ardından, 1990'da, bu modelin dijital/analog devrelerle gerçekleştirilmesinin, ağırlık hassasiyeti nedeniyle sınırlı olduğunu Stevenson göstermiştir.

Biyolojik nöron yapısına dayanan yapay nöron modeli, yapay sinir ağlarının temel elemanlarıdır. Bu hücrelerden oluşan sinir katmanlarının birkaç tanesinin arka arkaya sıralanması ile çok katmanlı sinir ağları oluşturulur.

En basit bir yapay sinir ağı modeli tek bir nörondan oluşur. Bu yapı ise en fazla iki ayrı doğrusal ayrıştırılabilen şablon gurubunu sınıflandırabilir.

Tek katmanlı yapay sinir ağı modeli ise, birden fazla nörondan oluşmuş bir sinir katmanıdır ve doğrusal ayrıştırılabilir şablon guruplarını sınıflandırır.



Şekil 4.1. Yapay nöron ve aktivasyon fonksiyonu

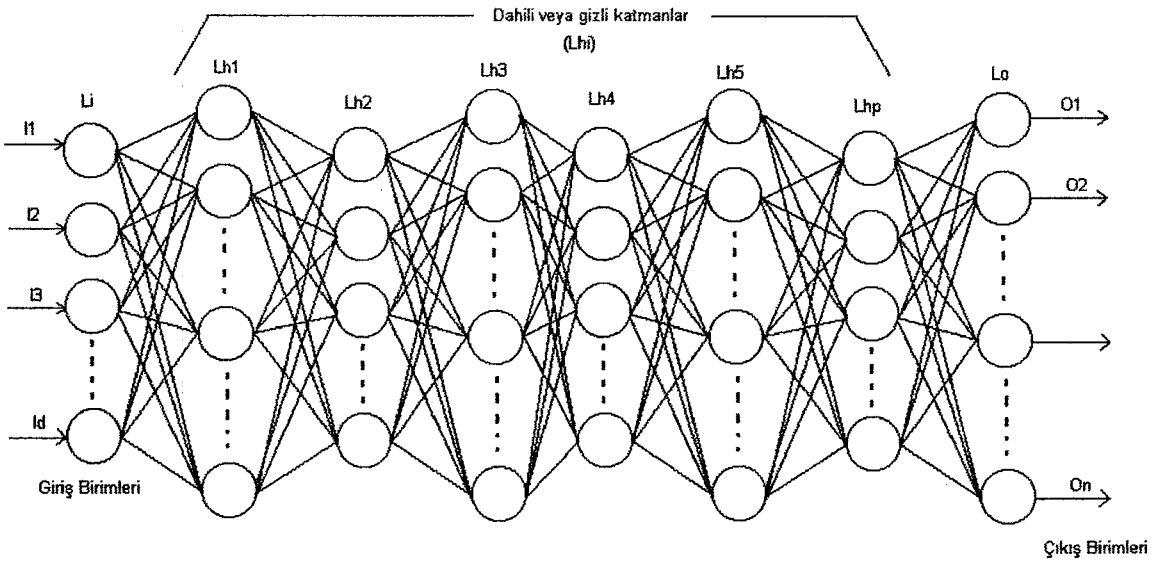
Bu iki model de, sadece doğrusal ayrıştırılabilir şablon guruplarını sınıflandırabilirler. Çünkü bir nöronun karar verebilmek için aşağıdaki basit denklemi uygulaması gerekir.

$$v_j = \sum_i W_{ji} x_i - \theta \qquad (4\text{-}1)$$

X giriş vektörü, nöronun W vektörüyle skalar çarpılarak doğrusal bir işleme tabi tutulup bulunan sonuç doğrusal olmayan bir aktivasyon fonksiyonuna girilir. Sistem tek katmanlı olduğunda doğrusal olmayan çıkışlar, başka bir nörona girmediği için sistem doğrusal oluyor.

Çok katmanlı, ileri beslemeli bir yapay sinir ağında, bu sinir katmanları, paralel bir bağlantı yapısıyla, kendinden sonraki sinir katmanına bağlanır. Giriş katmanı ve çıkış katmanı arasında en az bir ara katman (gizli katmanlar) bulunması gerekir.

Çok katmanlı bir yapay sinir ağı modeli doğrusal olmayan şablon guruplarını sınıflandırabilir yani kompleks giriş-çıkış haritalamasını (mapping) yapabilmektedir. Çünkü doğrusal olmayan bir hücre çıkışı, diğer hücrelere giriş olarak girildiği için sistem yüksek kompleks sınıflandırma performansına ulaşabilir.



Şekil 4.2. İleri beslemeli yapay sinir ağı

Çok katmanlı yapay sinir ağının üç temel karakteristiği vardır.

1. Sinir ağındaki her nöron, yumuşak bir doğrusal olmayan karakteristik gösterir.
2. Gizli katmanlar, karmaşık işleri öğrenmesine olanak sağlar, giriş uzayını kendi uzayına taşırken önemsiz ayrıntıları ihmal eder. Böylelikle, giriş vektörlerinden, birçok anlamlı özellikler ayrıştırılır.
3. Sinir ağı, hücrelerin birbiriyle bağlantı şekilleri nedeniyle yüksek derecede bağlamsallık (connectivity) özelliği gösterir.

4.1 Çok Katmanlı İleri-Beslemeli Yapay Sinir Ağının Eğitilmesi

4.1.1 Delta öğrenme kuralı

Bir nöronun eğitilmesi, bu nöronun ağırlık katsayılarının uygun değerlere ayarlanmasıdır. Nörona girilen giriş değerlerine karşılık, nöronun vermesi gereken çıkış değeriyle, nöronun verdiği çıkış değerinin farkı olan hata ile orantılı olarak değiştirilen bu katsayılar, hata minimuma indirgendiği zaman eğitilmiş sayılır.

Bir yapay sinir ağı da bu mantığa göre eğitilir. Sinir ağının giriş katmanındaki hücrelere, giriş vektörü uygulanıp, sinir ağının çıkışındaki hücrelerden, çıkış vektörü alınarak, olması gereken ideal çıkış vektörüyle farkı alınır ve hata hesaplanır. Bu hata oranı sinir ağındaki her hücrenin ağırlık ve eşik değerini etkiler ve hata minimuma indirgeninceye kadar bu işlem devam eder.

4.1.1.1 Yapay sinir ağının çıkış katmanındaki herhangi bir çıkış nöronunun eğitimi

Yapay sinir ağının son katmanındaki herhangi bir j'inci nöronu ele alalım; bu hücrenin, herhangi bir n'inci eğitim iterasyonunda verdiği hata $e_j(n)$, hücrenin verdiği çıkış $y_j(n)$ ile o hücrede olması gereken çıkış $d_j(n)$ arasındaki farktır.

$$e_j(n) = y_j(n) - d_j(n) \qquad (4\text{-}2)$$

Bu çıkış hücresindeki anlık kare hatayı $\frac{1}{2}e_j^{\,2}(n)$ denklemiyle ifade edebiliriz. Aynı şekilde, bu çıkış katmanındaki tüm çıkış nöronlarının toplam anlık hatalarının karesi aşağıdaki denklemle ifade edilebilir.

$$\varepsilon(n) = \frac{1}{2}\sum_{j \in C} e_j^{\,2}(n) \qquad (4\text{-}3)$$

Burada C kümesi, çıkış katmanındaki hücreleri içerir.

Çıkış katmanının, 'n' inci iterasyon anındaki ortalama kare hatası ise;

$$\varepsilon_{av}(n) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t) \qquad (4\text{-}4)$$

denklemiyle ifade edilir. Burada 'N' sayısı, toplam iterasyon sayısıdır. Bu ortalama kare hata değeri, yapay sinir ağının öğrenme performansının bir ölçüsü olarak kullanılır. Eğitim sürecinin amacı, $\varepsilon_{av}(n)$ değerini minimuma indirgemektir.

Çıkış katmanındaki herhangi bir j'inci nöronun girişi, bir önceki katmandaki nöronların çıkışıdır. Bu nedenle n'inci iterasyonda, bu j'inci nöronun aktivasyon fonksiyonuna girecek değeri, $v_j(n)$, aşağıdaki denklemle ifade edebiliriz.

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n) \qquad (4\text{-}5)$$

Bu denklemde, w_{ij} değeri, j'inci çıkış nöronunun i'inci ağırlık değeridir. Bu değerle, bir önceki katmandaki i'inci nöronun çıkışı çarpılır. Böylelikle bir önceki katmandaki tüm nöronların çıkışı, bir sonraki katmandaki nöronların ağırlık değerleriyle çarpılıp, $v_j(n)$ değeri hesaplanır ve bu değer nöronların aktivasyon fonksiyonuna girilir. Bu fonksiyonun çıkışı hücrenin çıkışını ifade eder. Bu denklemde 'p' sayısı, bir önceki katmandaki toplam nöron sayısıdır.

$$y_j(n) = \varphi(v_j(n)) \qquad (4\text{-}6)$$

Yapay sinir ağının çıkış katmanındaki tüm hücrelerin çıkış değerleri, yukarıdaki şekilde hesaplandıktan sonra, anlık hatanın ağırlık değerlerine göre değişim hızıyla orantılı olarak sinir ağının ağırlık değerleri ayarlanır. Bu ifadeye uyan aşağıdaki türev ifadesini zincir kuralına göre ifade edebiliriz.

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \qquad (4\text{-}7)$$

Zincir kuralı:

- Diferansiyeli alınabilen bir fonksiyonun parametre olduğu başka bir fonksiyonun da diferansiyeli alınabilir.
- $\zeta = \gamma(x, y, \ldots.)$, $\lambda = \xi(x, y, \ldots..)$, ... fonksiyonları, aynı parametrelere bağlı fonksiyonlar olsun. Eğer $f(\zeta, \lambda, \ldots..)$ fonksiyonun, $\zeta, \lambda, \ldots..$ fonksiyonlarına göre diferansiyeli alınabilir ise, x,y,..... parametrelerine göre kısmi türevleri aşağıdaki gibi olur.

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \zeta}\frac{\partial \zeta}{\partial x} + \frac{\partial f}{\partial \lambda}\frac{\partial \lambda}{\partial x} + \ldots\ldots\ldots \qquad (4\text{-}8)$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial \zeta}\frac{\partial \zeta}{\partial y} + \frac{\partial f}{\partial \lambda}\frac{\partial \lambda}{\partial y} + \ldots\ldots\ldots \qquad (4\text{-}9)$$

Zincir kuralına göre anlık hatanın ağırlık değerlerine göre türevi;

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n) \partial e_j(n) \partial y_j(n) \partial v_j(n)}{\partial e_j(n) \partial y_j(n) \partial v_j(n) \partial w_{ji}(n)} \qquad (4\text{-}10)$$

olur. Bu denklemin ana elemanlarını, denklem 4-3, 4-4, 4-5, 4-6'yı kullanarak sırasıyla aşağıdaki gibi belirleriz;

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \qquad (4\text{-}11)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \qquad (4\text{-}12)$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \qquad (4\text{-}13)$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \qquad (4\text{-}14)$$

Bu sonuçlara göre gradient denklemi tekrar düzenlenirse

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi'_j(v_j(n))y_i(n) \qquad (4\text{-}15)$$

elde edilir. Ağırlıklara uygulanacak düzeltme oranı $\Delta w_{ji}(n)$ ise;

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \qquad (4\text{-}16)$$

denklemiyle belirtilir. Burada η sabiti öğrenme oranıdır ve sinir ağının eğitim performansına (eğitim hızına ve eğitim oranına) etki eden bir sayıdır. Eksi işareti, ağırlık uzayında 'gradient'

eğimini takip etmek için konulmuştur. Yukarıdaki denklemleri kullanarak ağırlık düzeltme oranını tekrar yazacak olursak

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \qquad (4\text{-}17)$$

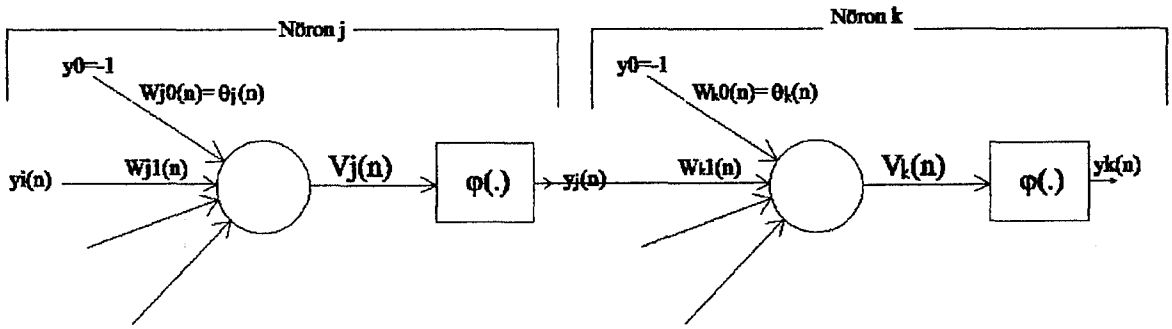
Burada, $\delta_j(n)$ fonksiyonu yerel gradient olarak tanımlanır ve aşağıdaki gibidir.

$$\delta_j(n) = -\frac{\partial \varepsilon(n) \partial e_j(n) \partial y_j(n)}{\partial e_j(n) \partial y_j(n) \partial v_j(n)} = e_j(n) \varphi_j'(v_j(n)) \qquad (4\text{-}18)$$

Şu ana kadar yazılan tüm denklemler ve bağıntılar, çıkış katmanındaki nöronlar için yazılmıştır ve bu durum oldukça basittir. Çünkü, her nöronun, sinir ağına girilen değerlere karşılık vermesi gereken değerleri vardır, yani bu nöronlar için hata kolaylıkla hesaplanmıştır.

4.1.1.2 Yapay sinir ağının gizli katmanındaki herhangi bir nöronunun eğitimi

Gizli bir katmandaki bir nöronu eğitmek için öncelikle bu nöronun çıkışının yanlış veya doğru yönlendiği zaman, bu nörona nasıl bir uyarı vereceğimizi belirlemeliyiz. Ayrıca bu nöron için olması gereken belirli bir çıkış değeri de elimizde bulunmamaktadır. Bu durumda bu gizli nöron için hata sinyali, bu nörona doğrudan bağlı tüm nöronların hata sinyallerinden hesaplanır. Çünkü eğer bu gizli nörona doğrudan bağlı diğer nöronların çıkışı hata yapmışsa, bu hatanın nedeni giriş değerlerine de bağlıdır, dolayısıyla bu giriş değerlerinin birer tanesi de bu gizli nörona ait olan çıkış değeridir, böylelikle, bu yakın ilişki nedeniyle bu gizli nörona ait hata sinyali, bu nörona doğrudan bağlı diğer nöronların hata sinyallerinden hesaplanır. Çıkış katmanındaki tüm nöronların hata sinyalleri ve düzeltilmiş ağırlık değerleri yukarıda belirtildiği gibi hesaplanmış idi, bu değerleri kullanarak bir önceki katmandaki nöronları eğitiriz ve böylelikle, katman katman geriye giderek tüm katmanlardaki nöronları eğitebiliriz.



Şekil 4.3. İleri beslemeli yapay sinir ağındaki ardışık iki nöron

Bu amaçla, yerel gradient denklemini çıkış katmanından bir önceki katmandaki j'inci nöron için tekrar düzenleyelim.

$$\delta_j(n) = -\frac{\partial \varepsilon(n) \partial y_j(n)}{\partial y_j(n) \partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \varphi_j'(v_j(n)) \qquad (4\text{-}19)$$

Bu denklemde bilinmeyen sadece $\frac{\partial \varepsilon(n)}{\partial y_j(n)}$ kısmi türevidir. Bunu bulmak için 4-20 denklemini şekil 4.3'den bakarak yazabiliriz.

$$\varepsilon(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n) \qquad (4\text{-}20)$$

Burada k'ıncı nöron çıkış nöronudur. Bu denklemin $y_j(n)$'e göre diferansiyelini alırsak;

$$\frac{\varepsilon(n)}{y_j(n)} = \sum_{k \in C} e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} \qquad (4\text{-}21)$$

$y_j(n)$'in toplam hataya olan etkisini yukarıdaki gibi ifade ettikten sonra zincir kuralını kullanarak bu denklemi aşağıdaki şekile sokabiliriz.

$$\frac{\varepsilon(n)}{y_j(n)} = \sum_{k \in C} e_k(n) \frac{\partial e_k(n) \partial v_k(n)}{\partial v_k(n) \partial y_j(n)} \qquad (4\text{-}22)$$

k'ıncı nöron çıkış nöronu olduğu için aşağıdaki denklemi yazabiliriz.

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \qquad (4\text{-}23)$$

Böylece;

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \qquad (4\text{-}24)$$

olur. Ayrıca, k'ıncı nöronun aktivasyon fonksiyonunun parametresi olan $v_k(n)$'i aşağıdaki gibi yazarız.

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \qquad (4\text{-}25)$$

Buradaki q sayısı k'ıncı nörona uygulanan toplam giriş sayısıdır. (Eşik değerini bir giriş olarak ele almıyoruz. Çünkü eşiğe uygulanan giriş değişmiyor. Daima -1'dir.) Yukarıdaki denklemin $y_j(n)$'e göre kısmi türevi;

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \qquad (4\text{-}26)$$

olur. Şimdi bu denklemi ve $\frac{\partial e_k(n)}{\partial v_k(n)}$ denklemini kullanarak $\frac{\partial \varepsilon(n)}{\partial y_j(n)}$ denklemini yeniden yazarsak;

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = -\sum_{k \in C} e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) \qquad (4\text{-}27)$$

$$= -\sum_{k \in C} \delta_k(n) w_{kj}(n) \qquad (4\text{-}28)$$

Son olarak bu denklemi kullanarak j'inci nöronun yerel gradient'ini yazarsak;

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \qquad (4\text{-}29)$$

Sonuç olarak, delta kuralını genelleştirirsek;

Nöron j'nin i'inci girişine bağlı olan bir önceki katmandaki i'inci nöronun, bu j'inci nörona ait w_{ji} ağırlık değerine yapacağı düzeltme miktarı aşağıdaki gibidir.

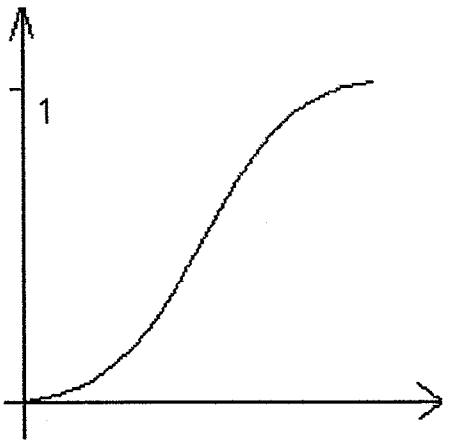
$$\begin{Bmatrix} \text{Ağırlık} \\ \text{Düzeltmesi} \\ \Delta w_{ji}(n) \end{Bmatrix} = \begin{Bmatrix} \text{Öğrenme} \\ \text{oranı} \\ \text{parametresi} \\ \eta \end{Bmatrix} . \begin{Bmatrix} \text{Yerel} \\ \text{Gradient} \\ \delta_j(n) \end{Bmatrix} . \begin{Bmatrix} \text{j'inci} \\ \text{nöronun} \\ \text{giriş} \\ \text{sinyali} \\ y_i(n) \end{Bmatrix}$$

Yerel gradient, iki faktöre bağlıdır.

1. Eğer j'inci nöron bir çıkış nöronu ise $\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$ (4-30)
2. Eğer j'inci nöron, bir gizli nöron ise $\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$ (4-31)

4.1.1.3 $\varphi_j'(v_j(n))$'in hesaplanması

Yerel gradient $\delta_j(n)$'i hesaplamak için aktivasyon fonksiyonunun türevinin $\varphi_j'(v_j(n))$ hesaplanması gerekir. Aktivasyon fonksiyonu sürekli ve türevi olan bir sigmoid fonksiyonu olmalıdır.



Şekil 4.4. Sigmoid fonksiyonu

Aktivasyon fonksiyonu olarak değişik fonksiyonlar kullanılmakla beraber, aşağıdaki iki doğrusal olmayan fonksiyon genel olarak kullanılmaktadır.

$$f(x) = \frac{1}{1+e^{-x}} \quad (4\text{-}32) \qquad \text{veya}$$

$$f(x) = \tanh\left[\frac{x}{2}\right] = \frac{1-e^{-x}}{1+e^{-x}} \quad (4\text{-}33)$$

İkinci fonksiyonun hesaplama zorluğu vardır, çünkü iki tane eksponansiyel işlem vardır. Bu çalışmada eğitim esnasında sayısal hassasiyetin korunması açısından ilk fonksiyon kullanılmıştır. Bu fonksiyonun türevi;

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} \qquad (4\text{-}34)$$

dır. Bu türev fonksiyonunu, aktivasyon fonksiyonu cinsinden yazarsak;

$$f'(x) = f(x)[1 - f(x)] \qquad (4\text{-}35)$$

dır. Aktivasyon fonksiyonu ve türevini tekrar yazarsak;

$$\varphi_j(v_j(n)) = \frac{1}{1+e^{-v_j(n)}} = y_j(n) \qquad (4\text{-}36)$$

$$\varphi_j'(v_j(n)) = y_j(n)[1 - y_j(n)] \qquad (4\text{-}37)$$

Bu iki denklemi kullanarak yerel gradient'i tekrar yazalım.

j'inci nöron çıkış nöronu ise;

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) = [d_j(n) - y_j(n)].y_j(n).[1 - y_j(n)] \qquad (4\text{-}38)$$

j'inci nöron gizli nöron ise;

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_{k\in C}\delta_k(n)w_{kj}(n) = y_j(n)[1 - y_j(n)]\sum_{k\in C}\delta_k(n)w_{kj}(n) \qquad (4\text{-}39)$$

olur.

4.1.2 Genelleştirilmiş delta kuralı

Öğrenme oranı parametresi η'nün öğrenme hızına ve miktarına etkisi büyüktür. Eğer η'yü küçük bir değer alırsak, nöronların ağırlık sabitlerindeki değişim miktarı da (Δw_{ji}) küçük olur, dolayısıyla sinir ağının öğrenme hızı ve miktarı düşük olacaktır. Diğer yandan, η sabitini büyük seçersek eğitimde kararsızlıklar ve salınımlar oluşacaktır.

Hem öğrenme hızını artırmak, hem de bu kararsızlıkları önlemek için delta kuralına ufak bir ekleme yapılır. Bir momentum sabiti (α) içeren aşağıdaki yeni Δw_{ji} terimine genelleştirilmiş delta kuralı denir.

$$\Delta w_{ji}(n) = \alpha\Delta w_{ji}(n-1) + \eta\delta_j(n)y_j(n) \qquad (4\text{-}40)$$

Momentum sabiti (α) genellikle pozitif bir sayıdır. Yeni Δw_{ji} teriminin etkisini incelemek için bu denklemi birinci derece diferansiyel denklem olarak kabul edip çözersek, aşağıdaki denklem elde edilir.

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_j(t) \qquad (4\text{-}41)$$

Bu denkleme $\delta_j(t) y_i(t) = -\frac{\partial \varepsilon(t)}{\partial w_{ji}(t)}$ terimini koyarsak;

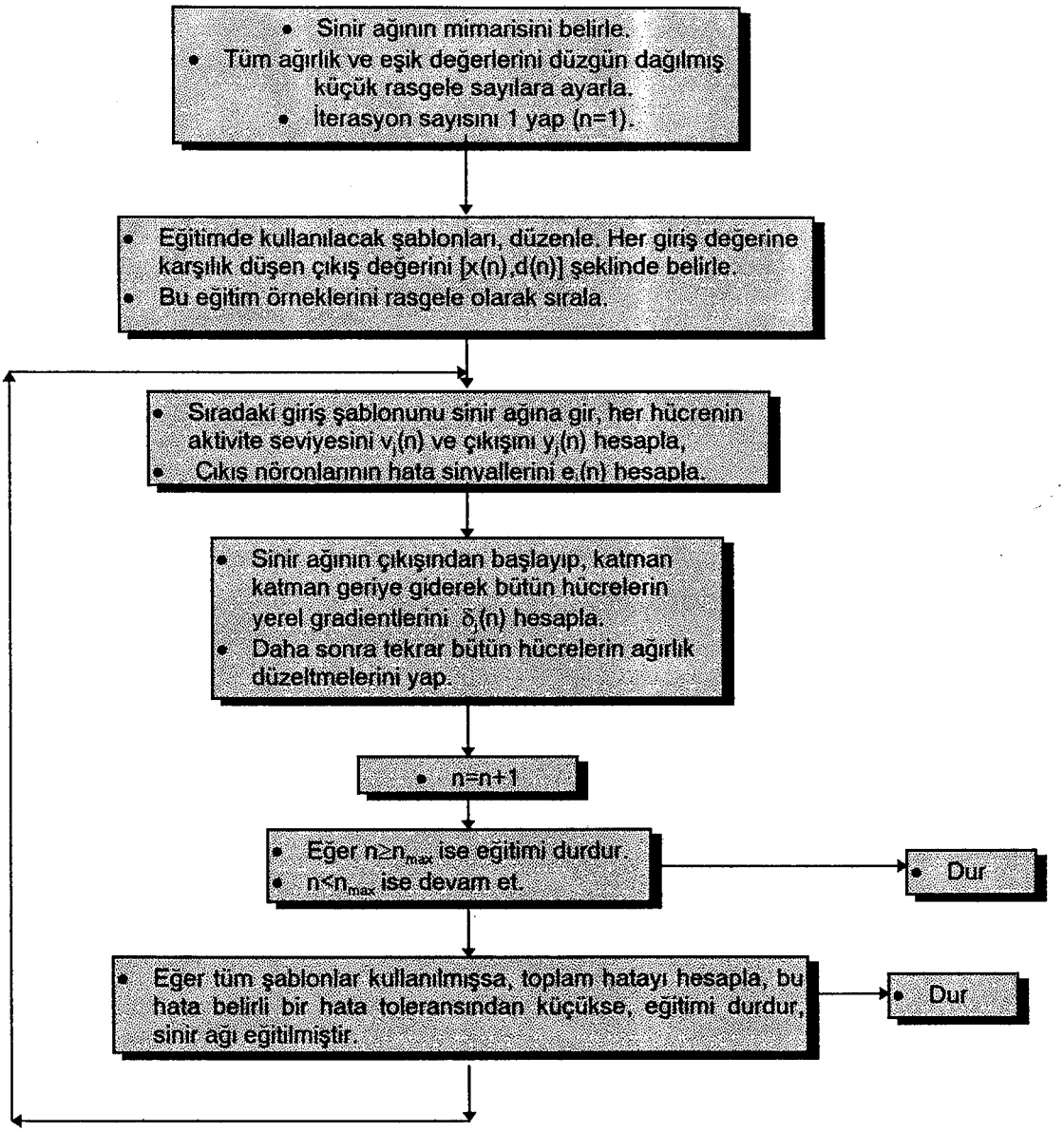
$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial \varepsilon(t)}{\partial w_{ji}(t)} \qquad (4\text{-}42)$$

olur. Bu denklemi aşağıdaki gibi yorumlayabiliriz.

- Δw_{ji} denklemi, eksponansiyel karakterli bir zaman serisidir. Bu serinin yakınsaması için momentum sabitinin [0 1] aralığında olması gerekir.
- Eğer bir nöronun çıkışındaki hata hızlı bir şekilde büyüyorsa, düzeltme miktarının da ($\Delta w_{ji}(n)$) hızlı büyümesi lazım ki hataya yetişip engellesin, bu nedenle ilaveten eklenen $\alpha \Delta w_{ji}(n-1)$ terimi bu hataya yetişmede yardımcı bir terimdir.
- Eğer bir nöronun çıkışındaki hata ardışık iterasyonlarda salınımlar yapıyorsa, düzeltme miktarı da (Δw_{ji}) ardışık iterasyonlarda işaret değiştirir. Dolayısıyla ilaveten eklenen $\alpha(\Delta w_{ji}(n-1))$ terimi, ($\Delta w_{ji}(n)$) terimini stabilize eder (salınımları frenler), böylelikle düzeltme miktarının salınımlarının önlenmesi, çıkıştaki salınımları da söndürür.

4.2 Çok Katmanlı Yapay Sinir Ağlarının Eğitimi İçin Geri Yayılım (Backpropagation) Algoritması

Yukarıda anlatılan genelleştirilmiş delta öğrenme kuralı, çok katmanlı, ileri beslemeli bir yapay sinir ağının eğitimi için aşağıda belirtilen geri yayılım algoritmasıyla kullanılır.



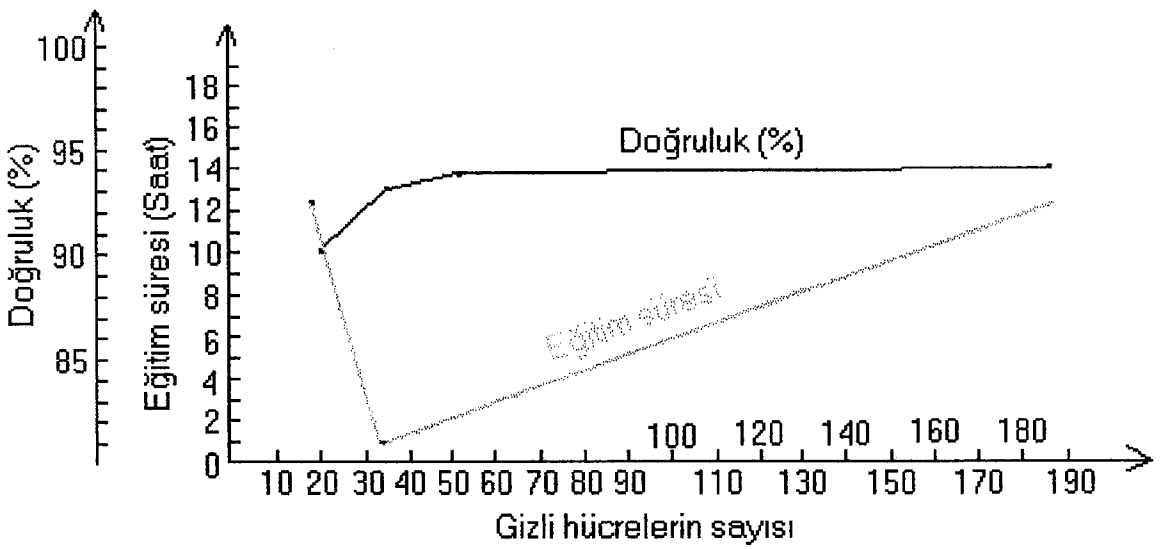
Şekil 4.5. Geri yayılım algoritması

4.2.1 Algoritmanın açıklaması

Algoritmada yapılması gereken ilk şey, yapay sinir ağının yapısını belirlemektir. Öncelikle, giriş katmanında, giriş vektörünün boyutu kadar nöron olmalıdır. Çıkış katmanında ise çıkış vektörünün boyutu kadar nöron olmalıdır. Gizli katmanların sayısı veya bu katmanlardaki nöron sayısı için optimal bir çözüm yoktur. Bu seçim için önemli bir kriter olarak ;

- Gizli katmanlar, kendilerine gelen sinyalleri tekrar haritalamalıdır (remap), yani, kendinden önceki katmanlarda sınıflandırılan veya ayrıştırılan verileri, daha çok sınıflandırabilmeli veya ayrıştırabilmelidir.

Optimal mimariye ulaşabilmek için, sinir ağının performansı deneysel olarak gözlenmelidir. Gizli katman ve nöron sayısını artırarak, performanstaki değişim gözlenir. Bu performans belirli bir sınırdan sonra düşmeye başlar. Aşağıdaki şekilde böyle bir çalışmaya bir örnek gösterilmiştir.



Şekil 4.6. Özel bir problem için, yapay sinir ağının eğitim süresi, tanıma doğruluğu ve gizli birimlerin sayısının karşılaştırılması. [5]

Yapay sinir ağının öğrenme oranı parametresi ve momentum parametresinin seçimi de eğitimin takip ettiği hata yüzeyine (error surface) bağlıdır. Bu hata yüzeyi, eğitici örneklere, sinir ağının mimarisi, başlangıç değerleri gibi birçok faktöre bağlıdır. İyi bir eğitim hızı ve öğrenme oranı elde edebilmek için, hatanın sıfıra çabuk yakınsaması ve değerinin düşük olması gerekir. Eğer öğrenme oranı parametresini (η) düşük seçersek eğitimin yakınsama hızı düşer, ama momentum parametresini 1'e yakın seçersek bu hızı artırırız.. Eğer büyük öğrenme oranı ve büyük momentum seçersek çıkıştaki ortalama hata, salınımlar

yapmaya başlayacaktır. Deneysel olarak, öğrenme oranı parametresinin 0.1'den küçük, momentum parametresinin de 0.8'den büyük seçilmesi genel bir çözüm olarak önerilebilir.

Yapay sinir ağının ağırlık ve eşik katsayılarının, düzgün dağılmış, küçük rasgele değerlere ayarlanması gerekir. Bu şekildeki bir ayarlama, sinir ağının çıkışındaki ortalama hatayı minimuma indirir. Yanlış bir ayarlama yapılması durumunda, hücre çıkışının daima sabit kalması gibi bir duruma neden olabilir. Örneğin, [-1,+1] aralığında değişen bir bipolar sigmoid fonksiyonlu bir çıkış hücresini ele alalım. Eğer bu hücrenin verdiği çıkış +1 iken vermesi gereken çıkış -1 ise ve o an için hücrenin ağırlık vektörü küçük rakamlardan oluşuyorsa, bu değerleri kolay kolay düzeltemeyecektir. Çünkü aktivite seviyesi de düşük olacağından ağırlık düzeltme miktarı da küçük olur. Bu duruma hücrenin doyuma uğraması denir. Bunu önlemek için başlangıçta ağırlık ve eşikler düzgün dağılmış küçük sabitlere ayarlanmalıdır.

Daha sonra algoritmanın döngüsüne girilmeden önce giriş ve çıkış şablonları ayarlanıp, düzene sokulmalıdır. Her giriş vektörünün vermesi gereken çıkış vektörü belirlenir, ve bu iki vektör tek bir eğitim şablonunu oluşturur. Tüm eğitim verileri bu şekilde düzenlendikten sonra tüm şablonların diziliş sırası rasgele olarak karıştırılmalıdır. Örneğin, bir X sınıfına ait birçok şablon arka arkaya sıralanmamalıdır. Tüm şablonlar rasgele olarak sıralandıktan sonra sırayla sinir ağına girilir. Bu şablonların hepsi kullanıldıktan sonra, tekrar rasgele olarak sıralanması, eğitim performansını artıracak bir sonuç verebilir çünkü sinir ağının ezberlemesini engelleyerek, bir genelleştirme yapmasını sağlayabilir.

Geri yayılım algoritmasının döngüsüne girildiğinde ilk önce, sıradaki giriş vektörü sinir ağına girilir ve tüm hücrelerin aktivite seviyeleri aşağıdaki denklemde belirtildiği gibi hesaplanır.

$$v_j^{(l)}(n) = \sum_{i=0}^{p} w_{ji}^{(l)}(n).y_i^{(l-1)}(n) \qquad (4\text{-}43)$$

Burada l 'inci katmandaki j'inci nöronun aktivite seviyesi hesaplanır. 'p' sayısı (l-1)'inci katmandaki nöron sayısıdır ve $y_i^{(l-1)}{}_{(n)}$ ifadesi, (l-1) inci katmandaki i'inci nöronun çıkışıdır. '$w_{ji}^{(l)}{}_{(n)}$ ifadesi ise j'inci nöronun i'inci ağırlık değeridir. 'i=0' için daima $y_0^{(l-1)}{}_{(n)}$=-1'dir, çünkü bu değer, j'inci hücrenin eşik seviyesinin işaretidir. Eğer l=1 ise yani ilk gizli katman ise, $y_j^{(0)}=x_j(n)$ olur. Buradan da anlaşılacağı gibi giriş katmanının ağırlık değerleri yoktur, bu katmanın görevi giriş vektörünü sonraki katmana dağıtmaktır. Tüm hücrelerin aktivite seviyesi hesaplandıktan sonra bu hücrelerin sigmoidal doğrusalsızlığı aşağıdaki gibi bulunur.

$$y_j^{(l)}(n) = \frac{1}{1+e^{-v_j^{(l)}(n)}} \qquad (4\text{-}44)$$

Eğer j'inci nöron giriş katmanında (l=0) ise $y_j^{(0)}=x_j(n)$ olur. Çünkü bu katman sadece girişleri sonraki katmana dağıtır. Ayrıca nöron j eğer çıkış katmanında ise çıkıştaki hata bulunmalıdır.

$$e_j(n) = y_j(n) - d_j^{(L)}(n) \qquad (4\text{-}45)$$

Burada 'L' sayısı sinir ağının toplam katman sayısıdır. Tüm hücreler için çıkış değerleri ($y_0^{(l)}{}_{(n)}$) hesaplanıp sinir ağının çıkışındaki hata bulunduktan sonra, bu sefer çıkış katmanından başlayarak katman katman geriye giderek tüm hücrelerin yerel gradientleri hesaplanır. Eğer yerel gradienti hesaplanacak j'inci hücre çıkış katmanında ise bu hücreye aşağıdaki yerel gradient denklemi uygulanır.

$$\delta_j^{(L)}(n) = e_j^{(L)}(n)\varphi_j'(v_j^{(L)}(n)) = e_j^{(L)}(n).y_j^{(L)}(n).[1 - y_j^{(L)}(n)] \qquad (4\text{-}46)$$

Eğer j'inci nöron bir gizli katmanda ise

$$\delta_j^{(l)}(n) = \varphi_j'(v_j^{(l)}(n))\sum_{k\in C}\delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n) \qquad (4\text{-}47)$$

$$\delta_j^{(l)}(n) = y_j^{(l)}(n)[1 - y_j^{(l)}(n)]\sum_{k\in C}\delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n) \qquad (4\text{-}48)$$

olur. Bu şekilde katman katman geriye giderek tüm hücrelerin yerel gradienti hesaplandıktan sonra, bu hücrelerin ağırlık vektörlerine aşağıdaki düzeltme denklemi uygulanır.

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n) - w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n)y_j^{(l)}(n) \qquad (4\text{-}49)$$

Daha sonra, ikinci giriş vektörünü, sinir ağına girmek için, algoritmanın üçüncü adımının başına dönülür. Tüm giriş vektörleri kullanıldıktan sonra, bu vektörlerin sırası rasgele olarak tekrar karıştırılmalıdır. Sinir ağının çıkışındaki toplam hata istenen değerin altına indiğinde eğitim durdurulur, sinir ağı eğitilmiş demektir. Eğer iterasyon sayısı belirlenen değerin üzerine çıkarsa yine algoritmadan çıkılmalıdır, aksi halde uzun süre eğitilemeyen bir sinir ağı, giriş verilerini ezberlemeye başlar. Böylelikle bir genelleştirme yapmaz. Yalnızca eğitimde kullanılan giriş vektörlerini ezberler. Bir sinir ağının ezberleme yolunu seçtiğini tespit etmek için, genelleştirme yeteneğini ölçmek gerekir. Eğitimde kullanılmamış başka bir giriş vektörü setiyle sinir ağı test edilir, eğer ezberlemişse, bu vektörlere, istenen tepkiyi gösteremez.

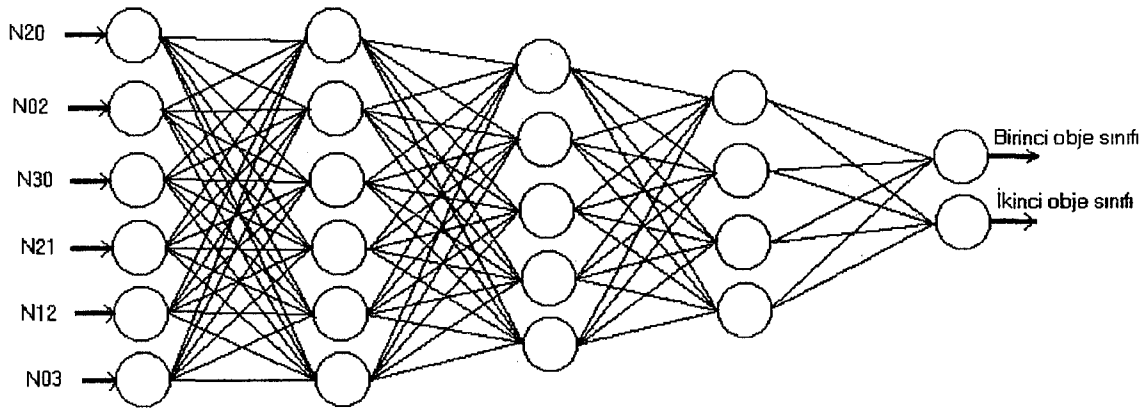
5. OBJE TANIMA UYGULAMALARI

Yukarıda tanıtılan sistemin değişik koşullara karşı olan performansını ölçmek amacıyla aşağıdaki dört uygulama yapılmıştır. Bu uygulamalarda kullanılan objelere ait bölütlenmiş görüntüler Ek-1'de gösterilmektedir. Bu uygulamalar, Ek-2'de listelenen ve C++ dilinde yazılmış programlarla gerçekleştirilmiştir.

5.1 Uygulama_1: Obje ışıklandırmasının, tanıma performansına olan etkisinin ölçümü

Bu uygulamada iki farklı koni objesi kullanılmıştır. Birinci sınıf şablonlar, üzerinde '1' rakamı bulunan ve farklı açılardan ve farklı şiddette ışıklandırılmış koni objelerinden oluşmaktadır. İkinci sınıfa ait objeler ise, üzerinde hiçbir desen bulunmayan ve aynı tip ışıklandırma ile aydınlatılmış konilerden oluşmaktadır.

Bu uygulamanın asıl amacı, ışıklandırma değişimlerinin, obje tanıma sisteminin performansını nasıl etkilediğini gözlemlemektir. Bu amaçla, birinci sınıftaki objeler farklı açılardan ışıklandırılmıştır. Bu iki sınıfa ait eğitici obje görüntüleriyle eğitilen aşağıdaki sinir ağı, bu objelerin farklı ışıklandırma, farklı ölçek, farklı açılar ve farklı konumlara sahip görüntüleriyle test edilmiştir.



Şekil 5.1. Uygulama_1'de kullanılan yapay sinir ağı

Kullanılan yapay sinir ağı, 5 katmanlıdır. İlk ve ikinci katmanında 6 nöron, üçüncüde 5, dördüncüde 4 ve beşinci katmanda da 2 nöron bulunmaktadır. Bu sinir ağı geri yayılım algoritmasıyla, ek_1'de gösterilen 138 adet eğitici obje görüntülerine ait moment tanımlayıcı vektörleri kullanılarak eğitilmiştir. Moment tanımlayıcı vektörlerinin elemanları, sinir ağına girilmeden önce f(x)=log(x) fonksiyonu ile dinamik aralığa indirgenmiştir. Her şablon sınıfı, sinir ağının çıkışında birer hücreyle temsil edilir. Giriş vektörü hangi sınıfa aitse, yapay sinir ağının çıkışında, o sınıfa tahsis edilmiş nöronun '1', diğer nöronların da '0' olması gerekir.

Her iki obje sınıfı da, koninin simetrik yapısı nedeniyle, yalnız y ekseni boyunca perspektif görüntüleri alınmıştır. Tüm görüntüler birer derecelik açılarla 45 dereceye kadar y ekseni boyunca örneklenmiştir. Birinci şablon sınıfında iki farklı ışıklandırma

olduğundan, 96 örnek kullanılmaktadır. Bu sınıfının ilk 46 tanesi, ışığı karşıdan almakta, diğer 46'sı ise sol üst köşeden almaktadır ve hepsinin üzerinde bir rakamı vardır. İkinci sınıfa ait objeler ise ışığı sadece karşıdan almaktadır ve düzgün bir yüzeye sahiptir.

Test sonucunda elde edilen performanstan da anlaşılacağı gibi, sinir ağı ezberleme yapmadan, standart moment vektörlerinden bir genelleme yapmıştır.

Eğitimde kullanılan 138 adet bölütlenmiş koni görüntüsü ek-1'den görülebilir. Bu görüntülerin standart moment vektörleri aşağıdaki tablolarda listelenmiştir.

Tablo 5.1. Birinci sınıfa ait, karşıdan ışıklandırılmış şablonların moment vektörleri

001)-	7,3375	7,64	11,5801	14,7065	12,4073	14,6542
002)-	7,3295	7,6376	11,5639	14,5668	12,3558	14,572
003)-	7,3134	7,6356	11,5292	14,4706	12,3159	14,693
004)-	7,3279	7,6453	11,5413	15,9547	12,3082	16,2711
005)-	7,3127	7,6384	11,5145	14,4045	12,2836	14,5638
006)-	7,3001	7,638	11,4942	14,5006	12,2478	14,6413
007)-	7,2999	7,6335	11,4724	14,0945	12,2508	14,2728
008)-	7,2957	7,6419	11,4699	14,4504	12,24	14,5804
009)-	7,2797	7,6378	11,4523	14,3564	12,202	14,5301
010)-	7,2845	7,6245	11,4362	14,188	12,1559	14,3516
011)-	7,2759	7,6311	11,4211	14,3472	12,1707	14,5743
012)-	7,2699	7,6107	11,3904	14,4215	12,1239	14,5762
013)-	7,2544	7,6106	11,3737	14,3023	12,0741	14,4811
014)-	7,2522	7,6139	11,369	14,5166	12,092	14,6127
015)-	7,2448	7,5873	11,34	14,4515	12,0539	14,5491
016)-	7,2406	7,5871	11,3507	14,6357	12,0059	14,6465
017)-	7,2344	7,5625	11,32	14,2851	11,9683	14,4373
018)-	7,2259	7,5576	11,3153	14,0319	11,9347	14,2626
019)-	7,2208	7,5639	11,3051	14,1334	11,9562	14,2636
020)-	7,2142	7,5508	11,2855	14,2695	11,9294	14,4284
021)-	7,2105	7,5395	11,2839	14,1398	11,8936	14,4058
022)-	7,1919	7,5123	11,2436	14,0554	11,8366	14,2708
023)-	7,1948	7,4933	11,2354	13,8844	11,7971	14,0086
024)-	7,1846	7,4971	11,2181	13,9927	11,8153	14,1163
025)-	7,1846	7,4977	11,252	14,0057	11,7812	14,2478
026)-	7,176	7,4707	11,2313	13,6185	11,7327	13,8976
027)-	7,1608	7,4203	11,1781	13,7911	11,6633	13,9093
028)-	7,1565	7,4046	11,1713	13,7093	11,6358	13,899
029)-	7,149	7,4024	11,1713	13,7412	11,6196	13,8857
030)-	7,427	7,5102	12,1165	13,7057	12,9933	13,8356
031)-	7,4293	7,5086	12,08	13,3278	12,999	13,5183
032)-	7,4221	7,5317	12,0353	13,7453	12,8756	13,9185
033)-	7,4175	7,5432	11,9855	13,8711	12,826	13,9324
034)-	7,4102	7,5551	11,9421	13,7767	12,7645	13,8913
035)-	7,4042	7,5617	11,8993	13,8608	12,756	14,0431
036)-	7,3992	7,5646	11,8667	13,9347	12,6972	14,0814
037)-	7,3949	7,584	11,851	14,1703	12,6619	14,1756
038)-	7,3888	7,5818	11,807	13,868	12,6248	14,0111
039)-	7,3891	7,6072	11,8122	14,1703	12,6038	14,3319
040)-	7,3829	7,6104	11,7624	14,3539	12,5833	14,3312
041)-	7,3791	7,6157	11,7489	14,1924	12,5289	14,3699
042)-	7,3726	7,6328	11,7396	14,0303	12,5381	14,2738
043)-	7,3659	7,6302	11,6946	14,1392	12,5029	14,2879
044)-	7,3508	7,6332	11,6436	14,3809	12,4888	14,5189
045)-	7,3441	7,6467	11,6241	14,5686	12,4471	14,6381
046)-	7,3308	7,6384	11,5938	15,0151	12,4143	14,9398

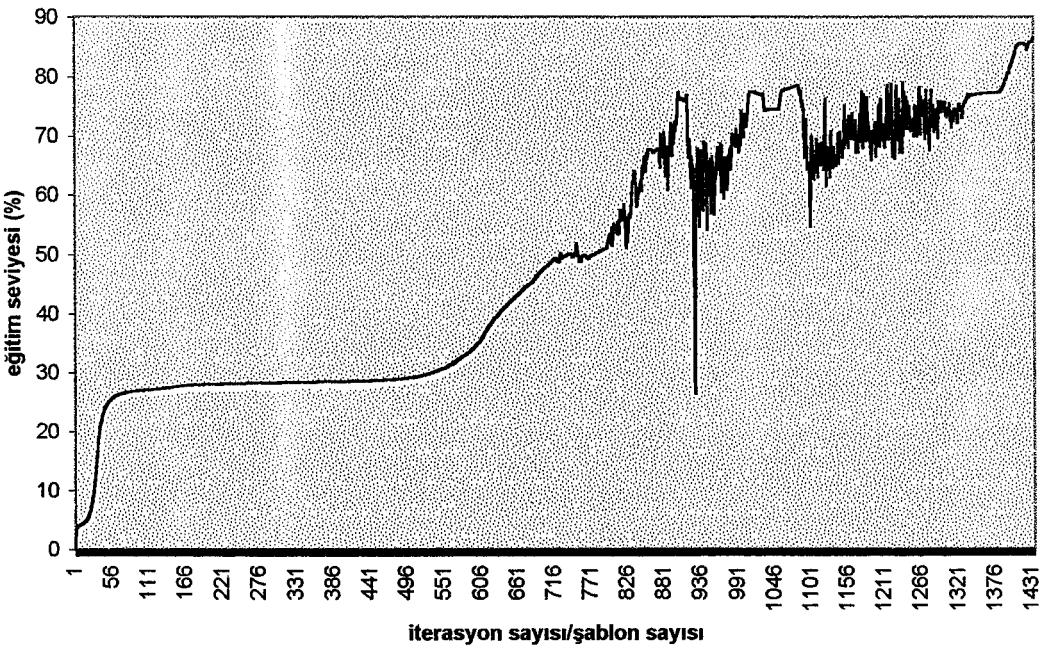
Tablo 5.2. Birinci sınıfa ait, sol üst köşeden ışıklandırılmış şablonların moment vektörleri

047)-	7,01	7,1907	11,4775	11,9615	13,4018	11,1044
048)-	7,006	7,2038	11,4536	11,9412	13,1517	11,1048
049)-	7,0016	7,2199	11,4368	11,9334	13,0349	11,1219
050)-	7,0278	7,362	11,1507	12,5518	12,2604	11,3785
051)-	7,0088	7,2387	11,4036	11,926	12,8977	11,1412
052)-	7,0038	7,2515	11,3546	11,9581	12,6529	11,1592
053)-	7,0062	7,262	11,3561	11,9117	12,7731	11,1606
054)-	7,0128	7,2711	11,3285	11,9652	12,5658	11,1898
055)-	7,0069	7,287	11,341	11,9372	12,5716	11,1973
056)-	7,0154	7,289	11,3665	11,8578	12,7144	11,1661
057)-	7,0169	7,3049	11,3158	11,9216	12,536	11,2192
058)-	7,0199	7,3058	11,3172	11,8812	12,5756	11,2064
059)-	7,0187	7,322	11,338	11,8605	12,5562	11,2136
060)-	7,0252	7,3329	11,2998	11,9403	12,4057	11,2585
061)-	7,0234	7,3298	11,2911	11,8818	12,4804	11,2417
062)-	7,0278	7,3421	11,3321	11,8627	12,4589	11,2371
063)-	7,0273	7,34	11,3239	11,8118	12,4873	11,2165
064)-	7,0318	7,3492	11,3826	11,775	12,5829	11,2144
065)-	7,036	7,3605	11,3155	11,8479	12,4154	11,2565
066)-	7,0385	7,3675	11,3144	11,8257	12,4295	11,2686
067)-	7,0477	7,3697	11,4049	11,7636	12,5915	11,2501
068)-	7,0447	7,3687	11,4265	11,7269	12,6502	11,2257
069)-	7,046	7,3667	11,4513	11,6694	12,7688	11,1957
070)-	7,0511	7,3779	11,3822	11,7366	12,5494	11,2486
071)-	7,06	7,3911	11,4725	11,7187	12,6001	11,2528
072)-	7,0639	7,3924	11,5814	11,6435	12,9469	11,2284
073)-	7,0612	7,3736	11,5864	11,5926	13,1332	11,1747
074)-	7,07	7,3747	11,7145	11,5585	13,6446	11,1686
075)-	7,0748	7,3868	11,7173	11,5779	13,3943	11,1822
076)-	6,951	7,094	11,4701	11,9636	12,3396	12,3169
077)-	6,9535	7,0931	11,5825	11,9122	12,1266	12,2773
078)-	6,9725	7,0904	11,7929	11,9484	11,9346	12,48
079)-	6,9842	7,0793	11,9947	11,9968	11,7693	12,6941
080)-	6,9942	7,0784	12,4734	12,2714	11,5572	14,9352
081)-	6,9988	7,0794	12,8484	12,6921	11,48	13,1802
082)-	7,0055	7,0824	13,1113	13,1495	11,4464	12,5601
083)-	7,0138	7,0862	13,2161	13,1831	11,5568	11,5925
084)-	7,008	7,0947	13,1164	12,7696	11,5799	11,4865
085)-	7,0191	7,1082	12,7564	12,3447	11,8225	11,298
086)-	7,0163	7,1096	12,2504	12,0735	12,2116	11,1396
087)-	7,0133	7,128	12,152	11,9964	12,3902	11,1182
088)-	7,0126	7,1456	11,9403	11,9688	12,9624	11,1037
089)-	7,0111	7,1479	11,8101	11,9365	13,3987	11,0748
090)-	7,0077	7,1578	11,6312	11,9616	18,0091	11,0813
091)-	7,0058	7,1807	11,5456	11,9807	13,7032	11,1025
092)-	7,0038	7,1847	11,4766	12,0328	13,0722	11,1065

Tablo 5.3. İkinci sınıfa ait şablonların moment vektörleri

093)-	7,3795	7,7048	11,7677	14,9559	12,397	14,7537
094)-	7,3676	7,7147	11,7435	14,4555	12,3934	14,5985
095)-	7,3671	7,703	11,7282	14,3747	12,3373	14,5701
096)-	7,3601	7,7078	11,6878	16,3055	12,3194	15,7043
097)-	7,3533	7,7051	11,6893	14,3141	12,3036	14,4551
098)-	7,3463	7,7117	11,68	14,5876	12,2992	14,6782
099)-	7,342	7,6993	11,6536	14,4018	12,2439	14,5042
100)-	7,3304	7,7107	11,6326	14,7864	12,2525	14,7122
101)-	7,3269	7,7071	11,6226	14,6076	12,2366	14,5694
102)-	7,3193	7,6976	11,601	14,511	12,1878	14,7319
103)-	7,3107	7,6983	11,5836	14,4569	12,1766	14,6644
104)-	7,3059	7,688	11,5663	14,4745	12,1423	14,5732
105)-	7,2973	7,6893	11,5599	14,5408	12,139	14,5512
106)-	7,2942	7,6641	11,5404	14,436	12,0715	14,4795
107)-	7,284	7,6592	11,5275	14,5318	12,0652	14,559
108)-	7,2798	7,652	11,522	14,599	12,0397	14,6574
109)-	7,2743	7,6479	11,5079	14,7544	12,0299	14,653
110)-	7,2691	7,6215	11,4921	14,3222	11,9564	14,395
111)-	7,2593	7,6236	11,4759	14,3905	11,9576	14,513
112)-	7,2496	7,6043	11,4551	14,2297	11,9108	14,4198
113)-	7,244	7,6028	11,4472	14,1235	11,9057	14,3528
114)-	7,2421	7,5985	11,4505	14,2159	11,9037	14,3468
115)-	7,2362	7,5735	11,4401	14,0935	11,8397	14,2285
116)-	7,2263	7,5536	11,4095	14,2152	11,7914	14,1963
117)-	7,2195	7,5536	11,4019	14,2675	11,8001	14,2804
118)-	7,2075	7,5284	11,3823	14,0102	11,7387	14,1415
119)-	7,1966	7,5132	11,3654	14,0107	11,7233	14,1513
120)-	7,1948	7,4883	11,3669	13,8275	11,6768	13,9398
121)-	7,1886	7,4749	11,3589	13,8374	11,6524	13,9733
122)-	7,461	7,575	12,248	13,913	12,9637	13,9469
123)-	7,4667	7,5756	12,2344	13,9217	12,8929	13,9904
124)-	7,4608	7,5843	12,183	14,0567	12,8295	14,0694
125)-	7,4565	7,5979	12,1527	14,1003	12,7944	14,1329
126)-	7,4508	7,6112	12,1084	14,4734	12,7547	14,3703
127)-	7,4502	7,6125	12,0738	14,3222	12,6842	14,2559
128)-	7,4404	7,6316	12,0503	14,1468	12,6874	14,1451
129)-	7,4325	7,6387	12,0068	14,3036	12,644	14,2915
130)-	7,4318	7,6459	11,9876	14,1417	12,6178	14,1341
131)-	7,4233	7,6612	11,9556	14,0235	12,5987	14,0804
132)-	7,4187	7,6696	11,9184	14,1673	12,5643	14,3236
133)-	7,4125	7,6801	11,8985	14,1436	12,5508	14,297
134)-	7,4119	7,6834	11,8894	14,3272	12,5306	14,3751
135)-	7,4078	7,684	11,8608	14,484	12,4833	14,4658
136)-	7,3952	7,6942	11,8244	14,4087	12,4633	14,4067
137)-	7,3892	7,7026	11,8112	14,6414	12,4552	14,6052
138)-	7,3834	7,7006	11,7784	14,5654	12,4018	14,5157

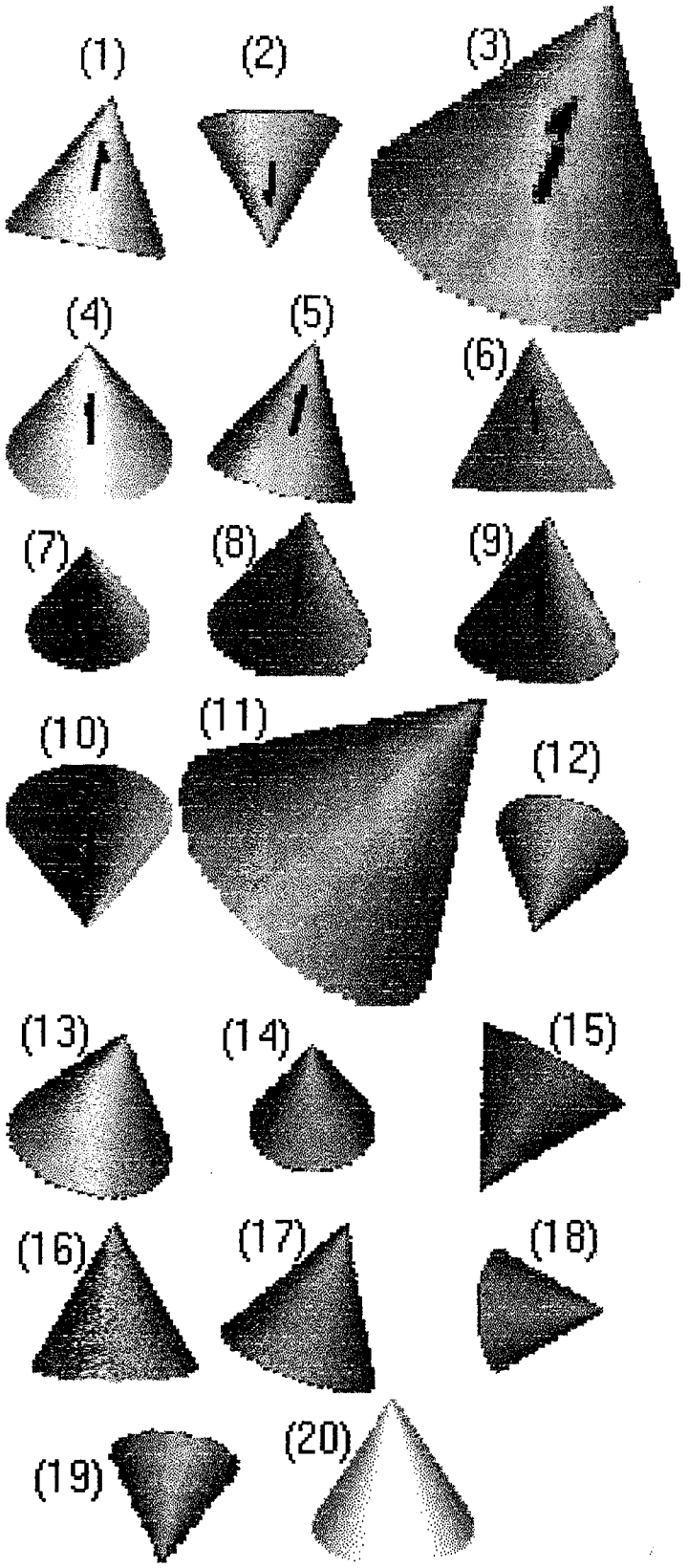
Yukarıdaki standart moment vektörleriyle eğitilen yapay sinir ağının öğrenme grafiği aşağıdadır. Bu grafik, sinir ağının çıkışındaki hatanın, yani nöronların verdiği çıkış değerleri ile olması gereken çıkış değerleri arasındaki farkın büyüklüğünün, ilk değerine göre yüzde olarak değişimidir. Eğitim yaklaşık 45 dakika sürmüştür. Eğitim esnasında program vasıtasıyla momentum ve öğrenme oranı parametrelerine performansı arttırmak amacıyla müdahale edilmiştir. Ama yaklaşık olarak 0.5 momentum ve 0.01 öğrenme oranları civarındaki değerler kullanılmıştır.



Şekil 5.2. Uygulama_1'de kullanılan sinir ağının eğitim grafiği

Sinir ağının eğitimi 100% başarıyla tamamlanmıştır, yani eğitimde kullanılan 138 objenin hepsi doğru bir şekilde sınıflandırılmıştır.

Eğitilmiş yapay sinir ağı, yukarıdaki test objeleri ile test edilerek, performansı ölçülmüştür. Bu test objeleri oluşturulurken, gerçek dünya ortamında gözlenebilecek çeşitli görüntü değişimleri göz önüne alınmıştır. Özellikle bu moment tanımlayıcılarının yüksek değişmezlik gösteremediği ışık değişimleri, objeler üzerinde çok kullanılmıştır.



Şekil 5.3. Uygulama_1'de eğitilmiş sinir ağını test etmek için kullanılan obje görüntüleri

Şekil 5.3'deki test objelerine uygulanmış deformasyonlar aşağıdaki listede açıklanmıştır.

1. Görüntü: 10% aydınlatılmış (brightness) objenin, döndürülmüş halinin ayna aksi.
2. Görüntü: 10% parlaklığı (kontrastı) artırılmış objenin, küçültülmüş halinin, düşey olarak yansımış görüntüsü.
3. Görüntü: Döndürülmüş ve büyütülmüş objenin görüntüsü.
4. Görüntü: 15% aydınlatılmış objenin görüntüsü.
5. Görüntü: 35% oranında parlaklığı arttırılmış objenin, döndürülmüş halinin görüntüsü.
6. Görüntü: 50% oranında parlaklığı azaltılmış objenin görüntüsü.
7. Görüntü: Küçültülmüş obje görüntüsü.
8. Görüntü: Döndürülmüş objenin, 10% oranında düzgün dağılmış gürültü eklenmiş görüntüsü.
9. Görüntü: Döndürülmüş objenin, 20% oranında parlaklığı arttırılmış halinin görüntüsü.
10. Görüntü: 5% aydınlatılmış objenin, düşey yansımış görüntüsü.
11. Görüntü: Döndürülmüş ve büyütülmüş obje görüntüsü.
12. Görüntü: Döndürülmüş ve küçültülmüş obje görüntüsü.
13. Görüntü: 10% aydınlatılmış ve döndürülmüş objenin görüntüsü.
14. Görüntü: 10% oranında kontrastı artırılmış ve küçültülmüş objenin görüntüsü.
15. Görüntü: 10% oranında karartılmış obje görüntüsü.
16. Görüntü: 20% oranında düzgün dağılımlı gürültü eklenmiş görüntü.
17. Görüntü: Döndürülmüş objenin görüntüsüne 10% düzgün dağılımlı gürültü eklenmiş.
18. Görüntü: 30% oranında parlaklığı artırılmış ve küçültülmüş objenin görüntüsü.
19. Görüntü: Döndürülmüş, küçültülmüş ve 7% gürültü eklenmiş görüntü.
20. Görüntü: 30% oranında aydınlatılmış obje görüntüsü.

Yukarda sıralanmış etkilerle değiştirilmiş obje görüntülerinin ilk on tanesi birinci obje sınıfındandır, diğerleri de ikinci sınıfa aittir.

Eğitilmiş sinir ağına, yukarıdaki 20 objeye ait standart moment vektörleri girildiğinde, sinir ağının çıkışındaki hücrelerin verdiği tepki sırasıyla aşağıdaki listededir.

Tablo 5.4. Şekil 5.3'deki test objelerinin yapay sinir ağındaki sonuçları

Obje no:	1. Çıkış nöronu	2. Çıkış nöronu	Sonuç
1'inci obje için output vektör:	0.104000	0.895578	YANLIŞ
2'inci obje için output vektör:	0.968936	0.031111	DOĞRU
3'inci obje için output vektör:	0.979756	0.020260	DOĞRU
4'inci obje için output vektör:	0.506098	0.501930	DOĞRU
5'inci obje için output vektör:	0.975154	0.024874	DOĞRU
6'inci obje için output vektör:	0.994209	0.006393	DOĞRU
7'inci obje için output vektör:	0.994137	0.006473	DOĞRU
8'inci obje için output vektör:	0.994387	0.006206	DOĞRU
9'inci obje için output vektör:	0.996851	0.003343	DOĞRU
10'inci obje için output vektör:	0.985228	0.016289	DOĞRU
11'inci obje için output vektör:	0.051372	0.948496	DOĞRU
12'inci obje için output vektör:	0.092918	0.906811	DOĞRU
13'inci obje için output vektör:	0.030418	0.969949	DOĞRU
14'inci obje için output vektör:	0.026253	0.974562	DOĞRU
15'inci obje için output vektör:	0.085789	0.913876	DOĞRU
16'inci obje için output vektör:	0.045159	0.954744	DOĞRU
17'inci obje için output vektör:	0.072191	0.927586	DOĞRU
18'inci obje için output vektör:	0.055269	0.944597	DOĞRU
19'inci obje için output vektör:	0.901756	0.098392	YANLIŞ
20'inci obje için output vektör:	0.041442	0.958634	DOĞRU

Birinci obje sınıfını birinci çıkış hücresi temsil ettiğinden, ilk 10 test vektörünün vermesi gereken çıkış vektörü $[1\ 0]^T$ olmalıdır. Bu 10 vektörden sadece birinci objeye ait olan ilk vektör yanlış tanınmıştır. İkinci obje sınıfı ise, ikinci çıkış hücresi ile temsil edildiğinden, 11-20 vektörlerinin vermesi gereken çıkış vektörü $[0\ 1]^T$ olmalıdır. Bu objeler arasında ise sadece ondokuzuncu obje yanlış tanınmıştır.

Bu sonuçlara göre tanıma işlemi 90% başarılı olmuştur. Aslında bu oran, eğitimdeki ortalama kare hatanın daha da azaltılmasıyla, daha da küçültülebilir fakat eğitim süresinin uzaması nedeniyle belirli bir orandaki hataya göz yumulmuştur. Ayrıca sinir ağının tanıma performansını artırmak için, eğitimde kullanılan örnekler zenginleştirilebilir, yani değişik ışık ortamlarındaki objeler ve değişik deformasyonlara uğramış objeler de kullanılarak sinir ağının genelleştirme yeteneği arttırılabilir. Tabi ki eğitici örnek sayısı arttıkça, eğitim süresi uzar, ayrıca sistemin hafıza gereksinimi büyür. Yukarıdaki sonuçlar incelendiğine birinci ve ondokuzuncu objelerin tanınamamasının nedeni açıkca görülememektedir. Fakat birinci objenin diğerlerinden tek farkı ayna aksi görüntüsü olması nedeniyle, birinci objenin yanlış tanınma nedeni bu olabilir. Ondokuzuncu obje için ise, küçültme ve gürültü ekleme işlemlerinin birlikte uygulanması, ana neden olabilir.

Moment tanımlayıcılarının temel sorunlarından biri olan obje ışıklandırma şekli ve şiddeti problemine karşı alınabilecek en iyi önlemlerden biri objelerin siluet veya kenar ayrıştırılmış görüntülerini kullanmaktır. Tabi ki bu yaklaşımların da beraberinde getirdiği

dezavantajlar vardır. Kenar ayrıştırılmış görüntüler için, bu hatalar, kenar işleme algoritmalarından oluşabilecek hatalar ve görüntüdeki düşük frekanslı bölgelerin tespit edilememesinden meydana gelen bilgi kayıplarıdır. Siluet görüntüleri için ise, görüntüdeki bütün desen ve yüzeysel özelliklerin ihmal edilmesinden oluşan bilgi kaybı önemlidir. Çünkü herhangi bir objenin herhangi bir açıdan alınmış perspektif görüntüsünü birçok obje verebilir. Aslında bu sorun gri seviye ve kenar ayrıştırılmış görüntülerde de geçerlidir, çünkü bir perspektif görüntü aynı zamanda birçok objeye ait olabilir ama bu görüntüler yüzeysel bilgileri de içerdiğinden siluet görüntülerinde olduğu gibi büyük bir sorun oluşturmaz. Bu çalışmada tanıtılan sistem eğer birkaç objeyi ayırt etmek için kullanıldığı taktirde siluet görüntülerin kullanımı sorun yaratmaz fakat aynı siluet görüntüleri verebilecek birden fazla objenin sınıflandırılması durumunda sorun yaşanacağı için çok sayıda objeyi sisteme tanıtırken dikkat edilmelidir. Aslında moment tanımlayıcılarının ancak sonsuz bir seti, tek bir iki boyutlu fonksiyonu temsil edebilirdi ve bu momentlerden bu fonksiyonun geri elde edebileceği gösterilmişti. Bu sistemde ise f(x,y) fonksiyonunun sadece 6 adet birinci, ikinci ve üçüncü derece momentleri kullanılmaktadır. Bu da demektir ki, bu 6 moment tek bir f(x,y)'yi temsil etmeyebilir. Yani farklı fonksiyonların bu 6 momenti aynı olabilir çünkü bir fonksiyonu sadece sonsuz sayıdaki moment seti temsil edebilir tıpkı Fourier serisi gibi. İşte bu nedenle moment tanımlayıcılarının, obje tanımlayıcısı olarak kullanılması, sistemin aynı anda birçok objeyi sınıflandırma özelliğini riske atmaktadır. Bu nedenle ancak birkaç objeyi sınıflandırmak amacıyla sistemi kullanmak daha güvenilir olabilmektedir. Eğer çok sayıda objeyi sınıflandırmak istersek, ikinci ve üçüncü dereceden momentlere ilaveten dördüncü, beşinci hatta gerekirse daha yüksek dereceden moment setlerini de kullanmalıyız böylelikle bir görüntüye ait moment vektörünün, temsil edebileceği uzayı sınırlandırmış oluruz. Bu problemin daha iyi anlaşılması için aşağıdaki örnek verilmiştir.

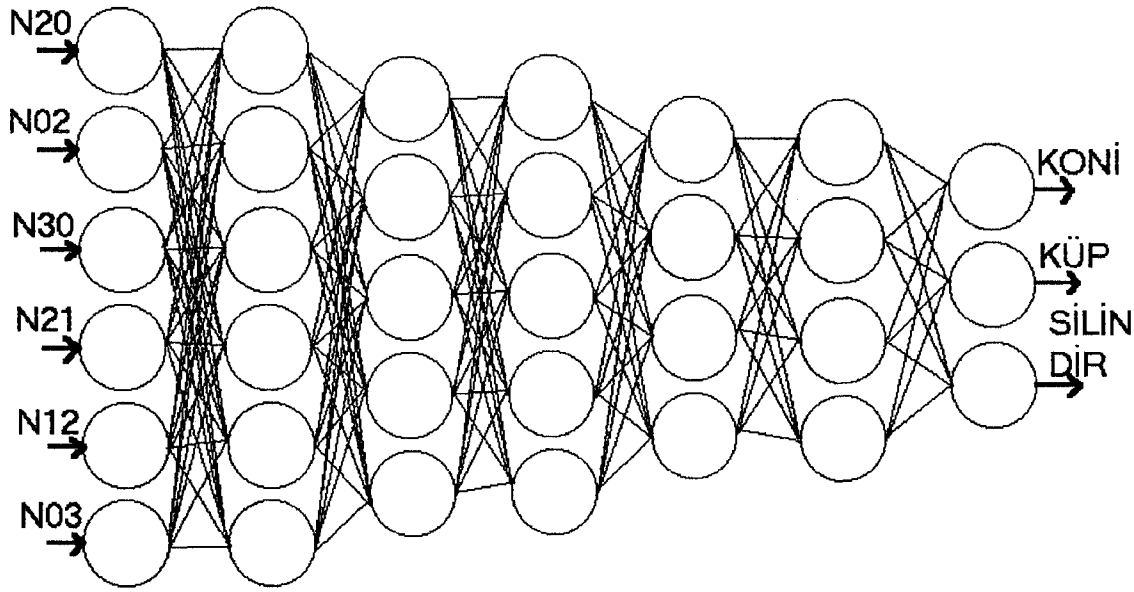
5.2 Uygulama_2: Sınıflandırılan Obje Sayısının Performansa Olan Etkisi

Aşağıda yapılan obje tanıma uygulamasında, küp, silindir ve koni objeleri yapay sinir ağına tanıtılmıştır. Sonuçlar incelendiğinde, yukarıda belirtilen sorun açıkça görülmektedir çünkü sinir ağı yaklaşık %80 oranında sınıflandırma yapabilmiştir. Sınıflandırılan obje sayısı arttıkça moment vektörlerinin boyutu arttırılmalıdır. Bu noktada önemli bir sorun olabilir ki, bu da yüksek dereceli momentlerde, örnekleme hatalarının büyük değerlere ulaştığıdır. Aşağıdaki uygulamada, eğitilen sinir ağı test edilirken kameradan alınan görüntüler de kullanılmıştır.

Eğitim için alınan örnek görüntüler, objelerin simetrik yapısı nedeniyle kısıtlı tutulmuştur. Moment tanımlayıcıları, yansıma (reflection) ve dönme (rotation) etkilerine karşı sabit olduğundan, örneğin bir küpün herhangi bir açıdan alınan görüntüsünün moment vektörü, yedi farklı açıdan alınan görüntülerinin moment vektörlerine eşit olur. Çünkü küp objesi bütün koordinat eksenleri boyunca simetriktir.

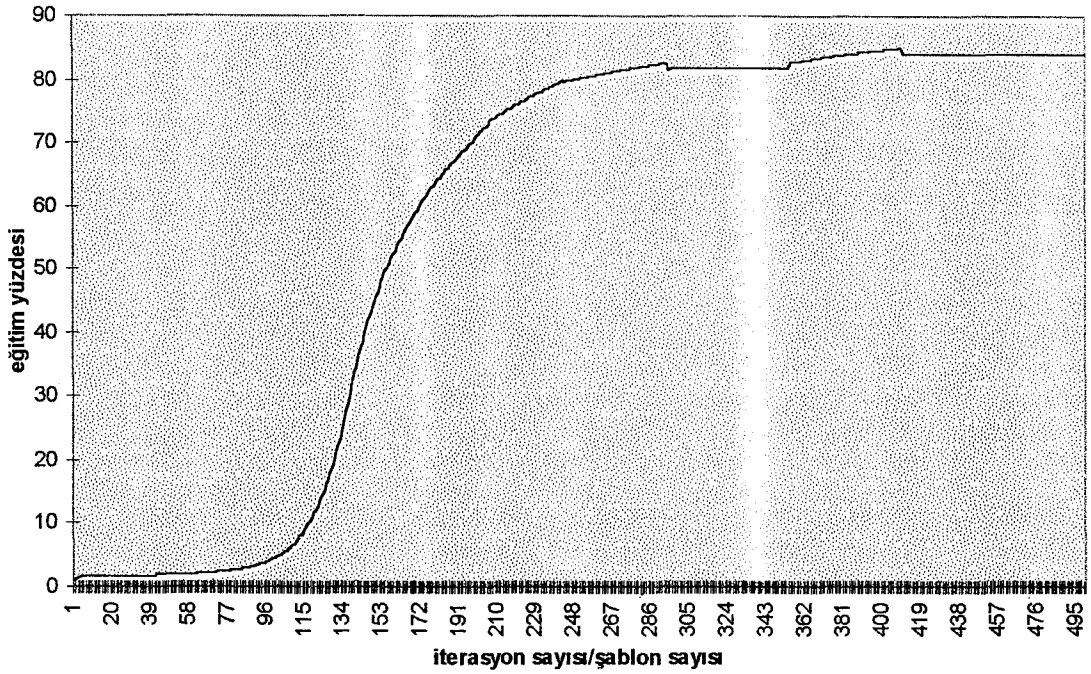
Ek-1'deki bu uygulamaya ait objelerle eğitilen sinir ağı, eğitimde kullanılmayan görüntülerle test edilmiştir. Test görüntüleri arasında kameradan alınan küp görüntüleri de vardır. Bu görüntüler özellikle, yapay sinir ağının performansını test etmek için bulanık ve gürültülü bir şekilde alınmıştır.

Uygulamada kullanılan sinir ağı yapısı aşağıdaki şekilde gösterilmiştir. Sinir ağının çıkışındaki her bir hücre bir obje sınıfını temsil etmektedir. Yani bir koni objesi için sinir ağının çıkışında $[1\ 0\ 0]^T$ vektörü, bir küp objesi için $[0\ 1\ 0]^T$ vektörü, bir silindir için ise $[0\ 0\ 1]^T$ vektörü olmalıdır.



Şekil 5.4. Uygulama_2'de kullanılan yapay sinir ağı

Yapay sinir ağının eğitim grafiği aşağıda gösterilmektedir. Belirli bir eğitim seviyesinden sonra salınımlar oluşmaktadır. Bu salınımların temel nedeni şudur; belirli bir seviyeye kadar eğitilen sinir ağının çıkışındaki hatalar küçülür dolayısıyla sinir ağının hücrelerine uygulanacak düzeltme miktarı da küçülür ama bu küçülme miktarı yeterince ufak olmadığı zaman hücrelere uygulanan düzeltmeler istenen seviyeden fazla olur böylece istenen seviye tutturulamaz ve salınımlar oluşur. Bunu önlemenin yolu öğrenme oranı ve momentum sabitlerinin değiştirilmesidir. Bu nedenle, eğitim esnasında sistemin bu parametrelerine müdahale edilmiştir.



Şekil 5.5. Uygulama_2'de kullanılan yapay sinir ağının eğitim grafiği

Yapay sinir ağının eğitim istatistikleri aşağıdadır.

- Sınıflandırılan obje sayısı: 182
- Yanlış sınıflandırılan obje sayısı: 22
- Objelerin %89.2'i doğru sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %31.3'ü koni sınıfına aittir.
- Yanlış sınıflandırılan objelerin %59.1'i koni sınıfına aittir.
- Koni sınıfı, %81,4 başarıyla sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %30.2'i küp sınıfına aittir.
- Yanlış sınıflandırılan objelerin %40.9'u küp sınıfına aittir.
- Küp sınıfı, %85,9 başarıyla sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %38.5'i silindir sınıfına aittir.
- Yanlış sınıflandırılan objelerin %0.0'ı silindir sınıfına aittir.
- Silindir sınıfı, %100 başarıyla sınıflandırılmıştır.

Eğitilmiş sinir ağının yapısal verileri aşağıda sunulmaktadır. Bu veriler kullanılan programın çıktılarıdır.

*** EGITILMIS SINIR AGININ YAPISAL VERİLERİ ***

7 katmanlı yapay sinir ağı %78.487770 eğitilmiştir;

Yapay sinir ağına ait bu ağırlık değerleri 1964520 iterasyonda hesaplanmıştır.

NOT:1'inci katman giriş katmanı olduğundan bu katmandaki nöronların ağırlık değerleri yoktur.

Katman No=2:

#1'inci nörona ait ağırlık vektörü:
[-0.116935 -0.111153 -0.098597 -0.824449 -0.139391 -0.449197]
Treshold=0.657163
#2'inci nörona ait ağırlık vektörü:
[9.577397 -3.800653 -0.599445 -7.950129 8.315296 -2.364955]
Treshold=0.918819
#3'inci nörona ait ağırlık vektörü:
[-1.599240 1.057814 0.822606 0.368487 0.158039 -2.262166]
Treshold=0.019390
#4'inci nörona ait ağırlık vektörü:
[-0.146872 1.644756 -1.108939 0.307928 0.273707 -0.347756] Treshold=-1.184655
#5'inci nörona ait ağırlık vektörü:
[0.027946 0.546356 0.092847 -0.422312 1.133879 0.848281] Treshold=-2.415680
#6'inci nörona ait ağırlık vektörü:
[-13.102492 2.132860 0.640269 1.975434 0.621765 1.841377]
Treshold=0.082664

Katman No=3:

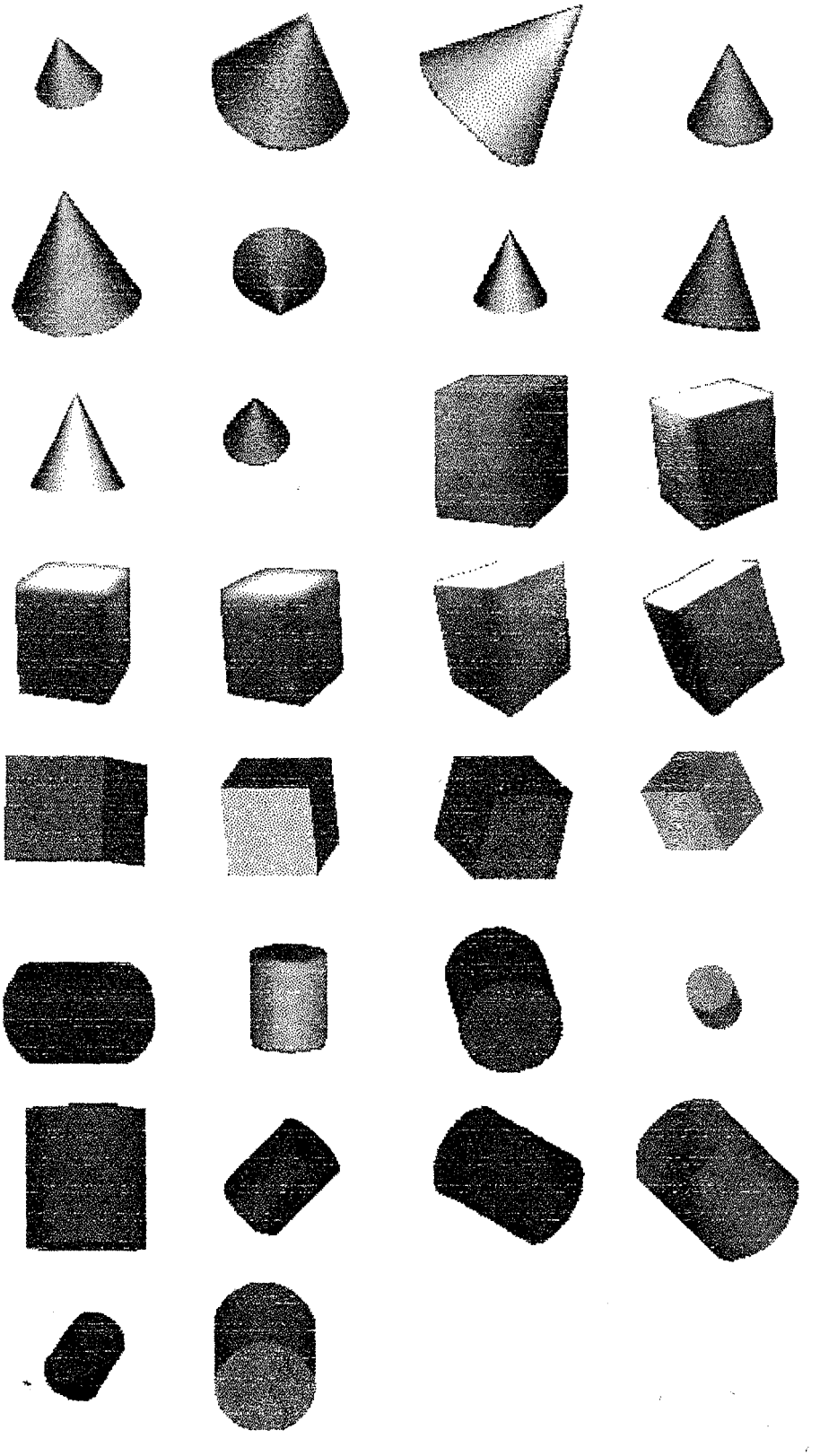
#1'inci nörona ait ağırlık vektörü:
[0.451618 -1.641177 0.285745 0.136520 -0.210913 -0.186980]
Treshold=-0.446910
#2'inci nörona ait ağırlık vektörü:
[-0.071516 1.823400 -0.876545 -1.808104 -0.473632 -2.736386]
Treshold=-1.328904
#3'inci nörona ait ağırlık vektörü:
[-0.900664 3.480591 -0.914702 -0.270685 -0.445630 -0.384134]
Treshold=0.202483
#4'inci nörona ait ağırlık vektörü:
[0.210911 4.071815 -1.189600 -0.924208 -0.116042 -6.100267]
Treshold=-1.700032
#5'inci nörona ait ağırlık vektörü:
[-0.953149 4.295892 -0.736959 -0.573880 -1.736463 -1.466353]
Treshold=-0.006656

Katman No=4:

#1'inci nörona ait ağırlık vektörü:
[-0.261497 -2.531835 0.580441 -5.986184 -0.223572] Treshold=-3.382757

```
#2'inci nörona ait ağırlık vektörü:
[-0.576223 -0.110575 0.033576 2.228022 0.405844 ] Treshold=-0.324152
#3'inci nörona ait ağırlık vektörü:
[-0.532560 0.384525 0.061061 1.536600 1.657131 ] Treshold=0.300959
#4'inci nörona ait ağırlık vektörü:
[0.777584 -0.023261 -2.178745 -1.154782 -1.018431 ] Treshold=-0.906683
#5'inci nörona ait ağırlık vektörü:
[2.384694 -1.874505 -3.622536 -2.402842 -3.590323 ] Treshold=-2.251533


Katman No=5:

#1'inci nörona ait ağırlık vektörü:
[-0.671971 0.692347 2.056813 -1.483237 -6.429526 ] Treshold=-1.789378
#2'inci nörona ait ağırlık vektörü:
[2.989286 -1.152990 -2.235123 1.895273 0.883349 ] Treshold=-0.250244
#3'inci nörona ait ağırlık vektörü:
[-6.746617 2.801825 0.262086 1.306159 -1.989761 ] Treshold=-1.386360
#4'inci nörona ait ağırlık vektörü:
[-0.301740 1.435868 0.043455 -1.321096 -2.762925 ] Treshold=-0.131109


Katman No=6:

#1'inci nörona ait ağırlık vektörü:
[-3.934109 1.438564 1.283442 -0.731520 ] Treshold=-0.066081
#2'inci nörona ait ağırlık vektörü:
[-0.669286 -1.589693 2.943088 0.465170 ] Treshold=0.241895
#3'inci nörona ait ağırlık vektörü:
[-4.999215 1.620900 -1.157618 -2.447623 ] Treshold=-1.820219
#4'inci nörona ait ağırlık vektörü:
[1.445823 -4.632150 7.043461 -0.409610 ] Treshold=1.985580


Katman No=7:

#1'inci nörona ait ağırlık vektörü:
[-2.467119 -2.803500 -3.491311 -6.717005 ] Treshold=-4.137976
#2'inci nörona ait ağırlık vektörü:
[2.434513 -1.607963 2.732997 -5.181793 ] Treshold=2.477095
#3'inci nörona ait ağırlık vektörü:
[0.358672 0.880228 -1.077006 6.636581 ] Treshold=4.230263
```

Eğitilmiş sinir ağının testinde kullanılan obje görüntüleri aşağıdadır. Bu görüntülerden ilk on tanesi koni, sonraki on tanesi küp ve son on tanesi de silindir görüntüleridir. Küp görüntülerinin ilk altı tanesi kameradan alınmış gerçek görüntüdür.



Şekil 5.6. Eğitilmiş sinir ağını test etmek için kullanılan obje görüntüleri

Eğitilmiş sinir ağına yukarıdaki bölütlenmiş test objelerine ait görüntülerin standart moment vektörleri sırasıyla girildiğinde, sinir ağının çıkışında aşağıdaki çıkış vektörleri görülmüştür.

Tablo 5.5. Şekil 5.6'daki test objelerinin yapay sinir ağındaki çıkış vektörleri

1'inci obje için çıkış vektörü :	0.921846	0.080576	0.032879	DOĞRU
2'inci obje için çıkış vektörü :	0.922130	0.081119	0.032701	DOĞRU
3'üncü obje için çıkış vektörü :	0.913739	0.070368	0.037042	DOĞRU
4'üncü obje için çıkış vektörü :	0.918031	0.074903	0.034982	DOĞRU
5'inci obje için çıkış vektörü :	0.917424	0.074184	0.035285	DOĞRU
6'ıncı obje için çıkış vektörü :	0.191072	0.822934	0.013208	YANLIŞ
7'inci obje için çıkış vektörü :	0.908103	0.065675	0.039583	DOĞRU
8'inci obje için çıkış vektörü :	0.917116	0.073830	0.035437	DOĞRU
9'uncu obje için çıkış vektörü :	0.914763	0.071292	0.036583	DOĞRU
10'uncu obje için çıkış vektörü :	0.923238	0.083500	0.031964	DOĞRU
11'inci obje için çıkış vektörü :	0.191103	0.822865	0.013210	DOĞRU
12'inci obje için çıkış vektörü :	0.191369	0.822264	0.013221	DOĞRU
13'üncü obje için çıkış vektörü :	0.191360	0.822336	0.013220	DOĞRU
14'üncü obje için çıkış vektörü :	0.191256	0.822532	0.013216	DOĞRU
15'inci obje için çıkış vektörü :	0.921308	0.082042	0.032597	YANLIŞ
16'ıncıobje için çıkış vektörü :	0.286845	0.728328	0.014119	DOĞRU
17'inci obje için çıkış vektörü :	0.191673	0.816823	0.013382	DOĞRU
18'inci obje için çıkış vektörü :	0.195372	0.817026	0.013288	DOĞRU
19'uncu obje için çıkış vektörü :	0.191539	0.822140	0.013221	DOĞRU
20'inci obje için çıkış vektörü :	0.191136	0.822797	0.013211	DOĞRU
21'inci obje için çıkış vektörü :	0.009410	0.000343	0.944770	DOĞRU
22'inci obje için çıkış vektörü :	0.868712	0.047910	0.055416	YANLIŞ
23'üncü obje için çıkış vektörü :	0.009391	0.000341	0.944857	DOĞRU
24'üncü obje için çıkış vektörü :	0.009408	0.000343	0.944776	DOĞRU
25'inci obje için çıkış vektörü :	0.009427	0.000344	0.944694	DOĞRU
26'ıncı obje için çıkış vektörü :	0.009407	0.000342	0.944783	DOĞRU
27'inci obje için çıkış vektörü :	0.009413	0.000343	0.944756	DOĞRU
28'inci obje için çıkış vektörü :	0.009422	0.000344	0.944712	DOĞRU
29'uncu obje için çıkış vektörü :	0.009430	0.000344	0.944673	DOĞRU
30'uncu obje için çıkış vektörü :	0.003708	0.022315	0.669993	DOĞRU

Yukarıdaki test sonuçlarına göre aşağıdaki istatiksel bilgiler çıkarılır.

- Doğru tanınan obje sayısı 27
- Yanlış tanınan obje sayısı 3
- Tanıma işlemi %90 başarılı olmuştur

- Doğru tanınan objelerin %33.3'ü koni sınıfına aittir.
- Yanlış tanınan objelerin %33.3'ü koni sınıfına aittir.
- Koni sınıfı, %90 başarıyla tanınmıştır.

- Doğru tanınan objelerin %33.3'ü küp sınıfına aittir.
- Yanlış tanınan objelerin %33.3'ü küp sınıfına aittir.
- Küp sınıfı, %90 başarıyla tanınmıştır.

- Doğru tanınan objelerin %33.3'ü silindir sınıfına aittir.
- Yanlış tanınan objelerin %33.3'ü silindir sınıfına aittir.
- Silindir sınıfı, %90 başarıyla tanınmıştır.

Tüm bu sonuçlardan görüleceği gibi sınıflandırılan obje sayısı arttıkça, moment vektörlerinin temsil ettikleri uzaylar çakışabilmektedir. Bu sorunu aşmak için daha yüksek dereceli momentleri de kullanmak gerekir. Fakat derece yükseldikçe kesikleştirmeden oluşan hatalar da belirginleşir, çünkü momentler hesaplanırken yapılan üssel işlemlerin derecesi de yükselir, bu da hatayı belirginleştiren bir faktör olur.

Ayrıntısı fazla olan objeler için bu sistem kullanıldığında yine yüksek dereceli momentlere ihtiyaç duyarız. Çünkü, bir f(x,y) iki boyutlu fonksiyonunu, ancak sonsuz sayıdaki momentlerini kullanarak geri elde edebileceğimiz için, f(x,y) fonksiyonuna ait ayrıntılı bilgiler için diğer yüksek dereceli momentlere ihtiyaç duyarız. Özellikle sınıflandırılacak ayrıntılı obje sayısı fazla olursa yüksek dereceli momentler mutlaka kullanılmalıdır.

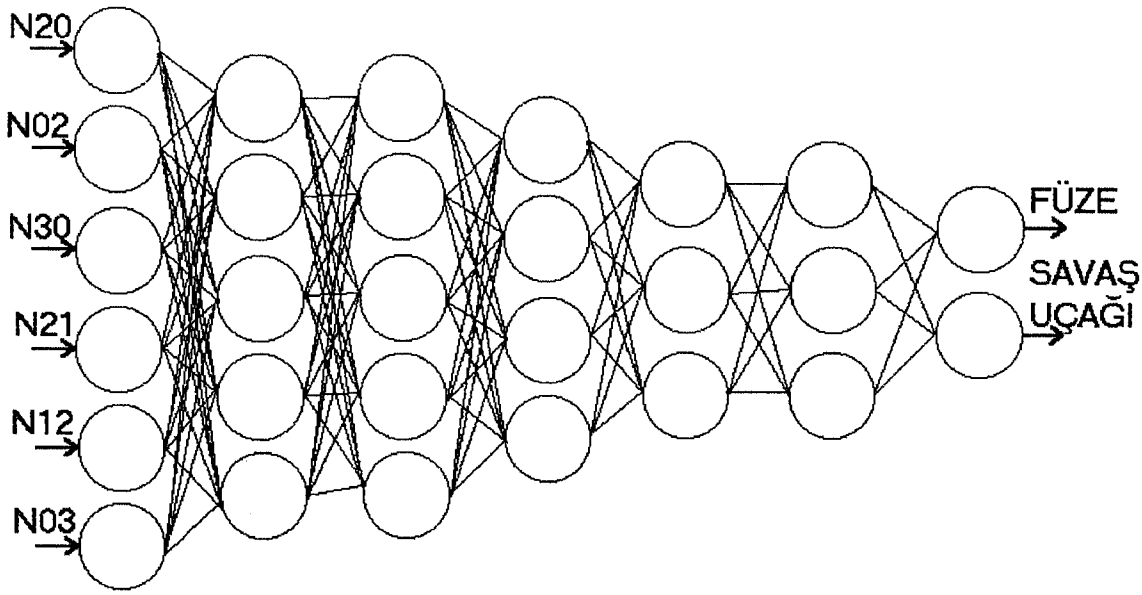
5.3 Uygulama_3: Kompleks Objelerin Sınıflandırılması

Aşağıdaki örnekte yukarıdaki uygulamalara göre daha ayrıntılı iki obje sınıflandırılmıştır. Birinci sınıf obje olarak bir füze diğer sınıfta da bir savaş uçağı kullanılmıştır. Ek-1'de görülen ve sinir ağının eğitiminde kullanılan füze objesinden 360 adet perspektif görüntü, savaş uçağından da 700 adet perspektif görüntü örneklenmiştir. Tüm görüntüler 5'er derecelik aralıklarla örneklenmiştir ve yine örnekleme işleminde, objelerin simetrik yapısı göz önüne alınmıştır. Bu nedenle savaş uçağı objesi, füzeye göre daha fazla örneklenmiştir.

Ayrıntılı objelerin eğitim için kullanılacak moment tanımlayıcı vektörleri bulunurken, perspektif görüntülerin boyutu önemli bir rol oynar. Çünkü objenin ayrıntılarının, görüntülerde doğru bir şekilde belirginleşmesi gerekir. Örneğin bir uçak objesine ait perspektif görüntülerde, görüntü boyutu ne kadar fazla olursa, ayrıntılar başına düşen piksel sayısı dolayısıyla bilgi miktarı fazla olur. Böylelikle, bu ayrıntıların, görüntünün moment vektörlerindeki etkileri belirginleşir. Bu çalışmada kullanılan sinir ağı gibi diğer bir çok sınıflandırıcı sistem için, böyle kesin değerler içeren vektörler her zaman performansı artırıcı bir faktör olur. Yukarıdaki eğitici obje görüntüleri, her biri 95x95 piksel boyutundaki resimlerden oluşmaktadır. Gerçekte bu boyutlardaki eğitici görüntülerin bir uçak veya füze gibi karmaşık objeler için kullanılması iyi sonuç vermez. Fakat bu uçak ve füze modelleri gerçeklerine göre gayet sade modeller olduğundan bu

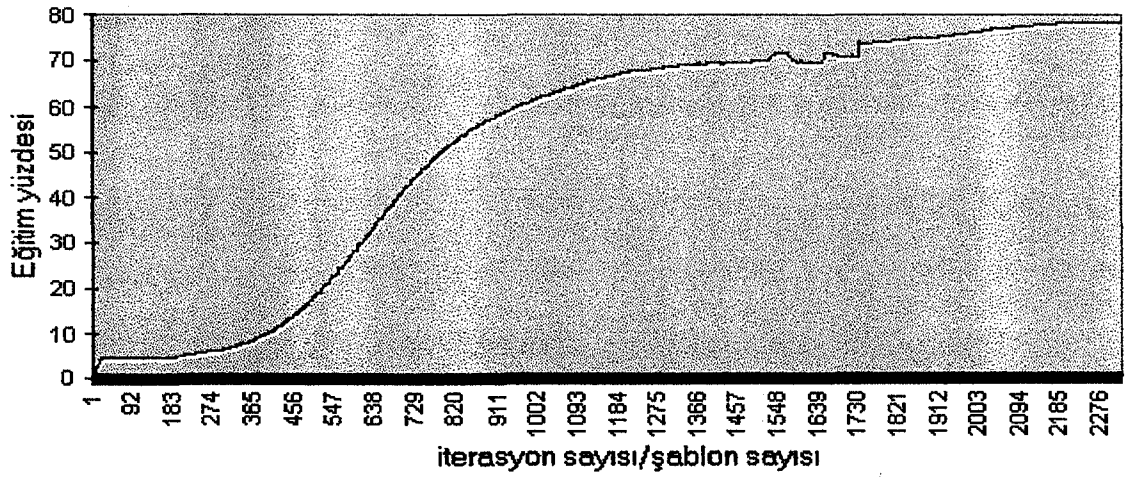
boyutlardaki görüntüler kullanılmıştır. Eğer resim boyutları yüksek tutulsaydı ve örnekleme açıları beşten küçük alınsaydı, tanıma performansı artacaktı.

Bu uygulamada kullanılan yapay sinir ağı modeli aşağıdaki gibidir. Diğer uygulamalarda olduğu gibi yine giriş vektörü olarak, eğitici görüntülerin standart moment vektörleri kullanılmıştır. Sinir ağının çıkışında ise, birinci obje sınıfı olan füzeler için $[1\ 0]^T$ çıkış vektörü, ikinci obje sınıfı olan savaş uçağı için ise $[0\ 1]^T$ vektörü atanmıştır.



Şekil 5.7. Uygulama_3'de kullanılan yapay sinir ağı

Sinir ağının eğitim grafiği aşağıda görülmektedir. Yaklaşık %78 civarında bir eğitim başarısı yakalanmıştır. Bu yüzde, yapay sinir ağının çıkışındaki ortalama hatayla orantılı bir değerdir. Bir önceki örnekte olduğu gibi, yine belli bir eğitim seviyesinden sonra salınımlar oluşmakadır, bu salınımlar, momentum ve öğrenme oranı parametrelerine müdahale edilerek önlenmiştir.



Şekil 5.8. Uygulama_3'deki yapay sinir ağının eğitim grafiği

Yapay sinir ağının eğitim istatistikleri aşağıdadır.

- Sınıflandırılan obje sayısı: 988
- Yanlış sınıflandırılan obje sayısı: 72
- Objelerin %93.2'i doğru sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %33.2'ü füze sınıfına aittir.
- Yanlış sınıflandırılan objelerin %44.4'i füze sınıfına aittir.
- Füze sınıfı, %91,1 başarıyla sınıflandırılmıştır.

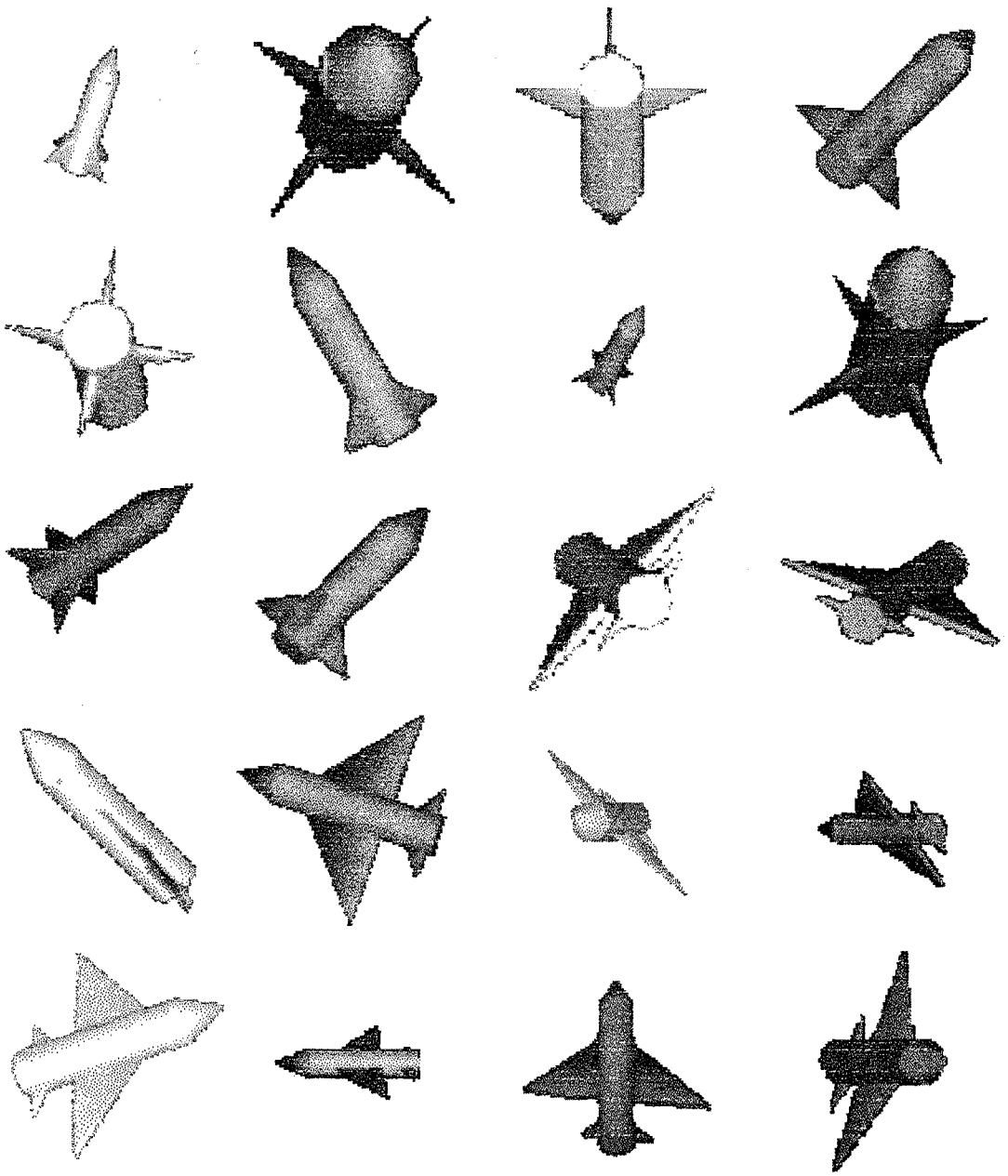
- Doğru sınıflandırılan objelerin %66.8'i uçak sınıfına aittir.
- Yanlış sınıflandırılan objelerin %55.6'u uçak sınıfına aittir.
- Uçak sınıfı, %94.3 başarıyla sınıflandırılmıştır.

Eğitilmiş sinir ağının yapısal verileri aşağıda sunulmaktadır.

```
*** EGITILMIS SINIR AGININ YAPISAL VERİLERİ ***

7 katmanlı yapay sinir ağı %78.838504 eğitilmiştir.
Yapay sinir ağına ait bu ağırlık değerleri 2312920 iterasyonda
hesaplanmıştır.

NOT:1'inci katman giriş katmanı olduğundan bu katmandaki nöronların
ağırlık değerleri yoktur.


Katman No=2:

#1'inci nörona ait ağırlık vektörü:
[12.150048 0.199956 -3.746634 12.658955 -16.355375 -4.114742 ]
Treshold=2.055157
#2'inci nörona ait ağırlık vektörü:
[0.340125 -0.088391 -0.865280 -0.098564 -0.255646 -0.034934 ]
Treshold=0.857111
#3'üncü nörona ait ağırlık vektörü:
[-16.111938 -37.575020 -1.205763 -0.512860 33.641296 1.073906 ]
Treshold=-6.285954
#4'üncü nörona ait ağırlık vektörü:
[-8.288000 15.962036 3.099607 -4.263890 -3.086939 -0.084161 ]
Treshold=0.864068
#5'inci nörona ait ağırlık vektörü:
[-2.189671 1.424799 1.942791 -3.687176 3.881570 0.083155 ]
Treshold=0.286146


Katman No=3:

#1'inci nörona ait ağırlık vektörü:
[1.964319 -1.153712 -4.224283 -2.588742 -1.518116 ] Treshold=1.369887
#2'inci nörona ait ağırlık vektörü:
[1.597044 -1.152612 -3.857963 -2.447867 -1.578545 ] Treshold=1.042587
#3'üncü nörona ait ağırlık vektörü:
```

```
[2.222512 -0.993034 -5.408119 -3.411349 -1.612864 ] Treshold=1.383582
#4'üncü nörona ait ağırlık vektörü:
[0.218614 -0.473206 -0.748347 -1.017093 -0.061397 ] Treshold=0.517991
#5'inci nörona ait ağırlık vektörü:
[-1.961862 -1.572217 6.122194 -1.196605 -1.114057 ] Treshold=0.860784


Katman No=4:

#1'inci nörona ait ağırlık vektörü:
[-1.739406 -1.499820 -2.382647 -0.796791 4.536244 ] Treshold=2.033478
#2'inci nörona ait ağırlık vektörü:
[-0.551398 -0.856373 -1.113036 -0.038481 1.044968 ] Treshold=0.943139
#3'üncü nörona ait ağırlık vektörü:
[-1.448218 -1.244284 -2.262962 0.156631 1.602163 ] Treshold=-1.209055
#4'üncü nörona ait ağırlık vektörü:
[2.181829 1.782563 3.118786 0.194034 -1.825530 ] Treshold=2.519014


Katman No=5:

#1'inci nörona ait ağırlık vektörü:
[-3.232698 -1.082301 -2.362039 3.662648 ] Treshold=0.818362
#2'inci nörona ait ağırlık vektörü:
[-5.038906 -1.266050 -0.513429 2.099803 ] Treshold=-2.883966
#3'üncü nörona ait ağırlık vektörü:
[2.398613 -0.204509 2.568625 -2.347171 ] Treshold=0.101516


Katman No=6:

#1'inci nörona ait ağırlık vektörü:
[-1.723336 -2.666308 1.534296 ] Treshold=-0.814157
#2'inci nörona ait ağırlık vektörü:
[1.347691 4.904063 -2.356057 ] Treshold=0.692472
#3'üncü nörona ait ağırlık vektörü:
[4.315633 2.386422 -3.537412 ] Treshold=2.714826


Katman No=7:

#1'inci nörona ait ağırlık vektörü:
[2.806000 -3.526745 -4.588274 ] Treshold=-2.628647
#2'inci nörona ait ağırlık vektörü:
[-2.851895 3.501261 4.572609 ] Treshold=2.588185
```

Eğitilmiş yapay sinir ağı, aşağıda belirtilen ve eğitimde kullanılmamış 20 adet füze ve savaş uçağı perspektif görüntüsüyle test edilmiştir. Ayrıca bu görüntüler, önceki uygulamalarda olduğu gibi çeşitli deformasyonlara ve ışıklandırma değişimlerine maruz bırakılmıştır.



Şekil 5.9. Eğitilmiş sinir ağını test etmek için kullanılan obje görüntüleri

Eğitilmiş sinir ağına yukarıdaki objelerin standart moment vektörleri sırasıyla girildiğinde, sinir ağının çıkışında aşağıdaki çıkış vektörleri görülmüştür.

Tablo 5.6. Şekil 5.9'daki test objelerinin yapay sinir ağındaki çıkış vektörleri

1'inci obje için çıkış vektörü :	0.984515	0.015464	DOĞRU
2'inci obje için çıkış vektörü :	0.630326	0.369591	DOĞRU
3'üncü obje için çıkış vektörü :	0.981492	0.018486	DOĞRU
4'üncü obje için çıkış vektörü :	0.984505	0.015474	DOĞRU
5'inci obje için çıkış vektörü :	0.041724	0.958263	YANLIŞ
6'ıncı obje için çıkış vektörü :	0.984515	0.015464	DOĞRU
7'inci obje için çıkış vektörü :	0.984515	0.015464	DOĞRU
8'inci obje için çıkış vektörü :	0.983742	0.016237	DOĞRU
9'uncu obje için çıkış vektörü :	0.984515	0.015464	DOĞRU
10'uncu obje için çıkış vektörü :	0.984515	0.015464	DOĞRU
11'inci obje için çıkış vektörü :	0.630991	0.368924	YANLIŞ
12'inci obje için çıkış vektörü :	0.042597	0.957392	DOĞRU
13'üncü obje için çıkış vektörü :	0.038385	0.961599	DOĞRU
14'üncü obje için çıkış vektörü :	0.559027	0.441056	YANLIŞ
15'inci obje için çıkış vektörü :	0.038384	0.961601	DOĞRU
16'ıncı obje için çıkış vektörü :	0.038385	0.961599	DOĞRU
17'inci obje için çıkış vektörü :	0.038391	0.961593	DOĞRU
18'inci obje için çıkış vektörü :	0.038542	0.961442	DOĞRU
19'uncu obje için çıkış vektörü :	0.045040	0.954951	DOĞRU
20'inci obje için çıkış vektörü :	0.038386	0.961598	DOĞRU

Yukarıdaki test sonuçlarına göre aşağıdaki istatiksel bilgiler çıkarılır.

- Doğru tanınan obje sayısı 17
- Yanlış tanınan obje sayısı 3
- Tanıma işlemi %85 başarılı olmuştur

- Doğru tanınan objelerin %52.9'ü füze sınıfına aittir.
- Yanlış tanınan objelerin %33.3'ü füze sınıfına aittir.
- Füze sınıfı, %90 başarıyla tanınmıştır.

- Doğru tanınan objelerin %47.1'ü uçak sınıfına aittir.
- Yanlış tanınan objelerin %66.7'ü uçak sınıfına aittir.
- Uçak sınıfı, %80 başarıyla tanınmıştır.

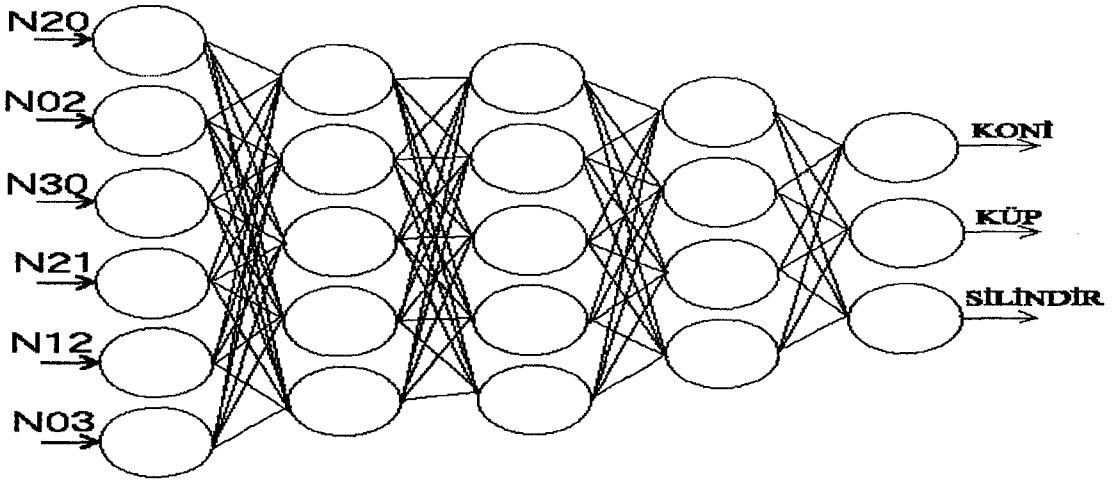
5.4 Uygulama_4: Farklı Işıklandırmalara Sahip Objelerin Eğitimde Kullanılmasıyla Performans Artışı Sağlamak

Standart momentlerin, en önemli dez avantajlarından biri olan obje ışıklandırma sorunu daha önceki bölümlerde açıklanmıştı. Bu uygulamada, bu soruna çözüm olarak gösterilebilecek bir yaklaşım sunulmaktadır. Bu yaklaşıma göre, sisteme tanıtılacak olan

objelerin farklı ışıklandırmalara sahip görüntüleri eğitimde kullanılırsa, sistemin tanıma performansı artacaktır, başka bir ifadeyle, olası farklı ışıklandırma şekillerine sahip objelerin yanlış tanınma ihtimali azalacaktır.

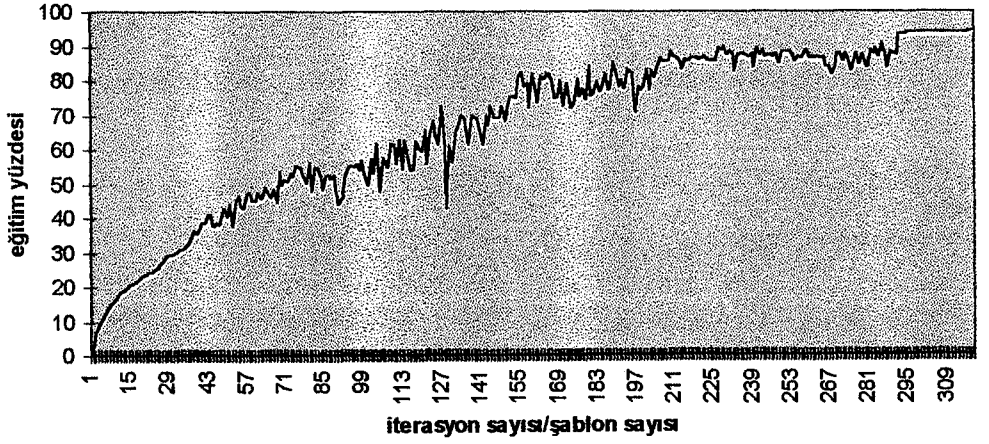
Bu yaklaşımın geçerliliğini açık bir şekilde göstermek için uygulama_2'de kullanılan koni, küp ve silindir objeleri farklı ışıklandırmalarla bu uygulamada da kullanılmıştır ve iki uygulamanın performansı karşılaştırılmıştır.

Tüm objeler, ayrı ayrı dört farklı açıdan ışıklandırılmıştır. Önden, sağdan, soldan ve yukarıdan ışıklandırılmış objelere ait bu görüntüler ek_1'den görülebilir. Bu objelerden, koni ve silindir objeleri y ekseni boyunca 45 dereceye kadar birer derecelik aralıklarla örneklenmiştir. Küp ise simetrik yapısından dolayı x ve y eksenleri boyunca 5'er derecelik aralıklarla 45 dereceye kadar örneklenmiştir.



Şekil 5.10. Uygulama_4'de kullanılan yapay sinir ağı

Eğitimin başarı grafiği şekil 5.11'deki gibidir. Bu grafikten de görüleceği gibi eğitimde salınımlar olmuştur fakat grafik genel olarak yakınsama göstermiştir. Eğitim 90% seviyelerine ulaştığında momentum ve öğrenme parametresine müdahale edilerek salınımlar önlenmiştir.



Şekil 5.11. Uygulama_4'de kullanılan sinir ağının eğitim grafiği

Eğitilmiş sinir ağının yapısal verileri aşağıdaki gibidir.

```
*** EĞİTİLMİŞ SİNİR AĞININ YAPISAL VERİLERİ ***

5 katmanlı yapay sinir ağı %98.035856 eğitilmiştir.

Yapay sinir ağına ait bu ağırlık değerleri 195625 iterasyonda
hesaplanmıştır.

NOT:1'inci katman giriş katmanı olduğundan bu katmandaki nöronların
ağırlık değerleri yoktur.


Katman No=2:

#1'inci nörona ait ağırlık vektörü:
[0.577484 0.595839 0.000087 1.542229 0.692880 0.402264 ]
Treshold=1.342433
#2'inci nörona ait ağırlık vektörü:
[-14.892017 14.511015 -0.037596 2.634346 -1.218091 -1.733855 ]
Treshold=1.443228
#3'inci nörona ait ağırlık vektörü:
[0.146765 4.444609 1.693335 0.401163 -0.738548 -2.049724 ]
Treshold=0.534975
#4'inci nörona ait ağırlık vektörü:
[-0.087525 -2.065536 0.306265 0.384535 -1.730265 0.867983 ]
Treshold=0.272582
#5'inci nörona ait ağırlık vektörü:
[-12.307194 -9.085290 4.243309 -3.959233 8.139186 4.282993 ]
Treshold=-2.481400


Katman No=3:

#1'inci nörona ait ağırlık vektörü:
[3.580075 -1.487130 -0.450744 4.278759 0.110063 ] Treshold=-4.109016
#2'inci nörona ait ağırlık vektörü:
[0.117381 1.490765 3.848721 0.030708 0.251820 ] Treshold=2.785451
#3'inci nörona ait ağırlık vektörü:
[-0.447133 4.824188 1.832010 -2.540710 -1.291124 ] Treshold=0.149238
#4'inci nörona ait ağırlık vektörü:
[2.882844 3.423315 0.782250 -1.835984 4.828193 ] Treshold=0.248688
#5'inci nörona ait ağırlık vektörü:
[0.279233 2.273941 0.632893 2.355269 -3.610978 ] Treshold=2.075786


Katman No=4:

#1'inci nörona ait ağırlık vektörü:
[1.402394 -1.394158 -4.527216 -4.323297 1.461300 ] Treshold=-1.462943
#2'inci nörona ait ağırlık vektörü:
[2.433661 0.538416 -0.149701 0.330661 4.646989 ] Treshold=-1.290125
#3'inci nörona ait ağırlık vektörü:
[-0.912476 1.255072 3.055787 1.782345 3.680773 ] Treshold=4.206014
#4'inci nörona ait ağırlık vektörü:
[-0.765392 0.946174 1.974316 -5.409609 5.888568 ] Treshold=-1.096858
```

```
Katman No=5:

#1'inci nörona ait ağırlık vektörü:
[-1.064465 1.442535 -1.486254 -7.798101 ] Treshold=-3.029628
#2'inci nörona ait ağırlık vektörü:
[-4.975553 -3.030240 4.490657 3.841403 ] Treshold=0.768369
#3'inci nörona ait ağırlık vektörü:
[5.883193 -3.418739 -3.784777 4.488231 ] Treshold=1.867659
```

Eğitime ait bazı istatistikler şöyledir.

- Sınıflandırılan obje sayısı: 619
- Yanlış sınıflandırılan obje sayısı: 6
- Objelerin %99 u doğru sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %29.1'i koni sınıfına aittir.
- Yanlış sınıflandırılan objelerin %83.3'ü koni sınıfına aittir.
- Koni sınıfı, %97.3 başarıyla sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %41.4'ü küp sınıfına aittir.
- Yanlış sınıflandırılan objelerin %0.0'ı küp sınıfına aittir.
- Küp sınıfı, %100.0 başarıyla sınıflandırılmıştır.

- Doğru sınıflandırılan objelerin %29.6'ı silindir sınıfına aittir.
- Yanlış sınıflandırılan objelerin %16.7'i silindir sınıfına aittir.
- Silindir sınıfı, %99.5 başarıyla sınıflandırılmıştır.

Eğitilen sinir ağı, uygulama_2'deki test objeleriyle test edilmiştir ve sinir ağının bu objelere verdiği tepki vektörleri tablo 5.7'deki gibidir.

Tablo 5.7. Şekil 5.6'daki test objelerinin yapay sinir ağındaki çıkış vektörleri

1. obje için çıkış vektörü:	0.970790	0.032969	0.013482	DOĞRU
2. obje için çıkış vektörü:	0.971552	0.026162	0.016832	DOĞRU
3. obje için çıkış vektörü:	0.971911	0.016274	0.027515	DOĞRU
4. obje için çıkış vektörü:	0.972045	0.016911	0.026314	DOĞRU
5. obje için çıkış vektörü:	0.972002	0.016705	0.026691	DOĞRU
6. obje için çıkış vektörü:	0.971983	0.016561	0.026954	DOĞRU
7. obje için çıkış vektörü:	0.971879	0.016169	0.027730	DOĞRU
8. obje için çıkış vektörü:	0.971945	0.016407	0.027252	DOĞRU
9. obje için çıkış vektörü:	0.971924	0.016325	0.027414	DOĞRU
10. obje için çıkış vektörü:	0.972322	0.018591	0.023609	DOĞRU
11. obje için çıkış vektörü:	0.018724	0.963349	0.025692	DOĞRU
12. obje için çıkış vektörü:	0.018781	0.954003	0.032865	DOĞRU
13. obje için çıkış vektörü:	0.018735	0.966831	0.022993	DOĞRU
14. obje için çıkış vektörü:	0.018731	0.966310	0.023399	DOĞRU
15. obje için çıkış vektörü:	0.968931	0.017929	0.027021	YANLIŞ
16. obje için çıkış vektörü:	0.018731	0.966272	0.023429	DOĞRU
17. obje için çıkış vektörü:	0.018752	0.957696	0.030034	DOĞRU
18. obje için çıkış vektörü:	0.018729	0.965958	0.023672	DOĞRU
19. obje için çıkış vektörü:	0.018737	0.960013	0.028257	DOĞRU
20. obje için çıkış vektörü:	0.018727	0.965595	0.023955	DOĞRU
21. obje için çıkış vektörü:	0.026094	0.021030	0.972791	DOĞRU
22. obje için çıkış vektörü:	0.025605	0.022233	0.971660	DOĞRU
23. obje için çıkış vektörü:	0.026124	0.020996	0.972806	DOĞRU
24. obje için çıkış vektörü:	0.025482	0.022588	0.971314	DOĞRU
25. obje için çıkış vektörü:	0.025801	0.021722	0.972146	DOĞRU
26. obje için çıkış vektörü:	0.026087	0.021047	0.972775	DOĞRU
27. obje için çıkış vektörü:	0.025928	0.021413	0.972436	DOĞRU
28. obje için çıkış vektörü:	0.025434	0.022718	0.971192	DOĞRU
29. obje için çıkış vektörü:	0.025764	0.021817	0.972057	DOĞRU
30. obje için çıkış vektörü:	0.026178	0.020981	0.972770	DOĞRU

Yukarıdaki sonuçlardan, on beşinci objenin yanlış tanındığı görülmektedir. Bu obje uygulama_2'de de yanlış tanınmıştı. Altıncı ve yirmi ikinci objeler uygulama_2'de yanlış tanınmasına rağmen, bu uygulamada doğru tanınmıştır, dolayısıyla bu objelere uygulanan ışık değişimleri tanımaya engel olmamıştır. O zaman on beşinci objenin yanlış tanınma nedeni ışıksal değil yapısal olabilir, çünkü bu obje bir küpten çok bir dikdörtgenler prizmasına benzemektedir. Gerçekte bir küpe ait olan bu görüntü, kameranın stereo olmasından dolayı bu şekilde örneklenmiştir.

Bu sonuçlara göre, uygulama_2'den daha iyi bir performans elde edilmiştir. Bu performans aşağıdaki istatistiklerden de görülebilir.

- Doğru tanınan obje sayısı 29
- Yanlış tanınan obje sayısı 1

- Tanıma işlemi %96.7 başarılı olmuştur

- Doğru tanınan objelerin %34.5'i koni sınıfına aittir.
- Yanlış tanınan objelerin %0.0'ı koni sınıfına aittir.
- Koni sınıfı, %100 başarıyla tanınmıştır.

- Doğru tanınan objelerin %31.0'ı küp sınıfına aittir.
- Yanlış tanınan objelerin %100.0'ı küp sınıfına aittir.
- Küp sınıfı, %90 başarıyla tanınmıştır.

- Doğru tanınan objelerin %34.5'i silindir sınıfına aittir.
- Yanlış tanınan objelerin %0.0'ı silindir sınıfına aittir.
- Silindir sınıfı, %100 başarıyla tanınmıştır.

6. SONUÇLARIN DEĞERLENDİRİLMESİ VE YORUMLAR

Bu çalışmada, üç boyutlu objelerin iki boyutlu perspektif görüntülerinden tanınması için yapılan yaklaşım, yukarıdaki uygulamalardan da görüleceği gibi belirli oranlarda başarılı sonuçlar vermektedir. Optimal performansın elde edilebilmesi için, sistemin yapısında bazı değişiklikler yapılabilir. Yapay sinir ağının yapısal şekli ve eğitimde kullanılan parametreleri önemli olduğu gibi, eğitici örneklerin seçimi de performansı etkileyen faktörler arasındadır.

Bu çalışmada kullanılan standart moment vektörleri, birinci, ikinci ve üçüncü derece momentlerden elde edilen düşük seviyeli moment tanımlayıcılarıdır. Daha yüksek dereceli momentleri içeren moment vektörleri, tanınacak obje sınıfının ve sayısının fazla olduğu problemlerde veya ayrıntılı objelerin sınıflandırılacağı problemlerde kullanılmalıdır.

Sistemin eğitiminde kullanılan eğitici obje görüntüleri, objenin ayrıntılarının net bir şekilde moment vektörlerinde temsil edilebilecek kadar büyük boyutlu olmalıdır. Aksi durumda, bu ayrıntılar, moment vektörleri içinde eksik temsil edileceğinden, sistemin tanıma performansı azalır.

Yapay sinir ağlarının, moment vektörleri gibi yüksek hassasiyet isteyen verileri sınıflandırma işleminde başarılı olabilmesi için, eğitim algoritmasında kullanılan bütün matematiksel işlemler yüksek hassasiyette yapılması gerekir. Aksi durumda, matematiksel işlemlerde yapılan ihmaller ve kayıplar milyarlarca işlem sonucunda büyük kayıplara neden olabilmektedir.

Moment vektörlerinin, büyüklük normalizasyonu işlemi nedeniyle çok küçük değerler alması, yine yukarıda belirtilen matematiksel işlem kayıplarına ve hatalarına neden olur bu nedenle moment vektörlerinin elemanları, yapay sinir ağının hatasız işlem yapabileceği, dinamik bir aralığa aktarılması gerekir ki bu da bir logaritmik fonksiyonla sağlanabilir.

Standart momentlerin büyüklük normalizasyonuna rağmen, büyük ışıklandırma şiddet ve yön değişimlerine karşı değişim göstermesi ve analog resmin kesikleştirilme hatalarından kaynaklanan kayıpların yüksek dereceli momentlerde belirginleşmesi nedeniyle oluşan sınıflandırma hataları, eğitimde değişik ışıklandırma şiddet ve yönüne sahip görüntüler kullanılarak ve görüntü boyutu artırılarak azaltılabilir. Eğitici görüntülerin boyutlarının arttırılması, görüntüdeki obje bilgisinin artmasına neden olacağı için, kesikleştirme hatalarından kaybolan bilgiler azaltılmış olur. Böylelikle yüksek dereceli momentlerde, bu hataların belirginleşmesi de azaltılmış olur. Objenin ışıklandırma problemi ise siluet veya kenar ayrıştırılmış objeler kullanılarak giderilebilir fakat uygulama-1 de belirtildiği gibi bu yaklaşımın da dezavantajları vardır. Bu soruna, sistemin eğitiminde farklı ışıklandırmalara sahip objeler kullanarak belirli bir miktarda çözüm getirmiş oluruz. Uygulama_4'de bu durum açıkça gösterilmiştir.

7. KAYNAKLAR

1. De Wilde, P., *Neural Network Models, Second Edition.* Springer, 1997

2. Schalkoff R., *Pattern Recognition, Statistical, Structural, Neural Approaches.* John Wiley&Sons, Inc., 1996

3. R. Castleman K., *Digital Image Processing.* Prentice Hall,1996

4. Jain R., Kasturi R., G. Schunck B., *Machine Vision.* McGraw-Hill, 1995

5. C. Gonzalez R., E. Woods R., *Digital Image Processing.* Addison_Wesley Publishing Company, 1992

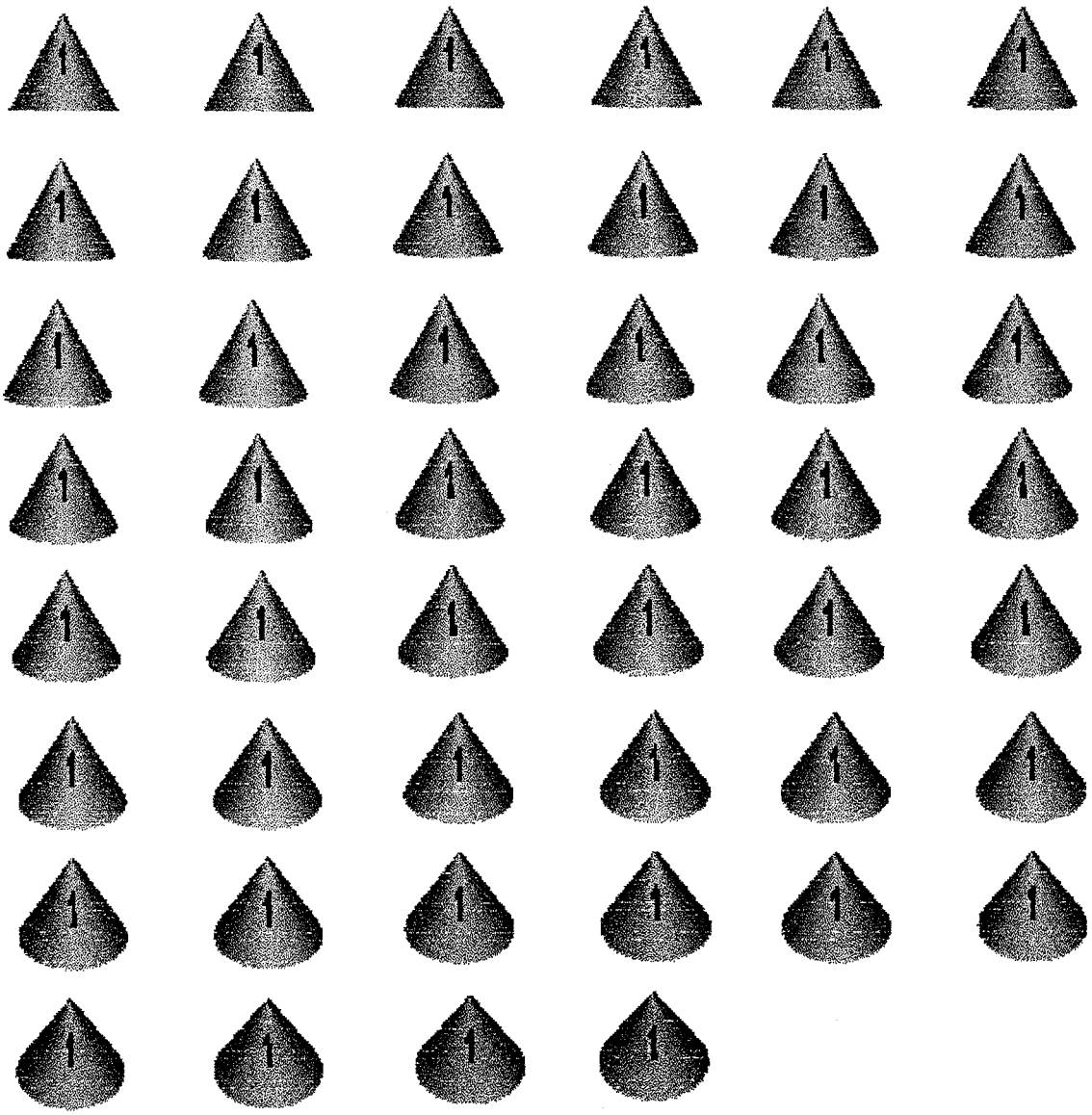
6. D. Levine M., *Vision In Man And Machine.* McGraw-Hill, 1985

7. M. Brady J., *Computer Vision, Vol.17.* Elsevier Science Publishers B.V., 1981

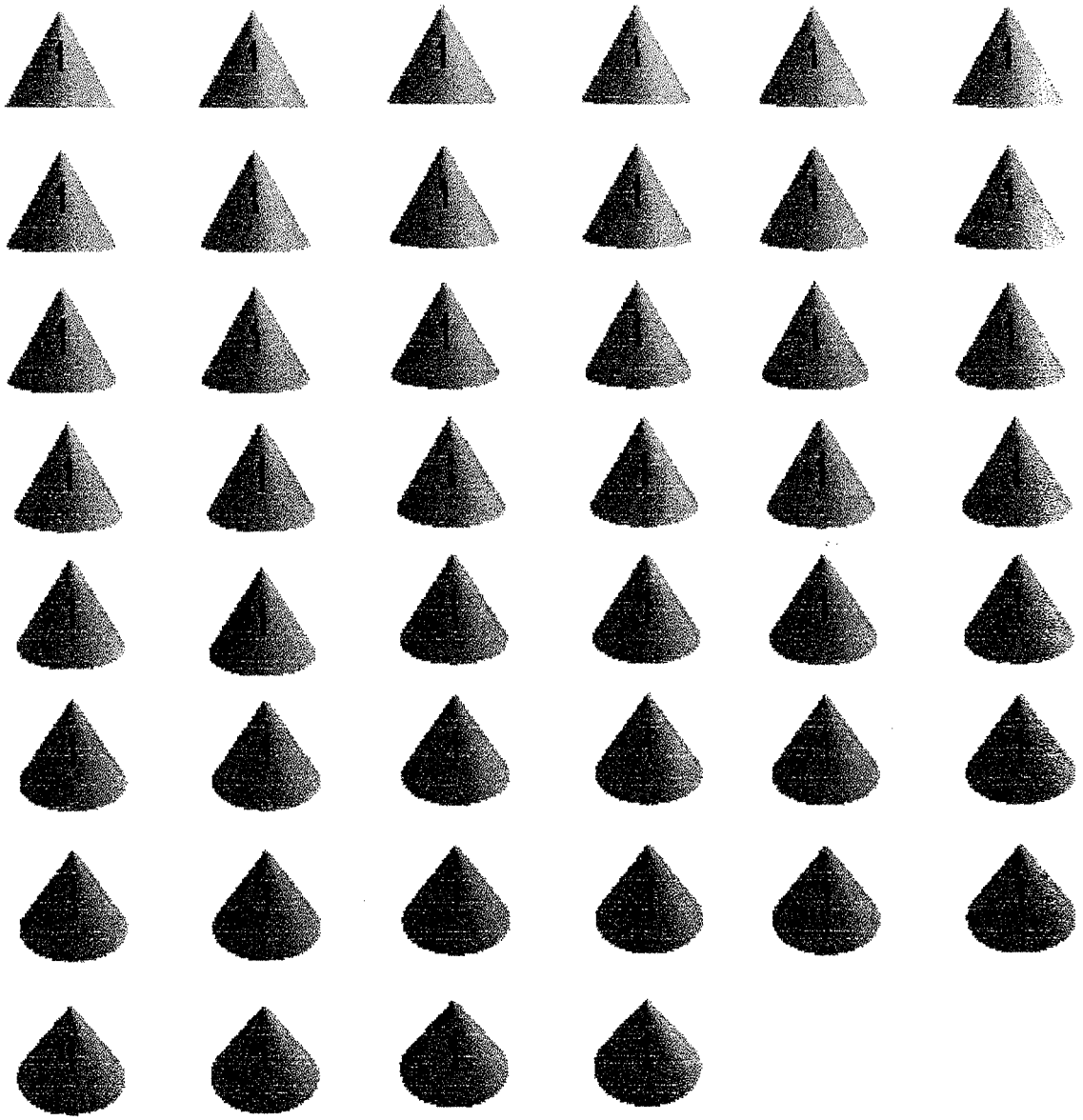
8. EKLER

8.1.Ek-1:Obje Tanıma Uygulamalarında Kullanılan Obje Görüntüleri

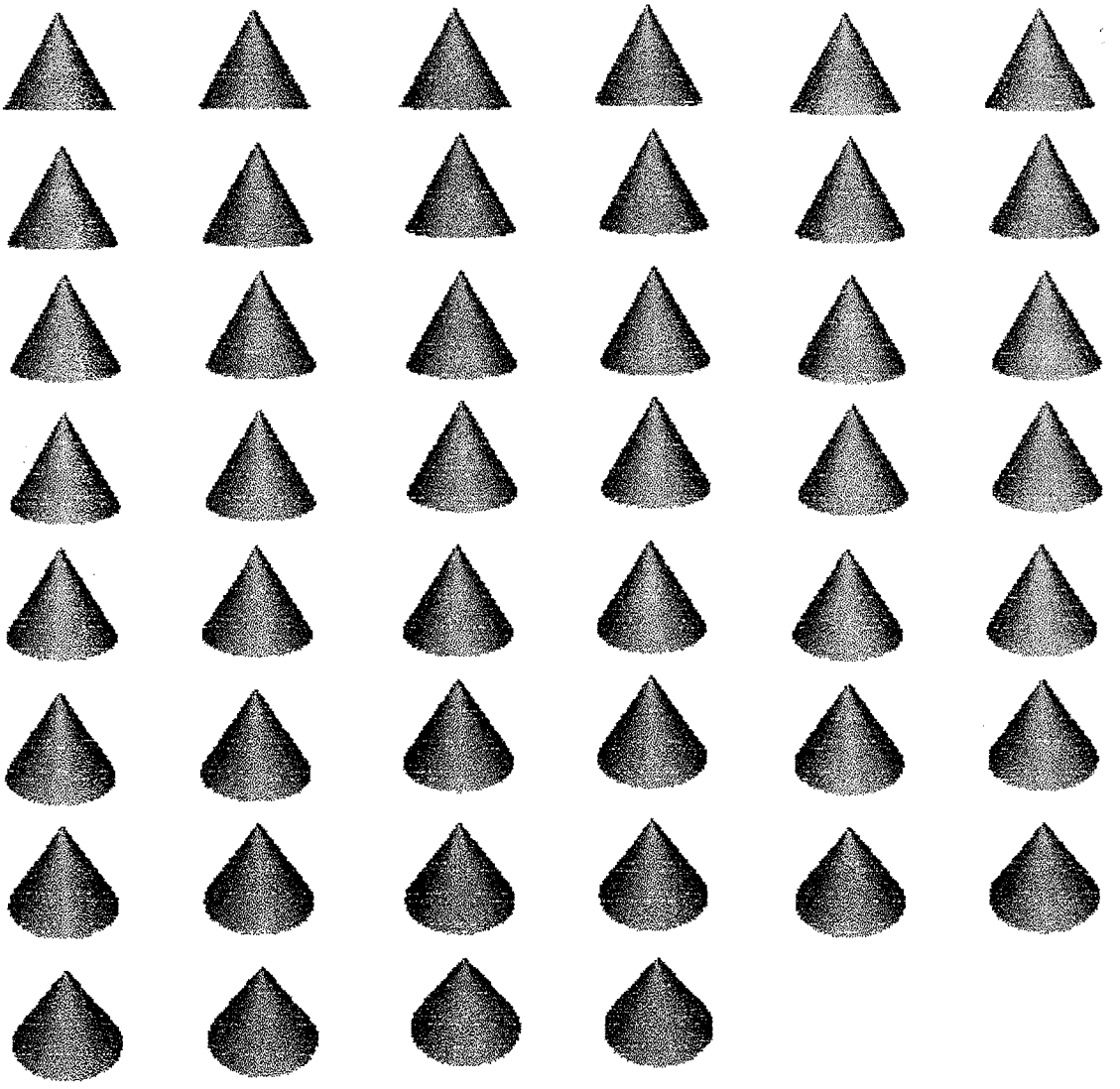
8.1.1. Uygulama_1'de kullanılan objeler



Şekil 8.1. Eğitimde kullanılan birinci obje sınıfı

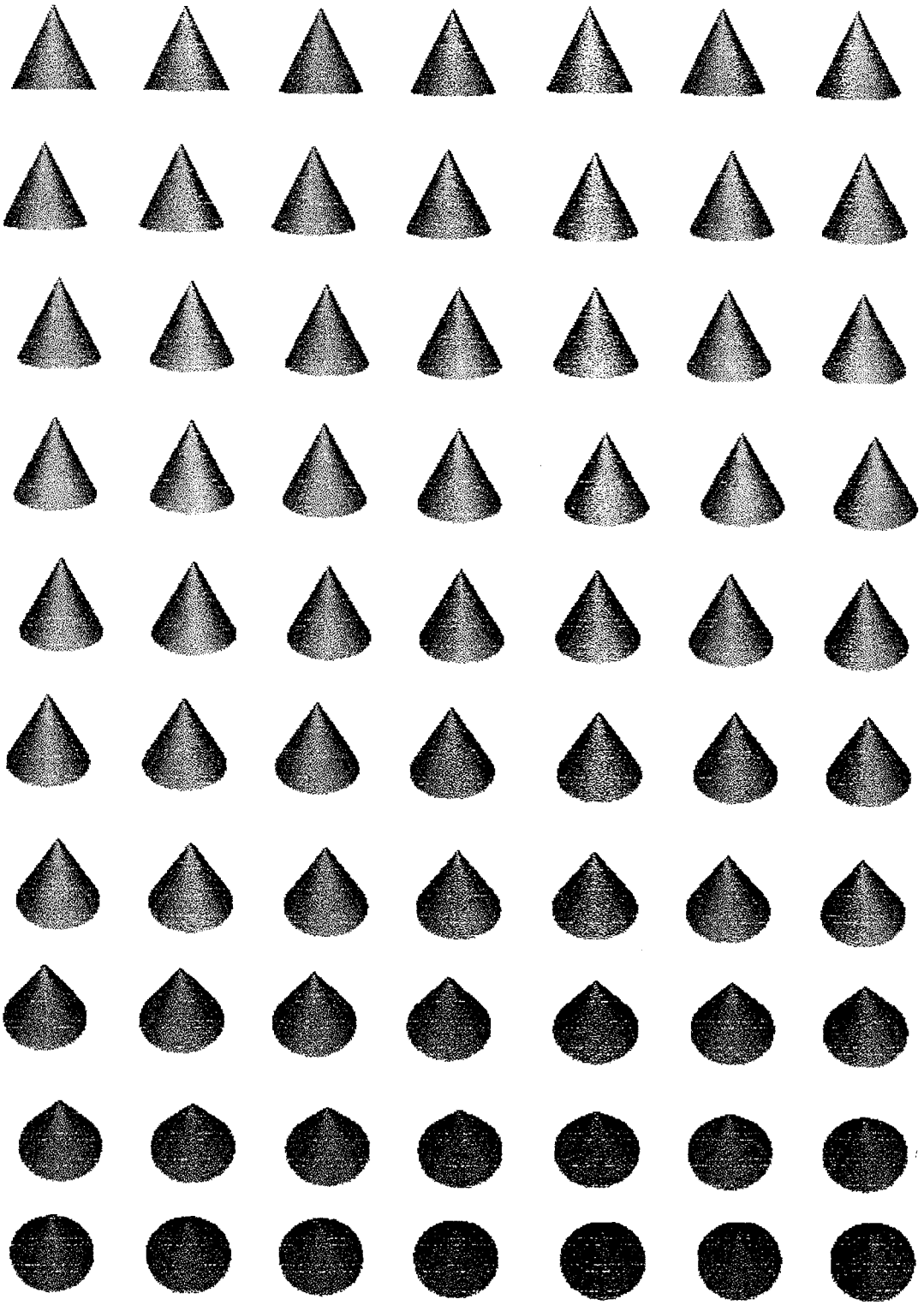


Şekil 8.1.(devam) Eğitimde Kullanılan Birinci Obje Sınıfı

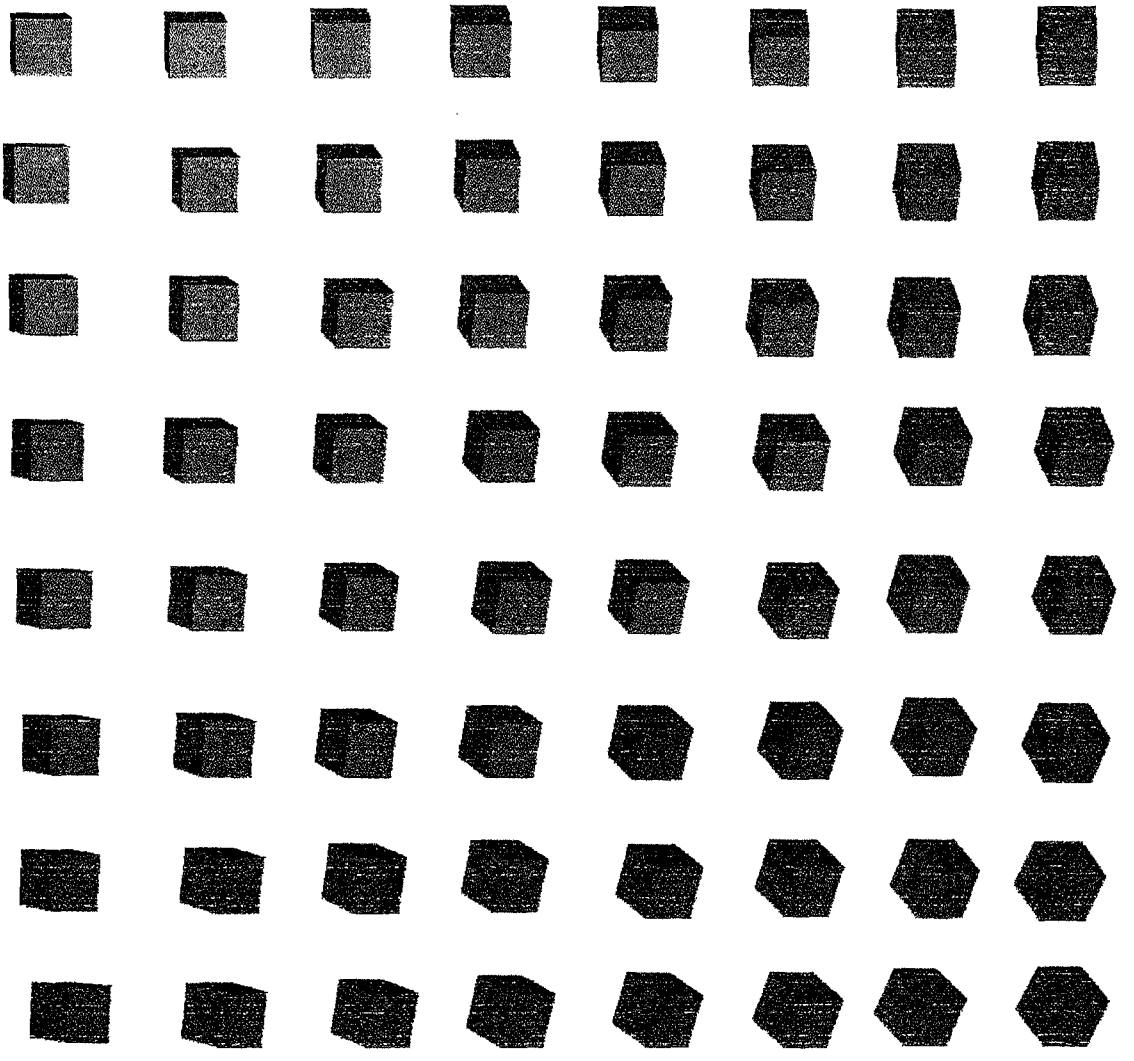


Şekil 8.2. Eğitimde Kullanılan İkinci Obje Sınıfı

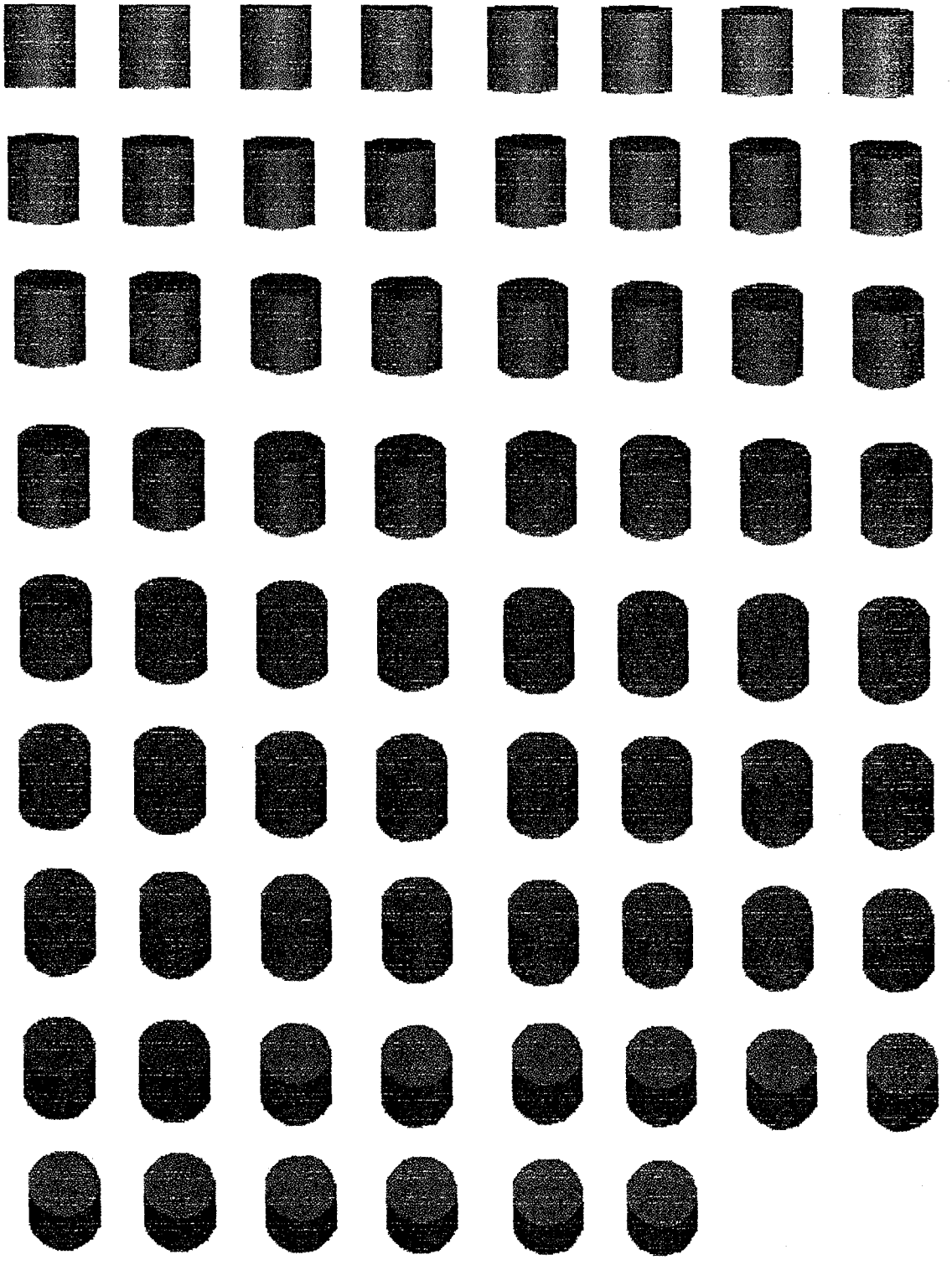
8.1.2. Uygulama_2'de kullanılan objeler



Şekil 8.3. Sinir ağının eğitiminde kullanılan 70 adet koni görüntüsü

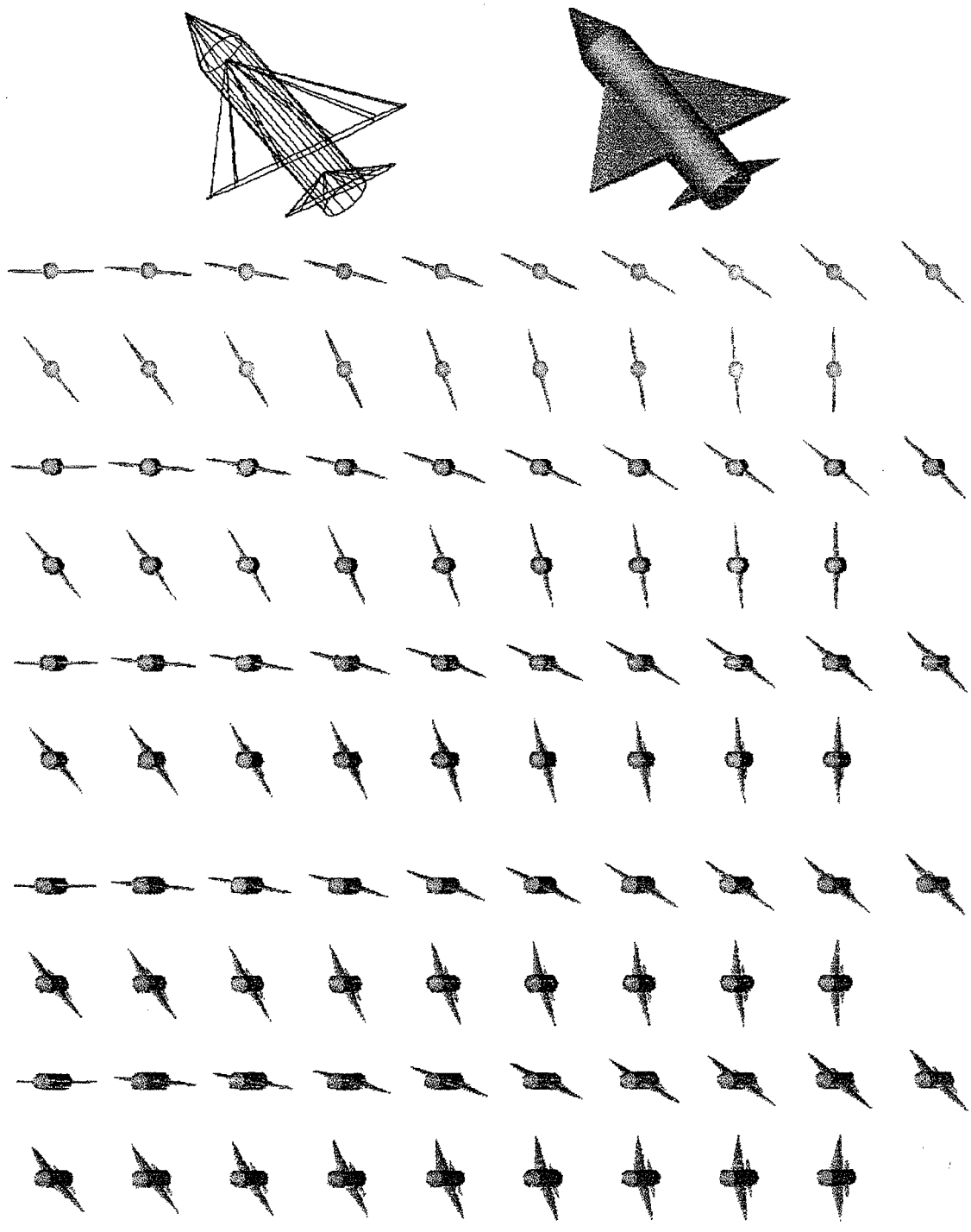


Şekil 8.4. Eğitimde kullanılan 64 adet küp görüntüsü

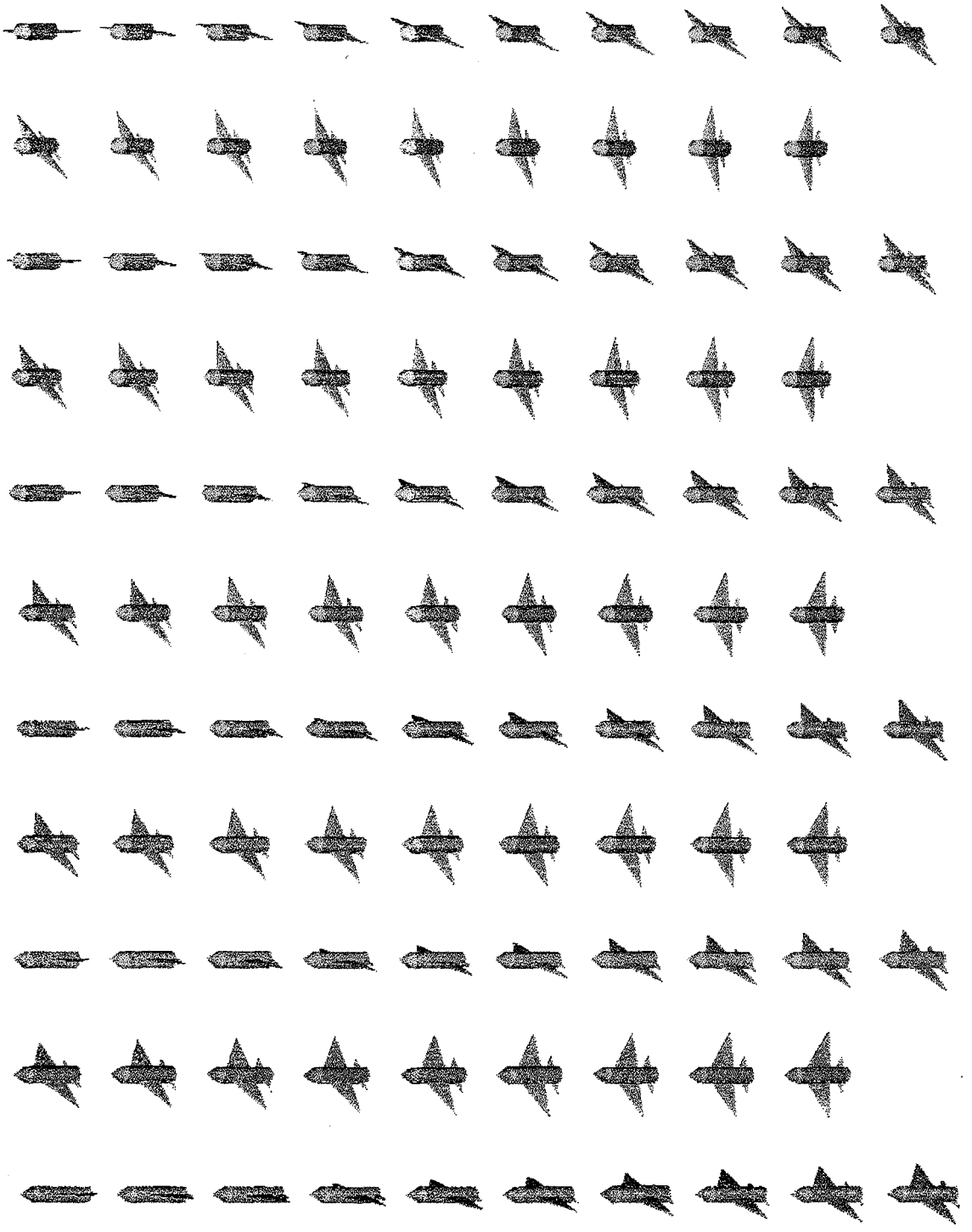


Şekil 8.5. Eğitimde kullanılan 70 adet silindir görüntüsü

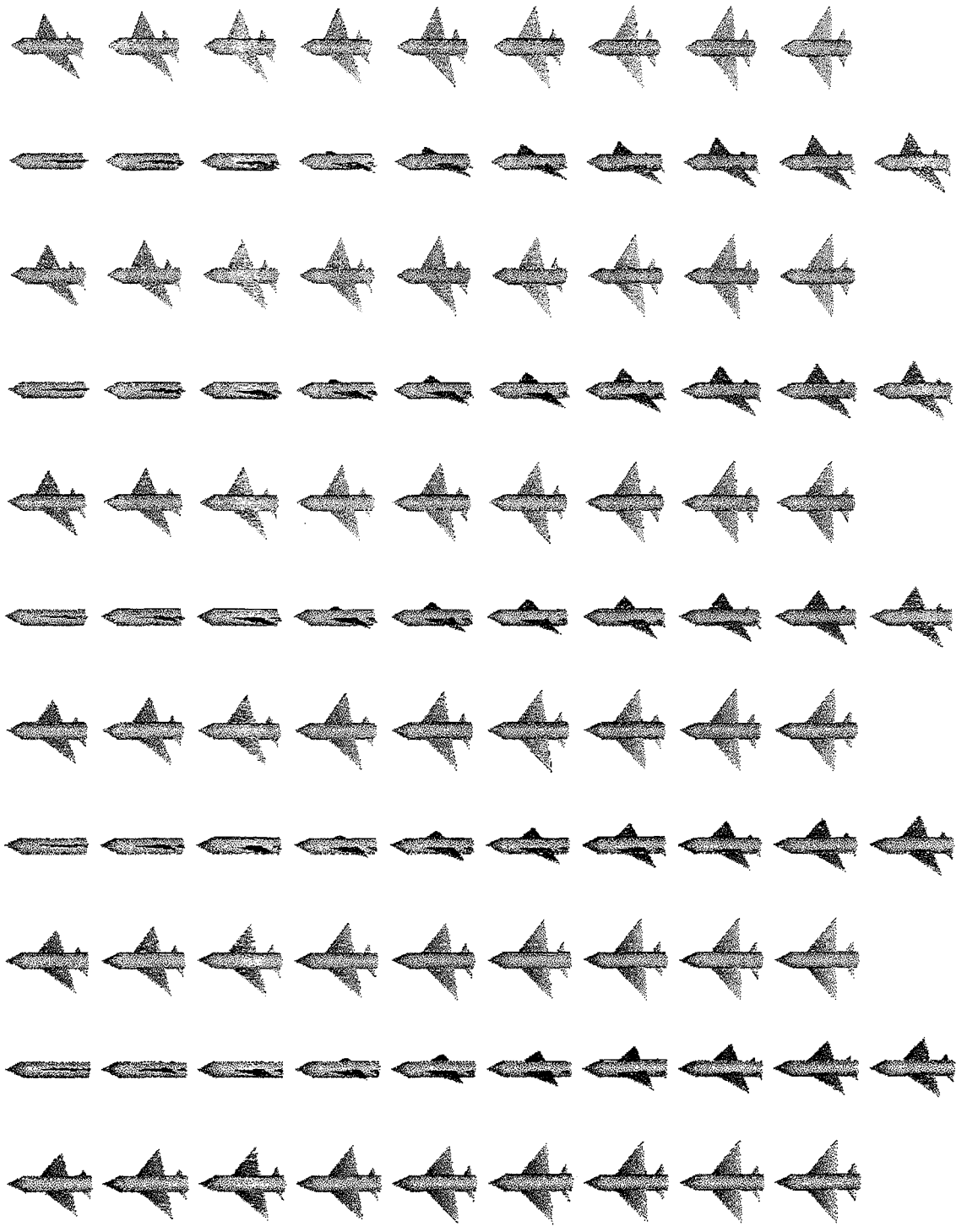
8.1.3. Uygulama_3'de kullanılan objeler



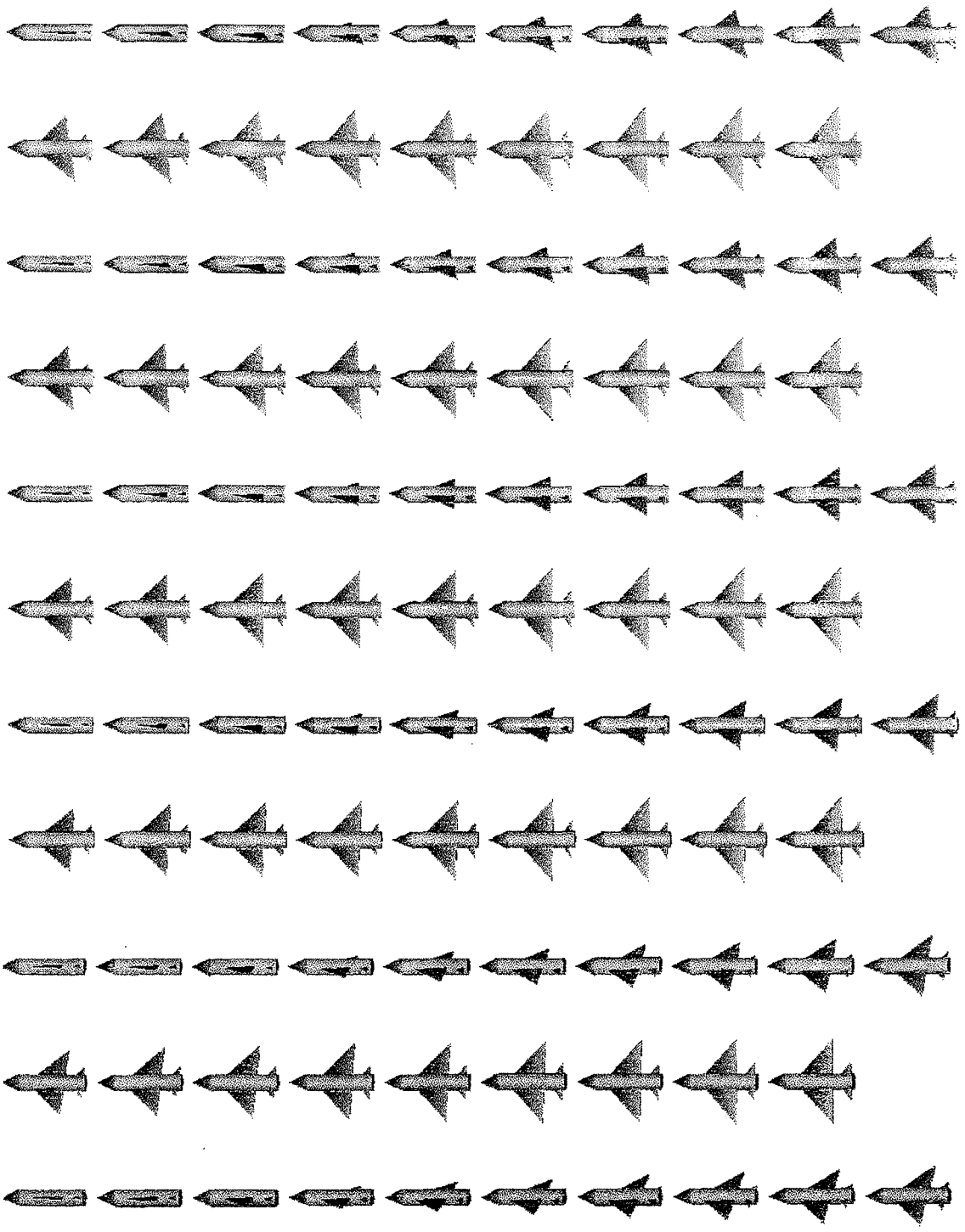
Şekil 8.6. Savaş uçağına ait perspektif görüntüler



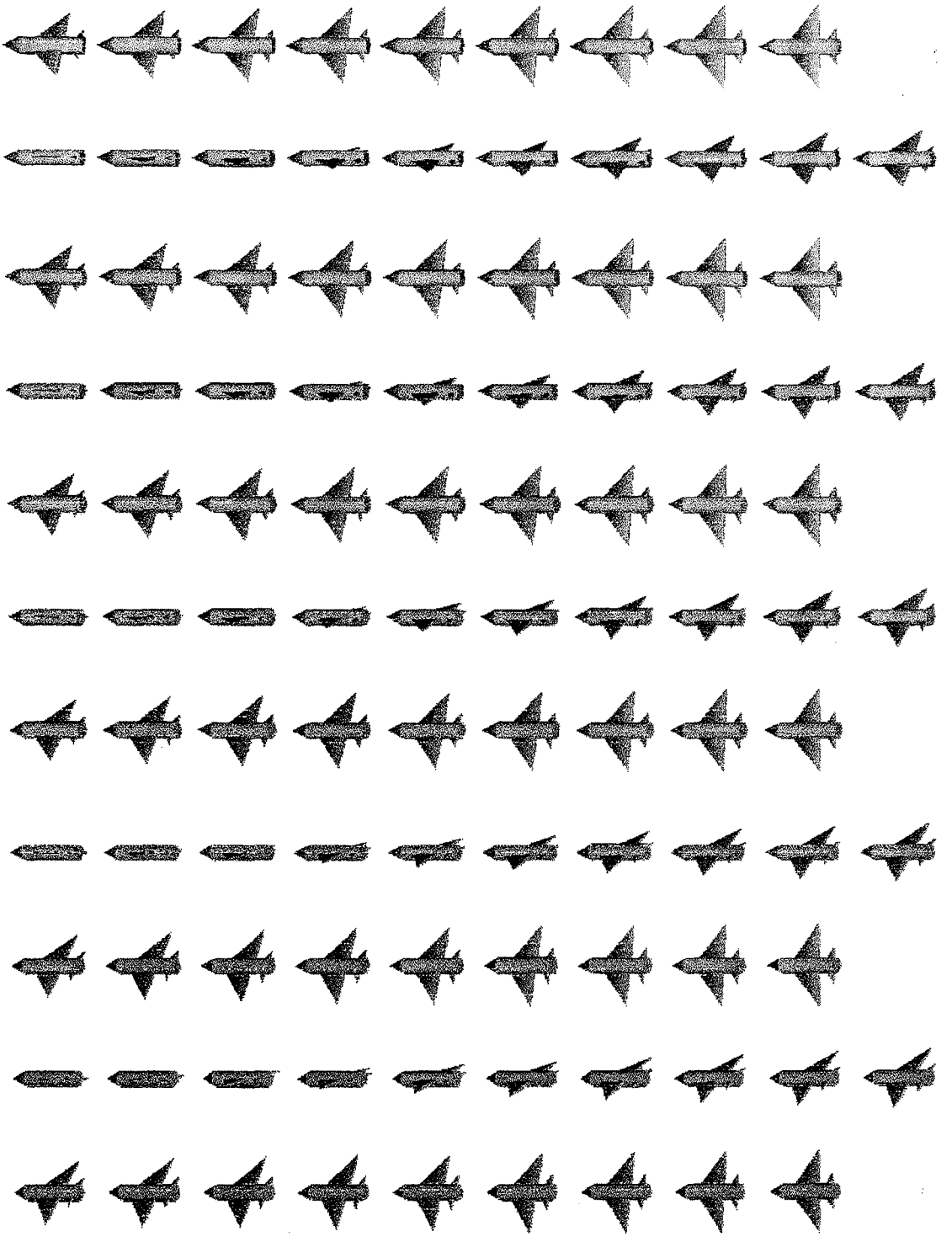
Şekil 8.6.(devam) Savaş uçağına ait perspektif görüntüler



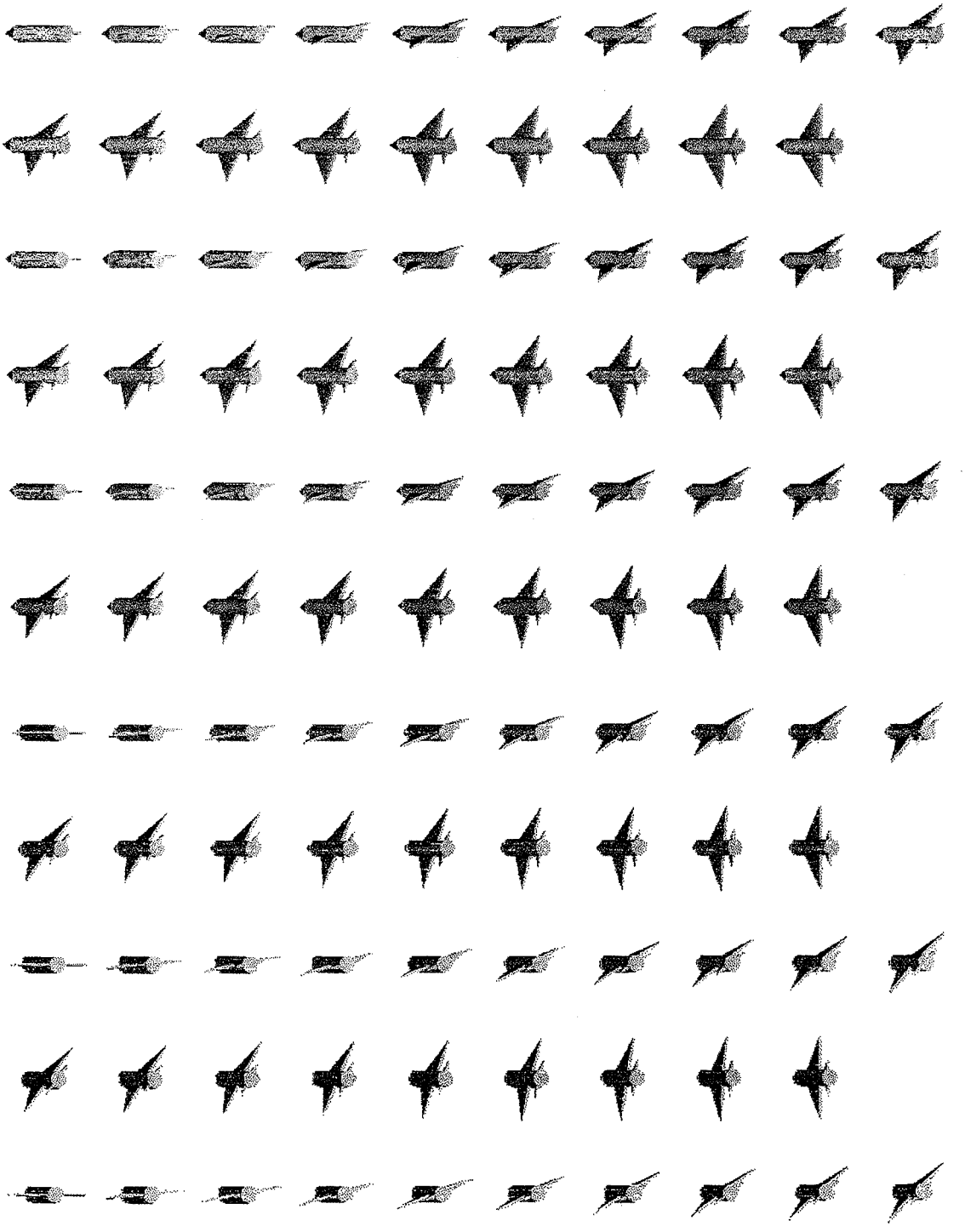
Şekil 8.6.(devam) Savaş uçağına ait perspektif görüntüler



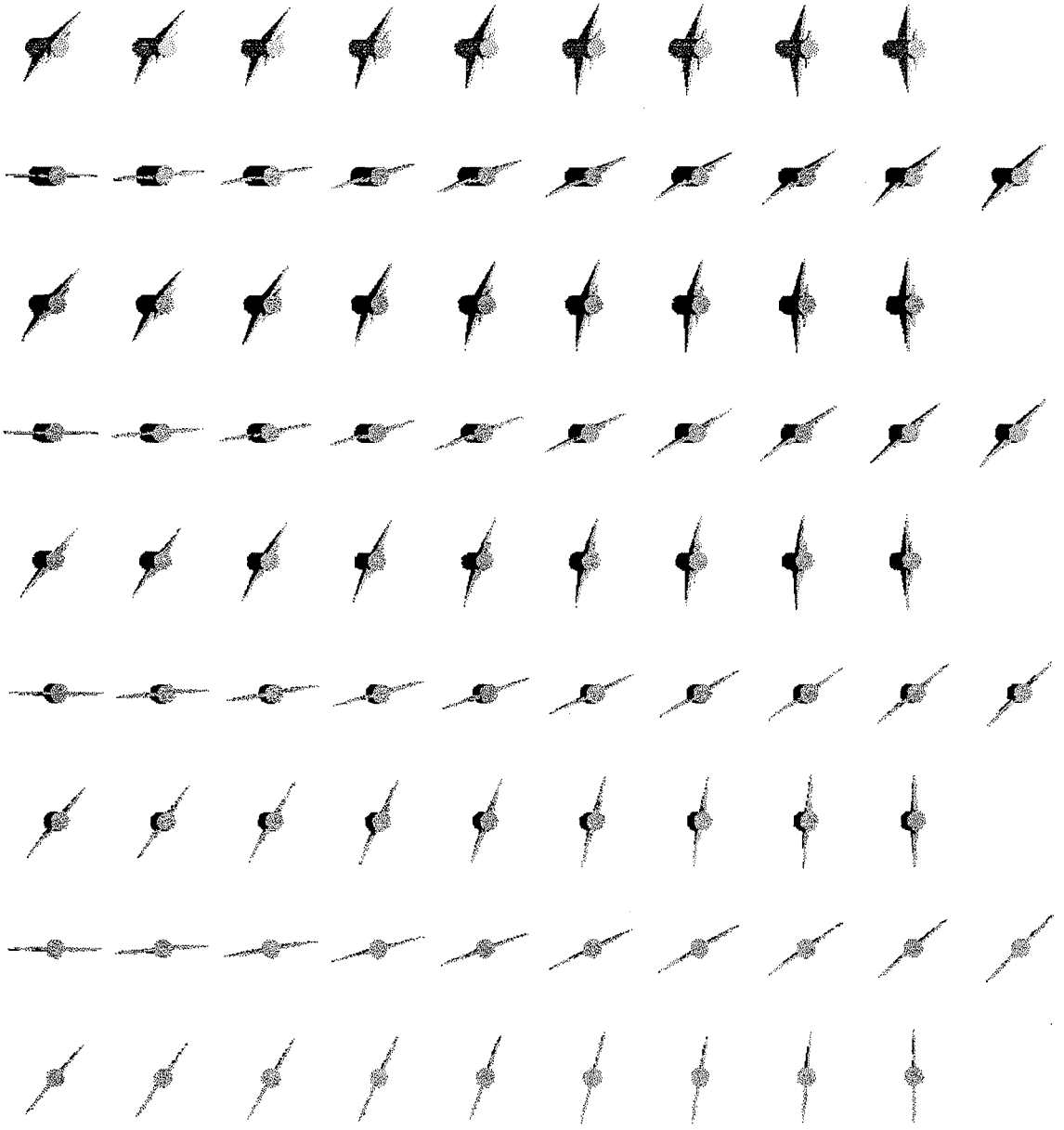
Şekil 8.6.(devam) Savaş uçağına ait perspektif görüntüler



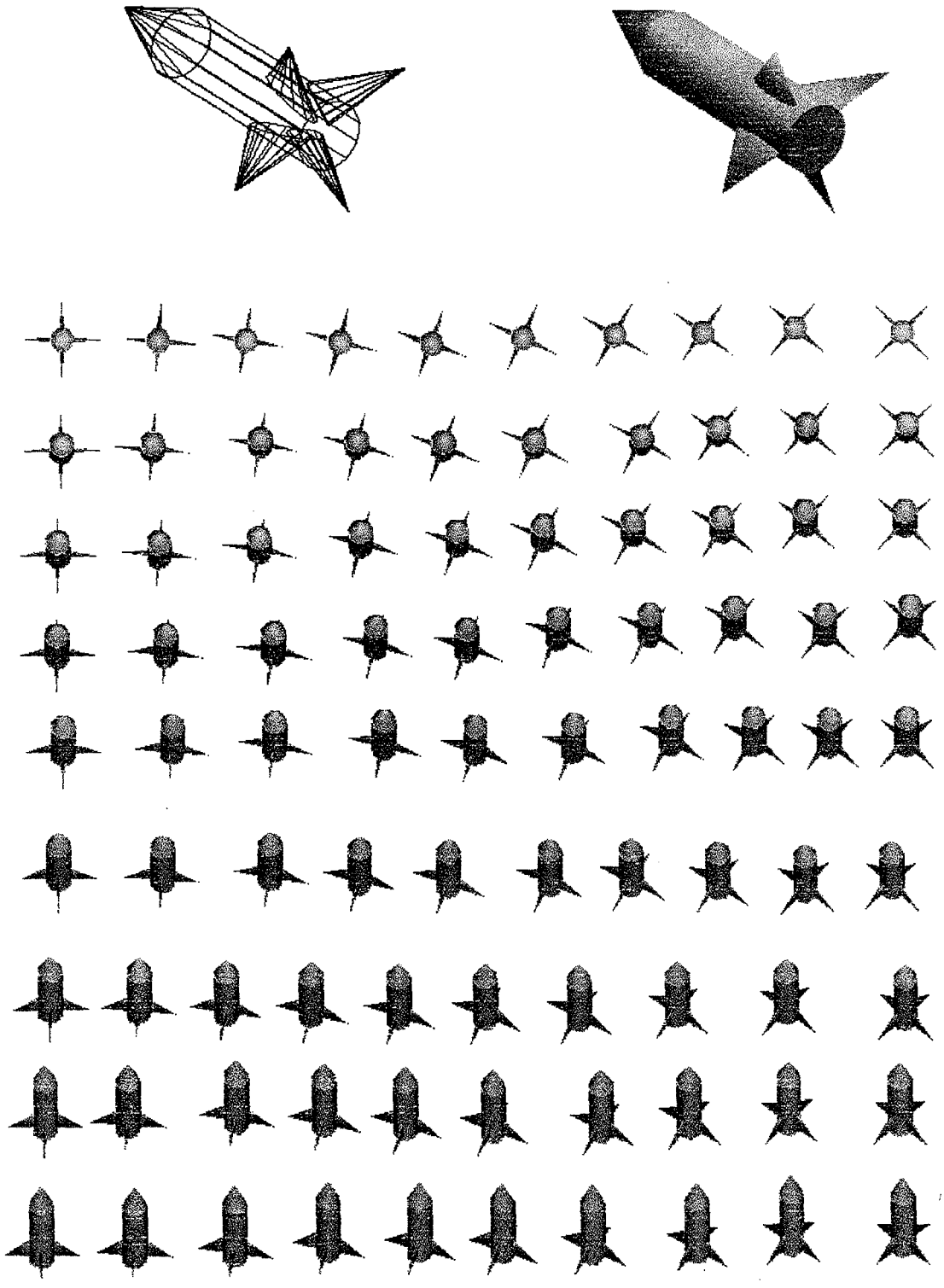
Şekil 8.6.(devam) Savaş uçağına ait perspektif görüntüler



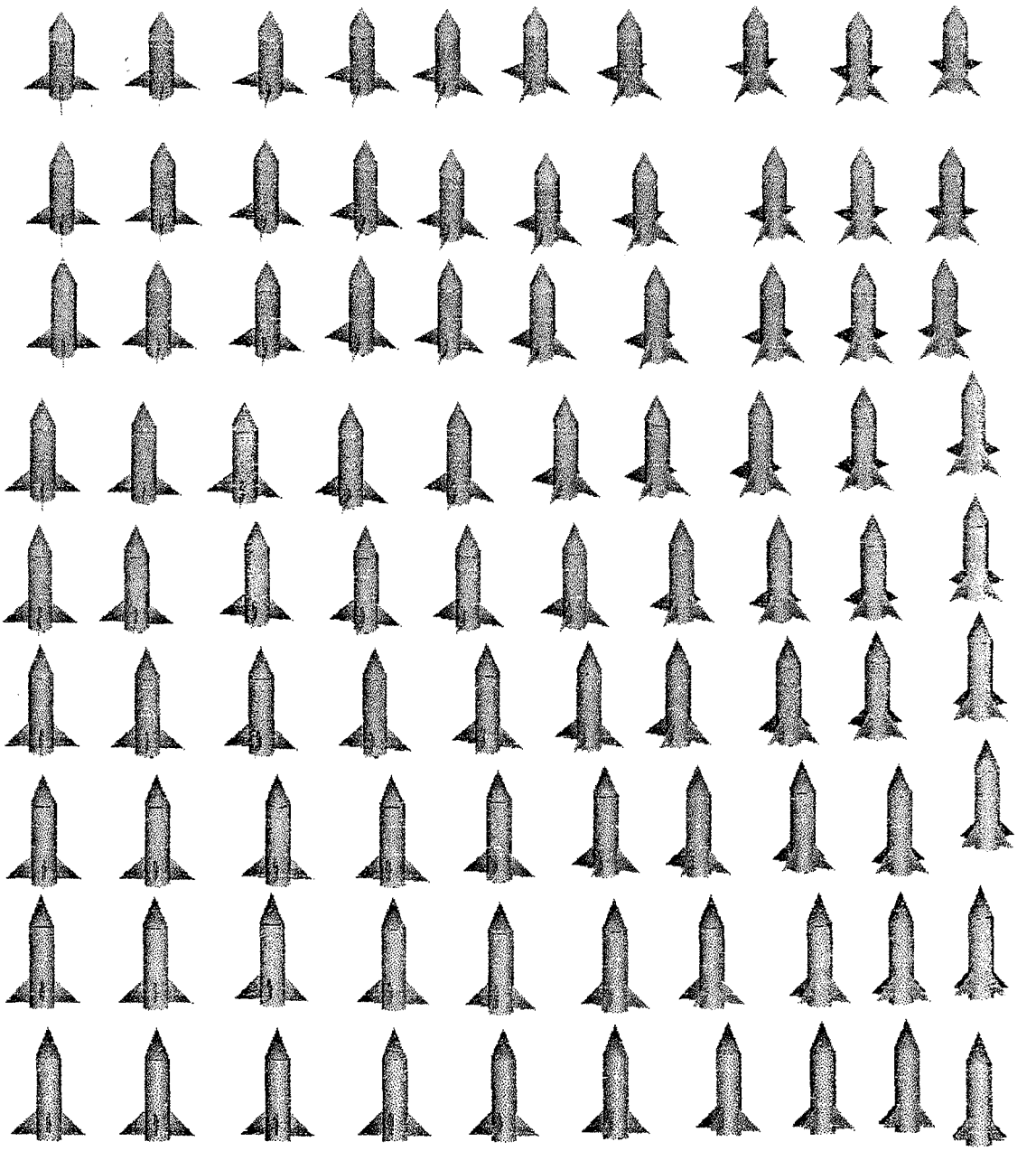
Şekil 8.6.(devam) Savaş uçağına ait perspektif görüntüler



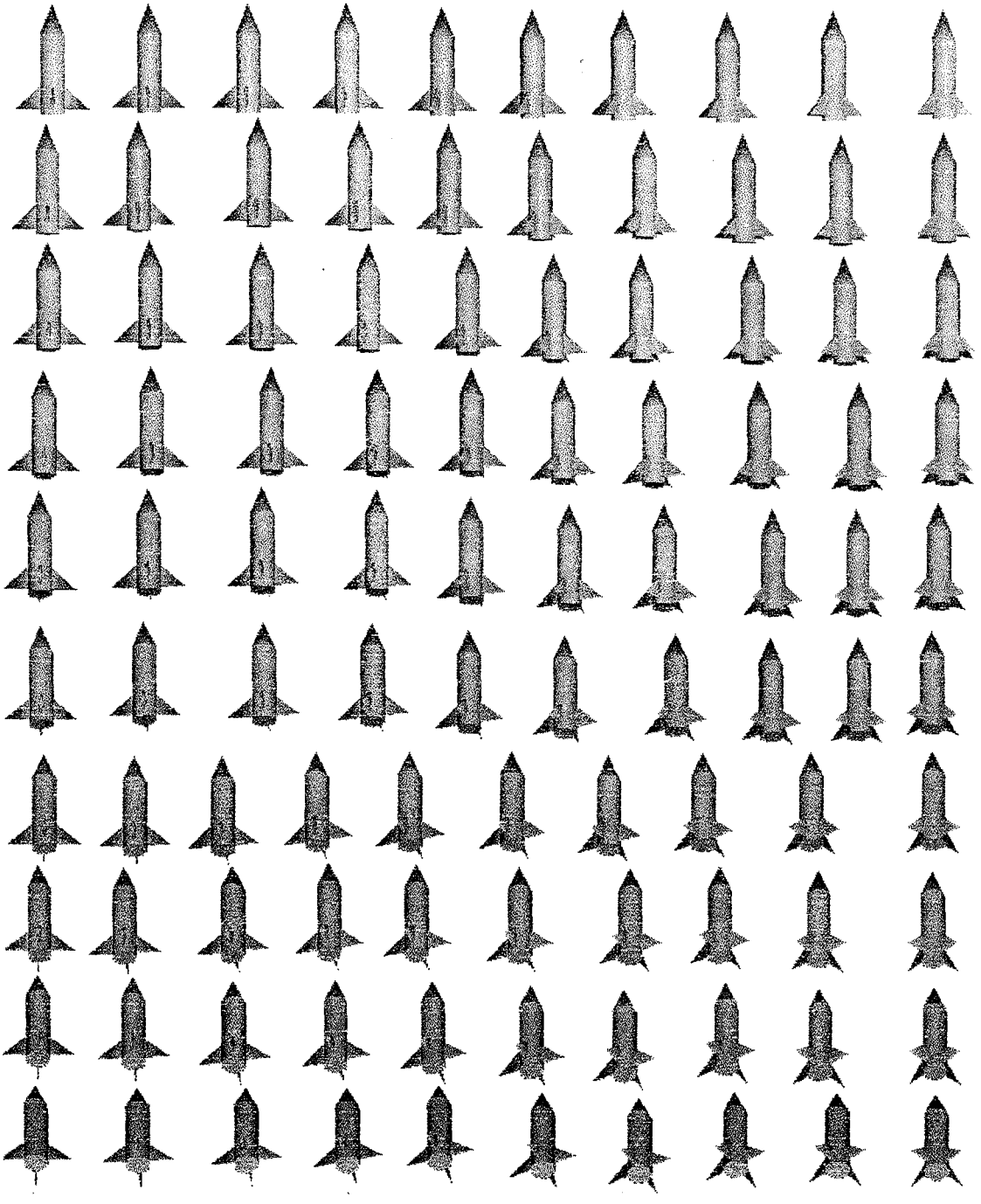
Şekil 8.6.(devam) Savaş uçağına ait perspektif görüntüler



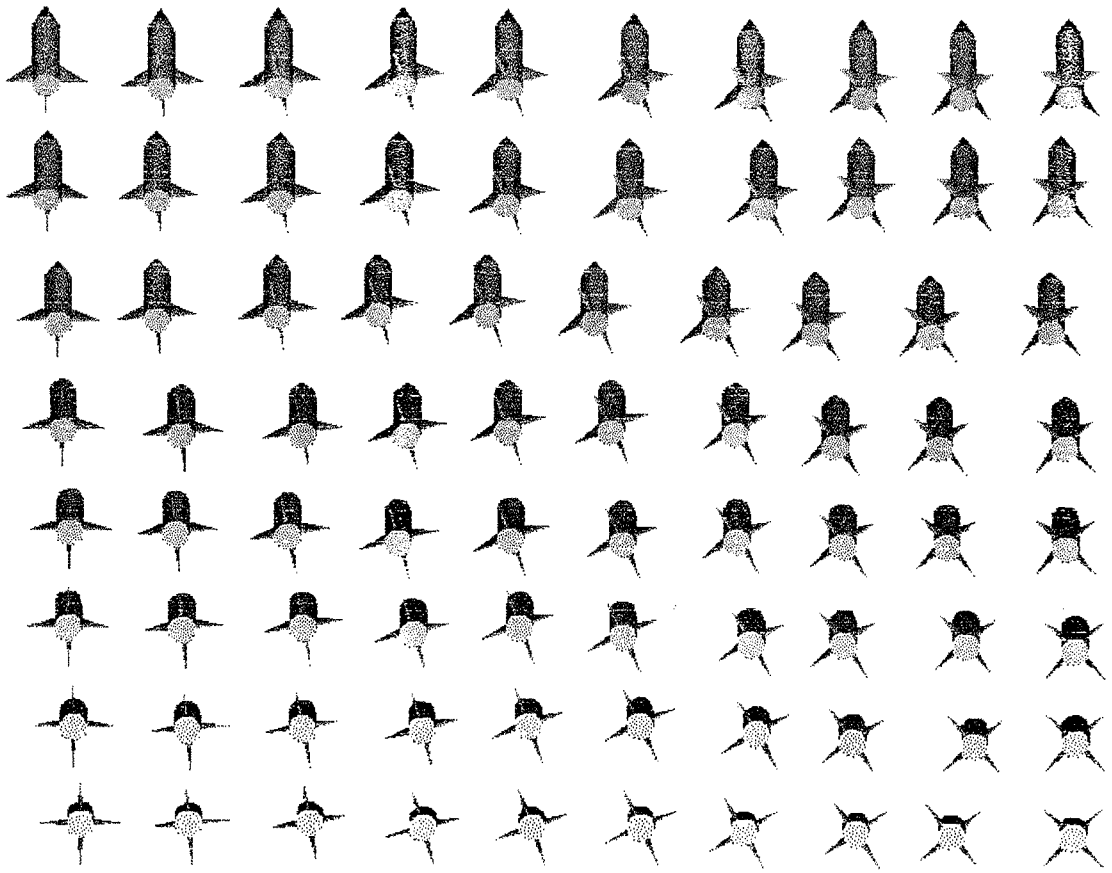
Şekil 8.7. Füzeye ait perspektif görüntüler



Şekil 8.7.(devam) Füzeye ait perspektif görüntüler

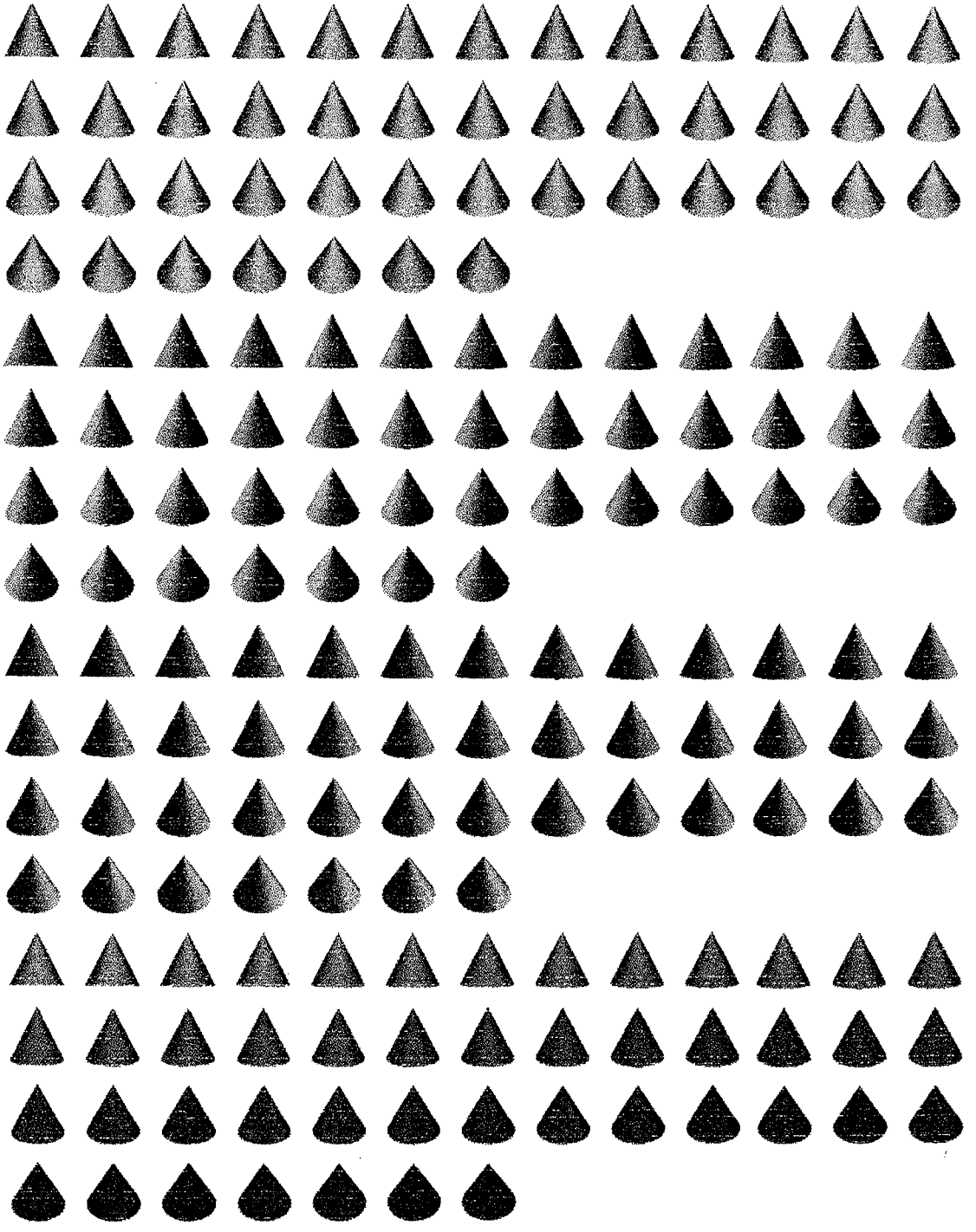


Şekil 8.7.(devam) Füzeye ait perspektif görüntüler

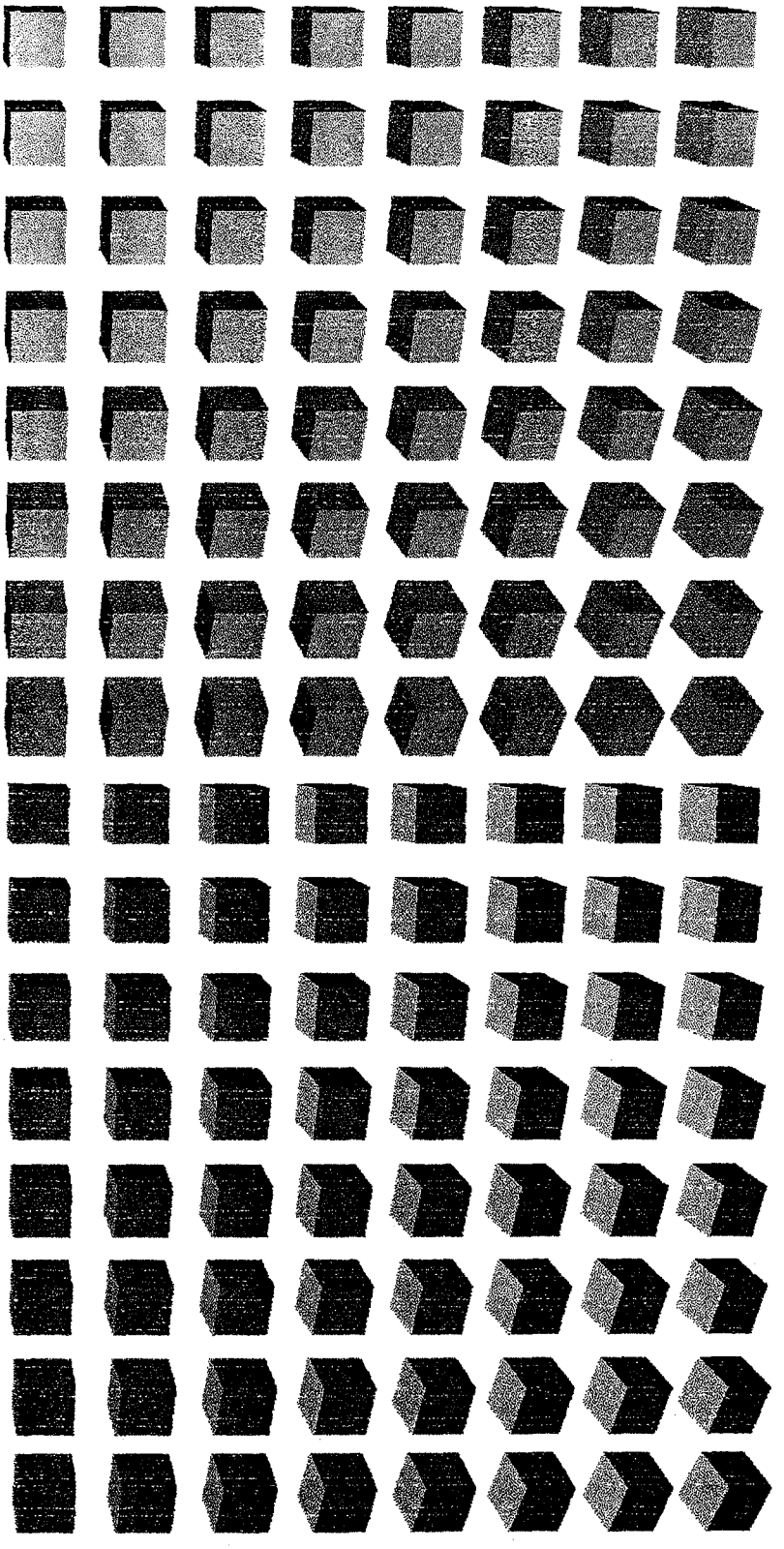


Şekil 8.7.(devam) Füzeye ait perspektif görüntüler

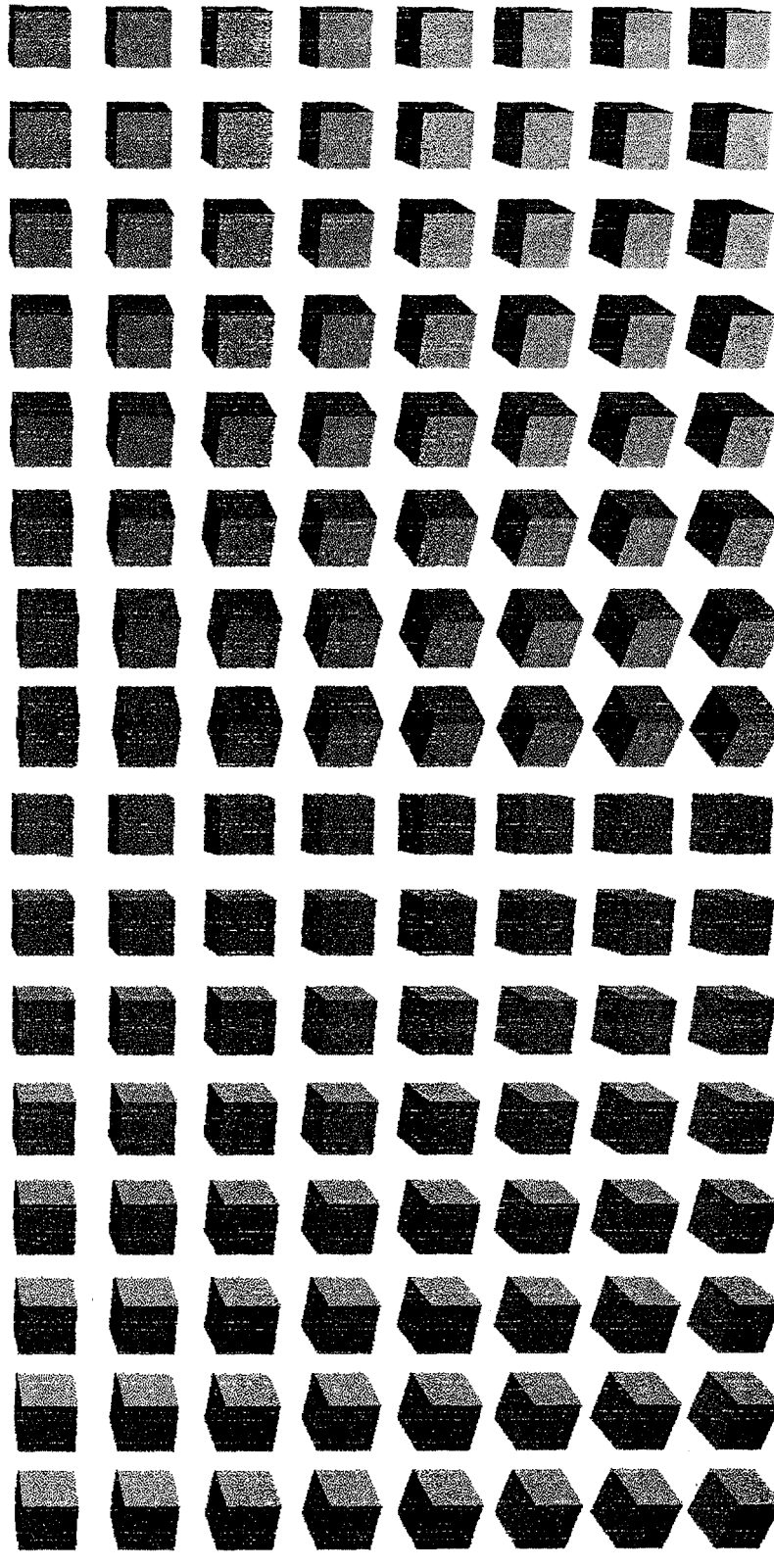
8.1.4. Uygulama_4'de kullanılan objeler



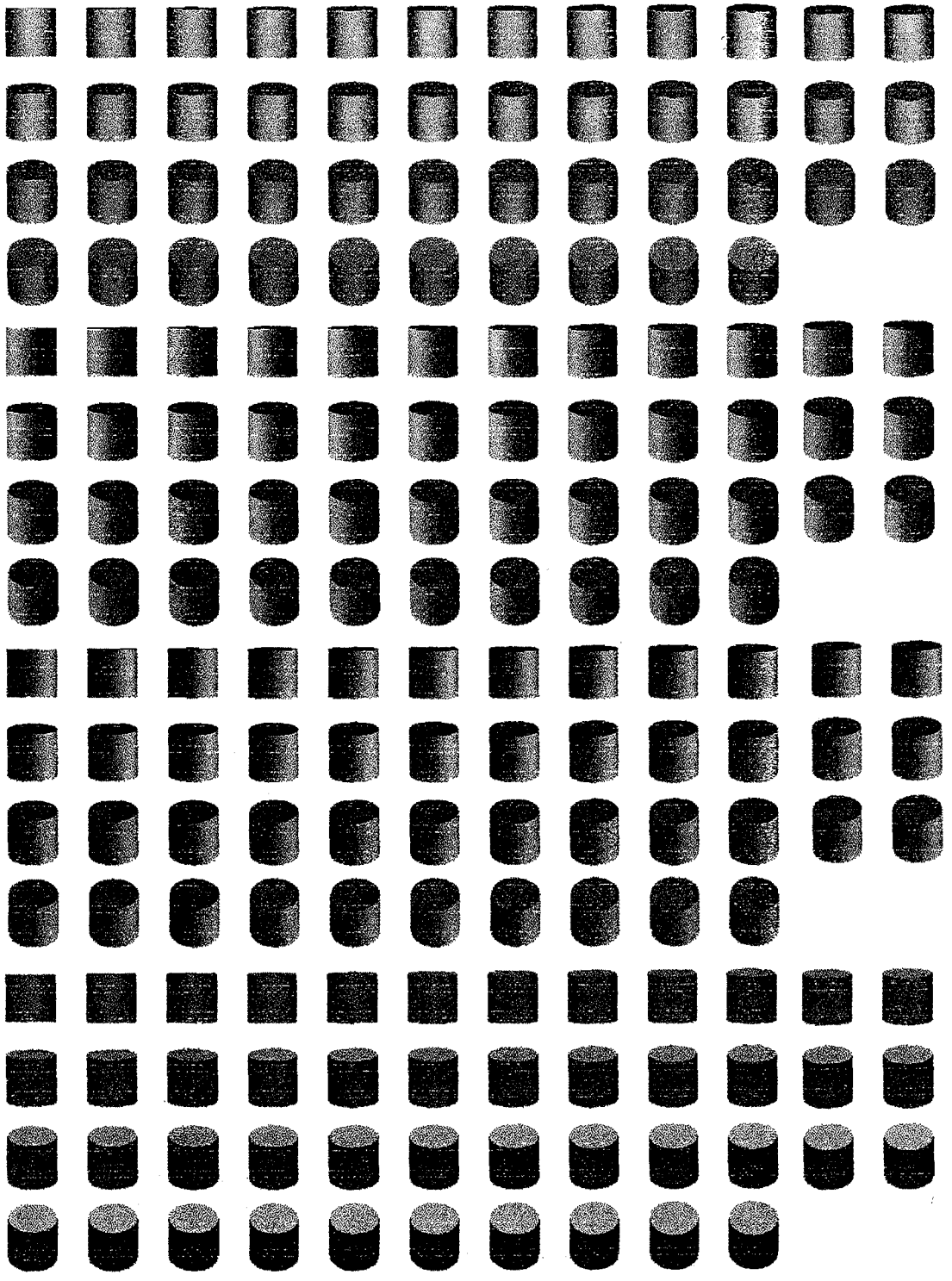
Şekil 8.8. Eğitimde kullanılan birinci obje sınıfı



Şekil 8.9. Eğitimde kullanılan ikinci obje sınıfı



Şekil 8.9.(devam) Eğitimde kullanılan ikinci obje sınıfı



Şekil 8.10. Eğitimde kullanılan üçüncü obje sınıfı

8.2. Ek-2: Obje Tanıma Sistemini Simule Eden Programlar

8.2.1 Programlar Hakkında Açıklamalar

Aşağıda listesi verilmiş olan programlar C++ dilinde yazılmıştır. ".inc" uzantılı program listeleri (mom_ext.inc ve neural.inc), ".cpp" uzantılı programların kullandığı kaynak kodlarıdır ve derleyicinin standart kaynak kütüpanelerinin bulunduğu klasöre koyulmalıdır.

"Momext.cpp " programı objelerin bölütlenmiş görüntülerini kullanarak, moment vektörlerini bulur. Bu görüntüler, herhangi bir grafik editörü kullanılarak,, programların kullandığı klasörlere ".raw" formunda kaydedilmiş 8 bit gri seviye görüntülerdir. Bu program, bir görüntünün normal gri seviye, siluet ve kenarları ayrıştırılmış olarak üç çeşit görüntüsünü bulur ve bunlara ait standart moment vektörlerini ve Hu'nun binary görüntüler için bulduğu moment vektörlerini bulur. Program, belirtilen herhangi bir obje sınıfına ait klasördeki görüntülerin yukarda belirtilen şekilde momentlerini bulup aynı klasörde aşağıdaki dosyalara yazar.

Bmv.wri : normal gri seviye görüntülerin Hu momentleri
Bmv_e.wri: Kenarları ayrıştırılmış görüntülerin Hu momentleri
Bmv_s.wri: Sillüyet görüntülerin Hu momentleri
Gmv.wri : normal gri seviye görüntülerin standart momentleri
Gmv_e.wri: Kenarları ayrıştırılmış görüntülerin standart momentleri
Gmv_s.wri: Sillüyet görüntülerin standart momentleri

Obje tanıma sistemin eğitiminden sonra kullanılan test objeleri ise yine her obje sınıfına ait olan klasördeki "test" isimli klasörde bulunur ve bu klasördeki test görüntülerinin de, yukarıda belirtildiği gibi moment tanımlayıcıları bulunup bu klasörün içine yazılır.

"Objrec.cpp" programı ise, "Momext.cpp" programının hesapladığı, yukarıdaki dosyalarda bulunan moment vektörlerini okur ve bu momentlerle yapay sinir ağını eğitir, ve test eder. Eğitim ve testlere ait tüm istatistiksel bilgileri sunar. Eğitimin performansı gerçek-zamanlı olarak grafiksel veya istatistiksel biçimde takip edilebilir ve eğitim algoritmasının parametrelerine müdahale edilebilir. Ayrıca eğitimin yakınsama eğrisini "conv.wri" dosyasına kaydeder. Eğitilmiş sinir ağının tüm verilerini "Net_dat.wri" dosyasına kaydeder. Program, birçok fonksiyonel işlevleri (Load, Save, vb..) bir menü yardımıyla gerçekleştirir.

8.2.2. Program rutinlerinin açıklaması

MOM_EXT.İNC

_long double Sqr(long double x)

x sayısının karesini hesaplar

_int fac(int a)

a sayısının faktoriyelini hesaplar

_int Combination(int p,int r)

p'nin r kombinasyonunu hesaplar

_long double momentpq(int M,int N,float *I,int p,int q)

'I' objesinin p+q'uncu derece momentini bulur.

_void CentralPositions(int M,int N,float *I,long double *Xu,long double *Yu)

'I' objesinin ağırlık merkezini bulur.

_long double CentralMomentpq(int M,int N,float *I,int p,int q)

'I' objesinin p+q'uncu derece merkezi momentini bulur.

_void ByMeansOfCentralMoments(int M,int N,float *I,long double *m,double *teta)

'I' objesinin merkezi momentlerini, direk denklemlerle hesaplar, böylelikle yukarıdaki fonksiyondan daha hızlı hesaplar.

_long double Fi(int p,int q,double t,long double *m)

'I' objesinin dönme etkilerine karşı sabitlenmiş p+q'uncu derece standart momentini bulur.

_void NormalizedByMeansOfCentralMoments(int M,int N,float *I,long double *m)

'I' objesinin dönme ve boyut değişimlerine karşı sabitlenmiş standart momentlerini hesaplar.

_void ByMeansOfMomentInvariants(int M,int N,float *I,long double *v)

'I' objesinin tüm değişimlere karşı sabitlenmiş standart moment vektörünü bulur.

_long double NormalizedCentralMomentpq(int M,int N,float *I,int p,int q)

'I' objesinin p+q'uncu dereceden boyutsal olarak sabitlenmiş merkezi momentini bulur.

_void MomentInvariants(int M,int N,float *I,long double *v)

Hu'nun bulduğu moment sabitlerinin hepsini hesaplar.

_void MomentsOfTheRawImage(long double *p,char *c)

'c' de belirtilen isimdeki '.raw' görüntüdeki objenin moment tanımlayıcılarını hesaplar.

_void ByMeansOfMomentsOfTheRawImage(long double *p,char *c)

'c' de belirtilen isimdeki '.raw' görüntüdeki objenin standart moment tanımlayıcılarını hesaplar.

_void MomentInvariantsForGreyImages(long double *m,char *s)

's' de belirtilen isimdeki '.raw' görüntüdeki objenin standart moment tanımlayıcılarını hesaplar ve altı boyutlu m vektörüne koyar. Bir görüntü dosyasındaki objenin standart moment tanımlayıcılarını bulmak için bu fonksiyonu çağırmak yeterlidir. Bu fonksiyon yukarıdaki fonksiyonları kullanarak 'm' vektörünü hesaplar.

_void MomentInvariantsForBinaryImages(long double *m,char *s)

's' de belirtilen isimdeki '.raw' görüntüdeki objenin moment tanımlayıcılarını Hu'nu denklemlerinde belirtilen şekliyle hesaplar ve yedi boyutlu m vektörüne koyar. Bir görüntü dosyasındaki objenin moment tanımlayıcılarını bulmak için bu fonksiyonu çağırmak yeterlidir. Bu fonksiyon yukarıdaki fonksiyonları kullanarak 'm' vektörünü hesaplar.

_void Gradient(int M,int N,float *p1,float *p2)

'p1' objesinin gradient kenarları ayrıştırılmış şeklini bulur ve 'p2' ye koyar.

_void RobertsEdgeDetection(int M,int N,float *p1,float *p2)

'p1' objesinin kenarlarını, robert maskeleriyle bulur ve sonucu 'p2' ye koyar

_void TresholdingToEdgeExtractedImage(int M,int N,float *p1,float t)

Kenarları ayrıştırılmış 'p1' objesini binary yapmak için 't' treshold değerini uygular ve sonucu 'p1' e koyar.

_void Silluet(int M,int N,float *p1)

'p1' objesinin sillüyet görüntüsünü bulup yine 'p1' e koyar.

MOMEXT.CPP

_void Mom_Extractor(int n,char *c,int k,float t)

'c' ile belirtilen obje klasöründeki n tane objenin normal, kenar ayrıştırılmış ve sillüyet görüntülerinin moment tanımlayıcılarını ve standart moment tanımlayıcılarını bulur ve daha önce açıklanan dosyalara kaydeder.

NEURAL.İNC

_int Round(float x)

x sayısını yuvarlar.

_void SetMode(int Mode)

Ekranı, grafik veya text moda ayarlar

_void PreCalc()

Grafik ekranı için ön hazırlık yapar.

_void Plot(int x, int y, Byte color)

Ekranın (x,y) noktasına color'da belirtilen renkte bir pixel koyar.

_void ClearPalette(PaletteRegister Color)

Renk paletini temizler.

_void SetPalette(PaletteRegister Hue)

Paletin renk sayısını ve başlangıç durumlarını ayarlar.

_void InitPalette(PaletteRegister Color,int Pal)

Renk paletinin renklerini ayarlar, 'pal' değişkenindeki sayı palet çeşidini belirtir. 1 ve 2 ise renkli paletler 3 ise gri seviye palet oluşturulur.
_void Swap(int *first, int *second)

İki değişkenin içeriklerini karşılıklı değiştirir.

_void InitGraphics(int Pal)

Grafik modu, 'pal' paletiyle açar.

_void WaitForKey()

Tuşa basılmasını bekler

_void ExitGraphics()

Grafik modu kapatır.

_void Draw(int xx1, int yy1, int xx2, int yy2, Byte color)

p1 noktasından p2 noktasına colar rengiyle çizgi çizer.

_void Grey_Grph(void)

Gri seviye grafik modunu açar.

_void Exit_Grph(void)

Grafik modu kapatır.

_double sqr(double x)

'x' sayısının karesini alır.

_int NearestFloor(double f)
'f' sayısını yuvarlar.

_void NoMemRem(void)

Belleğin bittiği uyarısını verir.

_void CantOpFile(void)

Dosya açılamadı uyarısını verir.

_int GetTheRawImageDim(char *strng)
'strng' ile belirtilen isimdeki '.raw' kare görüntüsünün boyutunu bulur.
_void GetTheRawImage(float *p,char *strng,int d)

'strng' isimli ve dxd boyutlu '.raw' kare görüntüyü okur ve 'p' ye koyar.

_void PutImagexy(int M,int N,float *p,int x,int y)

Ekranın (x,y) noktasına 'p' görüntüsünü koyar.

_void PutTheRawImage(int M,int N,float *p, char *strng)

Ekranın ortasına 'p' görüntüsünü koyar.

_float Extract(long double l)

'l' sayısının işaretini kaybetmeden logaritmasını bulur.

OBJREC.CPP

_long double vm(int d,long double *p1,long double *p2)
'd' boyutlu 'p1' ve 'p2' vektörlerini scalar olarak çarpar.

_void vg(int x,int d,float far *p1,long double *p2)

'p1' vektör dizisinin 'x' inci vektörünü 'd' boyutlu'p2' vektörüne koyar.

_int khit(void)

Klavye flag'ını, bir tuşa basılıp basılmadığını kontrol etmek için yoklar.

_int GetScanCode(void)

Klavyeden basılan bir tuşun 'Scan_Code' unu yollar.

_double ANumber(void)

Küçük değerli random sayılar üretir.

_void Hot_Keys(int i,int j)

Her an kullanılabilecek olan tuşların listesini ekrana basar.

_void Save_Report(NetworkModel NM,int NumberOfLayers,
int *NumberOfNeuronOfLayers,long double Momentum,
long double LearningRate,long double b,unsigned long int ite,char *c)

Kendine gelen sinir ağını ve parametreleri, 'c' de belirtilen isimle text modda dosyaya rapor eder.

_void Save(NetworkModel NM,int NumberOfLayers,
int *NumberOfNeuronOfLayers,long double Momentum,
long double LearningRate,long double b,
long double uy,unsigned long int ite,char *c)

Kendine gelen sinir ağını ve parametreleri, 'c' de belirtilen isimle dosyaya kendi okuyabileceği formatta kaydeder.

_NetworkModel Load(
NetworkModel NM,int *NumberOfLayers,int *NumberOfNeuronOfLayers,
long double *Momentum,long double *LearningRate,
long double *b,long double *uy,unsigned long int *ite,char *c)

Kendine gelen sinir ağına ve parametrelerine, 'c' de belirtilen isimdeki dosyadan okuduğu sinir ağını yükler.

_void LRowVecPut(int N,int n,float far *p,long double *v)

'p' vektör dizisinin 'n' inci vektörünü 'v' ye koyar

_void LRowVec(int N,int n,float far *p,long double *v)

'v' vektörünü 'p' vektör dizisinin 'n' inci elemanı olarak 'p' ye koyar.

_void Silluet(int M,int N,float *p1)

'p1' görüntüsünün sillüyetini bulur ve 'p1' e koyar.

_void RobEdDet(int M,int N,float *p1)

'p1' görüntüsünün kenarlarını, robert maskeleriyle bulur ve 'p1' e koyar.

_void GroupDisplayer(int k,int n,char *cs)

'cs' de belirtilen klasördeki n tane ekranda gösterir.

_void ObjectDisplayer(int k,int n,char *cs)

'cs' de belirtilen klasördeki n'inci objeyi ekranda gösterir.

8.2.3. Program listeleri

```
//                                    MOM_EXT.İNC

//      MOMEXT.CPP programının kullandığı kaynak dosya

#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <math.h>
```

```
#include <conio.h>
#include <time.h>
#include <stdlib.h>
#include <neural.inc>


long double Sqr(long double x){return(x*x);}


int fac(int a){register int k,s=1;
for(k=a;k>1;k--) s*=k;return(s);}


int Combination(int p,int r)
{return(fac(p)/(fac(r)*fac(p-r)));}


long double momentpq(
int M,int N,float *I,int p,int q)
{register int y,x;double xp,yq;long double l=0.;
for (y=0;y<M;y++)for(x=0;x<N;x++){
if((x==0&&p==0))xp=1.;else xp=pow(x,p);
if(y==0&&q==0) yq=1.;else yq=pow(y,q);
l+=*(I+y*N+x)*xp*yq;}return(l);}


void CentralPositions(
int M,int N,float *I,long double *Xu,long double *Yu){
long double m00;
m00=momentpq(M,N,I,0,0);
*Xu=momentpq(M,N,I,1,0)/m00;
*Yu=momentpq(M,N,I,0,1)/m00;}


long double CentralMomentpq(
int M,int N,float *I,int p,int q){
long double xp,yq,xu,yu,xx,yy,l=0.0;register int x,y;
CentralPositions(M,N,I,&xu,&yu);
for(y=0;y<M;y++)for(x=0;x<N;x++){
xx=1.*x-xu;yy=1.*y-yu;
if(xx==0.&&p==0) xp=1.;else xp=powl(xx,p);
if(yy==0.&&q==0) yq=1.;else yq=powl(yy,q);
l+=*(I+y*N+x)*xp*yq;}return(l);}
```

```
void ByMeansOfCentralMoments(
int M,int N,float *I,long double *m,double *teta){
long double m00,m20,m11,m02,m30,m21,m12,m03,cpx,cpy;
CentralPositions(M,N,I,&cpx,&cpy);
m00=momentpq(M,N,I,0,0);m11=momentpq(M,N,I,1,1);
m12=momentpq(M,N,I,1,2);m21=momentpq(M,N,I,2,1);
m03=momentpq(M,N,I,0,3);m30=momentpq(M,N,I,3,0);
m02=momentpq(M,N,I,0,2);m20=momentpq(M,N,I,2,0);
*m=m00;                                          /* M00 */
*(m+1)=0.;                                 /* M01 */
*(m+2)=0.;                                       /* M10 */
*(m+3)=m20-m00*Sqr(cpx);                                 /* M20 */
*(m+4)=m11-m00*cpx*cpy;                                  /* M11 */
*(m+5)=m02-m00*Sqr(cpy);                                 /* M02 */
*(m+6)=m30-3*m20*cpx+2*m00*cpx*Sqr(cpx);       /* M30 */
*(m+7)=m21-m20*cpy-2*m11*cpx+2*m00*Sqr(cpx)*cpy;/* M21 */
*(m+8)=m12-m02*cpx-2*m11*cpy+2*m00*cpx*Sqr(cpy);/* M12 */
*(m+9)=m03-3*m02*cpy+2*m00*Sqr(cpy)*cpy;       /* M03 */
m00=*(m+4)*2.;m11=*(m+3)-*(m+5);
if((m00==0.)&&(m11==0.))*teta=acos(-1.)/4.;
else *teta=0.5*atan2l(m00,m11);}


long double Fi(int p,int q,double t,long double *m){
long double k,M[4][4];register int r,s,uu,vv;
double u,v,a,b;
M[0][0]=*m;M[0][1]=*(m+1);
M[1][0]=*(m+2);M[2][0]=*(m+3);
M[1][1]=*(m+4);M[0][2]=*(m+5);
M[3][0]=*(m+6);M[2][1]=*(m+7);
M[1][2]=*(m+8);M[0][3]=*(m+9);k=0.0;
u=cos(t);v=sin(t);
for(r=0;r<p+1;r++)for(s=0;s<q+1;s++){
uu=p-r+s;vv=q-s+r;
if((u==0.)&&(uu==0))a=1.;
if((v==0.)&&(vv==0))b=1.;
if((u==0.)&&(uu<0))a=0.;
if((v==0.)&&(vv<0))b=0.;
if(!(u==0.&&uu<=0)) a=pow(u,uu);
if(!(v==0.&&vv<=0)) b=pow(v,vv);
k+=pow(-1.,q-s)*Combination(p,r)*Combination(q,s)*a*b*M[p-r+q-s][r+s];}
return(k);}


void NormalizedByMeansOfCentralMoments(
```

```
int M,int N,float *I,long double *m){
long double *fi,k;double t;register int i;
ByMeansOfCentralMoments(M,N,I,m,&t);
if((fi=(long double *)calloc(10,sizeof(long double)))==NULL)
{clrscr();textcolor(4);cprintf("Bellek yetersiz !!");getch();exit(0);}
k=Fi(0,0,t,m);
*(fi+0)=1.;
*(fi+1)=0.;
*(fi+2)=0.;
*(fi+3)=Fi(2,0,t,m)/pow(k,2.);
*(fi+4)=0.;
*(fi+5)=Fi(0,2,t,m)/pow(k,2.);
*(fi+6)=Fi(3,0,t,m)/pow(k,2.5);
*(fi+7)=Fi(2,1,t,m)/pow(k,2.5);
*(fi+8)=Fi(1,2,t,m)/pow(k,2.5);
*(fi+9)=Fi(0,3,t,m)/pow(k,2.5);
for(i=0;i<10;i++)*(m+i)=*(fi+i);free(fi);}


void ByMeansOfMomentInvariants(
int M,int N,float *I,long double *v){
long double *m;register int i;
if((m=(long double *) calloc(10,sizeof(long double)))==NULL)
{clrscr();textcolor(4);cprintf("Bellek yetersiz !!");getch();exit(0);}
NormalizedByMeansOfCentralMoments(M,N,I,m);
for(i=5;i<10;i++) *(v+i-3)=*(m+i);
*(v+1)=*(m+3);
*v=*m;free(m);}


long double NormalizedCentralMomentpq(
int M,int N,float *I,int p,int q){
float Gama;long double M00,Mpq;
Gama=(p+q)/2+1;M00=CentralMomentpq(M,N,I,0,0);
Mpq=CentralMomentpq(M,N,I,p,q);
return(Mpq/pow(M00,Gama));}


void MomentInvariants(
int M,int N,float *I,long double *v){
long double m11,m20,m02,m12,m21,m03,m30;
m11=NormalizedCentralMomentpq(M,N,I,1,1);
m02=NormalizedCentralMomentpq(M,N,I,0,2);
m20=NormalizedCentralMomentpq(M,N,I,2,0);
m12=NormalizedCentralMomentpq(M,N,I,1,2);
```

```
m21=NormalizedCentralMomentpq(M,N,I,2,1);
m30=NormalizedCentralMomentpq(M,N,I,3,0);
m03=NormalizedCentralMomentpq(M,N,I,0,3);
*(v)=m20+m02;*(v+1)=Sqr(m20-m02)+4*Sqr(m11);
*(v+2)=Sqr(m30-3*m12)+Sqr(m03-3*m21);*(v+3)=Sqr(m30+m12)+Sqr(m03+m21);
*(v+4)=(m30-3*m12)*(m30+m12)*(Sqr(m30+m12)-3*Sqr(m21+m03))+(m03-
3*m21)*(m03+m21)*(Sqr(m03+m21)-3*Sqr(m12+m30));
*(v+5)=(m20-m02)*(Sqr(m30+m12)-
Sqr(m21+m03))+4*m11*(m30+m12)*(m03+m21);
*(v+6)=(3*m21-m03)*(m30+m12)*(Sqr(m30+m12)-3*Sqr(m21+m03))-(m30-
3*m12)*(m21+m03)*(3*Sqr(m30+m12)-Sqr(m21+m03));}


void MomentsOfTheRawImage(long double *p,char *c){int j;float *f;
j=GetTheRawImageDim(c);
if((f=(float *) calloc(j*j,sizeof(float)))==NULL)
{clrscr();textcolor(4);cprintf("Bellek yetersiz !!");getch();exit(0);}
GetTheRawImage(f,c,j);
MomentInvariants(j,j,f,p);
if(*(p+6)<0.)*(p+6)*=-1.;
free(f);}


void ByMeansOfMomentsOfTheRawImage(long double *p,char *c){register int j;float
*f;
long double *p2;j=GetTheRawImageDim(c);
if((f=(float *) calloc(j*j,sizeof(float)))==NULL)
{clrscr();textcolor(4);cprintf("Bellek yetersiz !!");getch();exit(0);}
GetTheRawImage(f,c,j);
if((p2=(long double *) calloc(7,sizeof(long double)))==NULL)
{clrscr();textcolor(4);cprintf("Bellek yetersiz !!");getch();exit(0);}
ByMeansOfMomentInvariants(j,j,f,p2);free(f);
for(j=3;j<7;j++)
if(*(p2+j)<0.)*(p2+j)*=-1.;
for(j=0;j<6;j++)
*(p+j)=*(p2+j+1);
free(p2);}


void MomentInvariantsForGreyImages(long double *m,char *s)
{ByMeansOfMomentsOfTheRawImage(m,s);}


void MomentInvariantsForBinaryImages(long double *m,char *s)
{MomentsOfTheRawImage(m,s);}
```

```
void Gradient(int M,int N,float *p1,float *p2){
register int i,j;float x,y;
for(i=0;i<M;i++)for(j=0;j<N;j++){
if((i<M-1)&&(j<N-1)){
x=*(p1+i*N+j)-*(p1+i*N+j+1);
y=*(p1+i*N+j)-*(p1+(i+1)*N+j);
if(x<0.) x*=-1.;
if(y<0.) y*=-1.;
*(p2+i*N+j)=sqrt(x*x+y*y);}else{if(j>0)
x=*(p1+i*N+j)-*(p1+i*N+j-1);else
x=*(p1+i*N+j)-*(p1+i*N+j+1);if(i<M-1)
y=*(p1+i*N+j)-*(p1+(i+1)*N+j);else
y=*(p1+i*N+j)-*(p1+(i-1)*N+j);
if(x<0.) x*=-1.;if(y<0.) y*=-1.;
*(p2+i*N+j)=sqrt(x*x+y*y);}}
x=*p2;for(i=0;i<M;i++)for(j=0;j<N;j++)
if(x>*(p2+i*N+j))x=*(p2+i*N+j);
for(i=0;i<M*N;i++)*(p2+i)-=x;
x=*p2;for(i=0;i<M;i++)for(j=0;j<N;j++)
if(x<*(p2+i*N+j))x=*(p2+i*N+j);
for(i=0;i<M*N;i++)*(p2+i)*=(255./x);}


void RobertsEdgeDetection(int M,int N,float *p1,float *p2){
register int i,j;float a1,a2,a3,a4;
for(i=0;i<M;i++)for(j=0;j<N;j++){*(p2+i*N+j)=0.;
a1=*(p1+i*N+j);a2=*(p1+i*N+j+1);
a3=*(p1+(i+1)*N+j);a4=*(p1+(i+1)*N+j+1);
if(i==M-1) {a3=sqrt(a2*a1);a4=sqrt(a1*a2);}
if(j==N-1) {a2=sqrt(a3*a1);a4=sqrt(a1*a3);}
if(i==M-1&&j==N-1){a2=sqrt(a1*a1);
a3=sqrt(a1*a1);a4=sqrt(a1*a1);}
*(p2+i*N+j)=sqrt(Sqr((double)(sqrt(a1)-sqrt(a4)))+Sqr((double)(sqrt(a2)-sqrt(a3))));}
a1=*p2;for(i=0;i<M;i++)for(j=0;j<N;j++)if(a1>*(p2+i*N+j))a1=*(p2+i*N+j);
for(i=0;i<M*N;i++)*(p2+i)-=a1;
a1=*p2;for(i=0;i<M;i++)for(j=0;j<N;j++)if(a1<*(p2+i*N+j))a1=*(p2+i*N+j);
for(i=0;i<M*N;i++)*(p2+i)*=(255./a1);}


void TresholdingToEdgeExtractedImage(int M,int N,float *p1,float t){
int i;for(i=0;i<M*N;i++)if(*(p1+i)<t) *(p1+i)=0;else *(p1+i)=255;}
```

```
void Silluet(int M,int N,float *p1){
int i;for(i=0;i<M*N;i++)if(*(p1+i)<2.)*(p1+i)=0.;else *(p1+i)=255;}



//                                              MOMEXT.CPP

#include <mom_ext.inc>


void Mom_Extractor(int n,char *c,int k,float t){

int i,j;FILE *fse,*fhe,*fs,*fh,*fss,*fhs;float *p,*p2;
long double *m;char *ct,*ct2;

clrscr();

if((ct=(char *)calloc(100,sizeof(char)))==NULL) NoMemRem();
if((ct2=(char *)calloc(20,sizeof(char)))==NULL) NoMemRem();
if((m=(long double *)calloc(7,sizeof(long double)))==NULL)
NoMemRem();

strcpy(ct,c);strcat(ct,"gmv.wri");
if((fs=fopen(ct,"wb"))==NULL) CantOpFile();
strcpy(ct,c);strcat(ct,"bmv.wri");
if((fh=fopen(ct,"wb"))==NULL) CantOpFile();
strcpy(ct,c);strcat(ct,"gmv_e.wri");
if((fse=fopen(ct,"wb"))==NULL) CantOpFile();
strcpy(ct,c);strcat(ct,"bmv_e.wri");
if((fhe=fopen(ct,"wb"))==NULL) CantOpFile();
strcpy(ct,c);strcat(ct,"gmv_s.wri");
if((fss=fopen(ct,"wb"))==NULL) CantOpFile();
strcpy(ct,c);strcat(ct,"bmv_s.wri");
if((fhs=fopen(ct,"wb"))==NULL) CantOpFile();

Grey_Grph();

for(i=1;i<n+1;i++){

strcpy(ct,c);
itoa(i,ct2,10);
strcat(ct2,".raw");
strcat(ct,ct2);
```

```
gotoxy(1,1);printf("%s",ct);

j=GetTheRawImageDim(ct);

if((p=(float *)calloc(j*j,sizeof(float)))==NULL) {SetMode(3);
NoMemRem();}
if((p2=(float *)calloc(j*j,sizeof(float)))==NULL) {SetMode(3);
NoMemRem();}

GetTheRawImage(p,ct,j);
PutImagexy(j,j,p,1,50);

if(k=1)
RobertsEdgeDetection(j,j,p,p2);
else if(k=2)
Gradient(j,j,p,p2);

Silluet(j,j,p);
PutImagexy(j,j,p,j+1,50);
PutTheRawImage(j,j,p,"c:\\tmp2.tmp");free(p);
TresholdingToEdgeExtractedImage(j,j,p2,t);
PutImagexy(j,j,p2,j*2+1,50);
PutTheRawImage(j,j,p2,"c:\\tmp.tmp");free(p2);

MomentInvariantsForGreyImages(m,ct);
for(j=0;j<6;j++) fprintf(fs,"%Le\n",*(m+j));
MomentInvariantsForBinaryImages(m,ct);
for(j=0;j<7;j++) fprintf(fh,"%Le\n",*(m+j));
MomentInvariantsForGreyImages(m,"c:\\tmp.tmp");
for(j=0;j<6;j++) fprintf(fse,"%Le\n",*(m+j));
MomentInvariantsForBinaryImages(m,"c:\\tmp.tmp");
for(j=0;j<7;j++) fprintf(fhe,"%Le\n",*(m+j));
MomentInvariantsForGreyImages(m,"c:\\tmp2.tmp");
for(j=0;j<6;j++) fprintf(fss,"%Le\n",*(m+j));
MomentInvariantsForBinaryImages(m,"c:\\tmp2.tmp");
for(j=0;j<7;j++) fprintf(fhs,"%Le\n",*(m+j));

if(kbhit()) exit(0);}

SetMode(3);
free(m);free(ct2);free(ct);
fclose(fh);fclose(fs);
fclose(fhe);fclose(fse);
fclose(fhs);fclose(fss);
system("del c:\\*.tmp");}
```

```
void main(void){char *c;

int nt=10;

if((c=(char *)calloc(100,sizeof(char)))==NULL) NoMemRem();


// koniler

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic1\\");
Mom_Extractor(89,c,1,90.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic2\\");
Mom_Extractor(89,c,1,90.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic3\\");
Mom_Extractor(89,c,1,90.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic4\\");
Mom_Extractor(89,c,1,90.);


// küpler

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cube\\cube1\\");
Mom_Extractor(64,c,1,1.);


// silindirler

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin1\\");
Mom_Extractor(89,c,1,16.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin2\\");
Mom_Extractor(89,c,1,18.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin3\\");
Mom_Extractor(89,c,1,18.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin4\\");
Mom_Extractor(89,c,1,17.);
```

```
// Uçaklar

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\f16\\f16_1\\");
Mom_Extractor(700,c,1,18.);


strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\f16\\f16_2\\");
Mom_Extractor(700,c,1,19.);


// Füzeler

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\tc300\\tc300_1\\");
Mom_Extractor(360,c,1,22.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\tc300\\tc300_2\\");
Mom_Extractor(360,c,1,30.);


// test konileri

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic1\\test\\");
Mom_Extractor(nt,c,1,50.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic2\\test\\");
Mom_Extractor(nt,c,1,90.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic3\\test\\");
Mom_Extractor(nt,c,1,90.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\conic\\conic4\\test\\");
Mom_Extractor(nt,c,1,90.);


// test küpleri

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cube\\cube1\\test\\");
Mom_Extractor(nt,c,1,50.);


// test silindirleri

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin1\\test\\");
Mom_Extractor(nt,c,1,18.);
```

```
strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin2\\test\\");
Mom_Extractor(nt,c,1,18.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin3\\test\\");
Mom_Extractor(nt,c,1,18.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\cylin\\cylin4\\test\\");
Mom_Extractor(nt,c,1,17.);


// test Uçakları

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\f16\\f16_1\\test\\");
Mom_Extractor(nt,c,1,40.);


strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\f16\\f16_2\\test\\");
Mom_Extractor(nt,c,1,19.);


// test Füzeleri

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\tc300\\tc300_1\\test\\");
Mom_Extractor(nt,c,1,30.);

strcpy(c,"c:\\bc\\bin\\neural\\images\\synthtc\\tc300\\tc300_2\\test\\");
Mom_Extractor(nt,c,1,30.);

free(c);}



//                              NEURAL.İNC


//      OBJREC.CPP programının kullandığı kaynak dosyası


typedef enum {false, true} Boolean;
typedef unsigned char     Byte;
typedef unsigned int      Word;
```

```
int Round(float x){return((int)(x+0.5));}


union  REGS reg;
struct SREGS inreg;


void SetMode(int Mode)
{
 reg.h.ah=0;
 reg.h.al=Mode;
 int86(0x10,&reg,&reg);
                        }


#define MaxXres 320
#define MaxYres 200
#define MaxX (MaxXres-1)
#define MaxY (MaxYres-1)


int  XRes, YRes;
Word PreCalcY[MaxY+1];


void PreCalc()
{Word j;
 for(j=0; j<=MaxY; j++)
 PreCalcY[j]=0;
 for(j=0; j<=MaxY; j++)
 PreCalcY[j]=XRes*j;}


void Plot(int x, int y, Byte color)

{ Word Offset;
   char far *address;
   if(!((x<0) || (y<0) || (x>MaxX) || (y>MaxY)))
        {
   Offset = PreCalcY[y] + x;
   address = (char far *)  (0xA0000000L + Offset);
   *address = color;
                }
        }
```

```
typedef struct
{
 Byte Red;
 Byte Grn;
 Byte Blu;}RGB;


typedef RGB PaletteRegister[255];


PaletteRegister Color;


void ClearPalette(PaletteRegister Color){
Word i;
 for(i=0; i<=255; i++)
 {
  Color[i].Red=0;
  Color[i].Grn=0;
  Color[i].Blu=0;}}


void SetPalette(PaletteRegister Hue){
 reg.x.ax=0x1012;
 segread(&inreg);
 inreg.es=inreg.ds;
 reg.x.bx=0;
 reg.x.cx=256;
 reg.x.dx=(int)&Hue[0];
 int86x(0x10,&reg,&reg,&inreg);}


void InitPalette(PaletteRegister Color,int Pal){
 Word i,j;

if(Pal==1){
 for(i=0; i<64; i++){
  Color[i].Red=i;
  Color[i].Grn=i;
  Color[i].Blu=i;}
 for(i=64; i<128; i++){
  Color[i].Red=i-64;
  Color[i].Grn=0;
  Color[i].Blu=0;}
```

```
  for(i=128; i<192; i++){
    Color[i].Red=0;
    Color[i].Grn=i-128;
    Color[i].Blu=0;}
  for(i=192; i<=255; i++){
    Color[i].Red=0;
    Color[i].Grn=0;
    Color[i].Blu=i-192;}}
if(Pal==2){
  for(i=0; i<36; i++){
    Color[i].Red=0;
    Color[i].Grn=0;
    Color[i].Blu=Round(1.8*i);}
  for(i=36; i<72; i++){
    Color[i].Red=0;
    Color[i].Grn=Round(1.8*(i-36));
    Color[i].Blu=0;}
  for(i=72; i<108; i++){
    Color[i].Red=0;
    Color[i].Grn=Round(1.8*(i-72));
    Color[i].Blu=Round(1.8*(i-72));}
  for(i=108; i<144; i++){
    Color[i].Red=Round(1.8*(i-108));
    Color[i].Grn=0;
    Color[i].Blu=0;}
  for(i=144; i<180; i++){
    Color[i].Red=Round(1.8*(i-144));
    Color[i].Grn=0;
    Color[i].Blu=Round(1.8*(i-144));}
  for(i=180; i<216; i++){
    Color[i].Red=Round(1.8*(i-180));
    Color[i].Grn=Round(1.8*(i-180));
    Color[i].Blu=0;}
  for(i=216; i<252; i++){
    Color[i].Red=Round(1.8*(i-216));
    Color[i].Grn=Round(1.8*(i-216));
    Color[i].Blu=Round(1.8*(i-216));}}
if(Pal==3){
  for(i=0; i<64; i++){
        for(j=0; j<4; j++)  {
        Color[i*4+j].Red=i;
        Color[i*4+j].Grn=i;
        Color[i*4+j].Blu=i; }}}}
```

```
void Swap(int *first, int *second){ int temp;
 temp=*first;*first=*second;*second=temp;}


void InitGraphics(int Pal){
 XRes=MaxXres;YRes=MaxYres;
 PreCalc();SetMode(19);ClearPalette(Color);
 InitPalette(Color,Pal);SetPalette(Color);}


void WaitForKey(){while(!(getch()));}


void ExitGraphics()
{sound(1000);delay(500);nosound();WaitForKey();SetMode(3);}


void Draw(int xx1, int yy1, int xx2, int yy2, Byte color){
 int LgDelta, ShDelta, Cycle, LgStep, ShStep, dtotal;
 LgDelta=xx2-xx1;
 ShDelta=yy2-yy1;
 if(LgDelta<0){
  LgDelta=-LgDelta;
  LgStep=-1;}
 else
  LgStep=1;
 if(ShDelta<0){
  ShDelta=-ShDelta;
  ShStep=-1;}
 else
  ShStep=1;
 if(ShDelta<LgDelta){
  Cycle=LgDelta >> 1;
  while(xx1 != xx2){
   Plot(xx1, yy1, color);
   Cycle+=ShDelta;
   if(Cycle>LgDelta){
       Cycle-=LgDelta;
       yy1+=ShStep;}
   xx1+=LgStep;}
  Plot(xx1, yy1, color);}
 else{
  Cycle=ShDelta >> 1;
  Swap(&LgDelta, &ShDelta);
  Swap(&LgStep, &ShStep);
```

```
  while(yy1 != yy2){
   Plot(xx1, yy1, color);
   Cycle+=ShDelta;
   if(Cycle>LgDelta){
       Cycle-=LgDelta;
       xx1+=ShStep;}
   yy1+=LgStep;}
  Plot(xx1, yy1, color);}}


void Grey_Grph(void){InitGraphics(3);}


void Exit_Grph(void){ExitGraphics();}


double sqr(double x){return(x*x);}


int NearestFloor(double f){
return((int)(f+.5));}


void NoMemRem(void){
SetMode(3);textcolor(12);
cprintf("HEAP_BELLEK BİTTİ, PROGRAMDAN ÇIKILACAK. !");
getch();exit(0);}


void CantOpFile(void){
SetMode(3);textcolor(11);
cprintf("DOSYA AÇILAMADI, PROGRAMDAN ÇIKILACAK. !");
getch();exit(0);}


int GetTheRawImageDim(char *strng){int i,j,d,kr=0;FILE *f;
if((f=fopen(strng,"rb"))==NULL)
{clrscr();textcolor(4);printf("%s dosyası açılamadı",strng);
getch();exit(0);}
while(getc(f)!=EOF) kr++;fclose(f);d=sqrt(kr);return(d);}


void GetTheRawImage(float *p,char *strng,int d){int i,j;FILE *f;
if((f=fopen(strng,"rb"))==NULL)
{clrscr();textcolor(4);printf("%s dosyası açılamadı",strng);
```

```
getch();exit(0);}
for(i=0;i<d;i++)for(j=0;j<d;j++)*(p+i*d+j)=(float)getc(f);fclose(f);}


void PutImagexy(int M,int N,float *p,int x,int y){
register int i,j;
if(x>319||y>199||x<0||y<0)
{printf("koordinatlar yanlış");getch();exit(0);}
for(i=0;i<M;i++)for(j=0;j<N;j++)
Plot(j+x,i+y,*(p+i*N+j));}


void PutTheRawImage(int M,int N,float *p, char *strng)
{int i,j;FILE *f;
if((f=fopen(strng,"wb"))==NULL)
{clrscr();textcolor(4);printf("%s dosyası açılamadı",strng);
getch();exit(0);}
for(i=0;i<M;i++)for(j=0;j<N;j++)putc(*(p+i*N+j),f);fclose(f);}


float Extract(long double l){
long double a,b;a=l;
if(a<0.) l*=-1.;
if(a!=0.)b=logl(l);else b=0.;
if(a<0.) b*=-1.;
return((float)(b));}



//                                                    OBJREC.CPP

#include <stdio.h>
#include <dos.h>
#include <alloc.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <neural.inc>


#define scyc(i,Limit) for(i=0;i<Limit;i++)
#define MAX 7


typedef struct nm{long double W[MAX];
```

```
long double Y;long double V;long double Delta;
long double Error;long double Treshold;}NeuronModel;


typedef struct lm{NeuronModel N[MAX];}LayerModel;


typedef struct nt{LayerModel L[MAX];}NetworkModel;


long double vm(int d,long double *p1,long double *p2){
long double t=0.0;
register int i;scyc(i,d) t+=*(p1+i) * *(p2+i);return(t);}


void vg(int x,int d,float far *p1,long double *p2){
register int i;scyc(i,d)*(p2+i)=*(p1+x*d+i);}


int khit(void){union REGS i,o;
i.h.ah=1;int86(0x16,&i,&o);
if((o.x.flags&0x40))return(0);else return(1);}


int GetScanCode(void){union REGS i,o;
i.h.ah=0;int86(0x16,&i,&o);return(o.h.ah);}


double ANumber(void){double a,b;
a=log10(exp(sin(rand())-cos(rand())));
b=exp(sin(rand()))/exp(cos(rand()))-exp(cos(rand()))/exp(sin(rand()));
return(sin(a/b)*cos(b/a)*(sin(b)*exp(cos(a))-sin(a)*exp(cos(b))));}


void Hot_Keys(int i,int j){int l=1;
gotoxy(i+10,j);printf("HOT_KEYS:");
gotoxy(i,j+l*2);printf("Esc       :Menu");
gotoxy(i,j+l*3);printf("Ctrl_Q    :Exit from learning");
gotoxy(i,j+l*4);printf("1         :Text display");
gotoxy(i,j+l*5);printf("2         :Graphic display");
gotoxy(i,j+l*6);printf("Up_Arrow   :Increase the momentum parameter");
gotoxy(i,j+l*7);printf("Down_Arrow :Decrease the momentum parameter");
gotoxy(i,j+l*8);printf("Right_Arrow:Increase the learning rate parameter");
gotoxy(i,j+l*9);printf("Left_Arrow :Decrease the learning rate parameter");}
```

```
void Save_Report(NetworkModel NM,int NumberOfLayers,
int *NumberOfNeuronOfLayers,long double Momentum,
long double LearningRate,long double b,
unsigned long int ite,char *c){
int i,j,l;FILE *f1;
if((f1=fopen(c,"wt"))==NULL) CantOpFile();
fprintf(f1,"*** EGITILMIS SINIR AGININ YAPISAL DATALARI ***\n\n");
fprintf(f1,"%d katmanli ve Momentum=%Lf , Learning Rate=%Lf degerlerine
sahip\n",NumberOfLayers,Momentum,LearningRate);
fprintf(f1,"yapay sinir agi %%%Lf egitilmistir;\n",b);
scyc(i,NumberOfLayers){
fprintf(f1,"\n%d'inci katmaninda %d noron
vardir.\n",i+1,*(NumberOfNeuronOfLayers+i));}
fprintf(f1,"\nYapay sinir agina ait bu Weight degerleri %lu iterasyonda
hesaplanmistir.\n",ite);
fprintf(f1,"\nNOT:1'inci katman giris katmani oldugundan bu katmandaki noronlarin
weight degerleri yoktur.\n");
scyc(l,NumberOfLayers)if(l!=0){
fprintf(f1,"\n\n\nKatman No=%d:\n\n",l+1);
scyc(i,*(NumberOfNeuronOfLayers+l)){
fprintf(f1,"#%d'inci norona ait Weight vektoru:\n[",i+1);
scyc(j,*(NumberOfNeuronOfLayers+l-1))
fprintf(f1,"%Lf ",NM.L[l].N[i].W[j]);
fprintf(f1,"] Treshold=%Lf\n",NM.L[l].N[i].Treshold);}}fclose(f1); }


void Save(NetworkModel NM,int NumberOfLayers,
int *NumberOfNeuronOfLayers,long double Momentum,
long double LearningRate,long double b,
long double uy,unsigned long int ite,char *c){
int i,j,l;FILE *f1;
if((f1=fopen(c,"wt"))==NULL) CantOpFile();
fprintf(f1,"%lu,",ite);fprintf(f1,"%Lf,",b);
fprintf(f1,"%Lf,",uy);fprintf(f1,"%d,",NumberOfLayers);
fprintf(f1,"%Lf,",Momentum);fprintf(f1,"%Lf,",LearningRate);
scyc(i,NumberOfLayers)fprintf(f1,"%d,",*(NumberOfNeuronOfLayers+i));
scyc(l,NumberOfLayers)if(l!=0)scyc(i,*(NumberOfNeuronOfLayers+l))
scyc(j,*(NumberOfNeuronOfLayers+l-1))fprintf(f1,"%Lf,",NM.L[l].N[i].W[j]);
scyc(l,NumberOfLayers)if(l!=0)scyc(i,*(NumberOfNeuronOfLayers+l))
fprintf(f1,"%Lf,%Lf,%Lf,%Lf,%Lf,",NM.L[l].N[i].Treshold,
NM.L[l].N[i].Delta,NM.L[l].N[i].Y,NM.L[l].N[i].V,NM.L[l].N[i].Error);
fclose(f1);}
```

```
NetworkModel Load(
NetworkModel NM,int *NumberOfLayers,int *NumberOfNeuronOfLayers,
long double *Momentum,long double *LearningRate,
long double *b,long double *uy,unsigned long int *ite,char *c){
int i,j,l;FILE *f1;
if((f1=fopen(c,"rt"))==NULL) CantOpFile();
fscanf(f1,"%lu,",ite);fscanf(f1,"%Lf,",b);
fscanf(f1,"%Lf,",uy);fscanf(f1,"%d,",NumberOfLayers);
fscanf(f1,"%Lf,",Momentum);fscanf(f1,"%Lf,",LearningRate);
scyc(i,*NumberOfLayers)fscanf(f1,"%d,",(NumberOfNeuronOfLayers+i));
scyc(l,*NumberOfLayers)if(l!=0)scyc(i,*(NumberOfNeuronOfLayers+l))
scyc(j,*(NumberOfNeuronOfLayers+l-1))fscanf(f1,"%Lf,",&NM.L[l].N[i].W[j]);
scyc(l,*NumberOfLayers)if(l!=0)scyc(i,*(NumberOfNeuronOfLayers+l))
fscanf(f1,"%Lf,%Lf,%Lf,%Lf,%Lf,",&NM.L[l].N[i].Treshold,
&NM.L[l].N[i].Delta,&NM.L[l].N[i].Y,&NM.L[l].N[i].V,&NM.L[l].N[i].Error);
fclose(f1);return(NM);}


void LRowVecPut(int N,int n,float far *p,long double *v){
int i;for(i=0;i<N;i++) *(v+i)=*(p+n*N+i);}


void LRowVec(int N,int n,float far *p,long double *v){
int i;for(i=0;i<N;i++) *(p+n*N+i)=*(v+i);}


void Silluet(int M,int N,float *p1){
register int i;for(i=0;i<M*N;i++)
if(*(p1+i)<2.)*(p1+i)=0.;else *(p1+i)=255;}


void RobEdDet(int M,int N,float *p1){
register int i,j;float t,a1,a2,a3,a4;FILE *f;
if((f=tmpfile())==NULL) CantOpFile();
for(i=0;i<M;i++)for(j=0;j<N;j++){t=0.;
a1=*(p1+i*N+j);a2=*(p1+i*N+j+1);a3=*(p1+(i+1)*N+j);a4=*(p1+(i+1)*N+j+1);
if(i==M-1) {a3=sqrt(a2*a1);a4=sqrt(a1*a2);}
if(j==N-1) {a2=sqrt(a3*a1);a4=sqrt(a1*a3);}
if(i==M-1&&j==N-1){a2=sqrt(a1*a1);a3=sqrt(a1*a1);a4=sqrt(a1*a1);}
t=sqrt(sqr((double)(sqrt(a1)-sqrt(a4)))+sqr((double)(sqrt(a2)-sqrt(a3))));
fprintf(f,"%f,",t);}fseek(f,0,0);
for(i=0;i<M*N;i++) fscanf(f,"%f,",(p1+i));fclose(f);
a1=*p1;for(i=0;i<M*N;i++)if(a1>*(p1+i))a1=*(p1+i);
for(i=0;i<M*N;i++)*(p1+i)-=a1;
a1=*p1;for(i=0;i<M*N;i++)if(a1<*(p1+i))a1=*(p1+i);
```

```
for(i=0;i<M*N;i++)*(p1+i)*=(255./a1);}


void GroupDisplayer(int k,int n,char *cs){
register int i;int s,ss;float *p;char *c,*ct;clrscr();
if((c=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
if((ct=(char *)calloc(8,sizeof(char)))==NULL) NoMemRem();
s=0;Grey_Grph();for(i=1;i<n+1;i++){
strcpy(c,cs);itoa(i,ct,10);strcat(ct,".raw");strcat(c,ct);
ss=GetTheRawImageDim(c);
if((p=(float *)calloc(ss*ss,sizeof(float)))==NULL)NoMemRem();
GetTheRawImage(p,c,ss);if(k==1)RobEdDet(ss,ss,p);
else if(k==2)Silluet(ss,ss,p);PutImagexy(ss,ss,p,115,45);
free(p);if(s==1)goto atla;s=GetScanCode();atla:
}free(c);free(ct);SetMode(3);}


void ObjectDisplayer(int k,int n,char *cs){
float *p;char *c,*ct;int ss;
if((ct=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
if((c=(char *)calloc(8,sizeof(char)))==NULL) NoMemRem();
strcpy(ct,cs);
itoa(n,c,10);strcat(c,".raw");strcat(ct,c);free(c);Grey_Grph();
ss=GetTheRawImageDim(ct);
if((p=(float *)calloc(ss*ss,sizeof(float)))==NULL)NoMemRem();
GetTheRawImage(p,ct,ss);free(ct);
if(k==1)RobEdDet(ss,ss,p);else if(k==2)Silluet(ss,ss,p);
PutImagexy(ss,ss,p,115,45);getch();free(p);SetMode(3);}



void main(void){
Byte t[MAX];
FILE *f1;
NetworkModel NM,NM_1,NM_2;
int NumberOfLayers,NumberOfInputs,DimesionOfInputs,uyy;
long double err,Momentum,LearningRate,des,tro,trc,a,b,uy;
int *NumberOfNeuronOfLayers,syc,syc2,tt[MAX],nt;
register int i,j,k,l,n;
long double *Transient1,*Transient2,*Transient3,*DesiredVector;
float far *InputVector;
unsigned short int far *DesiredValue;
unsigned short int a1,a2;
unsigned long int ite=0;
char kr,*na,*N1,*N2,*Nt,*nn[7],*nT[7];
```

```
float lam;nt=10;


LearningRate=0.00001;Momentum=0.95;


if((N1=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
if((N2=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
if((Nt=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
if((na=(char *)calloc(10,sizeof(char)))==NULL) NoMemRem();

SetMode(3);

syc2=0;syc=0;a1=0;

dn:

strcpy(N1,"c:\\bc\\bin\\neural\\images\\synthtc\\");

printf("Please, Select object groups for classification.\n\n");
printf("'1'     CONICS\n");
printf("'2'     CUBES\n");
printf("'3'     CYLINDERS\n");
printf("'4'     ROCKETS\n");
printf("'5'     AIRPLANES\n");
printf("'Ctrl_Q  SELECTION COMPLETED\n");
printf("'Ctrl_E  QUIT FROM THE PROGRAM\n\n\n");

scyc(i,syc) printf("choice %d : %s\n",i+1,nn[i]);

k=getch();


clrscr();
if(k==49) {
printf("Please selec a conic sub_group\n\n");
printf("'1'  CONICS1: this group is lighting front.\n");
printf("'2'  CONICS2: this group is lighting from right.\n");
printf("'3'  CONICS3: this group is lighting front and\n");
printf("             there is a 'C' significant on each object.\n");
printf("'4'  CONICS4: this group is lighting from left and\n");
printf("             there is a 'C' significant on each object.\n");
l=70;
kr=getch();
if(kr!='1'&&kr!='2'&&kr!='3'&&kr!='4') goto dn;else{
```

```
*N2=kr;*(N2+1)=0;strcpy(Nt,"conic\\conic");strcat(Nt,N2);}}

else
if(k==50) {
printf("Please selec a cube sub_group\n\n");
printf("'1'  CUBES1: this group is lighting front.\n");
l=64;
kr=getch();
if(kr!='1') goto dn;else{
*N2=kr;*(N2+1)=0;strcpy(Nt,"cube\\cube");strcat(Nt,N2);}}

else
if(k==51) {
printf("Please selec a cylinder sub_group\n\n");
printf("'1'  CYLINDERS1: this group is lighting front.\n");
printf("'2'  CYLINDERS2: this group is also lighting front but lighted much more than
1.\n");
printf("'3'  CYLINDERS3: this group is lighting from left.\n");
printf("'4'  CYLINDERS4: this group is lighting from right and top.\n");
l=70;
kr=getch();
if(kr!='1'&&kr!='2'&&kr!='3'&&kr!='4') goto dn;else{
*N2=kr;*(N2+1)=0;strcpy(Nt,"cylin\\cylin");strcat(Nt,N2);}}

else
if(k==52) {
printf("Please selec a rocket sub_group\n\n");
printf("'1'  ROCKETS1: this group is lighting front.\n");
printf("'2'  ROCKETS2: this group is lighting from right.\n");
l=360;
kr=getch();
if(kr!='1'&&kr!='2') goto dn;else{
*N2=kr;*(N2+1)=0;strcpy(Nt,"tc300\\tc300_");strcat(Nt,N2);}}

else
if(k==53) {
printf("Please selec a airplane sub_group\n\n");
printf("'1'  AIRPLANES1: this group is lighting front.\n");
printf("'2'  AIRPLANES2: this group is lighting from right_up and left_up corners.\n");
l=700;
kr=getch();
if(kr!='1'&&kr!='2') goto dn;else{
*N2=kr;*(N2+1)=0;strcpy(Nt,"f16\\f16_");strcat(Nt,N2);}}

else
```

```
if(k==17) goto fin;else
if(k==5) exit(0);else goto dn;

if(syc2==1) goto dn4;

dn2:
clrscr();
printf("What kind image will you use\n\n");
printf("'1'  GREY LEVEL IMAGES\n");
printf("'2'  EDGE EXTRACTED IMAGES\n");
printf("'3'  SILHOUETTE IMAGES\n");
kr=getch();
if(kr==49) {strcpy(N2,"mv.wri");a1=0;}else
if(kr==50) {strcpy(N2,"mv_e.wri");a1=1;}else
if(kr==51) {strcpy(N2,"mv_s.wri");a1=2;}else goto dn2;


dn3:
clrscr();
printf("Which kind moment vector will you use\n\n");
printf("'1'  STANDARTIZED MOMENTS FOR GREY LEVEL IMAGES\n");
printf("'2'  HU'S MOMENTS FOR BINARY IMAGES\n");
kr=getch();
if(kr=='1') {strcpy(na,"g");DimesionOfInputs=6;}else

if(kr=='2') {strcpy(na,"b");DimesionOfInputs=7;}else goto dn3;
strcat(na,N2);

dn4:

syc++;
if(syc==8){clrscr();
printf("Maximum object group number can be 7 because of memory allocation");
syc--;goto fin;}

if((nn[syc-1]=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
if((nT[syc-1]=(char *)calloc(80,sizeof(char)))==NULL) NoMemRem();
strcat(N1,Nt);strcat(N1,"\\");strcpy(nn[syc-1],N1);
strcpy(nT[syc-1],N1);strcat(nT[syc-1],"test\\");
GroupDisplayer(a1,l,N1);
strcat(nn[syc-1],na);tt[syc-1]=l;
if(syc2==0) syc2=1;
goto dn;
fin:
free(N1);free(N2);free(Nt);
```

```
NumberOfInputs=0;
scyc(i,syc) NumberOfInputs+=tt[i];
lam=-.9;


clrscr();textcolor(12);randomize();


if(syc==2&&NumberOfInputs<250) NumberOfLayers=4;else
if(syc==2&&NumberOfInputs<500) NumberOfLayers=5;else
if(syc==2&&NumberOfInputs<750) NumberOfLayers=6;else NumberOfLayers=7;


if((NumberOfNeuronOfLayers=(int *)calloc(NumberOfLayers,sizeof(int)))==NULL)
NoMemRem();


if(NumberOfLayers==7){
*NumberOfNeuronOfLayers=DimesionOfInputs;
*(NumberOfNeuronOfLayers+1)=NearestFloor((DimesionOfInputs*5.+syc)/6.);
*(NumberOfNeuronOfLayers+2)=NearestFloor((DimesionOfInputs*4.+syc*2.)/6.);
*(NumberOfNeuronOfLayers+3)=NearestFloor((DimesionOfInputs*3.+syc*3.)/6.);
*(NumberOfNeuronOfLayers+4)=NearestFloor((DimesionOfInputs*2.+syc*4.)/6.);
*(NumberOfNeuronOfLayers+5)=NearestFloor((DimesionOfInputs*1.+syc*5.)/6.);
*(NumberOfNeuronOfLayers+6)=syc;} else
if(NumberOfLayers==6){
*NumberOfNeuronOfLayers=DimesionOfInputs;
*(NumberOfNeuronOfLayers+1)=NearestFloor((DimesionOfInputs*4.+syc)/5.);
*(NumberOfNeuronOfLayers+2)=NearestFloor((DimesionOfInputs*3.+syc*2.)/5.);
*(NumberOfNeuronOfLayers+3)=NearestFloor((DimesionOfInputs*2.+syc*3.)/5.);
*(NumberOfNeuronOfLayers+4)=NearestFloor((DimesionOfInputs*1.+syc*4.)/5.);
*(NumberOfNeuronOfLayers+5)=syc;} else
if(NumberOfLayers==5){
*NumberOfNeuronOfLayers=DimesionOfInputs;
*(NumberOfNeuronOfLayers+1)=NearestFloor((DimesionOfInputs*3.+syc)/4.);
*(NumberOfNeuronOfLayers+2)=NearestFloor((DimesionOfInputs*2.+syc*2.)/4.);
*(NumberOfNeuronOfLayers+3)=NearestFloor((DimesionOfInputs*1.+syc*3.)/4.);
*(NumberOfNeuronOfLayers+4)=syc;} else
if(NumberOfLayers==4){
*NumberOfNeuronOfLayers=DimesionOfInputs;
*(NumberOfNeuronOfLayers+1)=NearestFloor((DimesionOfInputs*2.+syc)/3.);
*(NumberOfNeuronOfLayers+2)=NearestFloor((DimesionOfInputs*1.+syc*2.)/3.);
*(NumberOfNeuronOfLayers+3)=syc;}
```

```
if((InputVector=(float far
*)farcalloc(DimesionOfInputs*NumberOfInputs,sizeof(float)))==NULL)
NoMemRem();
if((DesiredValue=(unsigned short int far *)farcalloc(NumberOfInputs,sizeof(unsigned
short int)))==NULL)
NoMemRem();



scyc(i,syc){
if((f1=fopen(nn[i],"rb"))==NULL) CantOpFile();free(nn[i]);
if(i>0){syc2=0;scyc(j,i)syc2+=tt[j]*DimesionOfInputs;}else syc2=0;
if(i>0){n=0;scyc(j,i)n+=tt[j];}else n=0;
scyc(k,tt[i]){scyc(l,DimesionOfInputs){
fscanf(f1,"%Le\n",&tro);
*(InputVector+syc2+k*DimesionOfInputs+l)=Extract(tro);    }
*(DesiredValue+n+k)=(unsigned short int)i;}
fclose(f1);}



clrscr();Hot_Keys(21,5);
i=21;j=5;l=1;gotoxy(i,j+l*14);textcolor(15|128);
cprintf("PLEASE WAIT, MOMENT VECTORS OF IMAGES IS LOADING");



if((Transient1=(long double *) calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
if((Transient2=(long double *) calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();



i=a1;for(ite=0;ite<100000;ite++){
n=random(NumberOfInputs);a1=*(DesiredValue+n);
l=random(NumberOfInputs);a2=*(DesiredValue+l);
LRowVecPut(DimesionOfInputs,n,InputVector,Transient1);
LRowVecPut(DimesionOfInputs,l,InputVector,Transient2);
LRowVec(DimesionOfInputs,l,InputVector,Transient1);
LRowVec(DimesionOfInputs,n,InputVector,Transient2);
*(DesiredValue+n)=a2;*(DesiredValue+l)=a1;}a1=i;
```

```
n=MAX;for(i=0;i<n;i++)for(j=0;j<n;j++){
for(k=0;k<n;k++)NM_1.L[i].N[j].W[k]=0.;
NM.L[i].N[j].Y=0.;NM.L[i].N[j].V=0.;
NM.L[i].N[j].Error=0.;NM.L[i].N[j].Delta=0.;
NM_1.L[i].N[j].Y=0.;NM_1.L[i].N[j].V=0.;
NM_1.L[i].N[j].Error=0.;NM_1.L[i].N[j].Delta=0.;
NM_1.L[i].N[j].Treshold=0.;}



a=.5;
ite=0;trc=-1.;err=0.;scyc(i,NumberOfLayers)
scyc(j,*(NumberOfNeuronOfLayers+i)){if(i==0){l=DimesionOfInputs;}else{
l=*(NumberOfNeuronOfLayers+i-1);}des=(long double)(2.0/(l*1.));scyc(k,l){
atla1:trc*=-1;tro=a*trc*ANumber();if(tro<0.) b=-1.*tro;else b=tro;
if (b<des){err+=tro;ite++;}else goto atla1; }scyc(k,l){atla2:
trc*=-1;tro=a*trc*ANumber();if(tro<0.) b=-1.*tro;else b=tro;
if (b<des){err+=tro;ite++;}else goto atla2; }}b=err/(ite*1.);
trc=-1.;scyc(i,NumberOfLayers)scyc(j,*(NumberOfNeuronOfLayers+i)){
if(i==0){l=DimesionOfInputs;}else{l=*(NumberOfNeuronOfLayers+i-1);}
des=(long double)(2.0/(l*1.));scyc(k,l){atla3:
trc*=-1;tro=a*trc*ANumber();;if(tro<0.) err=-1.*tro;else err=tro;
if (err<des)NM.L[i].N[j].W[k]=tro-b;else goto atla3; }scyc(k,l){atla4:
trc*=-1;tro=a*trc*ANumber();;if(tro<0.) err=-1.*tro;else err=tro;
if (err<des)NM.L[i].N[j].Treshold=tro-b;else goto atla4;}}



if((f1=fopen("c:\\bc\\bin\\neural\\Conv.wri","wt"))==NULL) CantOpFile();
if((DesiredVector=(long double
*)calloc(*(NumberOfNeuronOfLayers+NumberOfLayers-1),sizeof(long
double)))==NULL)
NoMemRem();
uyy=0;ite=0;syc2=0;textcolor(2);clrscr();
gotoxy(1,1);printf("Momentum=%Lf\nLearning Rate=%Lf",Momentum,LearningRate);
i=0;
kr=0;


if((Transient3=(long double *) calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
```

```
while(kr!=17){ite++;err=0.0;


for(n=0;n<NumberOfInputs;n++)      {
NM_2=NM_1;NM_1=NM;


vg(n,DimesionOfInputs,InputVector,Transient1);


for(i=0;i<*(NumberOfNeuronOfLayers+NumberOfLayers-1);i++)
if(i==*(DesiredValue+n)) *(DesiredVector+i)=1.;else *(DesiredVector+i)=0.;



for(l=0;l<NumberOfLayers;l++){
for(j=0;j<*(NumberOfNeuronOfLayers+l);j++){
if(l==0){NM.L[l].N[j].Y=*(Transient1+j);}else{
for(i=0;i<*(NumberOfNeuronOfLayers+l-1);i++){
*(Transient2+i)=NM.L[l-1].N[i].Y;*(Transient3+i)=NM.L[l].N[j].W[i];}
NM.L[l].N[j].V=(vm(*(NumberOfNeuronOfLayers+l-1),Transient2,Transient3)-
NM.L[l].N[j].Treshold);
NM.L[l].N[j].Y=1./(1.+expl(lam*NM.L[l].N[j].V));
if(l==NumberOfLayers-1) {NM.L[l].N[j].Error=*(DesiredVector+j)-NM.L[l].N[j].Y;}}
                                                                    }
                                                }


for(l=NumberOfLayers-1;l>-1;l--){
for(j=0;j<*(NumberOfNeuronOfLayers+l);j++){
if(l==NumberOfLayers-1)
{NM.L[l].N[j].Delta=NM.L[l].N[j].Error*NM.L[l].N[j].Y*(1.-NM.L[l].N[j].Y);}
else{for(k=0;k<*(NumberOfNeuronOfLayers+l+1);k++){
*(Transient2+k)=NM.L[l+1].N[k].Delta;
*(Transient3+k)=NM.L[l+1].N[k].W[j]; }
NM.L[l].N[j].Delta=NM.L[l].N[j].Y*(1.-
NM.L[l].N[j].Y)*vm(*(NumberOfNeuronOfLayers+l+1),Transient2,Transient3);
        }
                                                }
                                        }


for(l=0;l<NumberOfLayers;l++){if(l!=0){
```

```
for(j=0;j<*(NumberOfNeuronOfLayers+l);j++){
for(i=0;i<*(NumberOfNeuronOfLayers+l-1);i++)
{*(Transient1+i)=Momentum*(NM.L[l].N[j].W[i]-NM_2.L[l].N[j].W[i]);
*(Transient2+i)=NM.L[l].N[j].W[i];
*(Transient3+i)=LearningRate*NM.L[l].N[j].Delta*NM.L[l-1].N[i].Y;}
tro=NM_2.L[l].N[j].Treshold;trc=NM.L[l].N[j].Treshold;
for(i=0;i<*(NumberOfNeuronOfLayers+l-1);i++)
{NM.L[l].N[j].W[i]=*(Transient1+i)+*(Transient2+i)+*(Transient3+i);}
NM.L[l].N[j].Treshold=trc+Momentum*(trc-tro)-LearningRate*NM.L[l].N[j].Delta;
                                                    }
        }                       }



b=0.;
for(i=0;i<*(NumberOfNeuronOfLayers+NumberOfLayers-1);i++){
a=*(DesiredVector+i)-NM.L[NumberOfLayers-1].N[i].Y;b+=a*a;}


err+=b;
                                                    }


tro=err/(2.*NumberOfInputs);



if(ite>1){b=100.*(1-tro/uy);if(b<1.)b=1.;if(syc2==0){
gotoxy(25,10);printf("NeuralNet learned %%%Lf",b);}
else{uyy++;if(uyy>319) {uyy=1;Grey_Grph();
for(i=0;i<320;i+=2){Plot(i,48,255);Plot(i,148,255);Plot(i,98,128);}}
Draw(uyy,147,uyy,(int)(148-b),50+(int)(b*2));}
fprintf(f1,"%Lf\n",b);}else uy=tro;



i=khit();
if(i){
kr=getch();
if(syc2==0&&kr==72){
Momentum+=0.01;gotoxy(10,1);printf("%Lf",Momentum);}
else
if(syc2==0&&kr==80){
Momentum-=0.01;gotoxy(10,1);printf("%Lf",Momentum);}
else
```

```
if(syc2==0&&kr==77){
LearningRate+=0.0005;gotoxy(15,2);printf("%Lf",LearningRate);}
else
if(syc2==0&&kr==75){
LearningRate-=0.0005;gotoxy(15,2);printf("%Lf",LearningRate);}
else
if(kr==49){
SetMode(3);kr=0;syc2=0;gotoxy(1,1);
printf("Momentum=%Lf\nLearning Rate=%Lf",Momentum,LearningRate);}
else
if(kr==50){
Grey_Grph();kr=0;syc2=1;uyy=0;
for(i=0;i<320;i+=2){Plot(i,48,255);Plot(i,148,255);Plot(i,98,128);}}
else
if(kr==27){
SetMode(3);i=23;j=4;l=1;
gotoxy(i+10,j-1);printf("MENU:");
gotoxy(i,j+l);printf("F1:  Save The NeuralNet");
gotoxy(i,j+l*2);printf("F2:  Load A NeuralNet");
gotoxy(i,j+l*3);printf("F3:  Enter A New 'Momentum' Value");
gotoxy(i,j+l*4);printf("F4:  Enter A New 'Learning Rate' Value");
gotoxy(i,j+l*5);printf("F5:  Exit from learning");
gotoxy(i,j+l*6);printf("F6:  Exit from Program");
Hot_Keys(i-2,j+9);
kr=GetScanCode();clrscr();

if(kr==59){

clrscr();gotoxy(5,7);
printf("Enter the file name (MAX 4 letter): c:\\bc\\bin\\neural\\");
if((N1=(char *) calloc(100,sizeof(char)))==NULL) NoMemRem();
if((N2=(char *) calloc(100,sizeof(char)))==NULL) NoMemRem();
strcpy(N1,"c:\\bc\\bin\\neural\\");scanf("%s",N2);strcat(N1,N2);free(N2);
Save(NM,NumberOfLayers,NumberOfNeuronOfLayers,Momentum,LearningRate,b,uy,it
e,N1);
strcat(N1,"$");
Save(NM_1,NumberOfLayers,NumberOfNeuronOfLayers,Momentum,LearningRate,b,u
y,ite,N1);
strcat(N1,"$");
Save(NM_2,NumberOfLayers,NumberOfNeuronOfLayers,Momentum,LearningRate,b,u
y,ite,N1);
free(N1);clrscr();
}else
if(kr==60){
clrscr();gotoxy(5,7);
```

```
printf("Enter the file name (MAX 4 letter): c:\\bc\\bin\\neural\\");
if((N1=(char *) calloc(100,sizeof(char)))==NULL) NoMemRem();
if((N2=(char *) calloc(100,sizeof(char)))==NULL) NoMemRem();
strcpy(N1,"c:\\bc\\bin\\neural\\");
scanf("%s",N2);strcat(N1,N2);free(N2);
trc=Momentum;tro=LearningRate;
NM=Load(NM,&NumberOfLayers,NumberOfNeuronOfLayers,&Momentum,&Learning
Rate,&b,&uy,&ite,N1);
strcat(N1,"$");
NM_1=Load(NM_1,&NumberOfLayers,NumberOfNeuronOfLayers,&Momentum,&Lea
rningRate,&b,&uy,&ite,N1);
strcat(N1,"$");
NM_2=Load(NM_2,&NumberOfLayers,NumberOfNeuronOfLayers,&Momentum,&Lea
rningRate,&b,&uy,&ite,N1);
free(N1);err=((1.-b/100.)*uy*2.*NumberOfInputs);clrscr();
Momentum=trc;LearningRate=tro;
}else
if(kr==61){
gotoxy(30,12);printf("Momentum=");scanf("%Lf",&Momentum);clrscr();}
else
if(kr==62){
gotoxy(30,12);printf("Learning Rate=");scanf("%Lf",&LearningRate);clrscr();}
else
if(kr==63) kr=17;
else
if(kr==64) exit(0);}
gotoxy(1,1);
printf("Momentum=%Lf\nLearning Rate=%Lf",Momentum,LearningRate);

}

}fclose(f1);SetMode(3);free(DesiredVector);


textbackground(0);textcolor(10);


ite*=NumberOfInputs;


strcpy(N1,"c:\\bc\\bin\\neural\\Net_Dat.wri");
Save_Report(NM,NumberOfLayers,NumberOfNeuronOfLayers,Momentum,LearningRa
te,b,ite,N1);
```

```
clrscr();ite=0;

scyc(i,MAX) {tt[i]=0;t[i]=0;}
scyc(n,NumberOfInputs)                                    {

vg(n,DimesionOfInputs,InputVector,Transient1);


scyc(l,NumberOfLayers){
scyc(j,*(NumberOfNeuronOfLayers+l)){
if(l==0){NM.L[l].N[j].Y=*(Transient1+j);}else{
scyc(i,*(NumberOfNeuronOfLayers+l-1)){
*(Transient2+i)=NM.L[l-1].N[i].Y;
*(Transient3+i)=NM.L[l].N[j].W[i];}
NM.L[l].N[j].V=(-1.*NM.L[l].N[j].Treshold+vm(*(NumberOfNeuronOfLayers+l-
1),Transient2,Transient3));
NM.L[l].N[j].Y=1./(1.+expl(lam*NM.L[l].N[j].V));
                                                    }
                                          }
                                }


syc=1;
scyc(i,*(NumberOfNeuronOfLayers+NumberOfLayers-1))
if(NM.L[NumberOfLayers-1].N[*(DesiredValue+n)].Y<NM.L[NumberOfLayers-
1].N[i].Y)
syc=0;if(syc==1){ite++;
tt[*(DesiredValue+n)]++;
}else {t[*(DesiredValue+n)]++;}


        }


farfree(InputVector);farfree(DesiredValue);
free(Transient1);free(Transient2);free(Transient3);



tro=0.00000001;
printf("\nNUMBER OF TRUE CLASSIFIED OBJECTS=%lu   NUMBER OF FALSE
CLASSIFIED OBJECTS=%lu\n",ite,NumberOfInputs-ite);
printf("\nLEARNING WAS COMPLETED WITH %4.1f%%
SUCCESS.\n",(ite*100.)/(NumberOfInputs*1.));
scyc(i,*(NumberOfNeuronOfLayers+NumberOfLayers-1)){
```

```
printf("\n%4.1f%% OF CLASSIFIED OBJECTS WAS BELONG TO OBJECT
%d.",(float)(tt[i]*100./(ite*1.+0.00001)),i+1);
printf("\n%4.1f%% OF FALSE CLASSIFIED OBJECTS WAS BELONG TO OBJECT
%d.",(float)(t[i]*100./((NumberOfInputs-ite)*1.+0.00001)),i+1);
printf("\nOBJECT %d WAS CLASSIFIED WITH %4.1f%%
SUCCESS.\n",i+1,(tt[i]*100.)/((tt[i]+t[i])*1.));
                                    }getch();



if((Transient1=(long double *)calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
if((Transient2=(long double *)calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
if((Transient3=(long double *)calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
if((InputVector=(float far *)farcalloc(*(NumberOfNeuronOfLayers+NumberOfLayers-
1)*DimesionOfInputs*nt,sizeof(float)))==NULL)
NoMemRem();
if((DesiredValue=(unsigned short int far
*)farcalloc(*(NumberOfNeuronOfLayers+NumberOfLayers-1)*nt,sizeof(unsigned short
int)))==NULL)
NoMemRem();



if((Nt=(char *)calloc(*(NumberOfNeuronOfLayers+NumberOfLayers-
1)*nt,sizeof(char)))==NULL)
NoMemRem();
scyc(i,*(NumberOfNeuronOfLayers+NumberOfLayers-1)){
if((nn[i]=(char *)calloc(80,sizeof(char)))==NULL)NoMemRem();
strcpy(nn[i],nT[i]);strcat(nT[i],na);
if((f1=fopen(nT[i],"rb"))==NULL)CantOpFile();free(nT[i]);
scyc(j,nt){scyc(k,DimesionOfInputs){fscanf(f1,"%Le\n",&tro);
*(InputVector+j*DimesionOfInputs+k+nt*i*DimesionOfInputs)=Extract(tro);}
*(DesiredValue+j+i*nt)=(unsigned short int)i;
*(Nt+j+i*nt)=j;}fclose(f1);}free(na);



j=a1;
i=*(NumberOfNeuronOfLayers+NumberOfLayers-1)*nt;
for(ite=0;ite<i*100;ite++){
n=random(i);a1=*(DesiredValue+n);
l=random(i);a2=*(DesiredValue+l);
```

```
kr=*(Nt+n);*(Nt+n)=*(Nt+l);*(Nt+l)=kr;
LRowVecPut(DimesionOfInputs,n,InputVector,Transient1);
LRowVecPut(DimesionOfInputs,l,InputVector,Transient2);
LRowVec(DimesionOfInputs,l,InputVector,Transient1);
LRowVec(DimesionOfInputs,n,InputVector,Transient2);
*(DesiredValue+n)=a2;*(DesiredValue+l)=a1;}a1=j;



textcolor(14);clrscr();ite=0;syc2=0;scyc(i,MAX){tt[i]=0;t[i]=0;}
scyc(n,*(NumberOfNeuronOfLayers+NumberOfLayers-1)*nt)      {
vg(n,DimesionOfInputs,InputVector,Transient1);
scyc(l,NumberOfLayers){
scyc(j,*(NumberOfNeuronOfLayers+l)){
if(l==0){NM.L[l].N[j].Y=*(Transient1+j);}else{
scyc(i,*(NumberOfNeuronOfLayers+l-1)){
*(Transient2+i)=NM.L[l-1].N[i].Y;*(Transient3+i)=NM.L[l].N[j].W[i];}
NM.L[l].N[j].V=(-1.*NM.L[l].N[j].Treshold+vm(*(NumberOfNeuronOfLayers+l-
1),Transient2,Transient3));
NM.L[l].N[j].Y=1./(1.+expl(lam*NM.L[l].N[j].V));
                                        }
                        }
                }
clrscr();gotoxy(34,11);printf("TEST OBJECT %d.",n+1);getch();
free(Transient1);free(Transient2);free(Transient3);
ObjectDisplayer(a1,*(Nt+n)+1,nn[*(DesiredValue+n)]);
if((Transient1=(long double *)calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
if((Transient2=(long double *)calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();
if((Transient3=(long double *)calloc(MAX,sizeof(long double)))==NULL)
NoMemRem();

printf("\noutput for test object belonging to class %d: ",1+*(DesiredValue+n));
syc=1;
scyc(i,*(NumberOfNeuronOfLayers+NumberOfLayers-1)){
printf("%f ",(float)NM.L[NumberOfLayers-1].N[i].Y);
if(NM.L[NumberOfLayers-1].N[*(DesiredValue+n)].Y<NM.L[NumberOfLayers-
1].N[i].Y)
syc=0;}
if(syc==1){
printf(" TRUE");ite++;
tt[*(DesiredValue+n)]++;}
else{
t[*(DesiredValue+n)]++;
```

```
textcolor(14|128);
cprintf(" FALSE");
textcolor(14);}
getch();}


clrscr();textcolor(15|128);
cprintf("\r\n\r\n          ***    REPORT:    ***");
textcolor(15);
cprintf("\r\nNUMBER OF TRUE RECOGNIZED OBJECTS=%lu   NUMBER OF
FALSE RECOGNIZED
OBJECTS=%lu\r\n",ite,(*(NumberOfNeuronOfLayers+NumberOfLayers-1)*nt-ite));
cprintf("\r\nNEURAL NETWORK WAS SUCCEED IN RECOGNATING PROCESS
ABOUT %4.1f%%.\r\n",(ite*100.)/(*(NumberOfNeuronOfLayers+NumberOfLayers-
1)*nt*1.));
scyc(i,*(NumberOfNeuronOfLayers+NumberOfLayers-1)){
cprintf("\r\n%4.1f%% OF RECOGNIZED OBJECTS WAS BELONG TO OBJECT
%d.",(tt[i]*100.)/(ite*1.+0.00001),i+1);
cprintf("\r\n%4.1f%% OF FALSE RECOGNIZED OBJECTS WAS BELONG TO
OBJECT %d.",(t[i]*100.)/((*(NumberOfNeuronOfLayers+NumberOfLayers-1)*nt-
ite)*1.+0.00001),i+1);
cprintf("\r\nOBJECT %d WAS SUCCEED WITH
%4.1f%%.\r\n",i+1,(tt[i]*100.)/((tt[i]+t[i])*1.));

                                                    }
getch();

farfree(InputVector);farfree(DesiredValue);free(Nt);
free(Transient1);free(Transient2);free(Transient3);
                }
```