# ÖZGÜN BİR ŞEKİL TANIMLAYICISI TASARIMI, GERÇEKLEŞTİRİLMESİ VE TESTİ

YÜKSEK LİSANS TEZİ

Elif Ebru ÇAKI

Danışman
Dr. Öğr. Üyesi Celal Onur GÖKÇE

MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Temmuz 2023

**AFYON KOCATEPE ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**YÜKSEK LİSANS TEZİ**

# ÖZGÜN BİR ŞEKİL TANIMLAYICISI TASARIMI, GERÇEKLEŞTİRİLMESİ VE TESTİ

**Elif Ebru ÇAKI**

**Danışman**
**Dr. Öğr. Üyesi Celal Onur GÖKÇE**

**MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**Temmuz 2023**

**TEZ ONAY SAYFASI**

Elif Ebru ÇAKI tarafından hazırlanan "Özgün Bir Şekil Tanımlayıcısı Tasarımı, Gerçekleştirilmesi ve Testi" adlı tez çalışması lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri uyarınca 13 / 07 / 2023 tarihinde aşağıdaki jüri tarafından **oy birliği** ile Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü **Mekatronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

**Danışman** **:** Dr. Öğr. Üyesi Celal Onur GÖKÇE

**Başkan** **:** Doç. Dr. Durmuş ÖZDEMİR

Dumlupınar Üniversitesi, Mühendislik Fakültesi İmza ……

**Üye** **:** Dr. Öğr. Üyesi Celal Onur GÖKÇE

Afyon Kocatepe Üniversitesi, Teknoloji Fakültesi İmza ……

**Üye** **:** Dr. Öğr. Üyesi Güray SONUGÜR

Afyon Kocatepe Üniversitesi, Teknoloji Fakültesi İmza ……

Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..... /..... /..... tarih ve
……. sayılı kararıyla onaylanmıştır.

…….
Prof. Dr. İbrahim EROL
Enstitü Müdürü

**BİLİMSEL ETİK BİLDİRİM SAYFASI**

**Afyon Kocatepe Üniversitesi**

**Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;**

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

**beyan ederim.**

**13 / 07 / 2023**

**İmza**

**Elif Ebru ÇAKI**

# ÖZET

Yüksek Lisans Tezi

ÖZGÜN BİR ŞEKİL TANIMLAYICISI TASARIMI,
GERÇEKLEŞTİRİLMESİ VE TESTİ

Elif Ebru ÇAKI
Afyon Kocatepe Üniversitesi
Fen Bilimleri Enstitüsü
Mekatronik Mühendisliği Anabilim Dalı
**Danışman:** Dr. Öğr. Üyesi Celal Onur GÖKÇE

Bu araştırmada, nesne tanıma için özgün bir şekil tanımlayıcısı tasarlanmıştır. Gelişen bilgisayar teknolojisi ve yapay zeka ile nesne tanıma algoritmaları da geliştirilmeye başlanmıştır. Literatürde oldukça fazla nesne tanıma algoritmaları yer almaktadır. Çalışmamız özgün bir algoritmadır ve diğer algoritmalara göre az veri ile daha yüksek nesne tanıma başarısı elde edilmektedir. Ön işleme olarak görüntü işleme alanında oldukça kullanılan Canny kenar algoritması kullanılmaktadır. Görselin kenar çıktılarına uygulanan algoritmalar ile şekil tanımlayıcısı oluşturulmaktadır.

**2023, ix + 62 sayfa**

**Anahtar Kelimeler:** Şekil tanımlayıcı, Optimal, Optimal altı, Nesne tanıma, MNIST.

## ABSTRACT

M.Sc. Thesis

DESING, IMPLEMENTATION AND TEST OF A NOVEL SHAPE DESCRIPTOR

Elif Ebru ÇAKI

Afyon Kocatepe University

Graduate School of Natural and Applied Sciences

Department of Mechatronic Engineering

**Supervisor:** Asst. Prof. Celal Onur GÖKÇE

A novel shape descriptor for object recognition is designed in this research study. With the advances in computer technology and artificial intelligence, new object recognition algorithms are developed. Various object recognition algorithms exist in the literatüre. Our study is a novel one and compared with other algorithms our novel algorithm shows high performance with fewer data. As preprocessing, well known Canny edge detector is used. Shape descriptor is built using various algorithms applied to edge image of input.

**2023, ix + 62 pages**



**Keywords:** Shape Descriptor, Optimal, Sub-Optimal, Object Recognition, MNIST.

## TEŞEKKÜR

Lisans ve yüksek lisans eğitimim boyunca bilgileriyle ışık tutan, desteklerini asla esirgemeyen, akademik yolda yürürken bana şevk veren ve bu araştırmanın konusu, deneysel çalışmaların yönlendirilmesi, sonuçların değerlendirilmesi ve yazımı aşamasında yapmış olduğu büyük katkılarından dolayı tez danışmanım Sayın Dr. Öğr. Üyesi Celal Onur GÖKÇE'ye, lisans eğitimimden itibaren hep destekçim olan, bu çalışmam kapsamında araştırma ve yazım süresince yardımlarını esirgemeyen hocam Sayın Dr. Öğr. Üyesi Güray SONUGÜR'e, tüm bu süreçte yanımda olup, desteğini her konuda esirgemeyen Öğr. Gör. Tamer ASLAN'a, her konuda öneri ve eleştirileriyle yardımlarını gördüğüm hocalarıma ve arkadaşlarıma teşekkür ederim.

Hayatım boyunca beni daima destekleyen, haklarını asla ödeyemeyeceğim, maddi ve manevi her konuda aileme sonsuz teşekkür ederim.

Elif Ebru ÇAKI

Afyonkarahisar 2023

# İÇİNDEKİLER DİZİNİ

**Sayfa**

## SİMGELER ve KISALTMALAR DİZİNİ

**Kısaltmalar**

| | |
|---|---|
| CNN | Convolutional Neural Networks |
| CSHOT | Color Shot |
| DE | Derinlik Tamponu |
| DNN | Dekonvolüsyon Sinir Ağı |
| ESD | Açık Şekil Tanımlayıcıları |
| GB | Gigabyte |
| HDC | Hiper Boyutlu Bilgi İşlem |
| HOG | Boyutsal Gradyan Histogramı |
| ISS | İçsel Şekil İmzası |
| k-NN | k-En Yakın Komşu |
| LFD | Lightfield Tanımlayıcısı |
| LRF | Yerel Referans Çerçevesi |
| MNIST | Modified National Institute of Standards and Technology |
| NIST | National Institute of Standards and Technology |
| PSB | Princeton Shape Benchmark |
| SDBC | Bezier eğrileri |
| SI | Siluet Tabanlı |
| STN | Mekansal Trafo Ağları |
| SVM | Destek Vektör Makinesi |
| TOLDI | Üçlü Ortogonal Yerel Derinlik Görüntüleri |
| ZMD | Zernike Moment Tanımlayıcısı |

# ŞEKİLLER DİZİNİ

**Sayfa**

# ÇİZELGELER DİZİNİ

**Sayfa**

## RESİMLER DİZİNİ

**Sayfa**

# 1. GİRİŞ

Günümüzde gelişen teknoloji ile birlikte yaygınlaşan bilgisayarlı görü ve yapay zeka sistemleri nesne tanıma, ses tanıma, sınıflandırma alanlarında kullanılmaktadır. Bunların yanı sıra sağlık, eğitim, tarım, savunma, güvenlik bilgisayarlı görü konusunun ürünlerinin en sık kullanıldığı alanlardan bazılarıdır. Görüntü ve videolardan nesne tanıma ve takip etme de bu alanda yaygınlaşan başlıklar arasındadır (Arslan 2020, Hoşgör ve Güngördü 2022, Şeker 2020, Güzel ve Okatan 2022).

Nesne tanıma ve takip sistemlerinin en kritik parçalarından birisi öznitelik çıkarımıdır. Öznitelik çıkarımı sistemin performansına doğrudan etki etmektedir. Özellikle tek resimlerde kullanılabilecek özniteliklerin bir bölümü nesnenin şekline bağlı olarak tanımlanmaktadır. Bu tarz özniteliklere şekil tanımlayıcısı denilmektedir.

Çalışmamızda nesne tanıma ve takip sistemlerinde kullanılabilecek, daha hızlı ve doğru sonuç için özgün bir şekil tanımlayıcısı tasarlanmaktadır. Bu şekil tanımlayıcısı sayesinde görüntü üzerindeki şekil daha az veri ile temsil edilecektir. Temsil edilecek verilerin azalması ise hızlı işlemi beraberinde getirmektedir. Özellikle gerçek hayatta kullanılacak sistemlerin gerçek zamanlı çalışma kriterlerini sağlaması çok önemlidir. Bu bağlamda sistemin hızı, en az sistemin performansı kadar önem arz etmektedir. Hatta bazı durumlarda sistemin hızını artırmak için performanstan ödün vermek de kabul edilebilir bir tercih olmaktadır.

Çalışmamızda özellikle öznitelik çıkarımı üzerine yoğunlaşılmış, sınıflandırıcı olarak yapay sinir ağları kullanılmıştır. Çalışmamızın özgünlüğü öznitelik çıkarma kısmındadır, sınıflandırıcı olarak herhangi bir özgünlük getirilmemiş standart iki katmanlı yapay sinir ağları kullanılmıştır. Yapay sinir ağlarının parametreleri değiştirilerek kabul edilebilir bir performans elde edilmeye çalışılmış, onun dışında sınıflandırıcı kısmında bir çalışma yapılmamıştır.

Çalışmamızda MNIST veri seti kullanılmaktadır. El yazısı rakamlardan oluşan bu veri seti 70 bin görüntü içermektedir. Her bir rakam için önerilen özgün şekil tanımlayıcısı

kullanılmakta ve elde edilen öznitelikler iki katmanlı yapay sinir ağına girdi olarak verilmektedir.

Bu çalışmanın ilk bölümünde çalışma hakkında genel bilgilendirme yapılmaktadır. İkinci bölümde çalışma alanındaki diğer çalışmalar incelenmekte, ilgili referanslara atıf verilerek güncel bir literatür taraması sunulmaktadır. Üçüncü bölümde çalışmamızda kullandığımız materyal ve metotlar anlatılmaktadır. Özgün şekil tanımlayıcımızın çalışma sistemi de bu başlık altında detaylıca yer almaktadır. Dördüncü başlıkta çalışmamızdan elde ettiğimiz bulgulara, beşinci başlık altında ise çalışmamızın sonuçlarına ve bu sonuçlar doğrultusundaki tartışmalar bulunmaktadır. Çalışmamızın etkinliği ve kullanılabilirliği değerlendirilmiştir. Altıncı başlıkta ise çalışmamız kapsamında kullanılan kaynaklar yer almaktadır.

Çalışmamızın konuyla ilgili literatüre ve gerçek hayatta kullanılabilecek sistemlere önemli bir katkı sağladığını düşünüyoruz.

## 2. LİTERATÜR TARAMASI

Tombari vd. (2011) yapmış oldukları bu çalışmada, 3B özellikler için histogramların açıklaması için yeni bir formülasyon sunmuşlardır. Ardından doku ve şekillerin tanımlanması için de CSHOT'u önermişlerdir. Önerilen tanımlayıcı zor nesnelerin tanımlanmasında özellik eşleştirmenin gerçekleştirildiğini göstermektedir.

Fang vd. (2015) yapmış oldukları bu çalışmada, farklı şekil tanımlayıcılarının tanımlanmasında kullanılacak bir yöntem geliştirmişlerdir. Derin şekil tanımlayıcısı ismini verdikleri çalışmaları ile sınıflar arasındaki marjı yükseltmeyi hedeflemektedirler. DNN ile geliştirdikleri çalışmalarında çok sayıda alandan son teknolojiyi kullanmanın yanı sıra 3B verileri ve 3B verilerin tanımındaki yapısal çeşitlik ile tutarsızlıkları da ele alarak en zor tanımlanmaya sahip olduğunda dahi bunları tanıyarak bu zor verileri tanıma ile başa çıkmaktadır.

Xie vd. (2015) yapmış oldukları bu çalışmada, geometrik 3B şekillerin tanımlanmasında kullanılacak derin bir şekil tanımlayıcısı önermişlerdir. 3 adımda gerçekleştirdikleri çalışmalarında ilk olarak otomatik kodlayıcı uygulanmaktadır. Otomatik kodlayıcının ardından Fisher uygulanmakta ve son olarak da gizli katmandaki nöronlar tanımlayıcının oluşturulması için birleştirilmektedir. 3B şekillerin eşleştirilmesi için eğitilen otomatik kodlayıcıların başarıları çeşitli testlerde gösterilmek üzere çalışma sonlanmaktadır.

Xu vd. (2016) yapmış oldukları bu çalışmada, şekil eşleştirme ve nesne tanıma için çok ölçekli bir tanımlayıcı önermişlerdir. Önerilen yöntemde kontur baz alınır ve kontur nesnenin şeklini temsil etmektedir. Çalışma anlık el hareketlerini tanımaktadır. Yapılan deneyler sonucunda da önerilen tanımlayıcının doğruluğu ve verimliliği gösterilmektedir.

Shi vd. (2020) yapmış oldukları bu çalışmada, şekil tanımlayıcılarının hesabında kullanılmak üzere denetimsiz bir yöntem geliştirmişlerdir. Çalışmadaki temel yenilik derin sinir ağlarındaki mesafe kaybı ile noktaların dağılımı için şekil tanımlayıcının

ortaya konmasıdır.  Önerilen yöntem literatürdeki diğer denetimsiz tanımlayıcılara göre üstün başarı performansı göstermektedir.

Susan vd. (2019) yapmış oldukları bu çalışmada, kenarlardaki piksellerin devamını baz alarak nesnenin konturunu tanımlama çalışması gerçekleştirmişlerdir. Nesnenin konturunu bitişik kenar piksel sayısı ve kenarın bir sonraki bitişik pikseldeki sürekliliğini ölçmektedir. Kenar yönü ve kenar yönü süreklilik özelliklerini ele alarak yaptıkları çalışmada başarılı sonuçlar elde edilmiştir.

Xie vd. (2016) yapmış oldukları bu çalışmada, 3 boyutlu şekillerin tanımlanması için bir tanımlayıcı önerilmiştir. Çok ölçekli şekil dağıtımları hesaplanmakta ve ölçeklerin farklılığında özelliklerin çıkarılması için otomatik bir kodlayıcı eğitilmiştir. Tanımlayıcı farklı ölçekleri içeren üst düzey özniteliklerin birleştirilmesi ile gerçekleştirilmiştir. Farklı veri setlerinde sistem denenmiş ve deneysel sonuçlar 3 boyutlu şekil alma için tanımlayıcının etkinliğini göstermektedir.

Kunaver vd. (2005) yapmış oldukları bu çalışmada, farklı teknikleri ele alarak öznitelik çıkarımlarına genel bir bakış sunmaktadır. Görüntü işleme ve nesne tanımlama alanlarında öznitelik çıkarma aşaması önemli rol almaktadır. Bu bağlamda çalışmada da farklı öznitelik çıkarma düzeyleri anlatılmakta ve bu öznitelik çıkarımlarının farkları ele alınmaktadır.

Kim vd. (2000) yaptıkları çalışmada kontur tabanlı ve bölge tabanlı iki tip şekil tanımlayıcısı ile çalışılmaktadır. Kontur tabanlı tanımlayıcılar karmaşık şekiller için güvenilir değildir, bu nedenle bunun yerine Zernike momentleri gibi bölge tabanlı tanımlayıcılar kullanılır. Zernike momentleri, dönme değişmezliği, sağlamlık, ifade verimliliği, hızlı hesaplama ve çok düzeyli temsil özelliklerine sahiptirler.

Sohel vd. (2005) tarafından yapılan çalışmada Bezier eğrilerinin kullanımı, bilgisayar destekli tasarımdan kaligrafik karakter eskizlerine ve nesne şekli açıklamalarına kadar çok çeşitli uygulamalar için sağlam bir araç sunmaktadır. Ancak Bezier eğrileri, ani değişimlere veya keskin köşelere maruz kalan bir nesnenin şeklini tanımlamaya

geldiğinde hatalar üretebilmektedir. Bu sorunu ele almak için, kontrol noktaları oluşturmak üzere alana özgü bilgileri entegre etmek için Bezier eğrilerini (SDBC) kullanan bu çalışmada Şekil Açıklama Algoritması adlı yeni bir strateji tanıtılmaktadır. Bu algoritma aynı zamanda verimli kontrol noktası kodlaması için geliştirilmiş bir dinamik sabit uzunluklu kodlama şeması içermektedir. SDBC çerçevesi, çeşitli şekiller üzerinde kapsamlı bir şekilde test edilmiştir ve hem nitel hem de nicel analizler, mevcut algoritmalara kıyasla üstün performansını göstermektedir. Bu makale, bir nesnenin şeklini tanımlamada, yani kontrol noktalarının seçiminde Bezier eğrilerini kullanmanın kritik yönünü vurgulamaktadır. SDBC stratejisi, alana özgü bilgileri içeren ve daha etkili bir kodlama sistemine sahip daha doğru bir temsil sunar. Yeni algoritma, tasarım, robotik ve bilgisayar görüşü gibi çeşitli alanlarda potansiyel uygulamalara sahiptir.

Sparks vd. (2013) yapmış oldukları çalışmada orta dereceli prostat bezleri arasındaki ince morfolojik farklılıkları doğru bir şekilde ayırt edebilen yeni bir Açık Şekil Tanımlayıcıları (ESD) seti sunmaktadır. ESD'ler, Graph Embedding yoluyla bir şekil modeli, diffeomorfik kayıt ve doğrusal olmayan boyutluluk azaltma kullanılarak elde edilmektedir. Yüksek doğrulukla, ESD'ler farklı Gleason derecelerindeki prostat bezleri arasında ayrım yapabilmekte ve bunların prostat kanseri tanı ve tedavisini geliştirmedeki potansiyellerini sergileyebilmektedir.

Vranic vd. (2005) yapmış oldukları bu çalışmada, 3B ağ modelleri için şekil benzerliği aramasını tartışmaktadır ve DESIRE adlı bileşik bir 3B şekil öznitelik vektörünü tanıtmaktadırlar. Çokgen bir ağın derinlik arabelleği görüntülerini, silüetlerini ve ışın uzantılarını birleştirmekte ve LightField tanımlayıcısı (LFD) ile karşılaştırmaktadır. Deneyler, DESIRE'nin genel olarak, modellerin ikili hizalanmasına dayanan LFD'den daha iyi performans gösterdiğini göstermektedir. Sonuçların doğrulanması için Web tabanlı bir erişim sistemi ve yürütülebilir dosyalar sağlanarak DESIRE daha etkili, verimli ve daha az karmaşık olarak sergilenmektedir.

Derinlik Tamponu tabanlı (DE) ve Siluet tabanlı (SI) tanımlayıcılar gibi kullanılan 3B şekil tanımlama tekniklerinin ayrıntılı açıklamaları yer almaktadır. Yazarlar, 3D şekilleri ortogonal ve etkili bir şekilde karakterize etmek için farklı özellikleri

birleştirmenin önemini vurgulamaktadır. Princeton Shape Benchmark (PSB) ve yazarların kendi 3B model koleksiyonu üzerindeki deney sonuçları, DESIRE tanımlayıcısının sağlamlığını ve etkililiğini göstermektedir.

Çalışmada LightField tanımlayıcısını ve DESIRE'ı karşılaştırılarak, PSB 3D model setini kullanarak performanslarını değerlendirilmektedir. DESIRE'ın, erişim etkinliği ve karmaşıklığı açısından LightField tanımlayıcısından daha iyi performans gösterdiği görülmektedir.

Zhang vd. (2003) yapmış oldukları bu çalışmada, kontur tabanlı ve bölge tabanlı olmak üzere iki şekil tanımlayıcıyı, MPEG-7tarafından tanımlanan altı ilkeye göre diğer üç şekil tanımlayıcıya karşı değerlendirmektedir. Fourier tanımlayıcısı (FD), hesaplama karmaşıklığı açısından eğrilik ölçeği uzay tanımlayıcısından (CSSD) daha iyi performans göstermektedir. Makalede değerlendirilen bölge bazlı üç şekil tanımlayıcı arasından bölge bazlı şekil çıkarımı için en uygun Zernike moment tanımlayıcısı (ZMD) belirlenmiştir.

Wan vd. (2023) yapmış oldukları bu çalışmada, 2DLPP algoritması, 2B görüntü temsilini kullanarak çok boyutlu uzamsal verilerin manifold özelliklerini ve yerel bilgilerini koruyan popüler bir özellik çıkarma algoritması olarak çeşitli alanlara başarıyla uygulanmıştır. 2DLPP algoritması, gerçek dünya uygulamalarında ayrım yeteneği eksikliği, tekillik sorunları ve veri dağınıklığına ve gürültüye karşı hassasiyet gibi bazı zorluklarla karşı karşıya kalmaktadır. 2DLPP algoritmasını düşük dereceli iki boyutlu yerel ayırt edici grafik gömme ile birleştirerek bu sorunlara çözüm olarak LR-2DLDGE adlı yeni bir öznitelik çıkarma algoritması önerilmiştir. LR-2DLDGE algoritmasının üç avantajı vardır: bir grafik gömme kullanarak yerel komşuluk ayrım bilgilerini korumak, özellik uzayında farklı sınıflardan veri noktalarını olabildiğince bağımsız hale getirmek ve düşük seviyeli öğrenmeyi kullanarak bir kısıtlama olarak L1 normu aracılığıyla gürültü ve bozulmanın etkisini azaltmaktır. Algoritmanın teorik hesaplama karmaşıklığı ve yakınsaması açıklanmış ve kanıtlanmıştır, ayrıca karmaşık ve gürültülü üç görüntü veri kümesi üzerindeki kapsamlı deneysel sonuçlar, LR-2DLDGE algoritmasının etkinliğini ve sağlamlığını doğrulamaktadır.

Zhong vd. (2009) yapmış oldukları çalışmada, İçsel Şekil İmzası (ISS) adı verilen bir şekil tanımlayıcı kullanarak nokta bulutları olarak temsil edilen 3B nesneleri tanımak için yeni bir algoritma önermektedir. Algoritma hızlı poz tahminini kolaylaştırmakta ve verimli bir indeksleme içermektedir. ISS algoritması, sensör gürültüsü, bulanıklık ve sahne karmaşası varlığında 72 modelden oluşan bir veri tabanı kullanılarak farklı araç türlerini tanıma gibi zorlu bir görev üzerinde değerlendirilmiştir. Sonuçlar, diğer tanımlayıcı tabanlı 3B tanıma algoritmalarından daha iyi performans gösterdiğini ve gerçek dünyadaki 3B uygulamaları için umut verici bir çözüm olduğunu göstermektedir. ISS'nin performansı Spin Image ve 3D Shape Context gibi yerleşik 3D tanıma algoritmalarıyla karşılaştırılır ve ISS yaklaşımının, özellikle dağınıklık ve karartma varlığında daha iyi performans gösterdiği gözlemlenmektedir.

Yang vd. (2017) yapmış oldukları bu çalışmada, yeni bir yerel referans çerçevesi (LRF) ile TOLDI (üçlü ortogonal yerel derinlik görüntüleri) kullanan yerel şekil tanımlaması için yeni bir yöntem önermektedir. TOLDI yaklaşımı, çeşitli bağlamlarda önceki yöntemlere kıyasla verimliliği, ayırt ediciliği ve sağlamlığı önemli ölçüde artırır. Önerilen LRF, sırasıyla x ve z ekseni hesaplamaları için anahtar noktanın normalini ve komşularının ağırlıklı izdüşümlerinin vektör toplamını kullanarak önceki yöntemlerden farklıdır. TOLDI tanımlayıcısı, kapsamlı uzamsal ve geometrik bilgileri kodlayarak, LRF'deki üç ortogonal görünüm tarafından oluşturulan üç LDI imzasından gelen özellik vektörlerini birleştirmektedir. Yöntem, şekil eşleştirme, nesne tanımave 3B kayıt için çeşitli veri kümeleri üzerinde değerlendirilerek etkinliği, sağlamlığı ve verimliliği gösterilmektedir.

Taati vd. (2011) yapmış oldukları bu çalışmada, optimize edilmiş yerel şekil tanımlayıcıları, aralık verilerinde nesne tanıma ve yerelleştirme için bir optimizasyon problemi olarak formüle edildi. Değişken Boyutlu Yerel Şekil Tanımlayıcıları, özellik seçimi kullanılarak optimizasyon probleminin çözülebileceği bir platform oluşturmak için kullanıldı. Optimize edilmiş tanımlayıcıların, daha genel tanımlayıcılara kıyasla benzer işlem süreleriyle daha yüksek tespit oranları sağladığı deneysel olarak gösterilmiştir. Formülasyon ayrıca menzil görüntüleme için kullanılan sensöre dayalı

olarak tanımlayıcıların seçilmesine izin verir. Ön işleme modelleri, herhangi bir model tabanlı nesne tanıma için tanımlayıcıların optimize edilmesinden yararlanabilir sistemdir. Tek tek modelleri işlemek yerine nesneleri üst düzey şekil özelliklerine göre gruplandırmak, kümeleme olasılıklarını keşfetmek için alan bırakmaktadır.

Bober vd. (2011) yapmış oldukları bu çalışmada, özellikle MPEG-7 standardizasyonu bağlamında, şekil gösterimi ve eşleştirme için kullanılan teknikleri ve araçları açıklamaktadır. Nesne kimliği ve işlevselliğine ilişkin ipuçları sağlamada nesne şekli özelliklerinin rolü araştırılır ve kontur tabanlı şekil tanımlayıcısı, diğer çeşitli şekil tanımlayıcılarının yanı sıra ayrıntılı olarak sunulur. Makale ayrıca görsel nesne alma veya indeksleme, video güvenlik sistemleri ve e-ticaret dahil olmak üzere bu şekil tanımlayıcıların sahip olduğu farklı uygulamaları da araştırıyor. Benzer nesneleri tanımlamada ve görüntüleme koşullarından veya bölütleme süreçlerinden kaynaklanan bozulmalarla başa çıkmada etkili şekil tanımlayıcılarının önemi de vurgulanmaktadır. Kontur tabanlı şekil tanımlayıcının, Web tabanlı bir video klip arama uygulamasında örneklendiği gibi, özellikle sağlam, verimli ve etkili olduğu gösterilmiştir. Genel olarak, makale, nesneleri tanıma ve geri almada şekil özelliklerinin öneminin yanı sıra, varyasyonlar ve bozulmalarla başa çıkabilen etkili şekil tabanlı tekniklere ve araçlara olan ihtiyacı vurgulamaktadır.

Belongie vd. (2000) yapmış oldukları bu çalışmada, şekil tanımada farklı ölçümler karşılaştırılmaktadır. Çalışma, şekil mesafesi ölçümünün MNIST veri setinde SSD'den daha iyi performans göstermektedir. 3D nesne tanımada prototipleri daha verimli bir şekilde tahsis etmek için şekil mesafesine ve K-medoid kümelemeye dayalı yeni bir düzenleme algoritması geliştirilmiştir. Her üç boyutlu nesne için ortalama yalnızca 4 iki boyutlu görünümle iyileştirilmiş sınıflandırma performansı elde edilmektedir. Yaklaşım, yerleşik standart değişmezliklerle basit ve sağlamdır. Araştırma, ARO, NSF ve DFG dahil olmak üzere çeşitli hibelerle desteklenmektedir.

Medjahed vd. (2015) yapmış oldukları bu çalışmada, ikili ve çok sınıflı sınıflandırmalarda görüntü sınıflandırması için farklı sınıflandırıcılarla çeşitli özellik çıkarma tekniklerinin performansını değerlendirmektedir. PHOG, GABOR ve LBP

yöntemlerinin, Doğrusal SVM sınıflandırıcı ile en yüksek doğruluk oranını verdiğini ve PHOG'un çok sınıflı sınıflandırmada diğer tüm yöntemlerden daha iyi performans gösterdiğini bulmuştur. Sınıflandırma modelinin performansını değerlendirmek için sınıflandırma doğruluğunun yanı sıra kesinlik, hatırlama, f-measure ve G_mean gibi metrikleri değerlendirmenin önemini vurgulamaktadır.

Grappolini ve Subramoney (2023) yapmış oldukları bu çalışmada , artan sinir ağlarındaki zorlu görevleri çözmek için sinaptik ağırlıklar yerine sinaptik gecikmeleri kullanmanın fizibilitesini ve gücünü araştırmaktadır. Çalışma, geriye yayılım kullanarak yalnızca ileri beslemeli ani artış ağlarındaki gecikmelerin eğitiminin, ağırlık çalışmasıyla karşılaştırılabilir bir performans sağlayabildiğini göstermektedir. Üçlü değerler gibi kısıtlanmış ağırlıklar, ağların yalnızca sinaptik gecikmeleri kullanarak görevleri çözme yeteneğini önemli ölçüde etkilememektedir. Araştırma, geleneksel ağırlığa dayalı hesaplama modellerinden daha verimli olabilen, artan sinir ağlarını eğitmek için yeni bir paradigma önermektedir. Önceki çalışmalar, zaman kodlama şemalarını ve ani sinir ağlarında ani öğrenme zamanı ile bir SNN'deki eğitim gecikmelerini araştırmaktadır. Vekil gradyan yöntemiyle geri yayılım kullanan saf gecikme eğitiminin, ağırlık çalışmasıyla karşılaştırılabilir performans elde ettiği gösterilmektedir. Çalışma, ani yanıt modellerini kullanır ve sinaptik ağırlıklar ve gecikmeler, vekil gradyanlar kullanılarak eğitilmektedir. Deneyler, MNIST ve Fashion-MNIST veri kümelerinde yalnızca gecikmeli eğitimin görev performansını göstermektedir. Araştırma, daha iyi zamansal kodlama ve daha kesin gecikme eğitimi yöntemleri dahil olmak üzere, sinir ağlarını hızlandırmada öğrenme ve hesaplamanın temel bir bileşeni olarak gecikmelerin kullanılmasının daha fazla araştırılması için fırsatlar sunmaktadır.

Sakthimohan vd. (2023) yapmış oldukları bu çalışmada, basamak onayı üzerindeki etkiyi analiz etmek için bir CNN (Evrişimli Sinir Ağı) modelinin kullanımını tartışmaktadır. Model, el yazısı rakamlardan oluşan bir bilgisayar görüntü veri seti olan MNIST veri setini kullanır. CNN modeli, evrişim katmanları, havuzlama katmanları ve tamamen bağlı katmanlar dahil olmak üzere birkaç katmandan oluşmaktadır. Modelin performansını iyileştirmek için stokastik gradyan iniş ve bırakma düzenlileştirme

teknikleri kullanılmaktadır. Modelin doğruluğu, kesinlik ve hata oranı gibi metrikler kullanılarak değerlendirilmektedir. Çalışma, CNN modelinin rakam doğrulama görevlerinde yüksek doğruluk elde edebildiğini göstermektedir. Makale ayrıca, modelin performansını daha da iyileştirmek için küme standardizasyonu ve toplu normalleştirme tekniklerinin kullanımını da ele almaktadır. Sonuçlar, CNN modelinin rakam doğrulamada diğer geleneksel yöntemlerden daha iyi performans gösterdiğini göstermektedir.

Shao vd. (2023) yapmış oldukları bu çalışmada, evrişimli sinir ağları (CNN) gibi gelişmiş makine öğrenimi tekniklerini keşfederek, farklı bireyler arasında el yazısı stillerinin tanınma doğruluğunu ve el yazısı metnin düşük görüntü kalitesini iyileştirme üzerine çalışmaktadır. k-En Yakın Komşu (k-NN), boyutsal gradyan histogramı (HOG) özellikleri, istatistiksel sınıflandırma modelleri, destek vektör makinesi (SVM) ve kümeleme gibi hakim makine öğrenimi tekniklerinin el yazısı rakam tanıma ile kombinasyonu daha fazla olabilir sınıflandırma ve tanıma doğruluğunu geliştirmek için çalışılmıştır. Önerilen CNN modelinin diğer veri kümelerine veya tanıma görevlerine aktarılabilirliği, farklı senaryolarda uygulanabilirliğini belirlemek için değerlendirilebilmektedir.

Bettayeb vd. (2023) yapmış oldukları bu çalışmada, görüntü sınıflandırması için uzamsal trafo ağlarını (STN) ve Hiper Boyutlu Hesaplamayı (HDC) birleştiren bir çerçeve olan SpatialHD'yi tanıtmaktadır. SpatialHD, ağ içindeki verilerin uzamsal manipülasyonunu gerçekleştirmek için STN'yi, ardından STN çıkışı üzerinde çalışmak ve verileri sınıflandırmak için HDC'yi kullanılmaktadır. SpatialHD'deki STN ve HDC kombinasyonu, temel HDC'ye kıyasla doğruluğu yaklaşık %8 artırmakta ve verimliliği de 2,5 kat artırmaktadır. Önerilen çerçeve, etkili ve güvenilir bilişsel öğrenmeyi kolaylaştırma potansiyeline sahiptir.

Ponce vd. (2023) yapmış oldukları bu çalışmada, evrişimli sinir ağı (CNN) modellerini eğitmek için dağıtılmış bir yara tedavisi optimizasyonu (WTO) yönteminin kullanımını tartışmaktadır. Önerilen yöntem, el yazısı rakam tanıma için MNIST veri seti üzerinde değerlendirilmiştir. Deneysel sonuçlar, dağıtılmış DTÖ yönteminin, tek bir öğrenme

aracısı ile bir taban çizgisine kıyasla eğitim süresini %36,87 oranında iyileştirdiğini göstermektedir. Önerilen yöntem kullanılarak CNN modelinin doğruluğu da taban çizgisine göre %4,69 artmaktadır. Makale, yöntemin robotik, çok etmenli sistemler, birleşik öğrenme ve karmaşık optimizasyon problemleri gibi çeşitli alanlarda uygulanabileceğini önermektedir.

Literatür araştırmasında görüldüğü üzere öznitelik çıkarımına ait çalışmalar bulunmaktadır. Kullanılan veri setinin de oldukça başarılı sınıflandırılması gerçekleştirilmiştir. Çalışmamızda önerilen özgün öznitelik çıkarma algoritmasının hem özgün olması hem de hızlı öznitelik elde etmesi ile literatüre katkı sağlandığı düşünülmektedir.

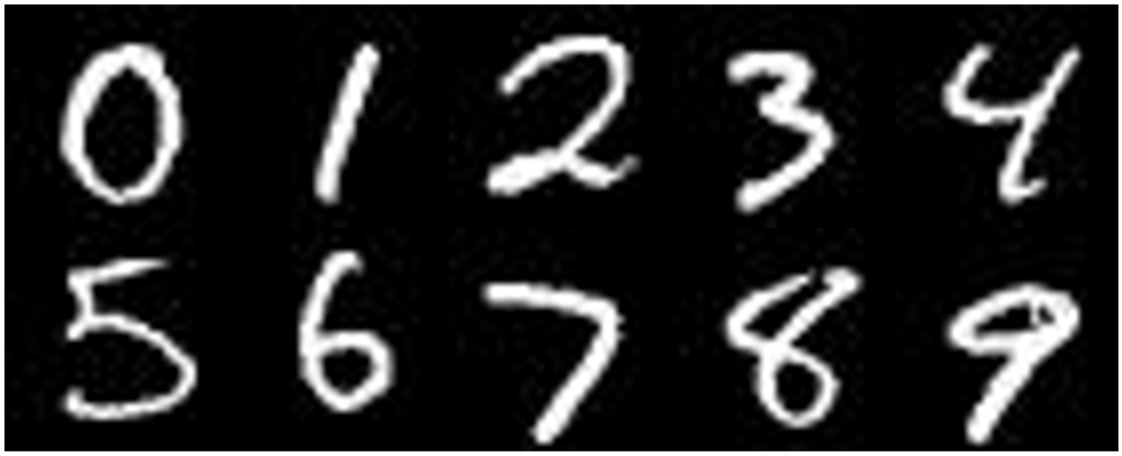
# 3. MATERYAL ve METOT

## 3.1 MNIST Veri Seti

MNIST (Modified National Institue of Standarts) veri seti literatürde farklı görüntü işleme tekniklerinde sıkça kullanılan bir veri setidir. 1998 yılında NIST'in kombinasyonu olarak oluşturulmuştur. 60.000 eğitim verisi, 10.000 test verisi içermektedir. Veri seti içeriğinde 0 ile 9 arasındaki rakamlardan oluşmaktadır. Rakamlar lise öğrencileri ve ABD Sayım Bürosu çalışanları tarafından yazılmıştır. Çizelge 3.1'de veri setinin rakamsal dağılımları verilmektedir. Veri seti içerisinde tüm rakam görüntüleri gri seviyeli ve 28x28 boyutunda, 784 pikseldir. Veri setinden alınan rakam örnekleri Şekil 1'de verilmektedir (Wu vd. 2010).

**Çizelge 3.1** MNIST veri seti dağılımları

| Rakam | Test | Training |
|---|---|---|
| 0 | 980 | 5923 |
| 1 | 1135 | 6742 |
| 2 | 1032 | 5958 |
| 3 | 1010 | 6131 |
| 4 | 982 | 5842 |
| 5 | 892 | 5421 |
| 6 | 958 | 5918 |
| 7 | 1028 | 6265 |
| 8 | 974 | 5851 |
| 9 | 1009 | 5949 |
| TOPLAM | 10000 | 60000 |



**Şekil 3.1** MNIST veri setinden alınan rakamlar.

## 3.2 Canny

1986 yılında John F. Canny tarafından geliştirilen Canny algoritması, görüntülerdeki kenarları algılamak için çok aşamadan oluşan bir algoritma kullanmaktadır. Değişik bilgisayarlı görme sistemlerinde yaygınca kullanılan bu algoritma görüş sistemlerindeki kenar algılamaların genelde benzer olduğunu bulmuştur. Algoritma optimaldir ve uygulama süreci oldukça basittir. Canny kenar algoritması süreci beş adımda gerçekleştirilmektedir. Öncelikle görüntülerdeki gürültünün giderilmesi için Gauss Filtresi uygulanmaktadır. Canny, bir Gauss fonksiyonunun birinci türevi tarafından iyi bir şekilde yaklaştırıldığında ortaya çıkan bir optimal kenar saptama operatörü türetmekte ve uygulamaktadır (J. Canny 1986). İkinci adım olarak yoğunluk gradyanları bulunmaktadır. Görüntüde kenar olmayanları ayrıştırmak için gradyan eşiği veya alt sınır kesme işlemi uygulanmaktadır. Kenarların potansiyel olanlarının tespiti için ise çift eşik uygulanmaktadır. Son adım olarak kenar histerezisi gözlemlenmektedir. Zayıf olan ve güçlü kenarları olmayan tüm kenarlar belirlenerek kenar tespiti gerçekleştirilmektedir.

Görüntüden gürültünün giderilmesi için bir Gauss filtresi kullanılmaktadır. Kenar algılama tarafından ani yoğunluk farkı sebebiyle bu gürültü kenar olarak kabul edilebilmekte olduğundan bu filtre uygulanmaktadır. Gauss çekirdeğindeki öğelerin toplamı 1'dir. Bu sebeple görüntüye uygulanmadan önce çekirdek normalleştirilmektedir. Gauss çekirdeği Denklem 3.1'de verilmektedir. Denklemde $\sigma$

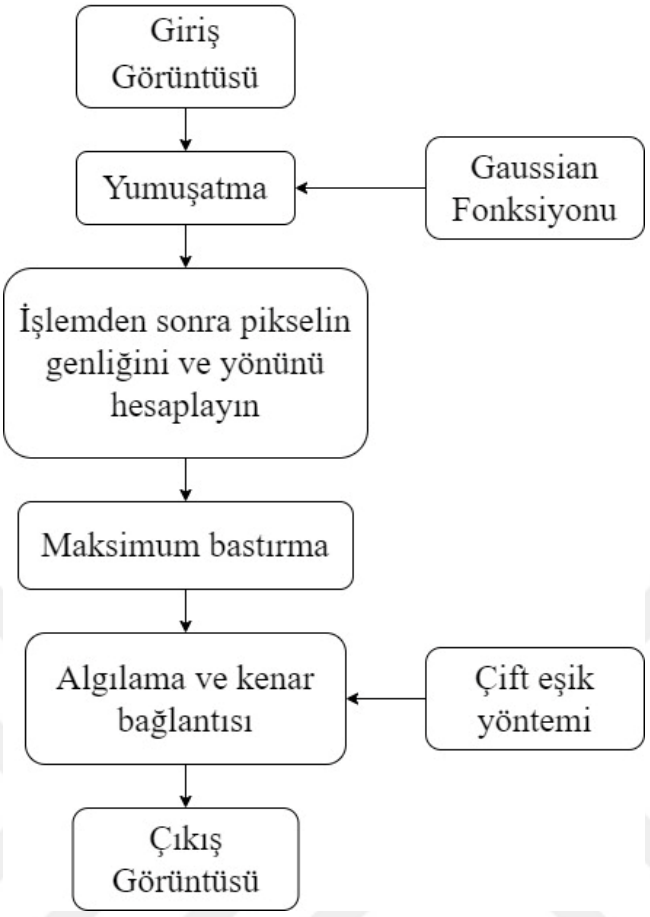
Gauss fonksiyonunun standart sapması ve $(x, y)$ görselin kartezyan koordinatlarıdır (D.Marr 1980).

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{3.1}$$

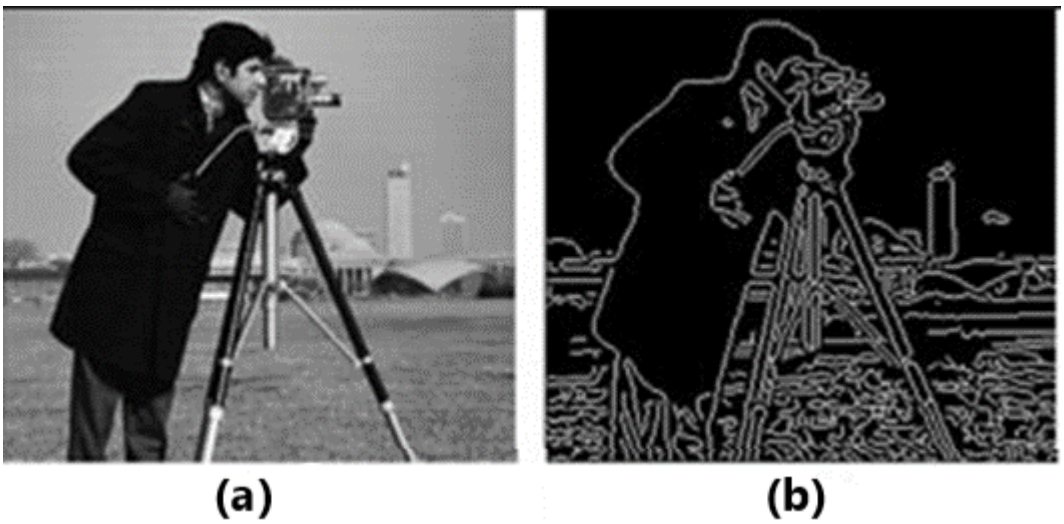
Canny kenar algoritmasının sözde kodu ve Şekil 3.2'de algoritmanın akış diyagramı verilmektedir.

Canny kenar algoritması sözde kodu;

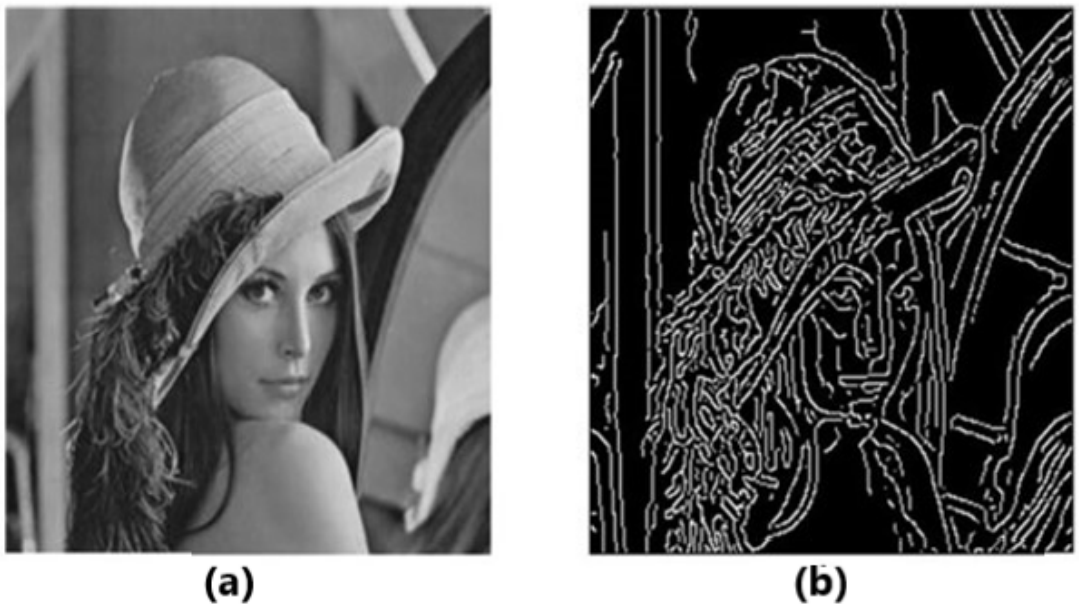
1. İşlenecek giriş görüntüsünü alın.
2. Canny kenar çıkarma işlemi için OpenCv kitaplığını yükleyin.
3. Görüntüyü dosyadan okur ve görüntüyü matris nesnesine kaydeder.
4. Resmi matris olarak okuyun.
5. İşlenen görüntüyü kaydetmek için boş bir matris oluşturun.
6. Kenarları algılamak için canny işlevini uygulayın.
7. Kenarlar = cv. Canny (görüntü,100,200);
8. cv.Canny (bu işlem için giriş görüntüsünü temsil eden bir nesne, bu işlem için hedefi (kenar) temsil eden bir matris nesnesi);
9. Görüntü işleme sonuçlarını oluşturun ve kaydedin.
   (Andayani vd. 2020)




**Şekil 3.2** Canny akış diyagramı.




**Resim 3.1** (a) Orijinal kameraman görüntüsü (b) Canny kenar dedektörü (Verma vd. 2013).




**Resim 3.2** (a) Orijinal Lena görüntüsü (b) Canny kenar dedektörü (Hussain vd. 2018).

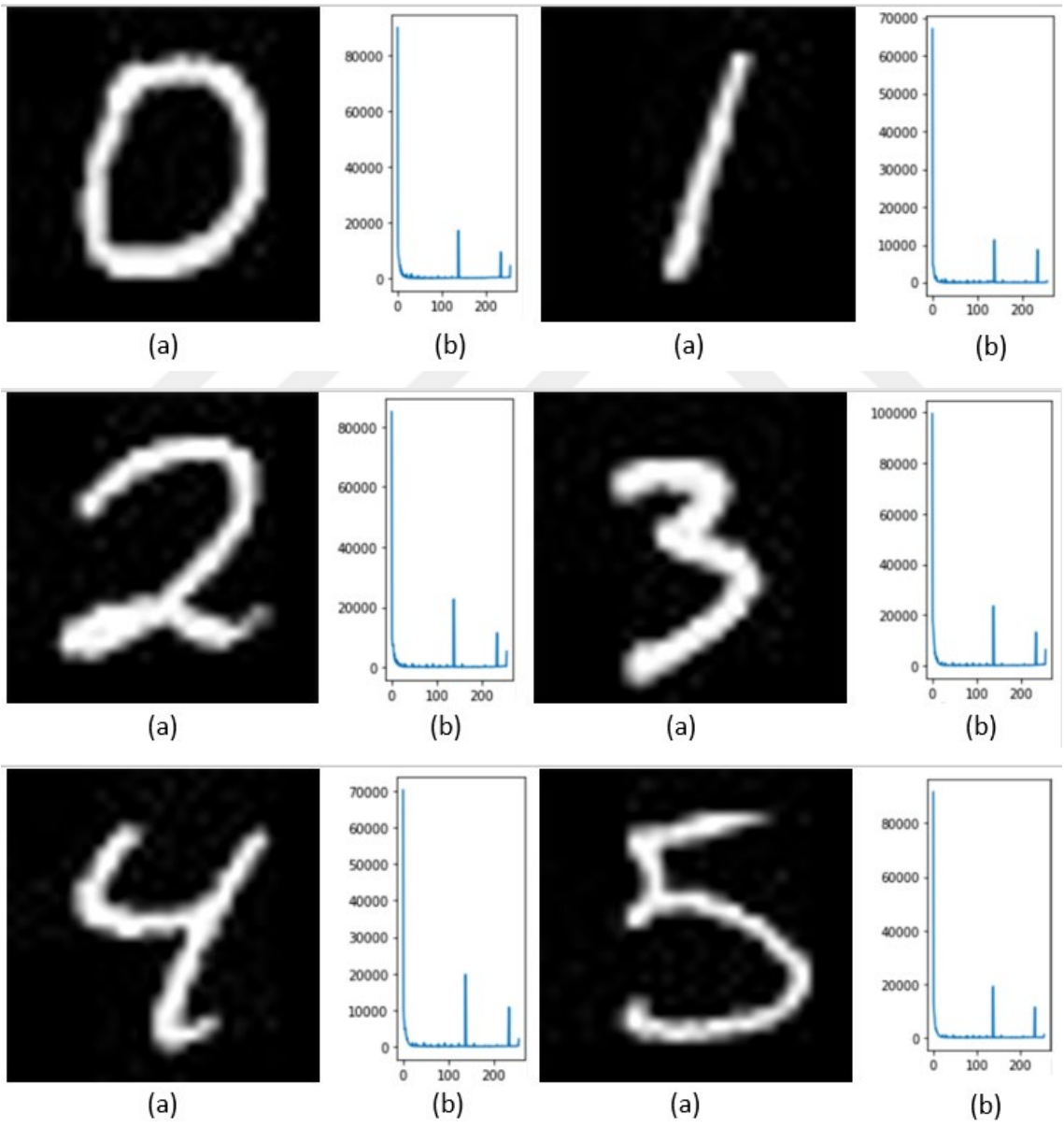
### 3.3 Threshold (Eşikleme)

Görüntü işleme tekniklerinden biri olan Threshold, görüntüleri ayrıştırmak için kullanılan, siyah ve beyaz tabanlı görüntülere dönüştürerek görüntü analizini sağlamaktadır. Çalışmada threshold görüntü işleme tekniğinden olan histogram eşikleme yöntemi kullanılmaktadır. Yapılan eğitimler sonucunda ağ başarısının yükseltilmesi hedefi doğrultusunda bu yöntem uygulanmaktadır.

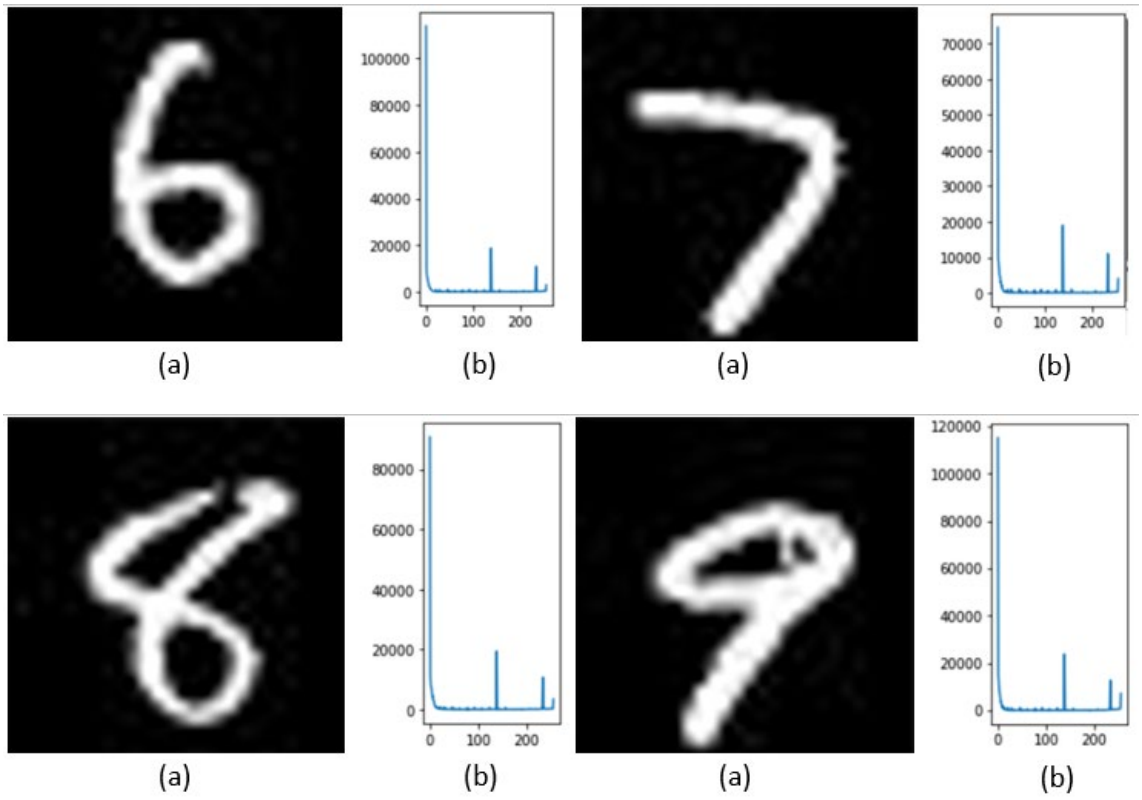
Histogram görüntünün renk dağılımlarının grafiksel gösterimini oluşturmaktadır. Grafiğin xy düzlemleri görüntü piksel değerini ve görüntüde o piksel değerinde bulunan piksel sayısını vermektedir. Histogramı alınan görüntünün kümülatif histogramı elde edilir. Kümülatif histogram piksellerdeki değerlerin bir önceki piksel değeri ile bulunulan piksel değerinin toplamdan elde edilmektedir. Toplam değerlerin piksel sayısına bölümü ile de görüntü normal seviyesine eşitlenmektedir. Yani ortalama piksel değeri ile dağılım yapılarak görüntünün iyileştirilmesi gerçekleştirilmektedir. Denklem 3.2'de $r$ değişkeni görüntünün piksel değeri olsun, yani $r$ [0, $L$ - 1] aralığındadır. $r = 0$ siyah ve $r = L - 1$ beyazdır. Çalışmamızda görsellerimiz 8 bit olduğu için maksimum piksel değeri 255'tir. Bu durumda $T(r)$ toplam dönüştürme fonksiyonu sayısını ve $s$ piksel değerlerini vermektedir.

$$s = T(r)\ 0 \leq r \leq L - 1 \tag{3.2}$$

Kullanılan veri setinden ele alınan verilerin histogram dağılımları Şekil 3.3'de verilmektedir. Şekillerde (a) veri seti içerisinden alınmış veri örneklerini, (b) görüntülerin histogram dağılımlarını vermektedir. Histogram dağılımları her veri için farklı değerleri içermektedir. Her veri değeri eşiklenerek işleme alınmakta ve ardından ağa girdi olarak verilmektedir.




**Şekil 3.3** (a) MNIST veri seti örnekleri, (b) verinin histogram dağılımları.




**Şekil 3.3 (Devam)** (a) MNIST veri seti örnekleri, (b) verinin histogram dağılımları.

## 3.4 Optimal Algoritma

Optimal algoritma, görüntüden alınan Canny kenar görselindeki koordinatları kullanmaktadır. Koordinatların x ve y eksenlerinin en uç noktalarından alınan değerlerden görseldeki doğu, batı, kuzey ve güney koordinatları bulunur. Kenar görüntüsündeki tüm noktalardan bu noktalar ayrıştırılmakta ve ardından bulunan bu 4 nokta baz alınarak bu noktalara en uzak diğer nokta bulunmaktadır. Şekil 3.5'de temsili doğu, batı, kuzey ve güney koordinatları verilmektedir. Uzaklık hesaplaması Öklid hesaplaması ile yapılmaktadır. Belirlenen en doğu, en batı, en kuzey ve en güney noktalarına en uzak değerdeki koordinat noktası bulunduğunda 5. nokta elde edilmektedir. Ardından algoritma elde edilmiş olan bu 5 noktaya en uzak diğer noktayı Öklid ile hesaplayarak 6. noktayı bulmaktadır.

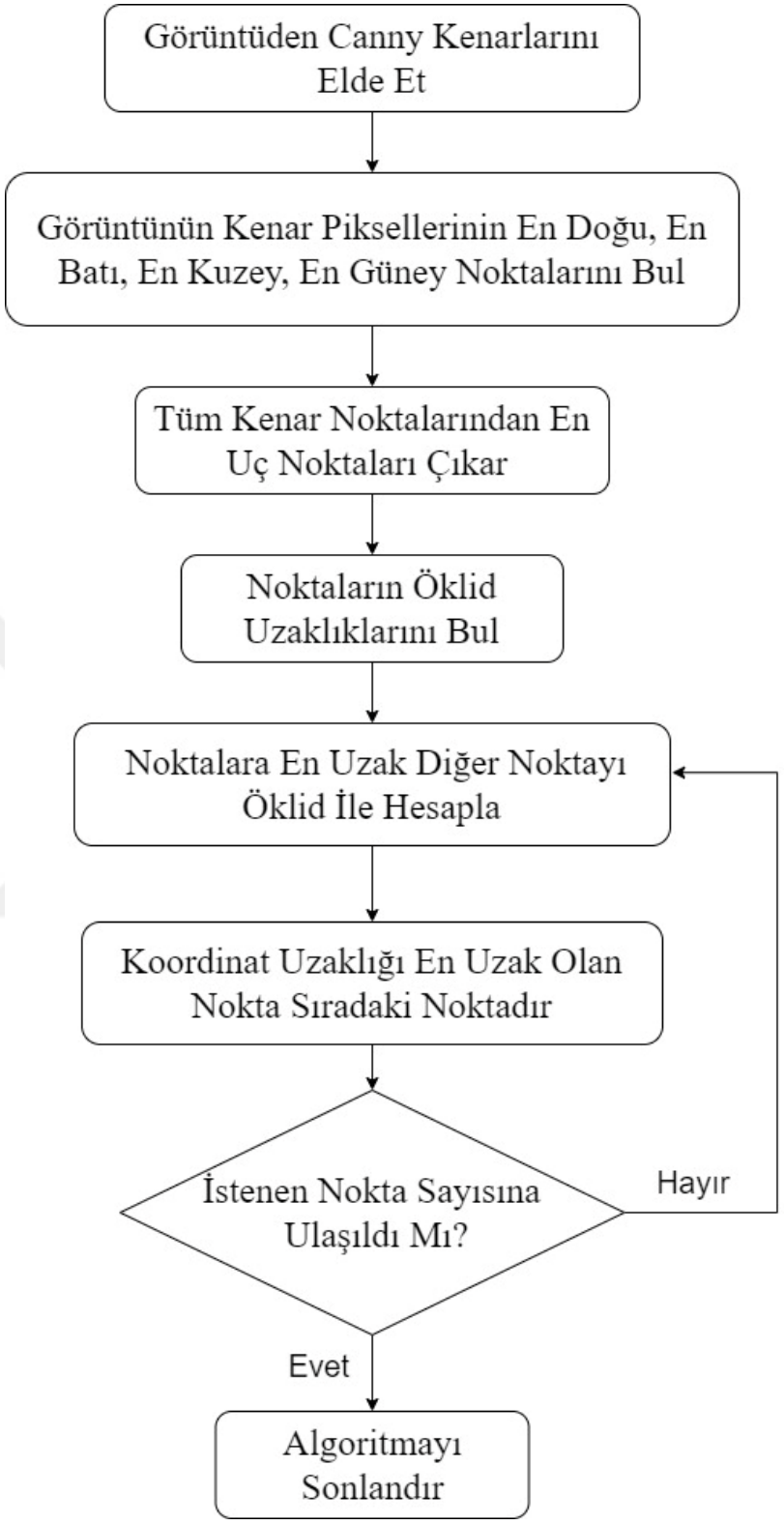
Tüm noktalar bir dizi içerisinde tutulmakta ve sırayla en uzak nokta bulunduğunda sıradaki noktanın yerine geçmektedir. En dış noktalardan sonra 5. nokta için algoritma

çalıştırıldığında bulunan koordinat değeri 5. sıraya gelmekte, 5.sırada yer alan koordinat değeri de en uzak bulunan koordinat değerindeki sıraya geçmektedir. Döngü indirgenecek nokta sayısına kadar devam etmektedir (Sonugür vd. 2022).

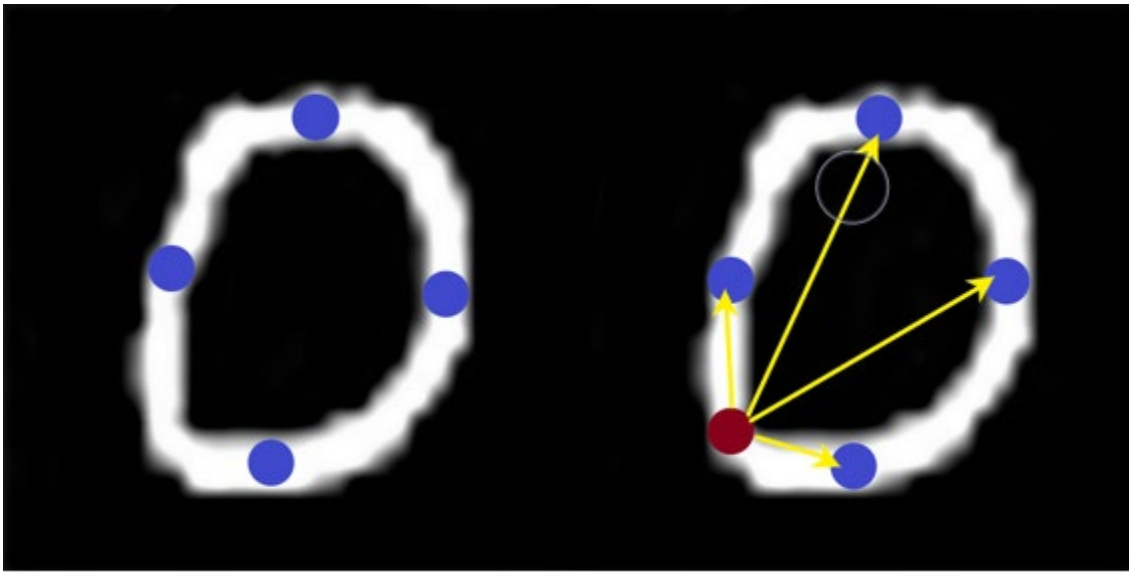
Optimal algoritmanın sözde kodu ve Şekil 3.4'da algoritmanın akış diyagramı verilmektedir.

Optimal algoritmanın sözde kodu;

1. Görüntü okunur ve Canny kenarları bul.
2. Kenarlar noktalarının koordinat değerleri diziye aktar.
3. En doğu, en batı, en kuzey ve en güney koordinatlardaki noktaları belirle, dizide ilk 4 satıra sabitle.
4. Tüm noktalardan bu noktalara en uzak noktayı Öklid ile bul.
5. Noktayı sıralı şekilde dizi içerisinde konumlandır.
6. İstenen nokta sayısına ulaşıldı mı?
    1. Ulaşılmadı ise 4.adıma git.
    2. Ulaşıldı ise 7.adıma git.
7. Nokta sayısına ulaşılınca algoritmayı durdur.




**Şekil 3.4** Optimal algoritma akış diyagramı.



**Şekil 3.5** Optimal algoritmadan elde edilen en dış noktalar ve onlara uzak sıralı diğer nokta.

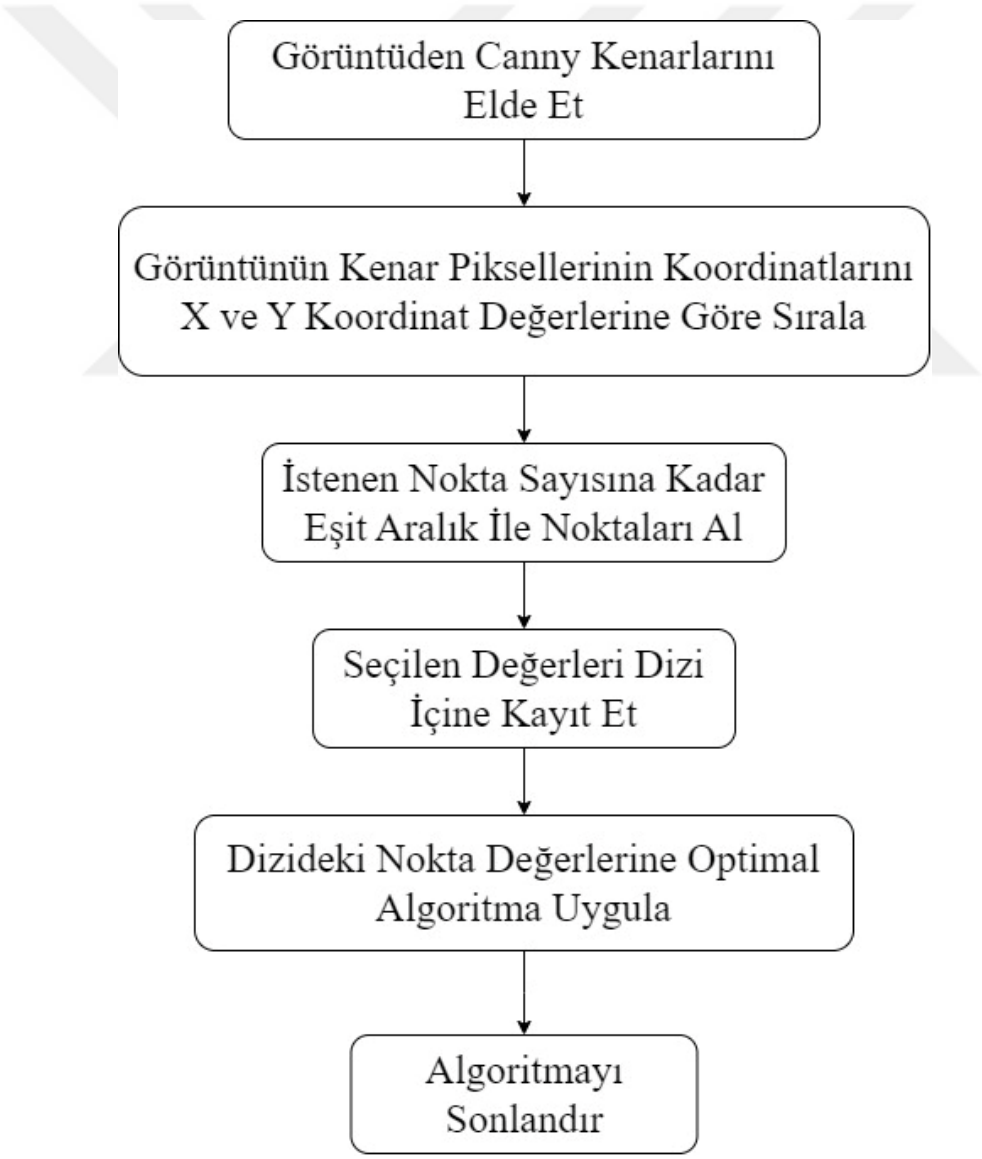
### 3.5 Sub-Optimal Algoritma

Bu algoritmada Canny kenar algoritmasından elde edilen kenar çıktı görselleri baz alınarak algoritma geliştirilmiştir. Kenar çıktısındaki pikseller belirlenen sayıya indirgenmektedir. Piksellerin koordinatları ele alınarak koordinatlar öncelikle x koordinatına göre sıralanmaktadır. Sıralanan koordinatlar indirgenecek piksel sayısınca eşit aralıklar ile seçilmektedir. Ardından kenar çıktı piksel koordinatları y koordinatlarına göre sıralanmaktadır ve indirgenecek piksel sayısınca eşit aralıklar seçilmektedir. Örneğin, ele alınan görüntü kenar çıktı piksel sayısı 100 ve indirgenecek piksel sayısı 20 olduğunda sıralanan koordinatlara göre her 5 pikselden 1'i, indirgenecek piksel sayısı 10 ise her 10 pikselden 1'i seçilmektedir. Seçilen piksellerin koordinat değerleri bir dizi içerisine kayıt edilmekte ve piksellere optimal algoritma yöntemi uygulanmaktadır (Sonugür vd. 2022).

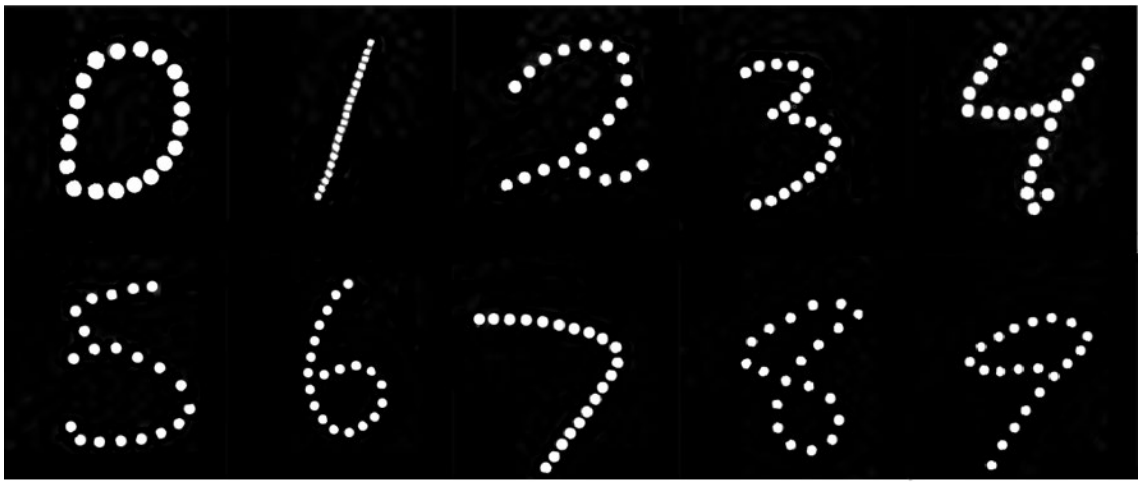
Optimal algoritma, temeli Öklid uzaklığına dayanmaktadır. Optimal algoritmada seçilen piksellerden öncelikle en doğu, en batı, en kuzey ve en güneyde bulunan noktalar belirlenmektedir. Bu noktalara en uzak konumda bulunan diğer nokta ise sıradaki noktayı temsil etmektedir. İndirgenecek piksel sayısı kadarınca bu algoritma devam etmektedir. Sub-optimal algoritmasının sözde kodu ve Şekil 3.6'de algoritmanın akış diyagramı verilmektedir.

Sub-Optimal algoritma sözde kodu;

1. Görüntü okunur ve Canny kenarları bul.
2. Görüntünün kenar piksellerinin koordinatlarını x ve y koordinat değerlerine göre sırala,
3. 20 nokta isteniyor ise toplam nokta sayısını eşit aralıklar ile bölecek şekilde 20 noktaya indirge,
4. Şeçilen noktaları değerleri ile diziye aktar.
5. Dizideki nokta değerlerine optimal algoritma uygula.
6. Algoritmayı sonlandır.




**Şekil 3.6** Sub-Optimal algoritma akış diyagramı.



**Şekil 3.7** MNIST veri seti Sub-Optimal belirlenen noktaların temsili gösterimi.

### 3.6 Özgün Algoritma

Özgün algoritmada öncelikle görsellerin canny kenar görselleri ele alınmaktadır. Sub-optimal algoritma ile noktalar belirlenmekte ve optimal algoritma kullanılarak kenarlardaki nokta sayısı azaltılmaktadır. Azalan kenar nokta sayıları ile daha hızlı işlem hedeflenmektedir. Noktaların ağırlık merkezi hesaplanmaktadır. Ağırlık merkezi Denklem 3.3'de yer almaktadır. Ağırlık merkezi bulunan kenar çıktı görselinden sub-optimal algoritma ile belirlenen noktaların hangi açılara denk geldiği bulunur.

Özgün algoritmada belirlenen noktaların ağırlık merkezine olan uzaklıkları hesaplanmaktadır. Bu hesaplama Öklid ile gerçekleştirilmektedir. Belirlenen tüm noktaların uzaklıklarının ortalaması bulunmakta ve bu ortalama ile ağırlık merkezinden baz alınarak çizilecek dairenin yarıçap uzunluğu belirlenmektedir. Ardından noktaların yaptıkları açılar bulunmaktadır ve noktaların merkeze olan açılarının ortalaması hesaplanmaktadır. Ortalama açı değeri Denklem 3.4'de yer almaktadır. Denklemde $x_{m,}$ ağırlık merkezinin x koordinatı değerini, $y_m$ ise ağırlık merkezinin y koordinatı değerini vermektedir. Şekil 3.9, Şekil 3.10 ve Şekil 3.11'de özgün algoritmada belirlenerek hesaplanan uzaklık ve açılar MNIST veri seti örneklerince temsili olarak gösterilmektedir (Çakı ve Gökçe 2023).

Ele alınan her nokta için hesaplanan ortalama uzaklıktan o noktanın ağırlık merkezine

olan uzaklığının farkının yüzdelik oranı ve hesaplanan ortalama açıdan o noktanın ağırlık merkezine olan açısının farkının yüzdelik oranı ele alınmaktadır. Denklem 3.5 ve Denklem 3.6'da hesaplamalar yer almaktadır. Denklem 3.5'de $r_m$, sub-optimal algoritma belirlenen noktaların ağırlık merkezine olan ortalama uzaklığının değerini ve Denklem 3.6'da Θm, Sub-optimal algoritma ile belirlenen noktaların ağırlık merkezine olan vektörünün x koordinatınca yaptığı açının ortalama değerini vermektedir. Şekil 3.8'de özgün algoritmanın akış diyagramı verilmektedir.

$$r_i = \sqrt{(x_i - x_m)^2 + (y_i - y_m)^2} \tag{3.3}$$

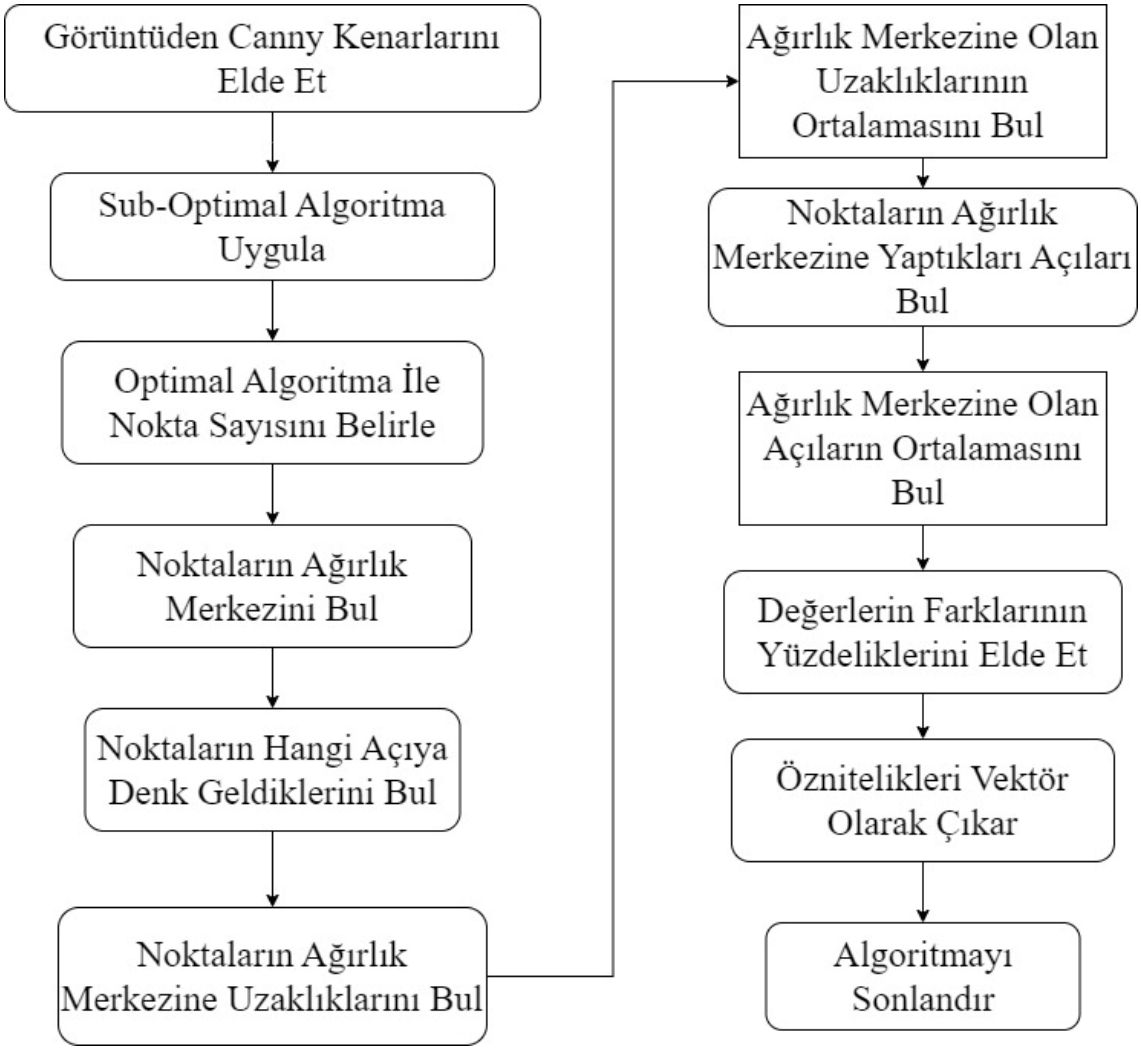
$$\theta_i = arctan(\frac{y_i - y_m}{x_i - x_m}) \tag{3.4}$$

$$\%\Delta r_i = \frac{r_i - r_m}{r_m} x100 \tag{3.5}$$

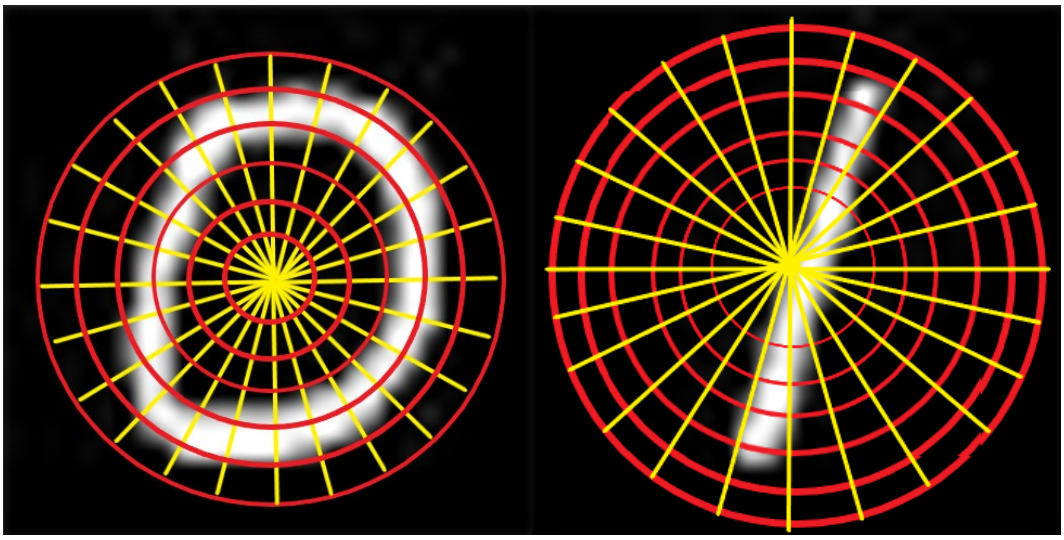
$$\%\Delta \theta_i = \frac{\theta_i - \theta_m}{\theta_m} x100 \tag{3.6}$$

Özgün algoritma sözde kodu;

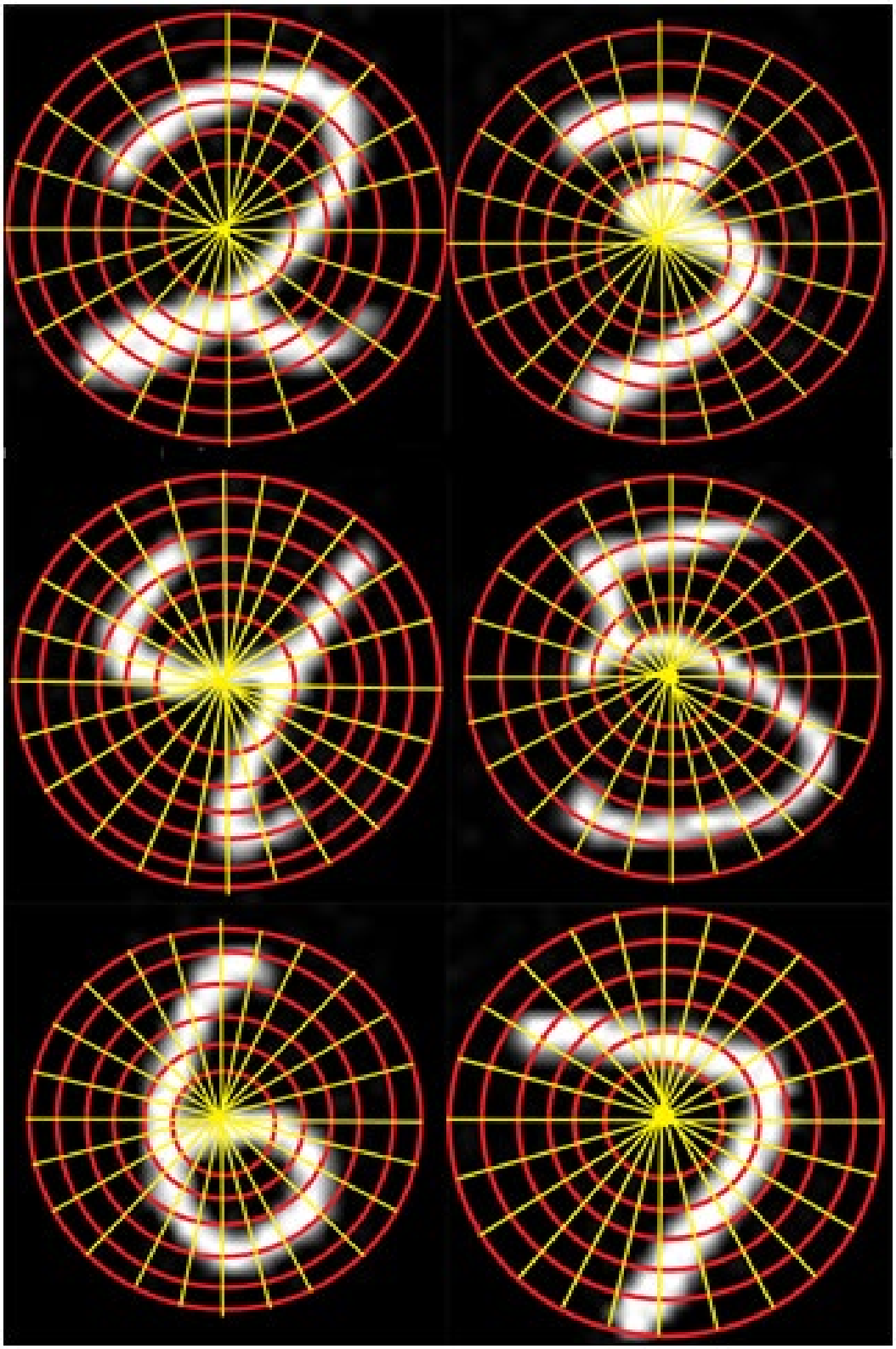
1. Görüntü okunur ve Canny kenarları bul.
2. Görüntüye Sub-Optimal algoritma uygula.
3. Optimal algoritma ile noktaları bul.
4. Noktaların ağırlık merkezini hesapla.
5. Noktaların ağırlık merkezi ile yaptıkları açıyı hesapla.
6. Noktaların ağırlık merkezine uzaklıklarını Öklid ile bul.
7. Uzaklıkların ortalamasını hesapla.
8. Noktaların ağırlık merkezine yaptıkları açıları bul.
9. Açıların ortalamasını hesapla.
10. Değerlerin farklarının yüzdeliklerini hesapla.
11. Öznitelikleri vektörel olarak çıkar.
12. İstenen nokta sayısına kadar döngüye devam et.
13. Nokta sayısına ulaşılınca algoritmayı durdur.



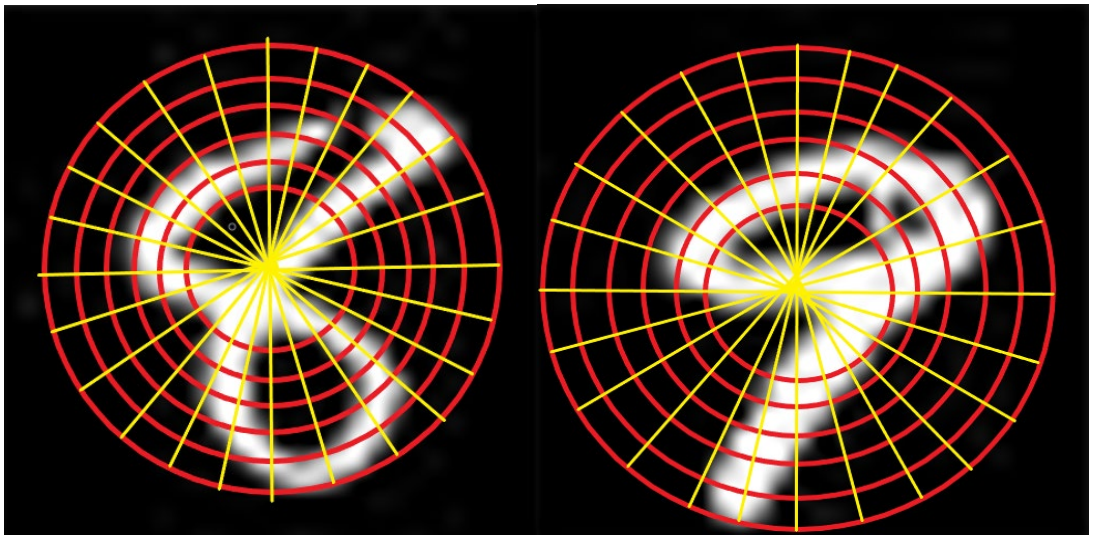

**Şekil 3.8** Özgün algoritma akış diyagramı.



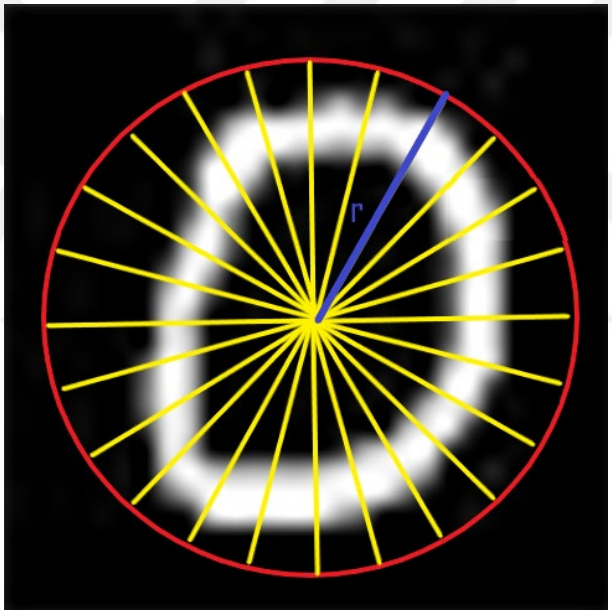
**Şekil 3.9** MNIST veri seti temsili özgün algoritma gösterimleri.



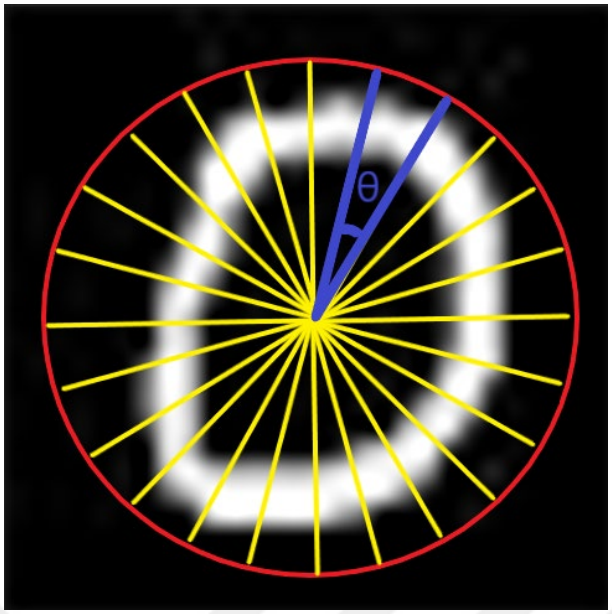
**Şekil 3.9 (Devam)** MNIST veri seti temsili özgün algoritma gösterimleri.



**Şekil 3.9 (Devam)** MNIST veri seti temsili özgün algoritma gösterimleri.




**Şekil 3.10** Ağırlık merkezine olan uzaklık temsili gösterimi.




**Şekil 3.11** Ağırlık merkezine olan vektörün X koordinatı açısı.

## 3.7 Yapay Zeka ile Eğitim

Çalışmalar sonucu elde edilen özniteliklerin tanınması için yapay zeka sistemlerinin bir birimi olan yapay sinir ağları kullanılmaktadır. Özniteliklerin vektörel olarak ağa girdi verilmesi sonucunda sınıflandırma işlemi yapılmaktadır.

### 3.7.1 Yapay Zeka

Yapay zeka; makineye veya bilgisayara akıl yürütme, anlam çıkartma, genelleme yapabilme gibi bir çok insani yetileri kazandırabilmektedir. İnsan beyni, yorumlama gibi olayların yanı sıra sayısal işlemleri de kısa zamanda gerçekleştirebilmektedir. Bilgisayarlar ise daha zorlu ve karmaşık sayısal işlemleri daha hızlı çözümleyebilmekte iken karar verme, öğrenme, yorumlama gibi niteliklerden yoksundurlar. Yani insan beyninin üstün olmasındaki asıl neden problemi deneyimleyebilme becerisi olmaktadır. Yapay zeka kavramı da tam bu deneyimleme alanında kullanılmaktadır. İnsanın karar verme yeteneklerini, davranışlarını taklit ederek bunları modellemesi işlemidir.

Yapay Zekânın konularından olan derin öğrenme yapay sinir ağlarını kullanmakta ve bu algoritmalar ile oluşturulan yapay zeka modellerinin en üstünü kapsamaktadır. Eğitilmiş

ağ kullanılarak girdiler yorumlanabilmekte ve bir sonuç elde edilmektedir. Elde edilen sonuçlar ile ön görülen sonuçların sapma değeri yapay sinir ağının çalışma prensibine optimize edilmesidir (Aksoylu, 2021). Derin öğrenmede girdi olarak verilen görüntülerin, çıkışta sınıflandırılmış olması istenmektedir. Sınıflandırma işlemi için Konvolüsyonel sinir ağları (Convolution Neural Network-CNN) kullanılmaktadır.

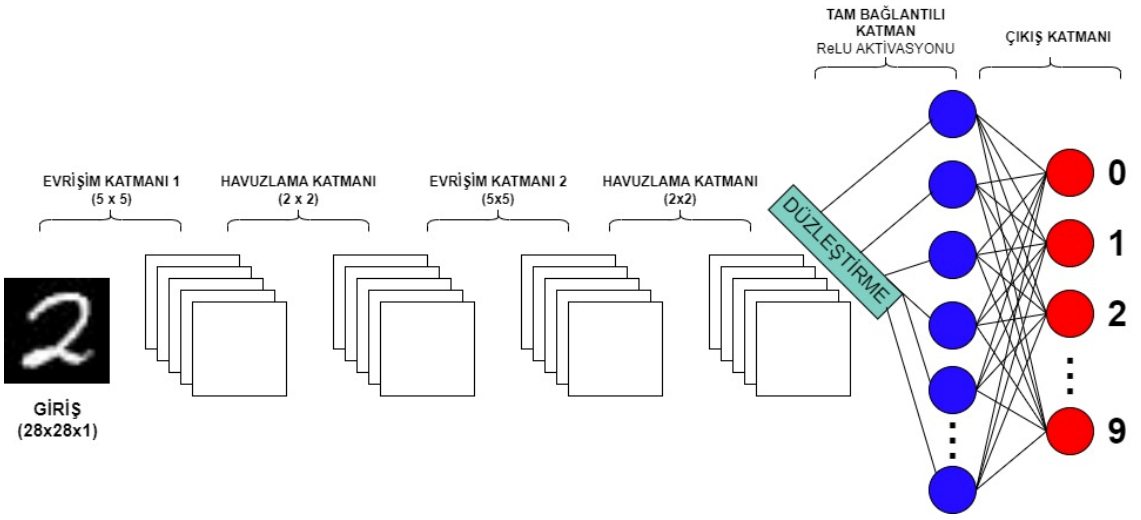
Özniteliklerin çıkarılmasının ardından modelin eğitimi için Intel(R) Core(TM) i7-7500U CPU, 2.90 GHz, 8,00 GB Ram, Intel(R) HD Graphics 620 özelliklerini içeren bilgisayar kullanılmıştır. Özniteliklerin çıkarılması Python dilinde programlanmıştır.

### 3.7.2 Evrişimli Sinir Ağları (Convolution Neural Network- CNN)

Evrişimli sinir ağları, bir diğer adı ile Konvolüsyonel sinir ağları, veri işlemek için kullanılmakta olan bir sinir ağıdır ve çok katmanlı yapıda sahiptirler. Nesne tanıma, sınıflandırma, görüntü işleme, biyomedikal teknolojileri, doğal dil işleme gibi alanlarda yaygın olarak kullanılmakta ve görüntü işleme alanında en başarılı sonuçları vermektedir (Tüfekçi vd. 2019). Diğer sinir ağlarından farklı olarak giriş verisinden öznitelik çıkartma işlemini kendisi yapmaktadır.

Evrişimli sinir ağları (CNN), genellikle evrişim (convolution), aktivasyon (activation), havuzlama (pooling) ve tam bağlantılı (fully connected) katmanlardan oluşan bir yapay sinir ağı mimarisidir. Evrişim katmanları, giriş verisine bir dizi filtre uygulayarak özellik haritalarını çıkaran katmanlardır. Her filtre, belirli bir özelliği algılamak için kullanılmaktadır. Evrişim katmanları, görüntünün özelliklerini vurgulamakta ve farklı ölçeklerde desenleri algılamaya yardımcı olmaktadır. Aktivasyon katmanları, evrişim katmanlarının çıktılarını sıkıştırmaktadır. Bu, sinir ağının daha karmaşık ilişkileri öğrenebilmesini sağlamaktadır. En yaygın kullanılan aktivasyon fonksiyonu ReLU (Rectified Linear Unit)'dır. ReLU, negatif değerleri sıfır olarak döndürerek pozitif değerleri korumaktadır. Aktivasyon fonksiyonları, evrişim katmanlarının çıktılarını sınırlamakta ve ağın daha genel özellikler öğrenmesini sağlamaktadır. Havuzlama katmanları, özellik haritalarını küçültmek için kullanılmaktadır. Havuzlama işlemi, özellik haritalarındaki bölgelerin özet istatistiklerini (genellikle maksimum değer)

alarak boyutlarını azaltmaktadır. Bu, özelliklerin küçük yer değişikliklerine karşı daha dirençli olmasını sağlamakta ve hesaplama yükünü azaltmaktadır. Maksimum havuzlama en yaygın kullanılan havuzlama yöntemlerinden olup çalışmamızda da bu yöntem kullanılmaktadır. Tam bağlantılı katmanlar, özellik haritalarının sınıflandırma veya tahmin işlemi için kullanıldığı katmanlardır. Özellik haritaları, düzleştirme işlemiyle tek bir vektöre dönüştürülmekte ve ardından tam bağlantılı katmanlara verilmektedir. Bu katmanlar, evrişimli sinir ağları (CNN) mimarisini oluşturan önemli bileşenlerdir. CNN, genellikle bu katmanların farklı sayıda ve sıralamada tekrarlandığı bir yapıya sahiptirler. Katman sayısı ve karmaşıklık, öğrenilecek görevin karmaşıklığına ve kullanılabilir kaynaklara bağlı olarak değişebilmektedir (Li vd. 2022, Eker ve Duru 2021, O'Shea ve Nash 2015).

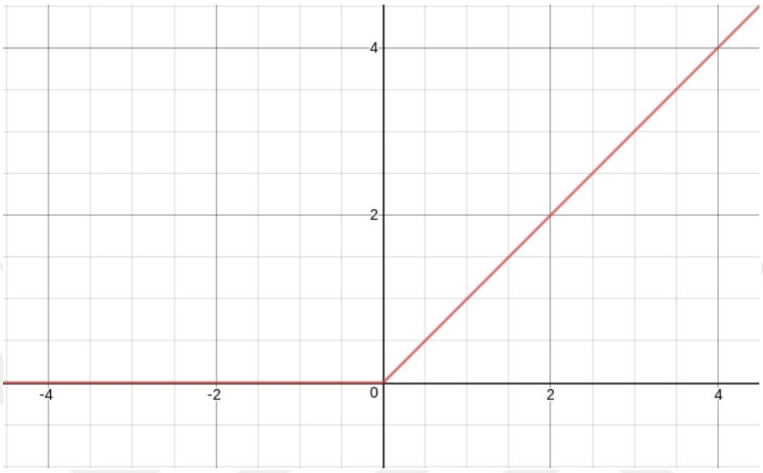



**Şekil 3.12** Evrişimli sinir ağı mimarisi (Akbulut ve Aslantaş 2023).

### 3.7.3 ReLU Aktivasyon Fonksiyonu

ReLU (Rectified Linear Unit), bir başka adı ile doğrultulmuş lineer birim fonksiyonu olarak da bilinmektedir. Yaygın olarak kullanılmakta olan doğrusal aktivasyon fonksiyonudur. Matematiksel olarak formülü Denklem 3.7'de verilmektedir. Matematiksel formülde nöronsal çıkış x, sıfırdan büyük ise sonuç x olmakta, nöronsal çıkış x, sıfırdan küçük ise sonuç 0 olmaktadır. Resim 3.3'de ReLU aktivasyon fonksiyonunun iki boyutlu koordinat sisteminde gösterimi bulunmaktadır (Agarap 2018).

$$ReLU(x) = \max(0, x) = \begin{cases} x, & eğer\ x > 0 \\ 0, & eğer\ x \leq 0 \end{cases} \tag{3.7}$$




**Resim 3.3** ReLU aktivasyon fonksiyonunun koordinat sisteminde gösterimi.

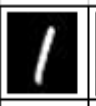
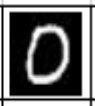
# 4. BULGULAR

Yapılan çalışmaların kodlarının çalışması sonucunda verilerden 160'ar adet öznitelik çıkarılmakta ve ilk gelen 10 verinin öznitelikleri Şekil 4.1'de verilmektedir. Belirtilen öznitelikleri, her rakam için sütunlarda yer alan değerler göstermektedir. Örneğin 1.satır ve 1. sütunda yer alan değer 1. sütundaki rakamın özniteliğini temsil etmektedir. Elde edilen öznitelikler koyu renk ile belirtilmiş ve 0 değerleri verideki noktanın öznitelik değeri içermediğini göstermektedir.

Örnek sayılardaki öznitelikler incelendiğinde çoğunlukla verilerin özniteliklerinin aynı öznitelik sıralamasında olduğu görülmektedir. Bu da çıkarılan özniteliklerin eşit olarak alındığını göstermektedir.

| Öznitelikler / Sayılar |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 4 | 0 | 0 | 13 | 9 | 0 | 17 | 17 | 17 | 16 |
| 3 | 6 | 8 | 10 | 2 | 10 | 11 | 3 | 3 | 4 | 7 |
| 4 | 0 | 4 | 4 | 1 | 0 | 2 | 2 | 2 | 0 | 1 |
| 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 2 | 5 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 12 | 19 | 7 | 3 | 11 | 5 | 3 | 3 | 10 | 6 |
| 13 | 12 | 4 | 1 | 14 | 8 | 3 | 14 | 14 | 11 | 13 |
| 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 23 | 2 | 1 | 0 | 3 | 0 | 0 | 3 | 3 | 4 | 3 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Şekil 4.1** Örnek özgün öznitelik verileri.

| Öznitelikler / Sayılar |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 35 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 39 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 43 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 44 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 45 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 46 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 47 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 48 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 49 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 50 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 51 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 52 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 53 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 54 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 55 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 57 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 58 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 59 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 61 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 62 | 1 | 0 | 1 | 9 | 3 | 0 | 1 | 1 | 5 | 4 |
| 63 | 1 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |
| 64 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 65 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 66 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 67 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 69 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 70 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 71 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 72 | 13 | 3 | 0 | 19 | 9 | 1 | 11 | 11 | 8 | 9 |
| 73 | 15 | 24 | 10 | 11 | 13 | 10 | 11 | 11 | 5 | 17 |
| 74 | 0 | 4 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 75 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 76 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 77 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 78 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 79 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 80 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Şekil 4.1 (Devam)** Örnek özgün öznitelik verileri.

| Öznitelikler / Sayılar |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 81 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 |
| 82 | 0 | 17 | 10 | 4 | 4 | 10 | 3 | 3 | 7 | 0 |
| 83 | 9 | 0 | 0 | 18 | 14 | 0 | 15 | 15 | 17 | 18 |
| 84 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 85 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 86 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 87 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 88 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 89 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 91 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 92 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 93 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 94 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 95 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 96 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 97 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 98 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 99 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 102 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 103 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 104 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 105 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 106 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 107 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 108 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 109 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 112 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 113 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 114 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 115 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 116 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 117 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 118 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 119 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 120 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 121 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 122 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 123 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 124 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 125 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 126 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 127 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 128 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 129 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 130 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Şekil 4.1 (Devam)** Örnek özgün öznitelik verileri.

| Öznitelikler / Sayılar |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| 131 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 132 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 133 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 134 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 135 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 136 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 137 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 138 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 139 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 140 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 141 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 142 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 143 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 144 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 145 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 146 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 147 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 148 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 149 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 151 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 152 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 153 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 154 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 155 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 156 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 157 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 158 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 159 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Şekil 4.1 (Devam)** Örnek özgün öznitelik verileri.

## 5. TARTIŞMA ve SONUÇ

Önerdiğimiz algoritmada nesne tanıma için hızlı bir şekil tanımlayıcısı tasarlanmıştır. Çalışmada MNIST veri kümesi kullanılmakta ve önerilen algoritma bu verilerde denenmektedir. Çalışma kapsamında uzaysal düzlemdeki noktalar belirlenmekte ve işlenebilmesi için kayıda alınmaktadır. Ağırlık merkezi hesaplanan noktaların, merkeze olan uzaklık ve açılarının baz alınarak elde edilen verilerin sınıflandırma eğitimi gerçekleştirilmiştir.

Önerdiğimiz algoritmamızdan elde ettiğimiz verilerin ReLU aktivasyon fonksiyonu, 1024 nöron sayısı ve 25 epoch ile %69.91 başarı elde edilmiştir. Çizelge 5.1'de ReLU aktivasyon fonksiyonunun farklı nöron sayı değerleri sonucunda elde edilen eğitim sonuçları verilmiştir. Çizelge 5.2'de farklı sayıları ve 1024 nöron sayısı ile elde edilen eğitim sonuçları verilmiştir.

**Çizelge 5.1** Farklı nöron sayısı eğitim sonuçları

| Aktivasyon Fonksiyonu Nöron Sayısı | Test Doğruluğu (Accuracy) | F1 Puanı | Toplam Kayıp (Loss) |
|---|---|---|---|
| 128 | 67.39 | 0.674 | 0.968 |
| 256 | 67.9 | 0.679 | 0.966 |
| 512 | 69.41 | 0.694 | 0.934 |
| 1024 | 69.91 | 0.699 | 0.942 |
| 2048 | 68.61 | 0.686 | 0.988 |

**Çizelge 5.2** Farklı nöron sayısı eğitim sonuçları

| Epoch Sayısı | Test Doğruluğu (Accuracy) | F1 Puanı | Toplam Kayıp (Loss) |
|---|---|---|---|
| 25 | 69.91 | 0.699 | 0.942 |
| 50 | 68.98 | 0.69 | 1.08 |
| 75 | 67.75 | 0.678 | 1.318 |
| 100 | 66.98 | 0.67 | 1.488 |

## 6. KAYNAKLAR


Agarap A F, 2018, Deep Learning Using Rectified Linear Units (Relu), Arxiv Preprint Arxiv:1803.08375.

Akbulut H, Aslantaş V, 2023, Evrişimli Sinir Ağı Kullanarak Çoklu Pozlamalı Görüntüyü Birleştirme, Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 38 (3), 1439–1452.

Aksoylu M Ü, 2021, Projelerle Yapay Zeka Ve Bilgisayarlı Görü, Kodlab,1–28, İstanbul.

Andayani U, Sumantri I B, Arisandy B, 2020, Classification of Zingiber Plants Based on Stomate Microscopic Images Using Probabilistic Neural Network (PNN) Algorithm, In Journal of Physics: Conference Series, Vol. 1566, No. 1, p. 012122, IOP Publishing.

Arslan K, 2020, Eğitimde Yapay Zeka ve Uygulamaları, Batı Anadolu Eğitim Bilimleri Dergisi, 11 (1), 71–88.

Belongie S, Malik J, Puzicha J, 2000, Shape Context: A New Descriptor For Shape Matching And Object Recognition, Advances İn Neural İnformation Processing Systems, 13.

Bettayeb M, Hassan E, Mohammad B, Saleh H, 2023, Spatialhd: Spatial Transformer Fused with Hyperdimensional Computing for AI Applications, In 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), 1-5, IEEE.

Bober M, 2001, MPEG-7 Visual Shape Descriptors. IEEE Transactions On Circuits And Systems For Video Technology, 11(6), 716–719.

Canny J, 1986, A Computational Approach To Edge Detection, IEEE Trans, Pattern Anal, Machine Intell, vol. PAMI-8, 679–698.

Çakı E E, Gökçe C O, 2023, A Novel Shape Descriptor For Object Recognition, International Journal Of Computational And Experimental Science And Engineering, 9(1), 1–5.

Eker A G, Duru N, 2021, Medikal Görüntü İşlemede Derin Öğrenme Uygulamaları, Acta Infologica , 5 (2) , 459–474. DOI: 10.26650/acin.927561.

Fang Y, Xie J, Dai G, Wang M, Zhu F, Xu T, Wong E, 2015, 3d Deep Shape Descriptor. In Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition, 2319–2328.

Grappolini E W, Subramoney A, 2023, Beyond Weights: Deep Learning İn Spiking Neural Networks With Pure Synaptic-Delay Training, Arxiv Preprint Arxiv:2306.06237.

Güzel B, Okatan E, 2022, Tarım Ve Yapay Zekâ, Yapay Zekânın Değiştirdiği Dinamikler, 199.

Hoşgör H, Güngördü H, 2022, Sağlıkta Yapay Zekanın Kullanım Alanları Üzerine Nitel Bir Araştırma, Avrupa Bilim ve Teknoloji Dergisi, (35), 395–407. DOI: 10.31590/ejosat.1052614.

Hussain M, Wahab A W A, Idris Y I B, Ho A T, Jung K H, 2018, Image Steganography İn Spatial Domain: A Survey. Signal Processing: Image Communication, 65, 46–66.

Kim W Y, Kim Y S, 2000, A Region-Based Shape Descriptor Using Zernike Moments, Signal Processing: Image Communication, 16(1-2), 95–102.

Kunaver M, Tasic J F, 2005, Image Feature Extraction-An Overview, In EUROCON 2005-The International Conference On Computer As A Tool, Vol. 1, 183–186, IEEE.

Li Z, Liu F, Yang W, Peng S, Zhou J, 2021, A Survey Of Convolutional Neural Networks: Analysis, Applications, And Prospects, IEEE Transactions On Neural Networks And Learning Systems.

Marr D, Hildreth E, “Theory of edge detection,” Proc. R. Soc. Lond. A, Math. Phys. Sci., vol. B 207, pp. 187–217, 1980.

Medjahed S A, 2015, A Comparative Study Of Feature Extraction Methods İn İmages Classification. International Journal Of İmage, Graphics And Signal Processing, 7(3), 16.

O'Shea K, Nash R, 2015, An Introduction To Convolutional Neural Networks, Arxiv Preprint Arxiv:1511.08458.

Ponce H, Moya-Albor E, Brieva J, 2023, Towards The Distributed Wound Treatment Optimization Method For Training CNN Models: Analysis On The MNIST Dataset, In 2023 IEEE 15th International Symposium On Autonomous Decentralized System (ISADS), 1–6, IEEE.

Shao H, Ma E, Zhu M, Deng X, Zhai S, 2023, MNIST Handwritten Digit Classification Based On Convolutional Neural Network With Hyperparameter Optimization, Intelligent Automation & Soft Computing, 36(3).

Shi Y, Xu M, Yuan S, Fang Y, 2020, Unsupervised Deep Shape Descriptor With Point Distribution Learning, In Proceedings Of The IEEE/CVF Conference On Computer Vision And Pattern Recognition, 9353–9362.

Siddhartha P V, 2023, Digit Recognition Of MNIST Handwritten Using Convolutional Neural Networks (CNN), In 2023 International Conference On Intelligent Systems For Communication, Iot And Security (Icıscoıs), 328–332, IEEE.

Sohel F A, Karmakar G C, Dooley L S, 2005, A Generic Shape Descriptor Using Bezier Curves, In International Conference On Information Technology: Coding And Computing (ITCC'05)-Volume II, 95–100, IEEE.

Sonugur G, Çakı E E, Akan S A, Gökçe C O, 2022, Gerçek Zamanlı İnsan Davranışı Anlamaya Doğru: Optimal-Altı Bir Şekil Tanımlayıcı, Afyon Kocatepe Üniversitesi Fen Ve Mühendislik Bilimleri Dergisi, 22(4), 769–777.

Sparks R, Madabhushi A, 2013, Explicit Shape Descriptors: Novel Morphologic Features For Histopathology Classification, Medical İmage Analysis, 17(8), 997–1009.

Susan S, Agrawal P, Mittal M, Bansal S, 2019, New Shape Descriptor In The Context Of Edge Continuity, CAAI Transactions On Intelligence Technology, 4(2), 101–109.

Şeker E, 2020, Yapay Zeka Tekniklerinin/Uygulamalarının Siber Savunmada Kullanımı, Uluslararası Bilgi Güvenliği Mühendisliği Dergisi, 6(2), 108–115.

Taati B, Greenspan M, 2011, Local Shape Descriptor Selection For Object Recognition İn Range Data, Computer Vision And Image Understanding, 115(5), 681–694.

Tombari F, Salti S, Di Stefano L, 2011, A Combined Texture-Shape Descriptor For Enhanced 3D Feature Matching, In 2011 18th IEEE İnternational Conference On İmage Processing, 809–812, IEEE.

Tüfekçi M, Karpat F, 2019, Derin Öğrenme Mimarilerinden Konvolüsyonel Sinir Ağları (CNN) Üzerinde Görüntü İşleme-Sınıflandırma Kabiliyetininin Arttırılmasına Yönelik Yapılan Çalışmaların İncelenmesi, In International Conference on Human-Computer Interaction, Optimization and Robotic Applications, 28–31.

Verma O P, Sharma R, Kumar M, Agrawal N, 2013, An Optimal Edge Detection Using Gravitational Search Algorithm, Lecture Notes On Software Engineering, 1(2), 148.

Vranic D V, 2005, DESIRE: A Composite 3D-Shape Descriptor, In 2005 IEEE International Conference On Multimedia And Expo, 4, IEEE.

Wan M, Chen X, Zhan T, Yang G, Tan H, Zheng H, 2023, Low-Rank 2D Local Discriminant Graph Embedding For Robust İmage Feature Extraction, Pattern Recognition, 133, 109034.

Wu M, Zhang Z, 2010, Handwritten Digit Classification Using The Mnist Data Set, Course Project CSE802: Pattern Classification & Analysis, 366.

Xie J, Dai G, Zhu F, Wong E K, Fang Y, 2016, Deepshape: Deep-Learned Shape Descriptor For 3d Shape Retrieval, IEEE Transactions On Pattern Analysis And Machine İntelligence, 39(7), 1335–1345.

Xie J, Fang Y, Zhu F, Wong E, 2015, Deepshape: Deep Learned Shape Descriptor For 3d Shape Matching And Retrieval, In Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition, 1275–1283.

Xu H, Yang J, Yuan J, 2016, Invariant Multi-Scale Shape Descriptor For Object Matching And Recognition, In 2016 IEEE International Conference On Image Processing (ICIP), 644–648, IEEE.

Yang J, Zhang Q, Xiao Y, Cao Z, 2017, TOLDI: An Effective And Robust Approach For 3D Local Shape Description, Pattern Recognition, 65, 175–187.

Zhang D, Lu G, 2003, Evaluation Of MPEG-7 Shape Descriptors Against Other Shape Descriptors, Multimedia Systems, 9, 15–30.

Zhong Y, 2009, Intrinsic Shape Signatures: A Shape Descriptor For 3D Object Recognition, In 2009 IEEE 12th İnternational Conference On Computer Vision Workshops, ICCV Workshops, 689–696, IEEE.

## EKLER

**EK 1.** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct  3 12:46:44 2022

@author: Elif Ebru ÇAKI
"""
import cv2
import numpy as np
import time
import imutils
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

def subOptimalv3(inputImage):
    img_blur = cv2.GaussianBlur(inputImage, (3,3), 0)
    edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny
Edge Detection
    numberOfEdgePoints = np.count_nonzero(edges)
    tempEdgeCoordinates = np.nonzero(edges)
    edgeCoordinates = np.zeros((numberOfEdgePoints, 2))
    for tempIndex in range(0,numberOfEdgePoints):
        edgeCoordinates[tempIndex,0] = tempEdgeCoordinates[0][tempIndex]
        edgeCoordinates[tempIndex,1] = tempEdgeCoordinates[1][tempIndex]

    edgeCoordinates = edgeCoordinates[np.argsort(edgeCoordinates[:,0])]
    edgeCoordinates = edgeCoordinates[np.argsort(edgeCoordinates[:,1])]
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
    edgeCoordinatesv2 = np.zeros((700,2))
    for tempIndex in range(0,numberOfEdgePoints):
        edgeCoordinatesv2[tempIndex,:]=edgeCoordinates[tempIndex,:]

    return edgeCoordinatesv2

def subOptimalv2(inputImage, numberOfSuboptimalPoints):
    img_blur = cv2.GaussianBlur(inputImage, (3,3), 0)
    edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny
Edge Detection
    numberOfEdgePoints = np.count_nonzero(edges)
    tempPadNumber = np.floor(numberOfEdgePoints / numberOfSuboptimalPoints)
    tempEdgeCoordinates = np.nonzero(edges)
    edgeCoordinates = np.zeros((numberOfEdgePoints, 2))
    for tempIndex in range(0,numberOfEdgePoints):
        edgeCoordinates[tempIndex,0] = tempEdgeCoordinates[0][tempIndex]
        edgeCoordinates[tempIndex,1] = tempEdgeCoordinates[1][tempIndex]

    edgeCoordinates = edgeCoordinates[np.argsort(edgeCoordinates[:,0])]
    edgeCoordinates = edgeCoordinates[np.argsort(edgeCoordinates[:,1])]
    #sorted_array = an_array[numpy.argsort(an_array[:, 1])]

    tempPointer = 0
    suboptimalEdgeCoordinates = np.zeros((numberOfSuboptimalPoints,2))
    for tempIndex in range(0, numberOfSuboptimalPoints):
        suboptimalEdgeCoordinates[tempIndex,:] = edgeCoordinates[int(tempPointer),:]
        tempPointer = tempPointer + tempPadNumber
    return suboptimalEdgeCoordinates

def novelFeature(subOptimalPoints):
    dimensionOfFeature = 2 + 2*subOptimalPoints.shape[0]
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
    #feature=np.zeros((dimensionOfFeature))
    feature=np.zeros((1402))
    totx=0
    toty=0
    numberOfPoints=int(np.count_nonzero(subOptimalPoints)/2)

    for tempIndex in range(0, numberOfPoints):
        totx=totx+subOptimalPoints[tempIndex,0]
        toty=toty+subOptimalPoints[tempIndex,1]

    centerX=totx/numberOfPoints
    centerY=toty/numberOfPoints

    difsFromCenter=np.zeros((numberOfPoints,2))

    for tempIndex in range(0, numberOfPoints):
        difsFromCenter[tempIndex,0]=subOptimalPoints[tempIndex,0]-centerX
        difsFromCenter[tempIndex,1]=subOptimalPoints[tempIndex,1]-centerY

    distsFromCenter=np.zeros((numberOfPoints))

    for tempIndex in range(0, numberOfPoints):

distsFromCenter[tempIndex]=np.sqrt(difsFromCenter[tempIndex,0]**2+difsFromCente
r[tempIndex,1]**2)

    anglesFromCenter=np.zeros((numberOfPoints))

    for tempIndex in range(0, numberOfPoints):

anglesFromCenter[tempIndex]=np.arctan2(difsFromCenter[tempIndex,1],difsFromCent
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
er[tempIndex,0])
    totDist = 0
    totAngle = 0
    for tempIndex in range(0, numberOfPoints):
        totDist=totDist+distsFromCenter[tempIndex]
        totAngle=totAngle+anglesFromCenter[tempIndex]

    averageDist=totDist/numberOfPoints
    if averageDist == 0:
        averageDist = 1e-6
    averageAngle=totAngle/numberOfPoints
    if averageAngle == 0:
        averageAngle = 1e-6

    #farkları
    distFeats=np.zeros((numberOfPoints))
    angleFeats=np.zeros((numberOfPoints))

    for tempIndex in range(0, numberOfPoints):
        distFeats[tempIndex]=(distsFromCenter[tempIndex]-averageDist)/averageDist
        angleFeats[tempIndex]=(anglesFromCenter[tempIndex]-
averageAngle)/averageAngle

    #novelFeaturesICCESEN202237=np.zeros((1+1+2*numberOfPoints))
    novelFeaturesICCESEN202237=np.zeros((1402))

    novelFeaturesICCESEN202237[0]=averageDist
    novelFeaturesICCESEN202237[1]=averageAngle

    for tempIndex in range(0, numberOfPoints):
        novelFeaturesICCESEN202237[tempIndex+2]=distFeats[tempIndex]
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
        novelFeaturesICCESEN202237[tempIndex+2+700]=angleFeats[tempIndex]

        feature=novelFeaturesICCESEN202237

    return feature

numberOfSuboptimalPoints=700
numberOfTrainImages = 60000
numberOfTestImages = 10000
dimensionOfFeatureSpace = 2+2*numberOfSuboptimalPoints
trainFeatures = np.zeros((numberOfTrainImages,dimensionOfFeatureSpace))
testFeatures = np.zeros((numberOfTestImages,dimensionOfFeatureSpace))

for tempImageIndex in range(0,numberOfTrainImages):
    if tempImageIndex % 100 == 0:
        print("Train Sample Number:",tempImageIndex)
    tempImage = train_images[tempImageIndex,:,:]
    tempSub=subOptimalv3(tempImage)
    tempFeat=novelFeature(tempSub)
    trainFeatures[tempImageIndex,:]=tempFeat[:]

for tempImageIndex in range(0,numberOfTestImages):
    if tempImageIndex % 100 == 0:
        print("Test Sample Number:",tempImageIndex)
    tempImage = test_images[tempImageIndex,:,:]
    tempSub=subOptimalv3(tempImage)
    tempFeat=novelFeature(tempSub)
    testFeatures[tempImageIndex,:]=tempFeat[:]

with open('trainFeatures.npy', 'wb') as f:
    np.save(f, trainFeatures)
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
with open('testFeatures.npy', 'wb') as f:
    np.save(f, testFeatures)
import numpy as np
import math
import matplotlib.pyplot as plt
import random
import matplotlib
import cv2
import time
import imutils

from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

with open('trainFeatures.npy', 'rb') as f:
    trainFeatures = np.load(f)
with open('testFeatures.npy', 'rb') as f:
    testFeatures = np.load(f)

def novelFeature2(novelFeature1):

    numberOfSuboptimalPoints = int((novelFeature1.shape[0]-2)/2)
    numberOfRangeBins = 5
    numberOfAngleBins = 5
    numberOfBins = numberOfRangeBins*numberOfAngleBins
    positiveRangeBinThreshold1 = 1e-2
    positiveRangeBinThreshold2 = 3e-1
    negativeRangeBinThreshold1 = -1e-2
    negativeRangeBinThreshold2 = -3e-1
    positiveAngleBinThreshold1 = 1e0
    positiveAngleBinThreshold2 = 1e1
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
    negativeAngleBinThreshold1 = -1e0
    negativeAngleBinThreshold2 = -1e1
    feature = np.zeros((numberOfBins))
    for tempIndex in range(2, numberOfSuboptimalPoints+2):
        tempRange = novelFeature1[tempIndex]
        if tempRange == 0:
            continue
        tempAngle = novelFeature1[tempIndex+numberOfSuboptimalPoints]
        if tempAngle == 0:
            continue
        if tempRange>=negativeRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold1 and
tempAngle>=negativeAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold1:
            feature[0] = feature[0] + 1
        if tempRange>negativeRangeBinThreshold2 and
tempRange<negativeRangeBinThreshold1 and
tempAngle>=negativeAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold1:
            feature[1] = feature[1] + 1
        if tempRange<negativeRangeBinThreshold2 and
tempAngle>=negativeAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold1:
            feature[2] = feature[2] + 1
        if tempRange>positiveRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold2 and
tempAngle>=negativeAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold1:
            feature[3] = feature[3] + 1
        if tempRange>positiveRangeBinThreshold2 and
tempAngle>=negativeAngleBinThreshold1  and
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
tempAngle<=positiveAngleBinThreshold1:
        feature[4] = feature[4] + 1

    if tempRange>=negativeRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold1 and
tempAngle>=negativeAngleBinThreshold2 and
tempAngle<negativeAngleBinThreshold1:
        feature[5] = feature[5] + 1
    if tempRange>negativeRangeBinThreshold2 and
tempRange<negativeRangeBinThreshold1 and
tempAngle>=negativeAngleBinThreshold2 and
tempAngle<negativeAngleBinThreshold1:
        feature[6] = feature[6] + 1
    if tempRange<negativeRangeBinThreshold2 and
tempAngle>=negativeAngleBinThreshold2 and
tempAngle<negativeAngleBinThreshold1:
        feature[7] = feature[7] + 1
    if tempRange>positiveRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold2 and
tempAngle>=negativeAngleBinThreshold2 and
tempAngle<negativeAngleBinThreshold1:
        feature[8] = feature[8] + 1
    if tempRange>positiveRangeBinThreshold2 and
tempAngle>=negativeAngleBinThreshold2 and
tempAngle<negativeAngleBinThreshold1:
        feature[9] = feature[9] + 1

    if tempRange>=negativeRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold1 and
tempAngle<negativeAngleBinThreshold2:
        feature[10] = feature[10] + 1
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
    if tempRange>negativeRangeBinThreshold2 and
tempRange<negativeRangeBinThreshold1 and
tempAngle<negativeAngleBinThreshold2:
        feature[11] = feature[11] + 1
    if tempRange<negativeRangeBinThreshold2 and
tempAngle<negativeAngleBinThreshold2:
        feature[12] = feature[12] + 1
    if tempRange>positiveRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold2 and
tempAngle<negativeAngleBinThreshold2:
        feature[13] = feature[13] + 1
    if tempRange>positiveRangeBinThreshold2 and
tempAngle<negativeAngleBinThreshold2:
        feature[14] = feature[14] + 1

    if tempRange>=negativeRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold1 and
tempAngle>positiveAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold2:
        feature[15] = feature[15] + 1
    if tempRange>negativeRangeBinThreshold2 and
tempRange<negativeRangeBinThreshold1 and
tempAngle>positiveAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold2:
        feature[16] = feature[16] + 1
    if tempRange<negativeRangeBinThreshold2 and
tempAngle>positiveAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold2:
        feature[17] = feature[17] + 1
    if tempRange>positiveRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold2 and t
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
tempAngle>positiveAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold2:
        feature[18] = feature[18] + 1
    if tempRange>positiveRangeBinThreshold2 and
tempAngle>positiveAngleBinThreshold1 and
tempAngle<=positiveAngleBinThreshold2:
        feature[19] = feature[19] + 1

    if tempRange>=negativeRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold1 and
tempAngle>positiveAngleBinThreshold2:
        feature[20] = feature[20] + 1
    if tempRange>negativeRangeBinThreshold2 and
tempRange<negativeRangeBinThreshold1 and
tempAngle>positiveAngleBinThreshold2:
        feature[21] = feature[21] + 1
    if tempRange<negativeRangeBinThreshold2 and
tempAngle>positiveAngleBinThreshold2:
        feature[22] = feature[22] + 1
    if tempRange>positiveRangeBinThreshold1 and
tempRange<=positiveRangeBinThreshold2 and
tempAngle>positiveAngleBinThreshold2:
        feature[23] = feature[23] + 1
    if tempRange>positiveRangeBinThreshold2 and
tempAngle>positiveAngleBinThreshold2:
        feature[24] = feature[24] + 1

  return feature

numberOfRangeBins = 10
numberOfAngleBins = 16
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
def novelFeature3(novelFeature1):

    numberOfSuboptimalPoints = int((novelFeature1.shape[0]-2)/2)
    numberOfBins = numberOfRangeBins*numberOfAngleBins
    feature = np.zeros((numberOfBins))

    rangeBinThresholds = np.zeros((numberOfRangeBins))
    angleBinThresholds = np.zeros((numberOfAngleBins))
    rangeBinThresholds[0] = -1e-1
    rangeBinThresholds[1] = -1e-2
    rangeBinThresholds[2] = -1e-3
    rangeBinThresholds[3] = 0
    rangeBinThresholds[4] = 1e-3
    rangeBinThresholds[5] = 1e-2
    rangeBinThresholds[6] = 1e-1
    rangeBinThresholds[7] = 1e0
    rangeBinThresholds[8] = 1e1
    rangeBinThresholds[9] = 1e1
    angleBinThresholds[0] = -1e3
    angleBinThresholds[1] = -1e2
    angleBinThresholds[2] = -1e1
    angleBinThresholds[3] = -1e0
    angleBinThresholds[4] = -1e-1
    angleBinThresholds[5] = -1e-2
    angleBinThresholds[6] = -1e-3
    angleBinThresholds[7] = 0
    angleBinThresholds[8] = 1e-3
    angleBinThresholds[9] = 1e-2
    angleBinThresholds[10] = 1e-1
    angleBinThresholds[11] = 1e0
    angleBinThresholds[12] = 1e1
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
angleBinThresholds[13] = 1e2
    angleBinThresholds[14] = 1e3
    angleBinThresholds[15] = 1e3

    for tempIndex in range(2, numberOfSuboptimalPoints+2):
        tempRange = novelFeature1[tempIndex]
        if tempRange == 0:
            continue
        tempAngle = novelFeature1[tempIndex+numberOfSuboptimalPoints]
        if tempAngle == 0:
            continue
        for tempRangeBinIndex in range(0, numberOfRangeBins):
            if tempRange<rangeBinThresholds[tempRangeBinIndex]:
                break
        for tempAngleBinIndex in range(0, numberOfAngleBins):
            if tempAngle<angleBinThresholds[tempAngleBinIndex]:
                break
        tempBinIndex = tempRangeBinIndex*numberOfRangeBins+tempAngleBinIndex
        feature[tempBinIndex]=feature[tempBinIndex]+1

    return feature

newTrainFeatures = np.zeros((60000,numberOfRangeBins*numberOfAngleBins))
newTestFeatures = np.zeros((10000,numberOfRangeBins*numberOfAngleBins))
for tempIndex in range(0, trainFeatures.shape[0]):
    if tempIndex % 1000 == 0:
        print("train index:", tempIndex)
    newTrainFeatures[tempIndex,:] = novelFeature3(trainFeatures[tempIndex,:])
for tempIndex in range(0, testFeatures.shape[0]):
    if tempIndex % 1000 == 0:
        print("test index:", tempIndex)
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
    newTestFeatures[tempIndex,:] = novelFeature3(testFeatures[tempIndex,:])

with open('newTrainFeatures.npy', 'wb') as f:
    np.save(f, newTrainFeatures)
with open('newTestFeatures.npy', 'wb') as f:
    np.save(f, newTestFeatures)

#install required libraries
#import pandas as pd
#keras packages
import keras
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Dropout
#model evaluation packages
#from sklearn.metrics import f1_score, roc_auc_score, log_loss
#from sklearn.model_selection import cross_val_score, cross_validate

import numpy as np
import math
import matplotlib.pyplot as plt
import random
import matplotlib
import cv2
import time
import imutils
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

with open('trainFeatures.npy', 'rb') as f:
    trainFeatures = np.load(f)
with open('testFeatures.npy', 'rb') as f:
    testFeatures = np.load(f)

#with open('newTrainFeatures.npy', 'rb') as f:
#    trainFeatures = np.load(f)
#with open('newTestFeatures.npy', 'rb') as f:
#    testFeatures = np.load(f)

onesForTrainFeatures = np.ones((trainFeatures.shape[0],1))
onesForTestFeatures = np.ones((testFeatures.shape[0],1))
trainFeatures = np.concatenate((onesForTrainFeatures, trainFeatures), axis=1)
testFeatures = np.concatenate((onesForTestFeatures, testFeatures), axis=1)
#initializing CNN model
classifier_e25 = Sequential()
#add 1st hidden layer
classifier_e25.add(Dense(input_dim = trainFeatures.shape[1], units = 256,
kernel_initializer='uniform', activation='relu'))

#add output layer
classifier_e25.add(Dense(units = 10, kernel_initializer='uniform', activation='softmax'))
#compile the neural network
classifier_e25.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
#model summary
classifier_e25.summary()
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
#fit training dataset into the model
classifier_e25_fit = classifier_e25.fit(trainFeatures, train_labels, epochs=25, verbose=0)

#install required libraries
#import pandas as pd
#keras packages
import keras
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Dropout
#model evaluation packages
#from sklearn.metrics import f1_score, roc_auc_score, log_loss
#from sklearn.model_selection import cross_val_score, cross_validate

import numpy as np
import math
import matplotlib.pyplot as plt
import random
import matplotlib
import cv2
import time
import imutils

from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

trainLabels = np.zeros((train_labels.shape[0], 10))
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
testLabels = np.zeros((test_labels.shape[0], 10))

for tempIndex in range(0, train_labels.shape[0]):
    trainLabels[tempIndex, train_labels[tempIndex]] = 1
for tempIndex in range(0, test_labels.shape[0]):
    testLabels[tempIndex, test_labels[tempIndex]] = 1

#with open('trainFeatures.npy', 'rb') as f:
#    trainFeatures = np.load(f)
#with open('testFeatures.npy', 'rb') as f:
#    testFeatures = np.load(f)



with open('newTrainFeatures.npy', 'rb') as f:
    trainFeatures = np.load(f)
with open('newTestFeatures.npy', 'rb') as f:
    testFeatures = np.load(f)

#onesForTrainFeatures = np.ones((trainFeatures.shape[0],1))
#onesForTestFeatures = np.ones((testFeatures.shape[0],1))

#trainFeatures = np.concatenate((onesForTrainFeatures, trainFeatures), axis=1)
#testFeatures = np.concatenate((onesForTestFeatures, testFeatures), axis=1)

#initializing CNN model
classifier_e25 = Sequential()
#add 1st hidden layer
classifier_e25.add(Dense(input_dim = trainFeatures.shape[1], units = 512,
kernel_initializer='uniform', activation='relu'))
#add output layer
classifier_e25.add(Dense(units = 10, kernel_initializer='uniform', activation='softmax'))
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
#compile the neural network
classifier_e25.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
#model summary
classifier_e25.summary()

#fit training dataset into the model
#classifier_e25_fit = classifier_e25.fit(trainFeatures, train_labels, epochs=25, verbose=0)
classifier_e25_fit = classifier_e25.fit(trainFeatures, train_labels, epochs=25, verbose=0)

#install required libraries
import pandas as pd
#model evaluation packages
from sklearn.metrics import f1_score, roc_auc_score, log_loss
from sklearn.model_selection import cross_val_score, cross_validate
#evaluate the model for testing dataset
test_loss_e25 = classifier_e25.evaluate(testFeatures, test_labels, verbose=0)
#calculate evaluation parameters
f1_e25 = f1_score(test_labels, np.argmax(classifier_e25.predict(testFeatures),axis=1), average='micro')
#roc_e25 = roc_auc_score(test_labels, np.argmax(classifier_e25.predict(testFeatures),axis=1), multi_class='ovo')
#create evaluation dataframe
stats_e25 = pd.DataFrame({'Test accuracy' :  round(test_loss_e25[1]*100,3),
                'F1 score'      : round(f1_e25,3),
                #'ROC AUC score' : round(roc_e25,3),
                'Total Loss'    : round(test_loss_e25[0],3)}, index=[0])
#print evaluation dataframe
display(stats_e25)
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
#install required libraries
#import pandas as pd
#keras packages
import keras
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Dropout
#model evaluation packages
#from sklearn.metrics import f1_score, roc_auc_score, log_loss
#from sklearn.model_selection import cross_val_score, cross_validate

import numpy as np
import math
import matplotlib.pyplot as plt
import random
import matplotlib
import cv2
import time
import imutils

from tensorflow.keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

trainLabels = np.zeros((train_labels.shape[0], 10))
testLabels = np.zeros((test_labels.shape[0], 10))
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
for tempIndex in range(0, train_labels.shape[0]):
    trainLabels[tempIndex, train_labels[tempIndex]] = 1
for tempIndex in range(0, test_labels.shape[0]):
    testLabels[tempIndex, test_labels[tempIndex]] = 1

#with open('trainFeatures.npy', 'rb') as f:
#    trainFeatures = np.load(f)
#with open('testFeatures.npy', 'rb') as f:
#    testFeatures = np.load(f)



with open('newTrainFeatures.npy', 'rb') as f:
    trainFeatures = np.load(f)
with open('newTestFeatures.npy', 'rb') as f:
    testFeatures = np.load(f)

#onesForTrainFeatures = np.ones((trainFeatures.shape[0],1))
#onesForTestFeatures = np.ones((testFeatures.shape[0],1))

#trainFeatures = np.concatenate((onesForTrainFeatures, trainFeatures), axis=1)
#testFeatures = np.concatenate((onesForTestFeatures, testFeatures), axis=1)

#initializing CNN model
classifier_e25 = Sequential()
#add 1st hidden layer
classifier_e25.add(Dense(input_dim = trainFeatures.shape[1], units = 1024,
kernel_initializer='uniform', activation='relu'))
#add output layer
classifier_e25.add(Dense(units = 10, kernel_initializer='uniform', activation='softmax'))
#compile the neural network
classifier_e25.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
```

**EK 1. (Devam)** Şekil Tanımlayıcısının Tasarımı ve Gerçekleştirilmesi

```
metrics=['accuracy'])
#model summary
classifier_e25.summary()

#fit training dataset into the model
#classifier_e25_fit = classifier_e25.fit(trainFeatures, train_labels, epochs=25, verbose=0)
classifier_e25_fit = classifier_e25.fit(trainFeatures, train_labels, epochs=25, verbose=0)

#install required libraries
import pandas as pd
#model evaluation packages
from sklearn.metrics import f1_score, roc_auc_score, log_loss
from sklearn.model_selection import cross_val_score, cross_validate
#evaluate the model for testing dataset
test_loss_e25 = classifier_e25.evaluate(testFeatures, test_labels, verbose=0)
#calculate evaluation parameters
f1_e25 = f1_score(test_labels, np.argmax(classifier_e25.predict(testFeatures),axis=1), average='micro')
#roc_e25 = roc_auc_score(test_labels, np.argmax(classifier_e25.predict(testFeatures),axis=1), multi_class='ovo')
#create evaluation dataframe
stats_e25 = pd.DataFrame({'Test accuracy' : round(test_loss_e25[1]*100,3),
                'F1 score'     : round(f1_e25,3),
                #'ROC AUC score' : round(roc_e25,3),
                'Total Loss'   : round(test_loss_e25[0],3)}, index=[0])
#print evaluation dataframe
display(stats_e25)
```