
GAZİ ÜNİVERSİTESİ
Gazi
1926

BÜYÜK HACİMLİ GÖRÜNTÜ VERİ TABANLARINDA HIZLI GÖRÜNTÜ ARAMA

Osman DURMAZ

DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARALIK 2019

Osman DURMAZ tarafından hazırlanan "BÜYÜK HACİMLİ GÖRÜNTÜ VERİ TABANLARINDA HIZLI GÖRÜNTÜ ARAMA" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Mühendisliği Ana Bilim Dalında DOKTORA TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Hasan Şakir BİLGE

Elektrik-Elektronik Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi ……..……..

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

Başkan: Prof. Dr. Erdoğan DOĞDU

Department of Computer Science, Angelo State University ...……

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

Üye: Prof. Dr. Suat ÖZDEMİR

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi ...……

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

Üye: Dr. Öğr. Üyesi Ahmet Ercan TOPCU

Bilgisayar Mühendisliği Ana Bilim Dalı, Ankara Yıldırım Beyazıt Ünversitesi .....……

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

Üye: Doç. Dr. Oktay YILDIZ

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi ...……

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

Tez Savunma Tarihi: 30/12/2019

Jüri tarafından kabul edilen bu çalışmanın Doktora Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..…….

Prof. Dr. Sena YAŞYERLİ

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Osman DURMAZ
30/12/2019

BÜYÜK HACİMLİ GÖRÜNTÜ VERİ TABANLARINDA HIZLI GÖRÜNTÜ ARAMA

(Doktora Tezi)

Osman DURMAZ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Aralık 2019

ÖZET

Bu tez çalışmasında, büyük hacimli görüntü veri tabanları üzerinde hızlı ve doğru bir şekilde görüntü arama yapılabilmesi için geliştirilen RDH (Randomized Distributed Hashing) yöntemi sunulmuştur. Büyük görüntü veri tabanlarında sorgulanan görüntülere yakın örneklerin bulunabilmesi için genellikle ANN (Approximate Nearest Neighbor) yöntemleri kullanılmaktadır. Bu yöntemlerde aranan örneklere benzer en yakın gerçek örneklerin bulunması yerine yakın olması muhtemel örnekler bulunmaktadır. Çoğu zaman özetleme yöntemleriyle gerçeklenen bu yöntemlerin kullanılmasıyla arama zamanı ciddi oranda azaltıbilmektedir. ANN arama yöntemleri genellikle merkezi olarak uygulanmaktadır. Ancak gerçek dünya uygulamalarında veriler genellikle dağıtık bir şekilde saklanmaktadır. Bu durum ANN arama yöntemlerinin dağıtık bir şekilde uygulanabilmesini gerektirmektedir. Bu amaçla önerdiğimiz yaklaşımda LSH (Locality Sensitive Hashing) dağıtık bir şekilde uygulanmıştır. Veri bir küme içindeki farklı düğümlere dağıtılmış sonrasında her bir düğümde aynı özet fonksiyon kümesi kullanılarak veri özetlenmiştir. Sorgu aşamasında sorgu örneği her bir düğümde yerel olarak aranmaktadır. Paralel sorgulardan faydalanıldığında sorgu süresi önemli oranda düşmüştür. Deneysel çalışmalarda 10 düğüm kullanıldığında sorgu hızı yaklaşık olarak 10 kat artırılmıştır. Sistemin başarısını değerlendirmek için kullanılan MAP (Mean Average Precision) değeri literatürdeki çalışmalarla kıyaslanabilecek ölçüde yüksek çıkmıştır. Bu çalışmada aynı zamanda düğümlerde aynı özet fonksiyonların kullanılması yerine farklı özet fonksiyonların ve seçilmiş özet fonksiyonların kullanımıyla LSH yönteminin dağıtık kullanımı detaylı bir şekilde irdelenmiştir. Seçilmiş özet fonksiyonları indeksleme yapılmadan önce veriyi bölme özelliğine göre oluşturulmuştur. LSH yöntemi veri bağımsız bir yöntem olduğundan düğümlerde aynı özet fonksiyonu kullanıldığında alınan sonuçlara benzer sonuçlar elde edilmiştir. Alınan sonuçlar son zamanlarda yayınlanan ve dağıtık özetleme konusunda farklı yöntemlere ait sonuçlar içeren bir çalışma ile karşılaştırılmıştır. Önerilen yöntem dağıtık olarak büyük boyutlu veri kümelerinde görüntü arama için umut vermektedir.

Bilim Kodu	:	92418
Anahtar Kelimeler	:	Hızlı görüntü alma, özetleme, görüntü arama, dağıtık işleme, yerel hassas özetleme
Sayfa Adedi	:	80
Danışman	:	Doç. Dr. Hasan Şakir BİLGE

FAST IMAGE SEARCH ON HIGH DIMENSIONAL IMAGE DATABASE

(Ph. D. Thesis)

Osman DURMAZ

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

December 2019

ABSTRACT

In this thesis, RDH (Randomized Distributed Hashing) method which is developed for fast and accurate image search on large scale image databases is presented. ANN (Approximate Nearest Neighbor) approaches are usually used to find the nearest samples to the queried images in large scale image databases. In these methods approximate nearest samples are found instead of finding the real nearest samples. Using these methods, which are often implemented by hashing methods, can significantly reduce the query time. ANN search methods are generally applied in centralized manner. However in real-world applications, data are often stored in a distributed manner. This situation requires to implement ANN search methods in a distributed manner. For this purpose in our proposed approach, LSH (Locality Sensitive Hashing) method is applied in a distributed way. Data are distributed to different nodes within a cluster, and then the data are hashed on each node using the same hash function set. In query phase, the query instance is searched locally on each node. By exploiting from parallelism, the query time is significantly decreased. In the experimental studies, we have a speed up of 10 for the query performance in the distributed scheme with 10 nodes. The level of MAP (Mean Average Precision) scores that are used to evaluate system performance are quite high which are comparable to other methods in literature. We have also investigated the usage of different and selected randomized hash functions in different nodes rather than using same indexing. By this way the distributed usages of LSH are scrutinized. We create selected hash functions according to their data division property before indexing. Since LSH is data independent method, we have obtained similar results with using same hash functions. We compared our experimental results with state-of-the-art methods given in a recent study. The proposed distributed scheme is promising for searching images in large datasets with multiple nodes.

Science Code	:	92418
Key Words	:	Fast image retrieval, hashing, image search, distributed processing, locality sensitive hashing
Page Number	:	80
Supervisor	:	Assoc. Prof. Dr. Hasan Şakir BİLGE

TEŞEKKÜR

Tez çalışmam süresince, aynı zamanda üniversite ve yüksek lisans eğitimim boyunca değerli fikir ve tecrübelerini bana aktarıp eğitimime yön veren çok değerli hocam Sayın Doç. Dr. Hasan Şakir BİLGE'ye teşekkür ederim. Tez izleme kurul toplantılarında fikirlerinden yararlandığım Sayın Prof. Dr. Erdoğan DOĞDU ve Sayın Prof. Dr. Suat ÖZDEMİR hocalarıma teşekkür ederim. Doktora çalışmalarım süresince sevgisini ve desteğini hiç esirgemeyen eşim Sulbiye DURMAZ ve çocuklarım Zehra Reyhan DURMAZ ile Ahmet Faruk DURMAZ'a teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
ms	Milisaniye
s	Saniye

Kısaltmalar	**Açıklamalar**
ABQ	Adaptive Binary Quantization
ADDM	Alternating Direction Method of Multipliers
AGH	Anchor Graph Hashing
ANN	Approximate Nearest Neighbor
BRE	Binary Reconstructive Embedding
CBIR	Content Based Image Retrieval
CV	Cross Validation
DGH	Distributed Graph Hashing
DH	Deep Hashing
DHT	Distributed Hash Table
DoG	Different of Gaussian
DVStH	Deep Variational and Structural Hashing
ICA	Independent Component Analysis
IH	Isometric Hashing
ITQ	Iterative Quantization
kNN	k-Nearest Neighbors
KSH	Kernel Based Supervised Hashing
LSH	Locality Sensitive Hashing
LSMH	Latent Semantic Minimal Hashing
MAP	Mean Average Precision
MCR	Min Cost Ranking

Kısaltmalar	Açıklamalar
NNS	Nearest Neighbor Search
PCAH	Principle Component Analysis Hashing
RDH	Randomized Distributed Hashing
SH	Spectral Hashing
SEH	Sparse Embedded Hashing
SIFT	Scale Invariant Feature Transform
SSH	Semi Supervised Hashing
SURF	Speeded Up Robust Transform

1. GİRİŞ

Son yıllarda görüntü yakalama cihazlarının yaygınlaşması ve İnternet'in geniş kitlelere ulaşmasıyla birlikte çok sayıda görüntü erişilebilir hale gelmiştir. Sosyal ağların insanlar tarafından kullanımının artmasıyla da pek çok insan yakaladıkları görüntüleri bu ortamlar üzerinden paylaşmıştır. Oluşan büyük hacimli görüntü verisi üzerinde sınıflandırma yapmayı zorlaştırmıştır. Benzer şekilde büyük veri kümesi üzerinde arama yapmak ve istenilen görüntülere ulaşmak da zorlaşmıştır [1]. Büyük veri kümeleri üzerinde işlem yapabilmek için özelleşmiş görüntü alma sistemleri gerekmektedir. Görüntü alma sistemleri etkili indeksleme ve arama yöntemlerine sahip olmalıdır [2, 3]. İyi bir indeksleme yöntemi arama hızını artırabilmeli, hafıza gereksinimi azaltabilmeli ve belli bir uzaklık ölçütüne göre başarılı sonuçlar sağlayabilmelidir [4].

CBIR (Content Based Image Retrieval - İçerik Tabanlı Görüntü Alma) sistemleri bu problem için çözüm sunmaktadır. CBIR sistemlerinde görüntüler sahip oldukları renk, doku, şekil ve uzaysal yerleşim gibi düşük seviyeli özniteliklerle değerlendirilmektedir [5-7]. Genellikle bu sistemler iki aşamadan oluşmaktadır. İlk aşama veri kümesi içindeki tüm görüntüler için öznitelik vektörlerinin oluşturulması ve bu vektörlerin indekslenmesidir. İkinci aşama ise indekslenen öznitelik uzayında verilen bir sorgu görüntüsüne en benzer örneklerin bulunmasıdır. CBIR sistemlerde insanların görsel algılarını taklit edebilecek sistemlerin tasarlanması oldukça zordur. Bazen birbirine yakın olarak bulunan iki görüntü gerçekte birbirlerinden tamamen farklı olabilmektedir.

Görüntülerin makineler tarafından görsel karakterlerine göre yorumlanması ile insanların görsel algıları arasındaki fark anlamsal boşluk (semantic gap) olarak adlandırılmaktadır [5, 8-15]. Bu iki görsel algı arasındaki boşluğu azaltmak için temel olarak NNS (Nearest Neighbor Search - En Yakın Komşu Arama) yöntemleri kullanılmaktadır. Bu yaklaşım benzerlik veya yakınlık arama olarak da bilinmektedir [16, 17]. NNS yönteminin bilgi alma, bilgisayarla görme, makine öğrenmesi gibi alanlar başta olmak üzerek pek çok alanda yaygın kullanımı vardır. NNS yönteminde amaç herhangi bir sorgu görüntüsü için bu görüntüye yakın görüntülerin bulunmasıdır. Benzer görüntüler, görüntü veri tabanında bulunan görüntülerin sorgu görüntüsüyle benzerliğinin belli bir benzerlik ölçütüne göre sıralanarak *k*

adet görüntünün seçilmesiyle bulunmaktadır. En yakın komşuların doğru bir şekilde bulunmasının yanı sıra bu görüntülerin elde edilme zamanı da çok önemlidir [13, 14].

Öklid uzaklığına göre en yakın komşuları belirleyen NNS yönteminin *n* örnek içeren bir veri kümesi üzerinde sorgu örneğiyle veri kümesi içindeki örnekler arasındaki uzaklıkların hesaplama maliyeti *O(n)*'dir. Bu durum arama süresinin küçük veri kümeleri için uygun olduğunu ancak büyük hacimli veri kümeleri için sorun olacağını göstermektedir [1-3, 16, 18-25]. Doğrusal arama işleminin performans probleminin çözülebilmesi için ağaç tabanlı (KD trees, BK trees, RN trees, R trees, M trees, cover trees, metric trees, S trees, SR trees) çözümler önerilmiştir. Bu algoritmalar indeksleme için ağaç yapısı kullanmaktadır. Ağaç tabanlı indeksleme yaklaşımları küçük veri kümelerinde logaritmik sorgu süresi sağlasa da büyük hacimli görüntü veri kümelerinde çok karmaşık bir yapı oluştuğundan doğrusal aramaya yakın arama zamanı sunmaktadır. Diğer yandan ağaç yapısının depolama gereksinimi doğrusal aramaya göre artmaktadır. Ağaç yapılarının kullandığı indeksler bazı durumlarda orijinal veri boyutundan bile büyük olabilmektedir [1, 4, 10, 19, 25-29]. Bu olumsuzluklardan dolayı ağaç yapıları büyük hacimli görüntü veri kümeleri üzerinde kullanışsızdır. Ağaç tabanlı yaklaşımlardan farklı olarak ANN (Approximate Nearest Neighbor - Yaklaşık En Yakın Komşu) yaklaşımları NNS yöntemlerinin ölçeklenebilirlik problemini etkili bir şekilde çözebilmek için önerilmiştir. Bu yaklaşımda gerçek komşuların bulunması yerine muhtemel komşular bulunmaktadır.

Özetleme (hashing) yöntemleri yaklaşık en yakın komşu arama için kullanılan etkili yöntemlerdir [3]. Özetleme yöntemlerindeki ana fikir orijinal öznitelik uzayını bu uzaydaki benzerliklerin korunarak ikili uzayda ifade edilebilmesi ve verinin bu uzayda indekslenebilmesidir [1, 29]. Depolama maliyetlerinin az olması ve hızlı sorgu süresi sunmalarından dolayı özetleme yöntemleri son yıllarda büyük ilgi çekmektedir [1, 2, 16, 19, 26, 30]. Arama özet kod tablosu veya özet kod derecelendirme şeklinde yapılabilmektedir. Özet kod tablosu yaklaşımında benzer örneklerin benzer özet kovalarında (hash bucket) bulunma ihtimali maksimize edilmektedir. Özet kod derecelendirme yaklaşımında sorgu örneği ile referans olarak kullanılan veri kümesindeki örneklerin birbiriyle uzaklıkları yeniden derecelendirilmektedir [2, 31, 32]. ANN yaklaşımları özetleme yöntemlerinin kullanılmasıyla hızlı ve doğru bir şekilde gerçekleştirilmektedir. Özetleme yöntemlerinde veri özet (hash) ismi verilen düşük boyutlu bit dizileriyle ifade edilmektedir. Bu yaklaşımla ilgili komşuların bulunması sabit veya alt doğrusal zaman almaktadır [22, 27, 33, 34].

Özetleme yöntemleri özet fonksiyonların öğrenilmesi ve ikili kodların oluşturulması olarak iki aşamadan oluşmaktadır. Literatürde özet fonksiyonların öğrenilmesi üzerinde yoğun olarak çalışılmaktadır [4, 19, 27]. Doğrusal fonksiyonlar özetleme için yaygın bir şekilde kullanılırken çekirdek fonksiyonları ve en yakın vektör atama tabanlı fonksiyonlarda arama doğruluğu için iyi sonuçlar verebilmektedir [27]. n noktadan oluşan d boyutlu veri uzayı $X = [x_1, x_2, x_3, \ldots x_n] \in d \times n$ şeklinde ifade edilebilmektedir. Örnek x_i için oluşturulan özet kodu y_i, $h(.)$ özet fonksiyonu kullanılarak $y_i = h(x_i)$ şeklinde elde edilebilmektedir.

Genellikle m adet özet fonksiyonu kullanılarak m adet özet kodu elde edilmektedir. Özet tabanlı yöntemler Hamming uzayında benzer örnekleri benzer ikili kodlarla eşleştirmektedir. Hamming uzayı düşük boyutlu olduğundan veri noktaları az sayıda bit ile kodlanmakta ve bu durum sorgu süresi ile hafıza gereksiniminin azalmasını sağlamaktadır [36]. Bu nedenle Hamming uzayında özet kodları kullanılarak yapılan arama hızlı olarak gerçekleştirilmektedir ve bu algoritmalar tarafından Hamming uzaklığı yaygın bir şekilde kullanılmaktadır [2, 21].

Özet yöntemleri genellikle merkezi olarak uygulanmaktadır. Gerçek dünya uygulamalarında ise büyük hacimli veri genellikle dağıtık olarak tutulmaktadır [36]. Bu yüzden dağıtık veri üzerinde çalışabilecek dağıtık özetleme yöntemlerinin gerçeklenmesi son zamanlarda yoğun olarak üzerinde çalışılan konular arasındadır [37-39]. Dağıtık özet kodu öğrenilmesinde veri düğümlere dağıtılmaktadır. Her bir düğüm kendisine gönderilen veri üzerinde çalışmaktadır. Bu tez çalışmasında LSH (Locality Sensitive Hashing - Yerel Hassas Özetleme) yönteminin dağıtık bir şekilde uygulanması önerilmiş ve önerilen bu yönteme RDH (Randomized Distributed Hashing - Rastgele Dağıtık Özetleme) ismi verilmiştir. LSH yöntemi veri bağımsız bir yöntem olduğundan merkezi kaynaklarla çalıştığı gibi dağıtık bir şekilde de kolaylıkla çalışabilmektedir. Bu şekilde tüm veriyi özet fonksiyonlar kullanarak merkezi bir düğümde indekslemek yerine, veriyi düğümlere dağıtıp düğümlerde birbirinin aynı özet fonksiyonları kullanarak indeksleme yapmak mümkündür. LSH yöntemi veri bağımsız bir yöntem olduğundan özet kodlarının oluşturulması esnasında düğümler arasında veri iletimine gerek yoktur. Ağ yalnızca sorgu aşamasında sorgu örneğinin düğümlere gönderilmesi ve düğümlerden gelen cevabın merkezi düğüme gönderilmesi sırasında kullanılmaktadır. Bu çalışmada LSH yöntemi dağıtık bir şekilde uygulanarak hafıza gereksinimi ile her bir düğümdeki sorgu ve eğitim süresi azaltılmıştır. Ayrıca her bir düğümde aynı özet fonksiyonlarının kullanılması yerine her bir düğümde birbirinden farklı

özet fonksiyonlarının kullanılması ve düğümlerdeki veriyi yaklaşık olarak eşit iki parçaya bölebilen özet fonksiyonlarının kullanım durumu incelenmiştir. Bu tez çalışmasının amacı büyük hacimli dağıtık görüntü kümeleri üzerinde herhangi bir sorgu örneği için bu örneğe en çok benzeyen örneklerin hızlı bir şekilde bulunması için gereken altyapının geliştirilmesidir. Tez çalışması kapsamında rastgele özet fonksiyonlar dağıtık görüntü öznitelikleri üzerinde uygulanmıştır. Bu tez çalışmasının temel katkıları şu şekildedir:

- Çoğu gerçek dünya uygulamasında veriler dağıtık bir şekilde tutulmaktadır. Bu çalışmada önerilen RDH yöntemiyle dağıtık özetleme problemine bir çözüm sunularak istenilen verilere hızlı erişim sağlanması hedeflenmiştir.
- RDH yönteminde veri her bir düğümde birbirinden bağımsız olarak indekslenmiştir. Bu şekilde tüm veri LSH yönteminden çok daha hızlı bir şekilde indekslenmiştir. İndeksleme aşamasında düğümlerin birbirleriyle iletişimde olması gerekmemektedir. Ağ yalnızca sorgu aşamasında kullanılmaktadır. Ek olarak veri indekslendikten sonra bile yeni düğümler esnek bir şekilde sisteme eklenebilmektedir.
- LSH yöntemi dağıtık bir şekilde uygulanmıştır. Bu şekilde sorgu ve eğitim süresi azaltılıp LSH yönteminden daha iyi sonuçlar elde edilmiştir.
- Bu alandaki çalışmalarda sıklıkla kullanılan üç veri kümesi üzerinde çok sayıda deneysel çalışma yapılmıştır. Önerilen sistem MAP (Mean Average Precision - Ortalama Kesinlik), hızlanma, genişleme ve çalışma zamanı ölçütlerine göre karşılaştırmalı bir şekilde değerlendirilmiştir. Bu şekilde RDH yöntemi için en uygun şartlar ortaya konmuştur. Elde edilen sonuçlar bu alanda farklı yaklaşımlardan elde edilen sonuçları bir arada sunan yakın zamanda yayınlanan bir çalışma [39] ile karşılaştırılmıştır. Deneysel sonuçlar bu çalışma ile benzer deneysel ortam oluşturularak alınmıştır.
- Düğümlerde aynı özet fonksiyonların kullanılmasının yanı sıra her bir düğümde birbirinden farklı özet fonksiyonları da kullanılmıştır. Ayrıca veriyi yaklaşık olarak iki eşit sayıda parçaya ayırabilen seçilmiş özet fonksiyonları ile de çalışmalar yapılmıştır. Bu özet fonksiyonları indeksleme aşamasından önce seçilmiş ve bu yaklaşım olumlu gelişmelere sebep olmuştur.
- LSH yöntemi yerine önerdiğimiz şekilde RDH yöntemi kullanılmasının daha tercih edilebilecek bir kullanımının olacağı tahmin edilmektedir. LSH yönteminin kullanımı bugünün gereksinimlerini karşılayabilecek şekilde geliştirilmiştir.

Tezin ikici bölümünde görüntü alma sistemleri, üçüncü bölümünde ise görüntü özetleme yöntemleri detaylı bir şekilde incelenmiştir. Dördüncü bölümde bu tez çalışması kapsamında önerilen yöntemin detayları anlatılmıştır. Beşinci bölümde benzerlik ve performans ölçütlerinden bahsedilmiştir. Tezin altıncı bölümünde yapılan deneysel çalışmalar anlatılmış ve alınan deneysel sonuçlar sunulmuştur. Tezin yedinci bölümünde yapılan çalışma özetlenmiş ve elde edilen çıkarımlar paylaşılarak gelecekte yapılabilecek çalışmalar hakkında bilgi verilmiştir.

2. GÖRÜNTÜ ALMA SİSTEMLERİ

Görüntü miktarının artması aranan görüntülere hızlı bir şekilde erişim ihtiyacı doğurmuştur [5]. Görüntü içeren veri tabanları üzerinde istenilen görüntülere etkili bir şekilde ulaşabilmek için genellikle metin tabanlı ve içerik tabanlı sistemler kullanılmaktadır. Metin tabanlı sistemlerde görüntüleri ifade eden anahtar kelimeler insanlar tarafından belirlenerek görüntüler ile ilişkili bir şekilde metin olarak saklanmaktadır. Daha sonra aranacak görüntü sahip olduğu özelliklerine göre metin uzayında aranarak benzer görüntüler elde edilmektedir. Metin tabanlı sistemleri oluştururken gereken insan emeği ve görüntülerin değerlendirilmesi sırasında insanların öznel yaklaşımları bu sistemlerin başarısını olumsuz olarak etkileyen etkenler olmuştur. Görüntülerin insanlar tarafından etiketlenmesi maliyeti artırmakla birlikte bazı görüntüler için de kimi zaman belirsiz anlamlar oluşturabilmektedir. Metin tabanlı sistemlerin olumsuzluklarını giderebilmek için içerik tabanlı görüntü arama sistemleri önerilmiştir [5, 8].

CBIR (Content Based Image Retrieval – İçerik Tabanlı Görüntü Alma) sistemlerinde görüntüler sahip oldukları renk, doku, şekil gibi düşük seviyeli görsel içeriklerine göre değerlendirilmektedir [5, 9, 40, 41]. Düşük seviyeli öznitelikleri kullanan içerik tabanlı görüntü alma yöntemleri literatürde yoğun ilgi gören bir alandır [8]. Bu sistemlerde görüntüye ait öznitelikler görüntünün tamamı kullanılarak veya yerel bir kısmından faydalanılarak oluşturulabilmektedir. Görüntünün yerel kısmını kullanan sistemlerde öncelikle görüntü bölütlere ayrılmaktadır. Renk, desen, şekil gibi öznitelikler oluşturulan bölütler kullanılarak belirlenmekte ve bu öznitelikler görüntülerin aranmasında kullanılmaktadır [5]. CBIR sistemlerinin hastalık teşhisi, suç önleme, coğrafi bilgi, uzaktan algılama gibi birçok alanda kullanımı vardır [10]. CBIR sistemler genellikle iki aşamalıdır. İlk aşama görüntü veri tabanındaki tüm görüntülerin öznitelik vektörlerinin oluşturulması ve etkili bir indeksleme ile bir veri yapısının oluşturulmasıdır. İkinci aşama indekslenen öznitelik uzayı kullanılarak sorgulanan görüntüye en yakın görüntünün bulunmasıdır. Görüntü öznitelik vektörleri görüntüler için anlamlı özelliklerin belirlenerek bu özellikleri bir dizi halinde ifade edilmesidir. Görüntüleri ifade eden anlamlı özellikleri bulmak, CBIR için önemli bir konudur [6-8, 11, 40]. CBIR uygulanarak sistemin eğitilmesi ve sorgu görüntülerinin eğitilen sistem üzerinde sorgulanması Şekil 2.1 ve Şekil 2.2'de gösterilmiştir.

Şekil 2.1. Eğitim modelinin oluşturulması

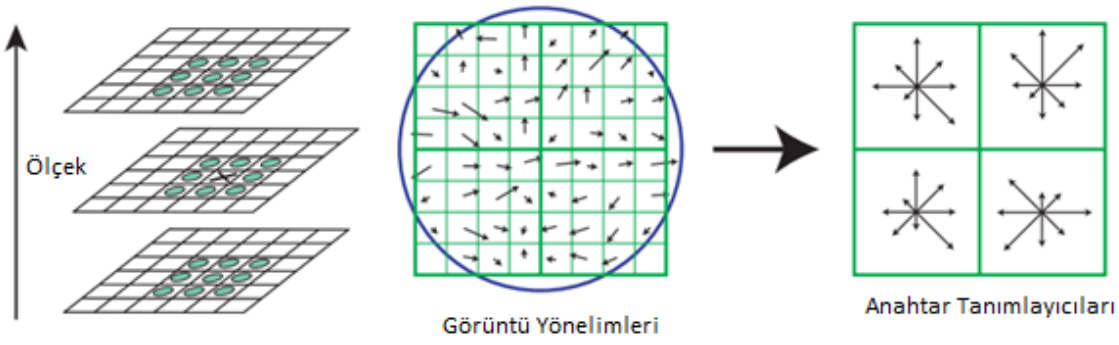
CBIR sistemlerinde öncelikle görüntü veri tabanı kullanılarak görüntülere ait öznitelikler çıkarılmaktadır. Çıkarılan bu öznitelikler sistemin eğitilmesinde kullanılmaktadır. Genellikle sistemin eğitilmesi çevrim dışı, sistemin testi ise çevrim için bir işlem olarak yapılmaktadır [42]. Herhangi bir sorgu görüntüsü sisteme gönderildiğinde yapılan ilk iş bu görüntünün eğitim modeli oluşturulurken kullanılan özniteliklerin çıkarılmasıdır. Sonrasında bu özniteliklere göre eğitim modeli kullanılarak sorgu örneğine yakın görüntüler bulunmaktadır.

Şekil 2.2. Sorgu görüntüsüne yakın örneklerin bulunması

Görüntü alma yöntemlerinde görüntünün tamamını temsil eden global öznitelikler veya görüntünün belli bölümlerini ifade eden yerel öznitelikler kullanılmaktadır. Her iki öznitelik için de görüntü üzerindeki görsel bilgilerin çıkarılması ortak bir işlemdir [9]. Global öznitelikler dönüşüm, dönme ve ölçekleme gibi işlemlerden bağımsızdır. Ancak global öznitelikler görüntü içindeki tekrarlı yapıları bulma konusunda yetersiz kalmaktadır. Yerel öznitelikler global özniteliklerde bulunan problemlerden dolayı önerilmiştir. Bu öznitelikler de dönüşüm, dönme ve kısmi aydınlanmaya karşı bağımsızdır. SIFT (Scale Invariant Feature Transform – Ölçekten Bağımsız Öznitelik Dönüşümü), SURF (Speeded Up Robust Features – Hızlandırılmış Sağlam Öznitelikler), HoG (Histogram of Oriented Gradients - Yönlü

Gradyan Histogramı) yerel öznitelik bulma için kullanılan temel yöntemlerden bazılarıdır [6, 43].

SIFT öznitelikleri en yaygın kullanılan düşük seviyeli özniteliklerden biridir [44]. SIFT öznitelikleri görüntü üzerinden çıkarılan ölçekten, döndürmesinden ve aydınlatmadan bağımsız özniteliklerdir. Bu özniteliklerin ayırt ediciliği yüksektir ve görüntü tanımlama için büyük veri tabanlarında yüksek doğrulukla eşleşme sağlamaktadır. Bu yöntemde öncelikle ölçek uzayı oluşturulmaktadır. DoG (Difference of Gaussian) fonksiyonu kullanılarak ölçek ve yönelimden bağımsız muhtemel ilgi noktaları bulunmaktadır. DoG fonksiyonu görüntülerdeki kenarların ve ilgili ayrıtların ortaya çıkarılmasında kullanılmaktadır. Her bir aday nokta için ölçek ve konumuna karar vermek için detaylı bir model uygulanmaktadır. Anahtar noktalar kararlılıklarına göre seçilmektedir. Her anahtar nokta konumuna yerel görüntü eğim doğrultularında bir veya birden fazla yönelim atanmaktadır. Son olarak yerel görüntü eğimleri her bir anahtar nokta çevresindeki alanda seçilen ölçeğe göre ölçülerek şekil bozulması ve ışık değişiminden bağımsız hale getirilmektedir. Ölçek uzayı farklı σ değerleri kullanılarak görüntüyü Gauss çekirdeği ile işleme sokulmasıyla oluşturulmaktadır. Oluşan farklı ölçekteki görüntüleri birbirinden çıkarılmakta ve yerel minimum veya maksimumların bulunması için her piksel kendi ölçeğindeki 8 komşusu ve yakın iki ölçekteki toplam 18 komşusuyla yani 26 nokta ile karşılaştırılmaktadır. Şekil 2.3'te oluşturulan ölçek uzayı gösterilmektedir [45]. Daha sonra bulunan noktaların konum ve ölçekleri belirlenmektedir [6, 43-45].



Şekil 2.3. SIFT tanımlayıcılarının oluşturulması [45]

SIFT yönteminde 10° aralıklarla 36 adet yönelim histogramı oluşturulmaktadır. Histograma eklenen her bir örnek σ değerinin 1,5 katı olan Gaussian ağırlıklıklı eğim değeri ile ağırlandırılmaktadır. Anahtar noktaların etrafındaki 16 × 16 alan kullanılarak aralarında 45°

açı farkı olan 8 yönelime sahip 4 × 4 boyutunda alan oluşturulur. Anahtar noktalar 8 yönelim ve 4 × 4 boyutunda alandan oluştuğundan toplamda 8 × 4 × 4 = 128 boyutunda öznitelik vektörleri ile temsil edilmektedir [45].

SURF yöntemi performans olarak SIFT yönteminin geliştirilmiş bir versiyonudur. SURF yönteminde anahtar noktaların en önemli özelliği tekrar edilebilir olmasıdır. Bu durum farklı görünüşler için anahtar noktaların benzer olmasını sağlamaktadır. HoG yöntemi SIFT yönteminin basitleştirilmiş bir uygulamasıdır [6, 43].

CBIR sistemlerinde insanların görsel algısını taklit edebilecek tasarımlar yapmak oldukça zor bir problemdir. Birbirine çok benzeyen iki görüntü aslında birbirinden farklı görüntüler olabilir. Örneğin Şekil 2.4'te gösterildiği gibi doğal ortamda benzer koşullarda çekilmiş leopar ve kaplan görüntülerinin ayırt edilmesi zor bir problemdir [9].

Şekil 2.4. Farklı hayvanların benzer görüntüsü [9]

Görüntü alma sistemlerinde görüntülerin sahip olduğu özniteliklerin belirlenmesinden sonra görüntü arama için kullanılacak olan öznitelikler belirlenmektedir. Teorik olarak çok sayıda özniteliğin olması sınıflandırmanın ayırıcılığını artırsa da pratikte tüm öznitelikleri kullanmak her zaman doğru sonuç vermemektedir. Bunun yanında büyük boyuttaki veri boyutsallık lanetine (curse of dimensionality) de neden olabilmektedir.

Öznitelik seçimi ile görüntüyü ifade edebilen daha az öznitelik bulunarak hesaplama maliyeti azaltılabilmektedir. Bu algoritmalar genellikle öznitelik dönüştürme ve öznitelik seçme olarak ikiye ayrılmaktadır. Öznitelik dönüştürme yönteminde görüntü veri tabanında

bulunan görüntülerden elde edilen öznitelik uzayı daha az öznitelikle ifade edilebilen yeni bir uzayda ifade edilirler. PCA (Principal Component Analysis – Temel Bileşen Analizi) ve ICA (Independent Component Analysis – Bağımsız Bileşen Analizi) bu alanda kullanılan yaygın yöntemlerdir [12]. Bu yöntemlerde görüntüler daha az öznitelik vektörleri ile ifade edilebilmesine rağmen bu yöntemlerin hesaplama maliyetinin yüksek olması olumsuz yanlarıdır.

Öznitelik seçme yönteminde ayırıcı özelliği yüksek öznitelikler belirlenerek görüntüler bu özniteliklere göre değerlendirilmektedir. Bu yöntemde orijinal öznitelik uzayı içinden ayırıcı özniteliklerin bulunduğu bir alt kümenin seçilmesiyle verinin kalitesinin artırılması hedeflenmektedir. Öznitelik dönüştürme yönteminden farklı olarak daha az öznitelik oluşturulmaktadır. Öznitelik seçme yöntemi görüntü alma ve metin sınıflandırma gibi pek çok konu üzerinde uygulanmaktadır [12].

2.1. Anlamsal Boşluk

İçerik tabanlı sistemlerde kullanılan düşük seviyeli öznitelikler genellikle insanların aklındaki yüksek seviyeli görüntü tanımlarına uymamaktadır [5, 6]. Öznitelik vektörleri renk, desen, şekil gibi düşük seviyeli öznitelikleri barındırmaktadır. İnsanların ifade edebildikleri yüksek seviyeli tanımlamalarla düşük seviyeli öznitelikler arasında ciddi bir fark bulunmaktadır [10, 11].

İnsanların görme sistemlerinin görüntüleri tanımlama ve seçmeye yarayan muntazam bir çalışma şekli vardır. Bu sistemin nasıl işlediği günümüzde tam olarak anlaşılamadığından görüntü algılama sistemlerini insanların görme sistemlerine benzetim zor bir konudur [9]. Düşük seviyeli görsel öznitelikler genellikle görüntüleri tam olarak ifade edememektedir [8]. Bilgisayar sistemleri kullanılarak elde edilen görsel özellikler ve insanların görüntüyü anlamlandırması arasındaki uyumsuzluk anlamsal boşluk olarak adlandırılmaktadır. İnsanlar görüntüleri yüksek seviyeli olarak değerlendirirken bilgisayar sistemleri görüntü üzerinde bulunan düşük seviyeli farklı öznitelikleri kullanarak görüntünün ne anlama geldiğini belirlemeye çalışmaktadır. Ayrıca yüksek seviyeli öznitelikler ile düşük seviyeli özniteliklerin birbiriyle doğrudan ilişkisinin bulunmaması problemin zorluğunu artırmaktadır [5].

2.2. Düşük Seviyeli Öznitelikler

Düşük seviyeli öznitelikler görüntünün sahip olduğu niteliklerin bilgisayar sistemleri kullanılarak farklı yöntemlere göre belirlendiği görüntü tanımlamalarıdır. Bu özniteliklerin elde edilmesi CBIR sistemlerin ele alması gereken temel konudur. Öznitelikler görüntünün tamamı ya da bir bölgesi kullanılarak elde edilmektedir. Görüntünün tamamı kullanılarak elde etme işlemi basit olsa da bölge tabanlı öznitelikler insan algısına daha yakın sonuçlar çıkarmaktadır [5]. Görüntüler renk, doku, şekil gibi sahip oldukları özniteliklere göre ifade edilebilmektedir [46].

2.2.1. Renk

Renk bilgisi görüntü işleme alanında kullanılan en yaygın özniteliklerden biridir [5, 7, 10, 41, 44, 47]. Renk bilgisi görüntünün boyutundan ve yönlendirmesinden bağımsızdır [8]. Görüntülerdeki renk bilgisine ulaşmak için RGB, HSV gibi farklı renk uzayları kullanılmaktadır. Ayrıca uzaysal bilgi içermeyen ve hesaplama maliyeti düşük renk histogramları veya renk histogramlarından daha iyi sonuç veren belli bir piksel mesafesinde renk çiftlerinin bulunma olasılıklarını tanımlayarak uzaysal bilgi sağlayan renk kollagramları gibi yöntemler kullanılmaktadır [5, 8, 13]. Renk bilgisini kullanırken görüntüler üzerindeki gürültünün azaltılması için kullanılacak filtreler başarımı artırabilmektedir [5]. Ayrıca renk uyum vektörü, renk momentleri, vektör niceleme gibi yöntemler de kullanılabilmektedir [12].

2.2.2. Doku

Görüntülerdeki doku bilgisi görüntünün yüzeysel özelliklerini belirten bir bilgidir. Bu bilgi renk özniteliği kadar kolay tanımlanamamaktadır [5, 7, 41, 47]. Bu öznitelik görüntü sınıflandırma için ayırıcı özellikler sağlamaktadır. Ayrıca görüntülerin yüksek seviyeli tanımlanabilmesi için doku özniteliği önemli bilgiler vermektedir. Bu öznitelik Gabor filtresi, GLCM (Grey-Level Co-Occurance Matrix - Gri Seviyeli Eş Oluşum Matrisi), dalgacık dönüşümü [7, 46], MRF (Markov Random Field - Markov Rastgele Alan) yöntemi, SAR (Simultaneous Auto Regressive - Eş Zamanlı Otomatik Gerileme) yöntemi ve istatistiksel öznitelik analizi yöntemleriyle elde edilebilmektedir. Gabor filtresi ve dalgacık

öznitelikleri dörtgensel görüntüler üzerinde çalışmak üzere tasarlanmıştır [5, 8, 9, 12]. GLCM ve Gabor filtresi yöntemleri yüksek performansa sahip olmalarına rağmen yüksek hesaplama karmaşıklığına sahiptir. GLCM yöntemi doku bilgisini yüksek frekanslı bileşenler üzerinde daha doğru bir şekilde elde ederken, Gabor filtresi yöntemi düşük frekanslı bileşenler üzerinde daha etkili sonuçlar vermektedir. Her iki yöntemin birleştirilmesiyle yüksek ve düşük seviyeli frekansların analiz edilmesi sağlanabilmektedir [13].

2.2.3. Şekil

Şekil bilgisi görüntülerin tanımlamasında kullanılan bir özniteliktir. En-boy oranı, dairesellik, Fourier tanımlayıcıları, sınır bilgilerinin tespiti için kullanılmaktadır [5, 7, 47]. Ayrıca normalize edilmiş hareketsizlik, Zernike momentleri, kenar yönelim histogramları ve kenar haritaları yöntemleri de bu özniteliği elde etmek için kullanılmaktadır [12]. Bölge tabanlı görüntü alma sistemlerinde renk ve doku öznitelikleri kadar yoğun kullanılmamasının yanında bu özniteliğinin elde edilmesi görüntünün bölütlenmesini gerektirdiği için maliyetlidir [9]. Herhangi bir görüntünün içeriği kenarlarıyla ifade edildiğinde tüm görüntü değil sadece anlamlı bölgenin ele alınabilmesi sağlanmaktadır [13].

2.2.4. Uzaysal yerleşim

Uzaysal yerleşim görüntülerdeki alanların ayrımı için kullanılan bir öznitelikdir. Deniz ve gökyüzü örneklerin düşünürse deniz resmin alt tarafında, gökyüzü ise üst tarafında olacaktır. Bu öznitelik için basitçe altta veya üstte şeklinde tanımlamalar yapılabilmektedir [5].

3. GÖRÜNTÜ ÖZETLEME

Benzerlik veya yakınlık arama herhangi bir sorgu örneğine belli bir uzaklık ölçütünü kullanarak bir veri tabanı üzerinde en yakın komşu olarak nitelendirilen örneklerin bulunması problemidir. Veri tabanı boyutu büyük olduğunda en yakın komşuların bulunması için sorgu örneğiyle veri tabanı noktaları arasındaki uzaklığın hesaplanmasının maliyeti ciddi oranda artmaktadır. Herhangi bir q sorgu örneğinin $X = \{x_1, x_2, .., x_n\}$ veri kümesi üzerinde en yakın olanın sorgulanması Eş. 3.1'de gösterildiği şekilde yapılmaktadır. Bu eşitlikte *NN* en yakın komşuyu, *dist* uzaklığı, *argmin* uzaklıklar içinde en küçük olanın seçilmesini ifade etmektedir.

$$\mathrm{NN} = \mathrm{argmin}_{\mathrm{x}\in\mathrm{X}}\, \mathrm{dist}(q, \mathrm{x}) \tag{3.1}$$

En yakın komşu arama yöntemlerinin büyük hacimli veriler üzerindeki doğrusal arama maliyetine alternatif olarak yaklaşık en yakın komşu arama yöntemi birçok problem için yeterli bir çözüm sunmaktadır. Yaklaşık en yakın komşu arama yöntemi düşük depolama maliyetinin yanı sıra hızlı sorgu süresi sağlamaktadır. Özetleme yaklaşımları sorgu örnekleri ile veri tabanı örneklerini birbirine eşleyerek yaklaşık en yakın komşuluk yönteminin etkili ve doğru bir şekilde uygulanmasını sağlamaktadır. Bu yöntemde ilgili komşuların bulunması sabit veya alt doğrusal bir zamanda bulunmaktadır [27, 33, 34].

Özetleme yöntemleri orijinal veri uzayında bulunan örnekler arasındaki uzaklığı koruyarak birbirine yakın örneklerin düşük Hamming uzaklığına sahip olmasını sağlamaktadır [48]. Özetleme yaklaşımında veriler özet denilen düşük boyutlu bit dizileri ile ifade edilmektedir. Benzer verilerin bir araya toplanmasıyla arama zamanının azaltılması hedeflenmektedir. Özetleme yöntemleri genelde özet fonksiyonların öğrenilmesi ve özet kodları için ikili kodların üretilmesi olarak iki aşamadan oluşmaktadır. Literatürde genellikle özet fonksiyonlarının öğrenilmesi aşamasına yoğunlaşılmıştır [4, 19, 27]. Özetleme için doğrusal fonksiyonlar etkili bir şekilde kullanılabilirken, çekirdek fonksiyonları ve en yakın vektör atama tabanlı fonksiyonlar da iyi arama doğruluğu sağlayabilmektedirler [27]. Görüntü ve video gibi büyük hacimli çoklu ortam verilerinin indekslenmesi özetleme yöntemlerinin temel uygulamalarıdır. Bu bağlamda görüntü arama ve görüntü alma alanlarında anlamsal boşluğa rağmen öğrenmeli ve öğrenmesiz özetleme alanlarında yoğun olarak kullanılmaktadır. Bunun yanında mobil ürün arama [48], görüntüde nesne yakalama [49],

görüntü sınıflandırma [50], yüz tanıma [51], nesne takibi [52], kopya algılama [53] gibi alanlarda kullanılmaktadır. Bunun yanında özetleme tabanlı algoritmalar makine öğrenmesi ve veri madenciliği uygulamalarında da kullanılmaktadır.

Herhangi bir örneğe ait olan özet kodu *y*, *h(x)* ile ifade edilebilen bir özet fonksiyonu kullanılarak hesaplanmaktadır. Özetleme yöntemlerinde genellikle özet kodunu hesaplayabilmek için çok sayıda özet fonksiyonu kullanılmaktadır. $\mathfrak{H}$ özet fonksiyon kümesini ifade etmek üzere *n* adet özet fonksiyonu $\mathfrak{H} = \{h_1, h_2, .., h_n\}$ şeklinde kullanıldığında $y = \{y_1, y_2 ,.., y_n\}$ olacak şekilde *n* adet özet biti elde edilmektedir. Herhangi bir örneğin özet kodu aynı zamanda $y = \{h_1(y_1), h_2(y_2), .., h_n(y_n)\}$ şeklinde de ifade edilebilmektedir. Özet kodları bulunarak yapılan arama Hamming uzayında çok hızlı bir şekilde gerçekleştirebilmektedir [2]. Eş. 3.2'de iki örnek arasındaki mesafenin hesaplanması gösterilmiştir.

$$\text{dist}\left(y_i,y_j\right) = \left|y_i - y_j\right| = \sum_{k=1}^{n} \left|h_k(x_i) - h_k\left(x_j\right)\right| \tag{3.2}$$

Özet kodları ile yapılacak yaklaşık en yakın komşu yöntemi özet tablosu arama ve özet kod derecelendirme olarak iki şekilde yapılmaktadır. Özet tablosu arama yönteminde benzer örnekler için özet tablosu denilen bir veri yapısı kullanılmaktadır. Bu yapıda örnekler kova denilen küçük hafıza birimlerinde tutulmaktadır. Bu yapıyı kullanan sistemlerde benzer örneklerin aynı kovada tutulmasıyla en yakın komşuların hızlı bir şekilde elde edilebilmesi sağlanmaktadır. Özet tablosu kullanılarak yapılan özetlemede yakın örneklerin aynı kova üzerinde çakışmalarının en üst düzeyde olması hedeflenmektedir. Bu yaklaşımda genellikle aynı kovalardaki örnekler gerçek uzaklık bilgisi ile yeniden derecelendirilmektedir. Tek bir özet tablo üzerinde özet kodlarını tutmanın alan maliyeti az olmakla birlikte iyi bir hassasiyet değeri için çok sayıda tablonun oluşturulması gerekmektedir. Çok sayıda tablo oluşturma ise hafıza maliyetini artırmaktadır [2, 27]. Genellikle büyük ölçekli veriler için özet kod derecelendirme yerine özet tablo üzerinden yapılan arama bilgi alma başarısı açısından benimsenmektedir [31]. Şekil 3.1'de örnek bir özet tablosu yapısı gösterilmiştir. x_1, x_2, x_3 örnekleri 01110 özet koduna sahiptir ve aynı özet kovasının içinde bulunmaktadır.

Özet Kodu	Veri
01110	x_1, x_2, x_3
11101	x_4, x_5, x_6
01101	x_7, x_8
11100	x_9
…	
10101	$x_{10}, x_{11}, x_{12}, x_{13}$

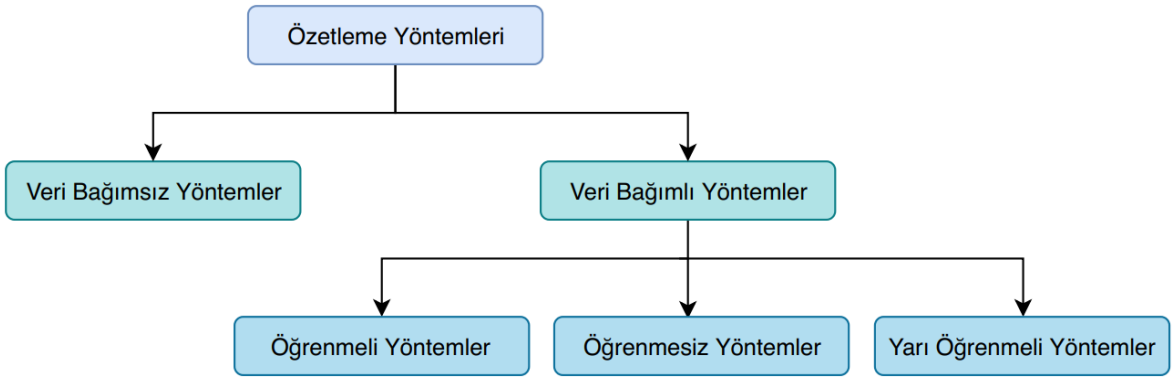
Şekil 3.1. Özet tablosu

Özet kod derecelendirme yönteminde eğitim veri tabanındaki örnekler benzerliklerine göre sıralanmakta ve birbirine yakın olanlar Hamming uzaklığıyla hızlı bir şekilde seçilebilmektedir [36]. Şekil 3.2'de örneklerin sahip oldukları özet kodlarına göre derecelendirilmesi gösterilmiştir.

Veri	Özet Kodu
x_1	01010
x_2	10101
x_3	11001
x_4	11101
…	
x_n	101011

Şekil 3.2. Özet kod derecelendirme

Özetleme yöntemleri veri bağımlı ve veri bağımsız olarak iki sınıfa ayrılmaktadır. Veri bağımsız yöntemlerde özet fonksiyonların üretilmesi için herhangi bir eğitim kümesine ihtiyaç yokken, veri bağımlı yöntemlerde özet fonksiyonların üretilmesi için eğitim kümesi kullanılmaktadır [2, 18, 27, 36].



Şekil 3.3. Özetleme yöntemlerinin sınıflandırılması

3.1. Veri Bağımsız Özetleme

Veri bağımsız yöntemler veri bağımlı yöntemlerle karşılaştırıldığında tatmin edici arama sonuçları elde edebilmek için çok daha uzun özet kodlarına ihtiyaç duymaktadır [30]. Veri bağımsız özetleme genellikle rastgele izdüşümler kullanılarak uygulanmaktadır. Rastgele seçilen özet fonksiyonları veri bağımlı yöntemlerden çok daha hızlı veri işleme süresi sağlamaktadır. Büyük hacimli veri kümeleri için tatmin edici sonuçlara ulaşabilmek için çok sayıda özet tablosunun kullanılması gerekmektedir [3, 4, 16, 28, 54, 55]. LSH özetleme konusunda kullanılan en temel yöntemlerden biridir [18, 21, 25, 29, 38, 42, 56, 57]. LSH yöntemi görüntü alma, nesne tanımlama, görüntü eşleme gibi büyük ölçekli bir çok gerçek dünya uygulamasında kullanılmaktadır [27, 42]. Bu yöntemde özet kodları normal Gauss dağılımına göre oluşturulan rastgele özet fonksiyonları kullanılarak elde edilmektedir. LSH yöntemi benzer örneklere büyük olasılıkla benzer özet kodlarını üretmektedir. LSH yönteminde kullanılan özet fonksiyonları Eş.3.3'de gösterildiği gibi hesaplanmaktadır [2].

$$h(x)=sgn(w^{T}x+b) \tag{3.3}$$

Bu eşitlikte *w* rastgele oluşturulan düzlemi *b* ise rastgele seçilen kesme değerini belirtmektedir. *sgn* ise işaret fonksiyonunu temsil etmektedir. LSH yönteminin bazı dezavantajları bulunmaktadır. Öncelikle veri bağımsız bir yöntem olduğundan yüksek kesinlik değerlerine ulaşılabilmesi için uzun özet kodlarına ihtiyaç duyulmaktadır. Uzun özet kodları kullanılması ise anma değerinin azalmasına sebep olmaktadır. Kesinlik ve anma değerlerini dengelemek için çok fazla sayıda özet tablosunun kullanılması gerekebilmektedir. Özet tablo sayısının fazla sayıda olması ise hafıza gereksinimlerini ve

sorgu süresini artırmaktadır [2]. Bu durum pek çok gerçek dünya uygulaması için sorun oluşturmaktadır [2, 16, 19, 26, 29]. Entropi LSH yöntemi temel LSH yönteminin hafıza gereksinimi azaltmak için önerilmiştir. Bu yöntem LSH yöntemiyle aynı özet fonksiyonlarını kullanmakla beraber benzer örneklerin sorgulanması farklı şekilde yapılmaktadır. Entropi LSH yönteminde orijinal sorguya offset eklenerek elde edilen sorgular kullanılmaktadır. Bu şekilde tüm özet kovalarına bakmaya gerek kalmadan yakın noktalar sorgu örneğiyle aynı özet kovasında bulunan örneklere veya offset eklenmiş sorgu örneğiyle aynı özet kovasında bulunan örneklere bakılarak bulunabilmektedir. Bu şekilde sorgulama bakılacak özet tablosu sayısını azalttığı için sorgu süresini azaltabilmektedir [58].

3.2. Veri Bağımlı Özetleme

Veri bağımlı özetleme yöntemlerinde orijinal veri uzayındaki benzerlikleri ikili veri uzayında da koruyacak özet fonksiyonların üretilmesi amaçlanmaktadır. Bu özet kodların üretilmesi için veri kümesi üzerinde öğrenme süreci uygulanır. Veri bağımsız yöntemlerle karşılaştırıldığında veri bağımlı yöntemler benzer sonuçları çok daha küçük kod boyutuyla sağlamaktadır. Veri bağımlı yöntemler öğrenmeli, öğrenmesiz ve yarı öğrenmeli olarak üç gruba ayrılmaktadır. Son yıllarda yapılan çalışmalarda veri bağımlı yöntemlere odaklanılarak daha efektif özet kodlarını üretecek özet fonksiyonlarının oluşturulabilmesi amaçlanmıştır [1-3, 16, 27, 55].

Öğrenmesiz yöntemler etiketlenmemiş veri üzerinde özet kodları üretmektedir. Bu yöntemde verinin dağılımı ve topolojik özellikleri kullanılır [16, 28]. Öğrenmesiz özet yöntemlerinden bazıları PCAH (Principle Component Analysis Hashing - Temel Bileşen Analizi), ITQ (Iterative Quantization - İteratif Niceleme), SH (Spectral Hashing - Spektral Özetleme) ve AGH (Anchor Graph Hashing - Çapa Graf Özetleme) yöntemleridir.

Öğrenmeli özetleme yöntemleri eğitim verisindeki sınıf veya etiket bilgisini kullanmaktadır. Ayrıca bu yöntemler veri kümesi küçük ve gürültülü olmadığı durumda anlamsal benzerliği de ortaya çıkarabilmektedir. Öğrenmesiz yöntemlerle kıyaslandığında öğrenme süreci yavaştır. KSH (Kernel Based Supervised Hashing - Çekirdek Tabanlı Öğrenmeli Özetleme), RSH (Ranking Based Supervised Hashing - Sıralama Tabanlı Öğrenmeli Özetleme), SDH (Supervised Discrete Hashing - Öğrenmeli Ayrık Özetleme) ve MLH (Minimal Loss

Hashing - En Az Kayıplı Özetleme) yöntemleri öğrenmeli özetleme yöntemlerinin bazılarıdır [16, 28, 56].

Yarı öğrenmeli (semi-supervised) özetleme yöntemleri etiketlenmiş veri kullanıldığında oluşabilecek hataları azaltarak bitlerin birbirinden bağımsız ve dengeli olması gibi istenilen özellikte özet kodlarının seçilmesi hedeflenmektedir. Bu amaçla SSH (Semi Supervised Hashing - Yarı Öğrenmeli Özetleme) yöntemi kullanılmaktadır [28, 56, 59].

3.3. Özet Yöntemleri

PCAH yönteminde rastgele izdüşümlerden daha iyi niceleme elde edilebilmesi amaçlanmaktadır [31, 56, 60]. PCAH orijinal veri uzayındaki en büyük kovaryansın korunmasıyla özet fonksiyonlarını öğrenilmektedir. Bu yöntem veri dağılımına göre benzerlikleri ortaya çıkarmaktadır. PCA yöntemi iyi bir özetleme performansına sahip olsa da veri elde hızı yetersizdir. Genellikle PCA tabanlı yöntemler izdüşüm bulma ve niceleme aşamaları olmak üzerek iki aşamadan oluşmaktadır. İzdüşüm aşamasında, orijinal vektör uzayı üzerinde izdüşüm uygulanarak düşük boyutlu vektör uzayı elde edilmektedir. Niceleme aşamasında bu vektörler eşikleme işlemiyle ikili kodlara dönüştürülmektedir. PCAH yönteminin izdüşüm sonucu elde edilen veri üzerinde varyansı maksimize etme eğilimi nedeniyle veri benzerlikleri tam olarak korunamamaktadır. PCAH yönteminde her bit farklı varyansa sahip olmasına rağmen bitlere aynı ağırlık verilmektedir. Hesaplama ve örnekleme karmaşıklığının yüksek olması başka bir problemdir [1, 19].

SH yöntemi orijinal veri uzayı ile kod uzayı arasındaki benzerlik ve uzaklık çarpımının maksimum olmasını hedeflemektedir. Orijinal uzayda benzerliğin büyük olduğu durumda kod uzayındaki uzaklığın küçük olması istenmektedir [27, 46, 61]. SH kodları dengeli olmasını yani her bitin %50 olasılıkla 1 veya 0 olmasını gerektirmektedir. Ayrıca bitler birbiriyle ilintisizdir [27, 56, 59]. Bu yöntemin amaç fonksiyonu Eş. 3.4'de gösterilmiştir.

$$\sum_{kij} S_{ij}(y(i,k) - y(j,k))^2 \qquad (3.4)$$

Bu formülde *y(i,k)* ve *y(j,k)* izdüşüm uzayındaki *i* ve *j* noktalarını ifade etmektedir. S_{ij}'nin büyük değere sahip olabilmesi için $(y(i,k) - y(j,k))^2$ değerinin küçük olması gerekmektedir

[36]. *N* boyutlu veri için öncelikle PCA yöntemi yardımıyla temel bileşenler bulunmaktadır. Her bir PCA düzlemi için *M* adet 1 boyutlu Laplasiyen öz fonksiyonu (eigen-function) *M* adet en küçük öz değer (eigen-values) kullanılarak hesaplanmaktadır. En küçük öz değere sahip M öz fonksiyonun seçilip sıfır değerine eşiklenerek ikili kodlar elde edilmektedir [27, 28]. SH yöntemi öncelikle temel bileşenleri çıkarmakta sonrasında izdüşürülmüş veriyi açısal frekansa göre ayırmaktadır. Sonrasında PCA yönelimleri boyunca önceden hesaplanmış açısal frekanslara sinüs fonksiyonu uygulanmaktadır [2, 61].

ITQ yöntemi basit ve etkili bir özetleme yöntemidir [62]. Bu yöntem niceleme hatalarını minimize etmeyi amaçlamakta ve izdüşürülmüş verideki yerel yapıları sıfır ortalamalı veriyi döndürerek korumaktadır [3, 62]. Bu yöntem eğitimli veya eğitimsiz olarak kullanılabilmektedir. Öncelikle *M* adet gerçek değerli izdüşüm fonksiyonu $N \times M$ boyutlu *S* matrisi elde etmek için kullanılmaktadır. Sonrasında *S* matrisindeki her bir vektör eşikleme uygulanarak ikili vektörlere dönüştürülmektedir. ITQ yöntemi PCA kullanarak dikey dönme matrisini bulmayı amaçlamaktadır [1]. Özet kodları 100 bitin altında olduğunda tatminkar bir doğruluk sağlayamamaktadır. Orijinal özellikler kullanıldığında elde edilecek performansa benzer performans elde edebilmek için ITQ yöntemi 320 bit gerektirmektedir. 1 milyar görüntü için 320 bit kullanımı yaklaşık 37 GB hafıza kullanımı gerektirmektedir [31].

BRE (Binary Reconstructive Embedding - İkili Yeniden Oluşum Katıştırma) orijinal örnekler arasındaki mesafeyi özet uzayında da korumayı amaçlamaktadır [63]. Bu yöntem çekirdek tabanlı olabilmektedir ve veri dağılımı dikkate alınmamaktadır. Yüksek depolama maliyetinden dolayı BRE yöntemi büyük hacimli veri kümelerinde yetersiz kalabilmektedir [2, 63]. Bu yöntemin amaç fonksiyonu Eş. 3.5'de gösterilmiştir. Bu formülde *K* çekirdek fonksiyonunu temsil etmektedir.

$$\mathrm{sgn}(w^{T}K(x)) \tag{3.5}$$

IH (Isometric Hashing - İzometrik Özetleme) yöntemi orijinal veri uzayında bulunan veri noktaları ve Hamming uzayındaki özet kodları arasında farkı minimize ederek benzerliği korumaktadır. Bu yöntemde ikili uzay ile orijinal veri uzayı arasında yeniden oluşturma hatası minimize edilmesi amaçlanmaktadır. Bu amaçla problem BRE yöntemindeki gibi çok sayıda optimizasyon problemine dönüştürülmektedir. Orijinal problem veri ve vektör uzayı

arasındaki mesafeyi minimize edilmesi ve vektör uzayı ile Hamming vektörleri arasındaki mesafenin minimize edilmesi şekilde iki parçaya ayrılmaktadır [1].

SEH (Sparse Embedded Hashing) yöntemi seyrek kodlama tekniğini kullanarak benzerlik korunması ve doğrusal katılma adımlarını tek bir amaç fonksiyonunda birleştirmektedir. Orijinal Öklid yapısını korumak için matris çarpanlarına ayırma yöntemi kullanılır. SEH eğitilmiş özet fonksiyonlarını kullanarak özet kodlarını elde etmektedir. Bu yöntem anlamsal benzerliği de dikkate almaktadır. Öğrenme zaman karmaşıklığı doğrusal olmakla birlikte diğer veri bağımlı özetleme yöntemleriyle kıyaslandığında ölçeklenebilirdir [35].

AGH (Anchor Graph Hashing - Çapa Graf Özetleme) yöntemi herhangi bir veri dağılımını dikkate almadan öğrenmesiz bir yaklaşımla yaklaşık komşuluk grafını kullanarak özet kodları üretmektedir [22]. Bu yöntemde tüm komşuluk grafını kullanmak yerine bu grafı temsil edebilecek daha az sayıda çapa noktası kullanılmaktadır. Bu yöntem anlamsal benzerlikleri ele alabilmektedir ve kısa özet kodları için iyi sonuçlar verebilmektedir [2, 22, 36].

LSMH (Latent Semantic Minimal Hashing) yöntemi matris çarpanlarına ayırma yöntemiyle veri noktalarını anlamsal kavramla eşleştirmektedir. Bu şekilde benzer anlamsal özelliklere sahip verileri için benzer kodların elde edilebilmesi amaçlanmaktadır [33].

MCR (Min Cost Ranking - En Az Maliyet Sıralama) yöntemi her bir boyut için ikili kodlar üretmektedir ve her bitin ayırt ediciliği belirlenen bir maliyet fonksiyonuna göre karar verilmektedir. Bu yöntemde en son ikili kodların elde edilmesi en az maliyete sahip bitlerin seçilip gruplanmasıyla elde edilmektedir. SSH yönteminden farklı olarak, her bitin öğrenilmesi birbirinden bağımsızdır ve birbirine paralel bir şekilde yapılabilmektedir [31].

Aynı zamanda derin öğrenme tabanlı özetleme alanında yapılan çalışmaların sayısı da artmaktadır. Derin sinir ağları karmaşık veri yapıları üzerinde özniteliklerin öğrenilmesi aşamasında sıklıkla kullanılmaktadır. Bu sinir ağları ilgili öznitelikleri öğrenirken eş zamanlı olarak özet fonksiyonların da ortaya çıkarılmasında kullanılabilmektedir [2]. SH (Semantic Hashing - Anlamsal Özetleme) bu konu üzerinde ortaya konan ilk yöntemlerden birisidir. SH giriş verisi için gizli ikili yapıyı bulmayı hedeflemektedir ve bunun için benzerlik bilgisini kullanmaktadır [2, 18, 36].

DH (Deep Hashing - Derin Özetleme) yöntemi veri üzerindeki çoklu hiyerarşik doğrusal olmayan ilişkileri ortaya çıkaran derin sinir ağları kullanarak özlü özet kodlarını üretebilmek amacıyla önerilmiştir [24, 64].

DVStH (Deep Variational and Structural Hashing) yöntemi derin öğrenme ağlarında gizli öznitelik yapısını ortaya çıkarabilmek için olasılıksal bir yapı sunarak özet fonksiyonlarını oluşturmak amaçlanmaktadır [23].

Merkezi özet yöntemlerinin yanında dağıtık veri üzerinde çalışabilecek yöntemlerin sayısıda gün geçtikçe artmaktadır. LSH yöntemini dağıtık bir şekilde uygulayan bir çalışmada P2P ağlar kullanılmış ve benzer örnekleri tutan benzer özet kovalarını birbirine komşu olacak bir şekilde tutup yapılacak ağ atlama sayısının azaltılması hedeflenmiştir. Bu şekilde k-NN aramanın ilgili uçta yapılması hedeflenmiştir. Ağdaki atlama sayının azaltılmasıyla sorgu süresinin kısaltılması hedeflenmiştir. Bu çalışmada düğümler birbirlerine DHT (Distributed Hash Table – Dağıtık Özet Tablosu) ile bağlanmıştır [26].

Dağıtık katmanlı LSH yönteminde özet kovaları benzer veri noktalarını aynı makinede birbirine benzemeyen noktaları farklı makinede tutacak bir şekilde dağıtılmaktadır. Bu yöntemde yeni bir katman eklenerek yeniden özetleme yapılmaktadır. Entropi LSH yöntemi dağıtık bir şekilde uygulanmaktadır. DHT ve MapReduce yaklaşımları bu uygulamalar için kullanılmaktadır [58].

DisH (Distributed Hashing - Dağıtık Özetleme) yönteminde özet kodların dağıtık bir yolla öğrenilmesi hedeflenmektedir. Vektör niceleme kullanan merkezi sistemlerde her bir veri noktasına bir kod atanmakta ve veri noktasıyla atanan kod arasındaki mesafe minimize edilmeye çalışılmaktadır. Bu çalışmada dağıtık bir yaklaşımla veri noktası ve ikili kod çarpımı kod güncellemesiyle minimize edilmesi amaçlanmaktadır. ADMM (Alternating Direction Method of Multipliers) yöntemi problemi çözmek için kullanılmaktadır. Sorgu örneği q düğümlere gönderilmekte her bir düğümdeki veriye göre benzer örnekler hesaplanmaktadır. Bu çalışma özet kodlarının dağıtık bir şekilde öğrenilebilmesiyle ilgili yapılan ilk çalışmalardan biridir [37].

ABQ (Adaptive Binary Quantization) yöntemi *K*-ortalamalar ve küresel özetleme gibi yetersiz özet kodu üreten prototip tabanlı özet yöntemleri için bir çözüm önermektedir. Prototip tabanlı ikili niceleme yönteminde belli sayıda prototip noktası seçilmekte ve bu noktaların ikili kodları oluşturulmaktadır. Prototip noktalar aynı zamanda özet fonksiyonların öğrenilmesinde ve optimizasyon probleminin çözülmesinde kullanılmaktadır. Önerilen algoritma sadece kısa kodlar üretebildiğinden çarpım niceleme yöntemi kullanılarak uzun kodların üretilebilmesi mümkün hale gelebilmektedir [38].

DGH (Distributed Graph Hashing - Dağıtık Graf Özetleme) yöntemi özet fonksiyonlarını dağıtık bir şekilde öğrenilebilmesi amacıyla önerilmiştir. Bu yöntemde komşuluk graf matrisi yerine ağ içi iletişimi ve hesaplama maliyetini azaltabilmek amacıya çapa grafı tabanlı matrisi önerilmektedir. Çapa noktaları tüm düğümdeki noktalar üzerinden seçilmektedir. Yerel graf matrisi her düğüm için bağımsız olarak üretilmektedir [39].

Özet yöntemleri veri bağımlılıklarına, öğrenme yöntemlerine, özet kod tiplerine, benzerlik ölçütlerine ve özet platformlarına göre Çizelge 3.1'de karşılaştırılmıştır. Bu tez çalışması kapsamında önerilen RDH yöntemi son satırda karşılaştırma amacıyla sunulmuştur.

Çizelge 3.1. Özetleme yöntemleri

Yöntem	Veri Bağımlılığı	Öğrenme Yöntemi	Özet İşleyişi	Özet Platformu
LSH	Bağımsız	-	İzdüşüm	Merkezi
PCAH	Bağımlı	Öğrenmesiz	İzdüşüm	Merkezi
SH	Bağımlı	Öğrenmesiz	İzdüşüm	Merkezi
ITQ	Bağımlı	Öğrenmesiz, Öğrenmeli	Niceleme	Merkezi
SSH	Bağımlı	Yarı-Öğrenmeli	İzdüşüm	Merkezi
BRE	Bağımlı	Öğrenmesiz, Öğrenmeli	İzdüşüm	Merkezi
AGH	Bağımlı	Öğrenmesiz	İzdüşüm	Merkezi
MCR	Bağımlı	Öğrenmesiz	İzdüşüm	Merkezi
DisH	Bağımlı	Öğrenmesiz	Niceleme	Dağıtık
ABQ	Bağımlı	Öğrenmesiz	Niceleme	Dağıtık
SDH/PDH	Bağımlı	Öğrenmesiz	İzdüşüm	Dağıtık
RDH (Önerilen Yöntem)	Bağımsız	-	İzdüşüm	Dağıtık

4. ÖNERİLEN YÖNTEM

Bu bölümde, tez çalışması kapsamında önerilen RDH (Randomized Distributed Hashing – Rastgele Dağıtık Özetleme) yönteminin temelleri ve RDH yöntemi anlatılmıştır. RDH yöntemi temel olarak LSH (Locality Sensitive Hashing – Yerel Hassas Özetleme) yönteminin dağıtık uygulamasıdır. LSH yöntemi literatürde bir çok çalışmada merkezi olarak uygulanmıştır [16, 18, 19, 21, 27, 37-39, 54, 65]. LSH yöntemini RDH yönteminde önerildiği gibi kullanmak eğitim süresini ciddi oranda azalttığı gibi sorgu süresini de azaltmaktadır. Bununla birlikte LSH yöntemiyle benzer başarım elde edilmektedir.

4.1. LSH Yöntemi

LSH yöntemi *n* örnek içeren $X = \{x_1, x_2, .., x_n\}$ veri kümesindeki her bir elemanı *k* adet özet fonksiyonu kullanarak *k*-bit özet koduna eşlemektedir. Her bir özet fonksiyonu veriyi 0 yada 1 değerlerinden birine eşlemektedir. LSH yönteminin önemli özelliklerinden birisi orijinal veri uzayında birbirine yakın olan iki noktanın aynı özet koduna sahip olma ihtimalinin yüksek olmasıdır. *P* olasılık fonksiyonu olmak üzere birbirine benzer örneklerin özet kodlarının aynı olma olasılığı Eş. 4.1'de gösterilmiştir. Bununla birlikte LSH yöntemi büyük bir oranda yerel benzerliği de korumaktadır [34, 59, 66].

$$P(H(x_1)=H(x_2))=\text{Benzerlik}(x_1, x_2) \quad (4.1)$$

LSH yönteminde kullanılan temel özet fonksiyonu Eş. 4.2'de gösterilmiştir. Bu eşitlikte *w* rastgele oluşturulan düzlemi *b* ise rastgele seçilen kesme değerini belirtmektedir. *sgn* ise işaret fonksiyonunu temsil etmektedir [59, 67]. *w* rastgele vektörünün her bir elemanı veri bağımsız olarak standart Gauss dağılımına göre elde edilmektedir [2, 27, 59].

$$H(x)=\text{sign}(w^T x+b) \quad (4.2)$$

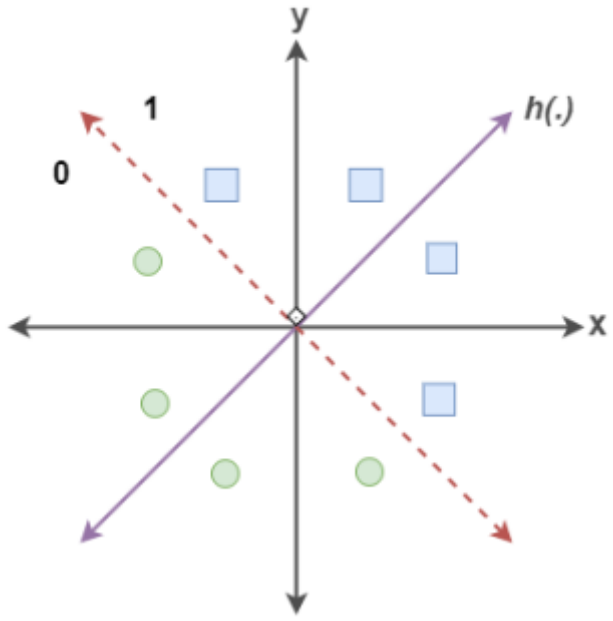
İki vektör arasındaki nokta çarpım değeri Eş. 4.3 veya Eş. 4.4'de gösterildiği şekilde hesaplanabilmektedir. Nokta çarpım değeri skaler bir değerdir. Eş. 4.3 kullanıldığında *x* ve *y* vektörlerinin ilgili sıradaki her bir elemanın çarpımlarının toplanmasıyla bulunmaktadır.

$$x.y = \sum_{i=0}^{n} x_i y_i \tag{4.3}$$

Eş. 4.4 kullanıldığında ise *x* ve *y* vektörlerinin büyüklüklerinin ve vektörler arasındaki açının kosinüs değerinin çarpılmasıyla bulunmaktadır. Nokta çarpım değeri pozitif olduğunda vektörler arasında dar açı, negatif olduğunda ise geniş açı oluşmaktadır. İki vektörün birbirine dik olduğu durumda ise cos(90º) değeri 0 olduğu için nokta çarpım değeri de 0 olmaktadır.

$$x.y = \|x\| \, \|y\| \cos\alpha \tag{4.4}$$

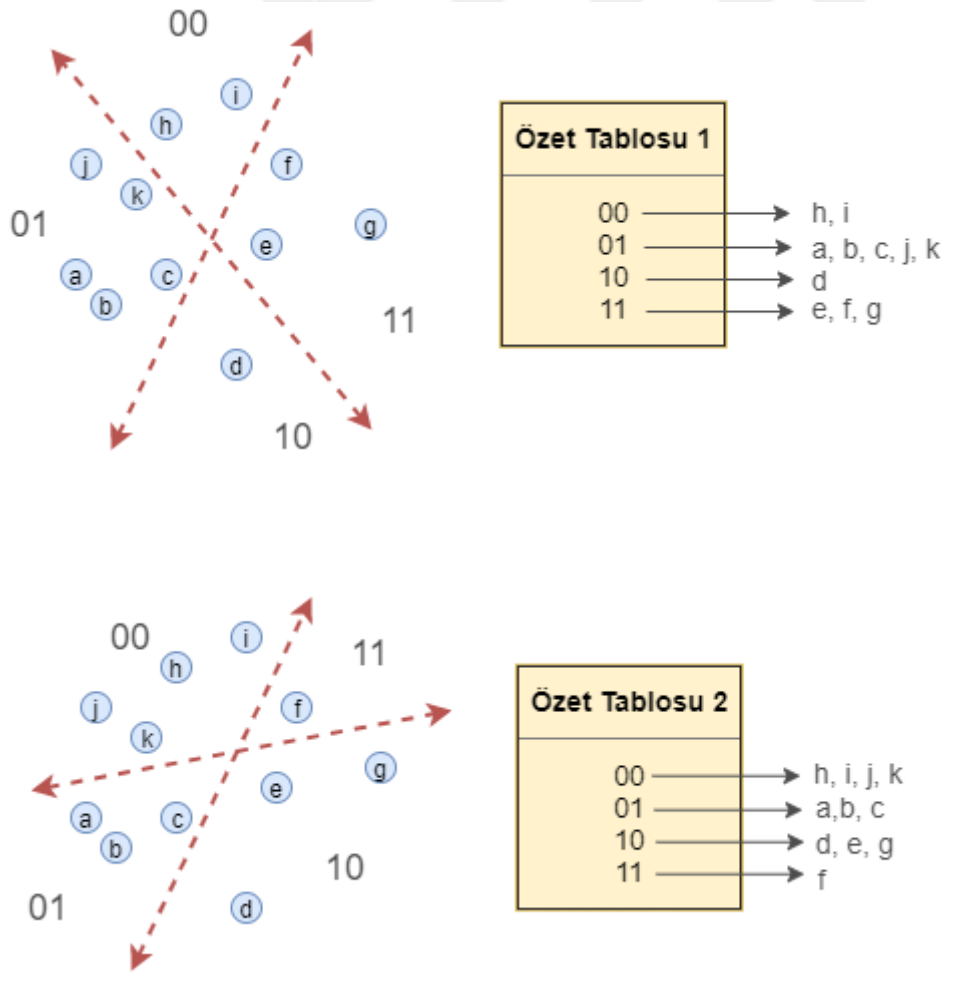
LSH yöntemi nokta çarpım değerinin pozitif veya negatif olmasına göre uzayı iki parçaya ayırmaktadır. Şekil 4.1'de iki boyutlu bir veri uzayı üzerinde *h(.)* özet fonksiyonun veriyi ikiye ayırması gösterilmiştir. Veri kümesindeki örnekler özet fonksiyonu ile nokta çarpım değerlerinin pozitif veya negatif olmasına göre 1 veya 0 değerlerini almıştır. İlgili şekilde mavi ile işaretlemiş örnekler 1 değerini alırken, yeşil ile işaretlenmiş örnekler 0 değerini almaktadır.



Şekil 4.1. Verinin özet fonksiyonu ile ikiye ayrılması

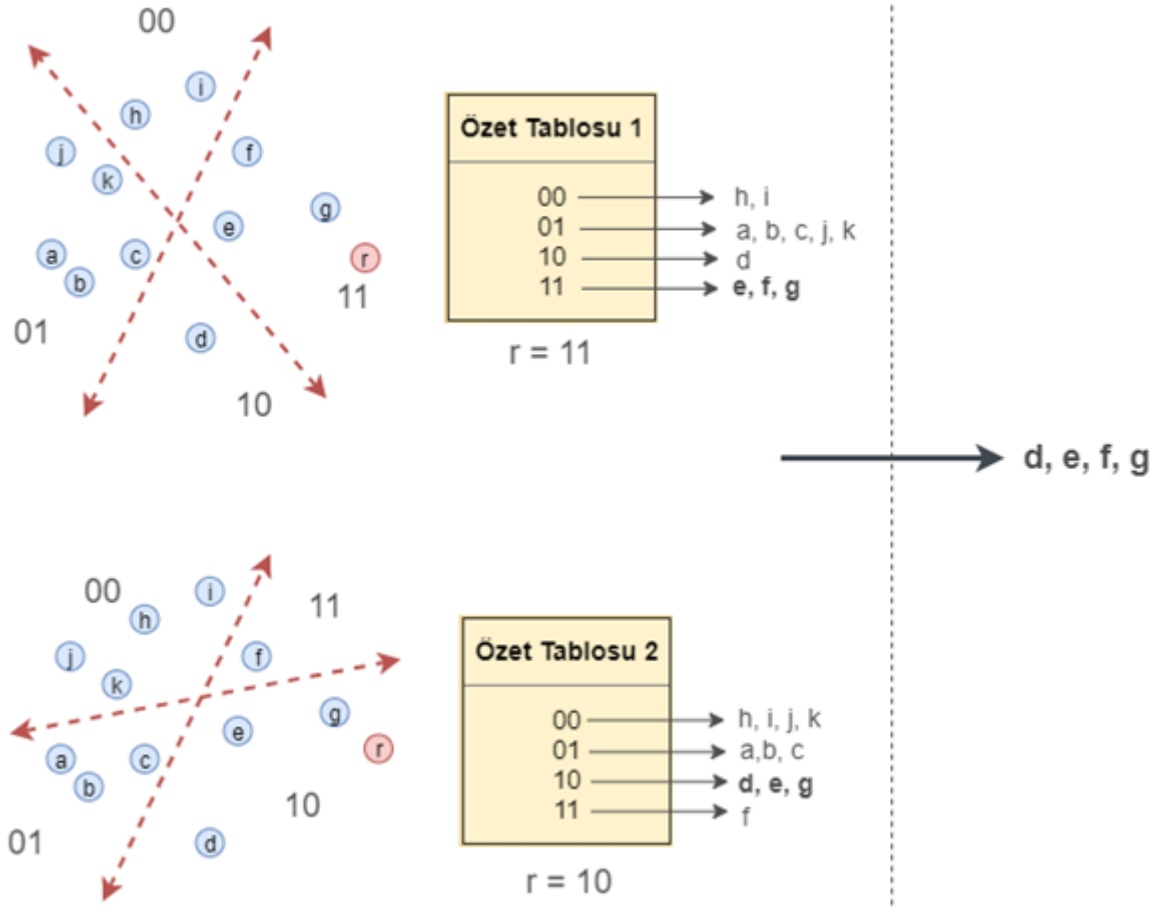
LSH yöntemi benzer noktaların aynı özet koduna sahip olma ihtimalini artırmakla birlikte bu yöntem veri bağımsız bir yöntem olduğu için üretilen özet kodların verimi düşüktür. Bu durum oluşturulan özet kodlarının verinin özelliklerinden bağımsız bir şekilde oluşturulmasından kaynaklanmaktadır. Bunun için genellikle çok sayıda özet tablosu ve uzun özet kodlarının kullanılması gerekmektedir. Bu durum ise hafıza ve hesaplama maliyetini artırmaktadır [16, 27, 28, 59, 66].

LSH yönteminde her bir özet tablosu birden fazla özet kovasından oluşmaktadır. İndekslenen her bir örnek özet tablolar içindeki bir kovaya yerleştirilmektedir [18]. Şekil 4.2'de birden fazla özet tablosu kullanıldığında oluşan özet tabloları gösterilmiştir. Bu şekilde iki özet fonksiyonu kullanıldığı için 2-bit uzunluğunda özet kodları elde edilmiştir. 2-bit özet kodu kullanıldığında ise toplamda 2^2 özet kovası oluşmaktadır. Dolayısıyla her bir özet tablo içerisinde 4 adet özet kovası bulunmaktadır.



Şekil 4.2. LSH yönteminde birden fazla özet tablo kullanımı

Veri kümesi LSH yöntemi kullanılarak indeksleme yapıldıktan sonra herhangi bir örneğin sorgulanması Şekil 4.3'de gösterilmiştir. Burada sorgulanan *r* örneği birinci tablonun oluşturulmasında kullanılan özet fonksiyonları kullanıldığında 11 özet değerini almaktadır. İkinci tablonun oluşturulmasında kullanılan özet fonksiyonları kullanıldığında ise 10 özet değerini almaktadır. LSH yönteminde sorgu aşamasında sırasıyla her bir özet tablosu sorgulanarak sorgulanan örnekle aynı özet koduna sahip örnekler döndürülmektedir.



Şekil 4.3. LSH yönteminde sorgulamanın yapılması

Tablolardan döndürülen değerler küme birleşiminde olduğu gibi birleştirilerek sorgu örneğine yaklaşık olarak en yakın komşular döndürülmektedir. Şekil 4.3'de *r* örneğine birinci tabloda *e*, *f* ve *g* örnekleri; ikinci tabloda ise *d*, *e*, *g* örnekleri yakın olarak belirlenmiştir. Sonuç olarak iki tablodaki benzer örneklerin birleştirilmesiyle *d*, *e*, *f* ve *g* örnekleri sorgu örneğine yakın olarak belirlenmiştir.

4.2. Önerilen Yöntemin Temelleri

Temel LSH yönteminde $\mathfrak{H}= \{h: S \rightarrow U\}$ özet fonksiyon ailesi Eş 4.5'de gösterilen ifade herhangi $q, v \in S$ noktaları ve $h(q), h(v) \in U$ özet kodları için sağlandığında (r_1, r_2, p_1, p_2) hassas olarak nitelendirilmektedir [18, 21, 34].

$$\begin{aligned} &\textit{if } \text{dist}(q, v) \leq r_1 \textit{ then } \Pr_H(h(q) = h(v)) \geq p_1 \\ &\textit{if } \text{dist}(q, v) > r_2 \textit{ then } \Pr_H(h(q) = h(v)) \leq p_2 \end{aligned} \tag{4.5}$$

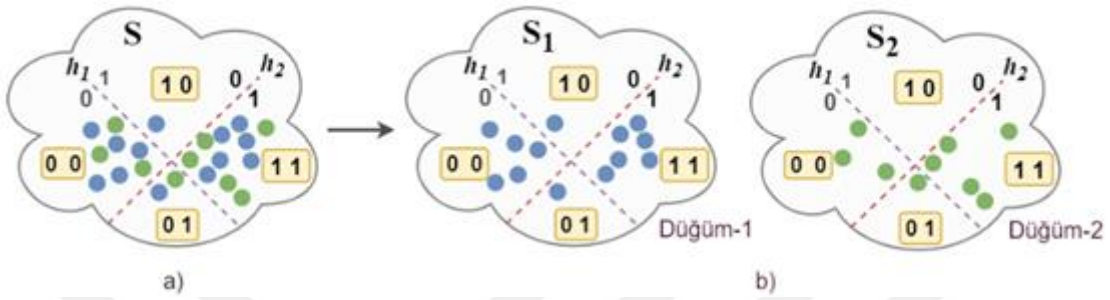
S ifadesi uzaydaki veri noktalarını ve *dist* ifadesi bu uzaydaki tanımlanmış uzaklık fonksiyonunu ifade etmektedir. Ayrıca $p_1 > p_2$ ve $r_1 < r_2$ eşitsizliklerinin sağlandığı kabul edilmektedir [26, 34].

Eş. 4.5 kullanıldığında *S* veri kümesi S_1 and S_2 olarak eşit sayıda eleman içeren iki alt kümeye bölündüğünde ve özet fonksiyon ailesi $\mathfrak{H}_1 = \{h_1: S_1 \rightarrow U_1\}$ ve $\mathfrak{H}_2 = \{h_2: S_2 \rightarrow U_2\}$ Eş. 4.6'daki belirtilen durumlar sağlandığında ayrı ayrı (r_1, r_2, p_1, p_2) ve (r_3, r_4, p_3, p_4) hassas olarak değerlendirilmekte, herhangi bir sorgu örneği *q* ve eğitim noktaları $v_1 \in S_1$, $v_2 \in S_2$ için de Eş. 4.6'da belirtilen durumlar sağlanmaktadır.

$$\begin{aligned} &\textit{if } \text{dist}(q, v_1) \leq r_1 \textit{ then } \Pr_H(h_1(q) = h_1(v_1)) \geq p_1 \\ &\textit{if } \text{dist}(q, v_1) > r_2 \textit{ then } \Pr_H(h_1(q) = h_1(v_1)) \leq p_2 \\ &\textit{if } \text{dist}(q, v_2) \leq r_3 \textit{ then } \Pr_H(h_2(q) = h_2(v_2)) \geq p_3 \\ &\textit{if } \text{dist}(q, v_2) > r_4 \textit{ then } \Pr_H(h_2(q) = h_2(v_2)) \leq p_4 \end{aligned} \tag{4.6}$$

v_1, v_2 noktalarının aynı noktalar; $\mathfrak{H}_1$ ve $\mathfrak{H}_2$ özet fonksiyonlarının aynı özet fonksiyonları olduğunu varsayıldığında p_1 değeri p_3 değerine, p_2 değeri p_4 değerine eşit olacaktır. Aynı zamanda r_1 değeri r_3 değerine ve r_2 değeri r_4 değerine eşit olacaktır. Sonuç olarak her bir alt kümede aynı özet fonksiyonları kullanıldığından $U_1 \cup U_2$ kümesi *U* kümesine eşit olacaktır. Aynı özet fonksiyonlar kullanıldığında verinin hangi düğümde olduğunun herhangi bir önemi olmamaktadır. Bu yolla örneğin bulunabileceği özet kovalar değiştirilmemektedir. Bir kovada olabilecek eleman sayısı azaltılmakta ve kova içeriği farklı düğümlere dağıtılmaktadır. Bu şekilde sistem paralel bir şekilde eğitilmekte ve sorgulanabilmektedir. LSH yönteminde sorgu aşamasında çok sayıda özet tablosu kullanmaktadır ve yaklaşık en yakın komşulara erişim sabit ya da alt doğrusal zamanda olmaktadır. Çok sayıdaki özet

tablosundan yaklaşık en yakın komşular elde edildikten sonra istenilen sayıda komşuyu elde edebilmek için k-NN (k-Nearest Neighbor – k-En Yakın Komşuluk) aramanın yapılması gerekmektedir. Bu işlemi daha az veriyle paralel bir şekilde gerçekleştirmek sorgu zamanını ciddi oranda azaltmaktadır.



Şekil 4.4. Verinin düğümlere dağıtılması ve özet fonksiyonların kullanımı

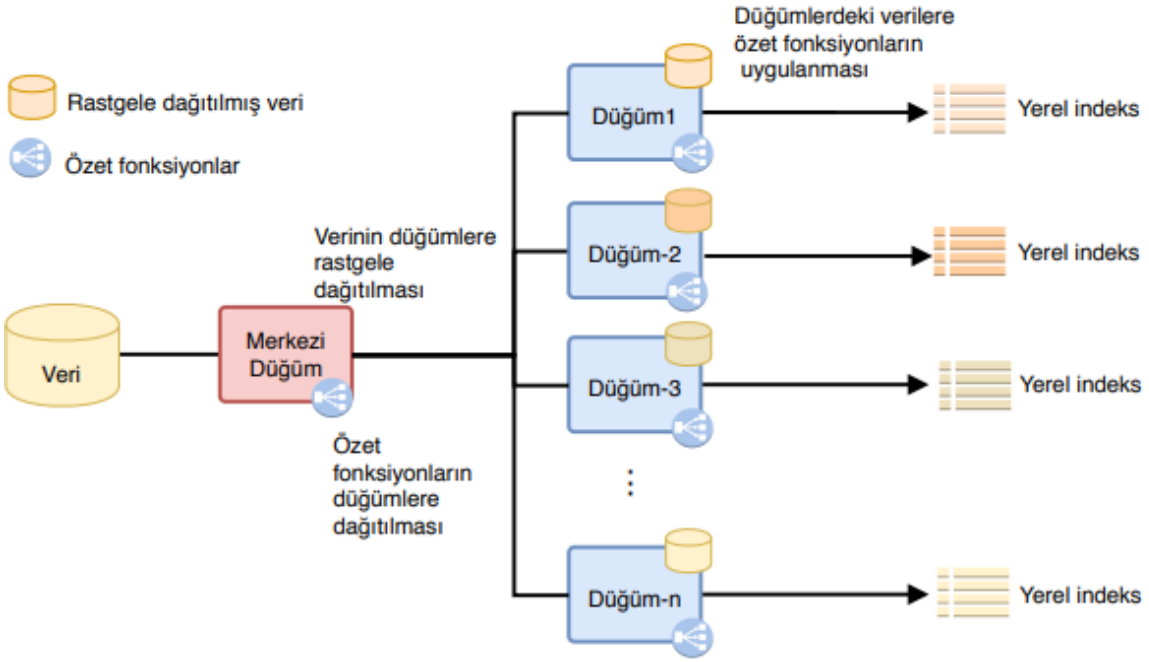
S veri kümesi öznitelik uzayında Şekil 4.4'de gösterilmiştir. Şekil 4.4.a'da tüm verinin tek bir düğümde tutulmasını ifade edilmektedir. h_1 ve h_2 özet fonksiyonları *S* veri kümesi üzerinde uygulandığında her bir örnek için 2-bit özet kodu elde edilmektedir. Şekil 4.4.b'de *S* veri kümesi S_1 and S_2 olarak iki alt kümeye ayrılmakta ve her bir alt küme bir düğüme dağıtılmaktadır. Her düğümde h_1 ve h_2 özet fonksiyonu kullanıldığında düğümlerdeki her bir örnek için Şekil 4.4.a'daki özet kodlarının aynısı elde edilmektedir. Sorgulama bu şekilde yapıldığında her bir düğüm daha az veriyle eş zamanlı olarak çalışabildiğinden sorgu zamanı azalmaktadır. Benzer şekilde eğitim süresi de ciddi oranda azalmaktadır. Bu tez çalışmasında önerilen yöntemin temelini bu yaklaşım oluşturmaktadır. Bu yaklaşım uygulanırken her bir düğümde tüm komşularının döndürülmesi yerine her bir düğümde elde edilen ilk *n* yaklaşık en yakın komşular döndürülerek bu komşularla çalışılmıştır.

4.3. Önerilen Yöntem

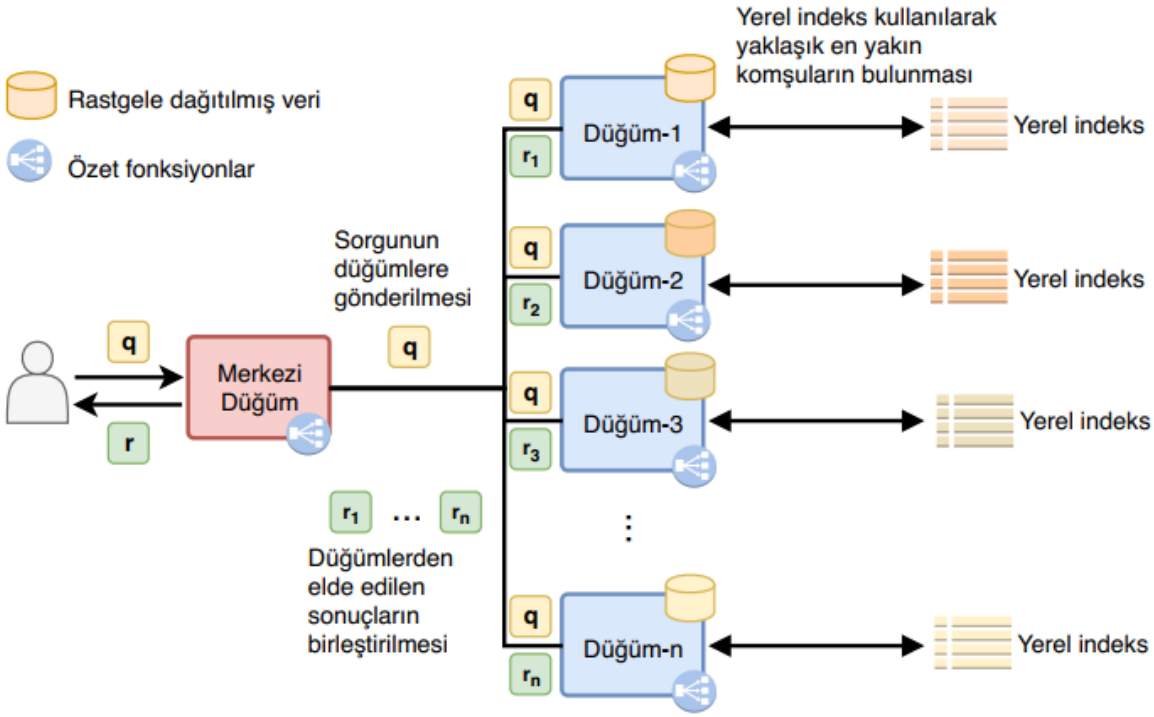
Bu bölümde önerilen yöntem sunulmuştur. Çok sayıda hesaplama düğümün olduğu ve bu düğümlerin birbirine bağlı olduğu varsayılmıştır. Tüm düğümlerde aynı özet fonksiyonlar kullanılmıştır. Bununla birlikte karşılaştırma amacıyla birbirinden farklı düğümlerde birbirinden farklı özet fonksiyonları da kullanılmıştır. Bu çalışmayla ilgili deneysel çalışmalara ve sonuçlara sonraki bölümde yer verilmiştir. Veri kümesinin sıfır ortalamalı olduğu varsayılmıştır [37-39].

n veri kümesindeki eleman sayısını, d her bir elemanın boyutunu ifade etmek üzere X ifadesinin $n \times d$ boyutlu veriyi ifade etmektedir ve $X \in \mathbb{R}^{n \times d}$. X kümesi eşit boyutlu m adet alt kümeye bölündüğünde aynı zamanda $X = [x_1, x_2, x_3, ..., x_m]$ şeklinde tüm alt kümelerin birleşimi şeklinde de ifade edilebilmektedir. Her bir alt küme x_i , n / m adet veri içermektedir. Bu alt kümeler rastgele olarak birbirinden farklı düğümlere dağıtılabilir. Tüm verinin tek bir merkezi düğümde indekslenmesi yerine, alt kümeler birbirinden ayrı olmak üzere dağıtık düğümlerde de bu verinin tamamı indekslenebilmektedir.

İndeksleme aşamasında merkezi düğümde özet fonksiyon kümesi oluşturulmakta ve bu fonksiyonlar diğer düğümlere gönderilmektedir. Her bir düğüm aynı özet fonksiyonlarına sahip olduğu için birbirleriyle aynı şekilde veriyi indeksleyebilmektedirler. Sorgu aşamasında sorgu tüm düğümlere dağıtılmakta ve her bir düğüm n adet yaklaşık en yakın komşuyu bularak merkezi düğüme göndermektedir. Bu çalışmada deneysel çalışmalar için n değeri 100 olarak seçilmiştir. Merkezi düğüme sadece vektörü tanımlayıcı ve uzaklık bilgisi gönderildiğinden ağ yükü azaltılmaktadır. Düğümlerden elde edilen sonuçlar merkez düğümde birleştirilerek nihai sonuçlar elde edilmektedir. Bu şekilde merkezi şekilde kurgulanmış sistemlerle benzer sonuçlar dağıtık sistemlerin avantajıyla daha iyi sorgu süresiyle elde edilmektedir. LSH yönteminde sorgu örneğine en yakın örneklerin bulunabilmesi için sorgu örneğiyle aynı kovada bulunan örnekler toplanmakta daha sonra bu örnekler üzerinde k-NN arama yapılmaktadır. Veri rastgele olarak düğümlere dağıtıldığında LSH yönteminin daha az veriyle aynı şekilde çalışması sağlanabilmektedir. Bu durum daha iyi bir sorgu süresi sağlamanın yanı sıra eğitim süresinin de ciddi oranda azalmasını sağlamaktadır. Önerilen yöntemin eğitim aşaması Şekil 4.5'de, sorgu aşaması ise Şekil 4.6'de gösterilmiştir.



Şekil 4.5. Verinin düğümlere dağıtılmasıyla düğümlerde yerel indeksin oluşturulması



Şekil 4.6. Sorgu örneğinin düğümlere gönderilmesi ve sonuçların merkezi düğümde birleştirilmesi

RDH yönteminde kullanılan algoritma Çizelge 4.1'de gösterilmiştir. Eğitim aşamasında tüm veri kümesi X, m adet parçaya bölünmekte ve her bir parça bir düğüme gönderilmektedir. Merkezi düğümde oluşturulan $\mathfrak{H}$ özet fonksiyon kümesi de benzer şekilde düğümlere

gönderilmektedir. Her bir düğüm LSH yöntemini merkezi düğümde oluşturulan özet fonksiyonlarını kullanarak sahip olduğu veri üzerinde uygulamaktadır. Bu işlemin sonunda tüm veri dağıtık bir şekilde indekslenmektedir ve artık sistem sorgu aşamasına hazır halde bulunmaktadır.

Çizelge 4.1. RDH yönteminin eğitim algoritması

Algoritma 1: RDH Eğitim
Giriş: Eğitim verisi X, özet fonksiyon kümesi $\mathfrak{H}$, düğüm sayısı m 1. X'i m parçaya böl 2. for i = 1, …, m do 3. X_i'i i. düğüme gönder 4. $\mathfrak{H}$'yi kullanarak LSH uygula 5. end for Çıktı: Her bir düğüm içinde özetlenmiş veri

LSH yönteminin algoritması Çizelge 4.2'de gösterilmiştir. Bu algoritmada eğitim verisindeki her bir örnek her bir özet tablosu için belirlenen rastgele özet fonksiyonlar kullanılarak özetlenmekte ve ilgili özet tablosuna yerleştirilmektedir.

Çizelge 4.2. LSH yönteminin algoritması

Algoritma 2: LSH
Giriş: Eğitim verisi X, eğitim verisindeki örnek sayısı s, özet tablo sayısı t, özet fonksiyon sayısı f 1. for i = 1, …, s do 1. for j = 1, …, t do 2. Rastgele özet fonksiyon oluştur 2. X_i'yi rastgele fonksiyonu kullanarak özetle 3. Oluşan özeti j. tabloya yerleştir 4. end for Çıktı: Özetlenmiş veri

Sorgu aşamasında sorgu örneği q tüm düğümlere gönderilmekte ve her bir düğümde eş zamanlı olarak sorgulanmaktadır. Sonrasında bulunan yaklaşık en yakın komşular merkezi düğüme gönderilmektedir. Son olarak düğümlerden alınan yerel sonuçlar birleştirilmekte ve nihai yakın örnekler elde edilmektedir. Uygulamamızda düğümler tüm yaklaşık en yakın komşular yerine belli sayıdaki komşuyu merkezi düğüme göndermişlerdir. Bu şekilde düğümlerden gelen sonuçların birleştirilmesi sorgu süresine oranla çok kısa sürmüştür.

Çizelge 4.3. RDH yönteminin sorgulama algoritması

Algoritma 3: RDH Sorgu
Giriş: Sorgu örneği *q* 1. *q* örneğini *m* düğüme gönder 2. Her düğümde *q* örneğini eş zamanlı sorgula 3. Yaklaşık en yakın komşuları merkezi düğüme gönder 4. Düğümlerden gelen komşuları birleştir 5. *k*-NN arama ile nihai komşuları bul Çıktı: Sorgu örneğine benzeyen örnekler

LSH yöntemi *d* boyutlu *N* eleman içeren veri kümesi *X* üzerinde *k* adet özet fonksiyonu kullanılarak uygulandığında uygun kovaların bulunması her bir eleman ve özet fonksiyonu arasında nokta çarpım işlemi uygulandığından elde edilecek süre Eş. 4.7'de gösterilmiştir.

$$d \times k \tag{4.7}$$

Tek bir özet tablosu için *k* özet fonksiyonu 2^k özet kovası oluşturacağından her bir kova içindeki karşılaştırma maliyeti Eş. 4.8'de gösterilmiştir.

$$\frac{d \times N}{2^k} \tag{4.8}$$

T adet özet tablosu kullanıldığında toplam hesaplama maliyeti Eş. 4.9'da gösterilmiştir.

$$T \times d \times k + T \times \frac{d \times N}{2^k} \tag{4.9}$$

k değerinin *log N* ifadesine eşit olduğu varsayıldığında her bir kova içerisinde sabit sayıda karşılaştırma yapılmaktadır ve algoritmanın hesaplama karmaşıklığı *O(log N)* olmaktadır. Hesaplama karmaşıklığı *N*'ye bağlıdır [18, 54].

Önerdiğimiz yöntem *N* değerini *m* parçaya böldüğü için her bir düğümdeki sorgu süresinin karmaşıklığı Eş. 4.10'da gösterilmiştir.

$$O\left(\log \frac{N}{m}\right) \tag{4.10}$$

5. BENZERLİK ve PERFORMANS ÖLÇÜTLERİ

Görüntülerin birbirleriyle olan benzerlikleri görüntünün bir bölümünü ya da görüntünün tamamı kullanılarak hesaplanabilmektedir. Öklid, Manhattan, Canberra, Angular, Czekanowski, iç çarpım ve kosinüs katsayıları gibi ölçütler benzerliğin belirlenebilmesi için kullanılmaktadır. *n* boyutlu $x = [x_1, x_2, ..., x_n]$ ve $y = [y_1, y_2, ..., y_n]$ görüntüleri için Minkowski ölçütünün hesaplanması Eş. 5.1'de gösterilmiştir. Bu eşitlikte iki vektörün aynı sıradaki elemanlarının farkı alınarak toplanmaktadır. Sonrasında eşitlikte belirtilen *r* değerine göre kök alınarak uzaklık bulunmaktadır. $r = 2$ olduğu durumda Öklid, $r = 1$ olduğu durumda ise Manhattan uzaklığı bulunmaktadır [5, 6].

$$d(x,y)=\left(\sum_{i=1}^{n}|x_i-y_i|^r\right)^{1/r} \tag{5.1}$$

Görüntü alma sistemlerinde performans genellikle kesinlik (precision) ve hassasiyet (recall) veya MAP (Mean Average Precison - Genel Ortalama Kesinlik) değerlerine göre değerlendirilmektedir [1, 3, 5, 6, 12, 14-16, 27, 30, 35-40, 62, 65].

5.1. Kesinlik

Kesinlik (precision) değeri, herhangi bir sorgu örneği için bulunan ilişkili örnek sayısının, sorgu sonucu bulunan toplam örnek sayısına oranlanmasıyla bulunmaktadır. Eş. 5.2'de kesinlik değerinin hesaplanması gösterilmiştir [7, 40, 41, 47].

$$\text{Kesinlik}=\frac{\text{Bulunan ilişkili nesne sayısı}}{\text{Toplam bulunan nesne sayısı}} \tag{5.2}$$

5.2. Hassasiyet

Hassasiyet (recall) değeri, herhangi bir sorgu örneği için bulunan ilişkili örnek sayısının, veri kümesinde toplamda var olan ilişkili örnek sayısına oranlanmasıyla bulunmaktadır. Eş. 5.3'de hassasiyet değerinin hesaplanması gösterilmiştir [7, 40, 41, 47].

$$\text{Hassasiyet}=\frac{\text{Bulunan ilişkili nesne sayısı}}{\text{Toplamda var olan ilişkili nesne sayısı}} \tag{5.3}$$

5.3. MAP (Mean Average Precision)

Görüntü üzerinde yapılan özetleme yöntemlerinin değerlendirilmesinde performans için kesinlik ve hassasiyet değerlerinin yanı sıra her bir sorgunun ortalamasını veren MAP (Mean Average Precison - Genel Ortalama Kesinlik) yöntemi de birçok çalışmada yoğun olarak kullanılmıştır [1, 3, 16, 27, 35, 36, 38, 39, 62, 65]. MAP yönteminde her bir sorgu için ortalama kesinlik değeri hesaplanmaktadır. Bu değer aynı zamanda kesinlik ve hassasiyet grafiğinin altında kalan alan olarak da ifade edilebilmektedir [3, 65]. Yüksek MAP değeri benzerlik sıralaması için iyi performans değerini ifade etmektedir. MAP değerinin hesaplanması için genellikle sorgu örneğiyle orijinal veri uzayındaki örneklerin Öklid uzaklığına göre sıralanmasıyla bulunan komşular referans olarak kullanılmaktadır [19, 31, 37-39, 62, 65]. MAP değerinin hesaplanması Eş. 5.4'de gösterilmiştir.

$$\text{MAP}=\frac{1}{k}\sum_{i=1}^{k}\text{AP(i)} \tag{5.4}$$

MAP değeri her bir görüntü için hesaplanan AP (Average Precision - Ortalama Kesinlik) değerlerinin ortalaması alınarak bulunmaktadır. AP değerinin hesaplanması Eş. 5.5'de gösterilmiştir. Bu eşitlikte *S* ifadesi sorgulama sonucu elde edilen kümedeki sorgu örneğiyle ilişkili örnek sayısını ifade etmektedir. P_r ifadesi sorgulama sonucu elde edilen kümedeki sorgu örneğiyle ilişkili ilk *r* örneğin kesinliğini ve I_r ifadesi ise *r*. sıradaki örneğin ilgili olup olmamasına göre 1 veya 0 değerlerinden bir tanesini veren bir fonksiyonu ifade etmektedir [35].

$$\text{AP(i)}=\frac{1}{S}\sum_{r=1}^{R}\text{P}_r\text{(i)I}_r\text{(i)} \tag{5.5}$$

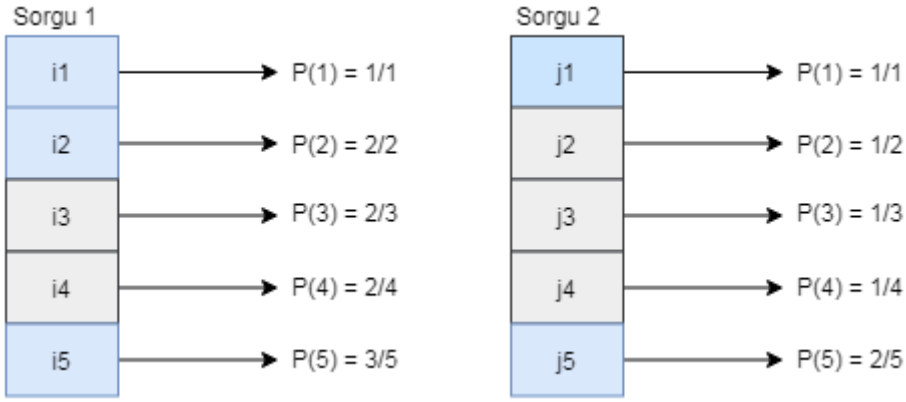
Şekil 5.1.'de MAP değerinin hesaplanması iki sorgu örneği kullanıldığı durum için örneklendirilmiştir. Şekilde sorgu örneğiyle ilişki örnekler mavi renk ile gösterilmiştir. Buna göre 1. sorgu örneği için ortalama kesinlik değeri 0,86 olarak bulunmuştur. Sorgu örneğiyle

ilişkili örnek sayısı olan *S* ifadesi 3 olmuştur. I_r fonksiyonu 3. ve 4. sıradaki örnekler sorgu örneğiyle ilgisiz olduğundan 0 değerini almış ve dolayısıyla bu sıradaki örnekler AP hesaplamasına katılmamıştır.

$AP_1 = [\ P(1) + P(2) + P(3)\ ] / 3 = (1 + 1 + 0.60)\ /\ 3 = 0{,}86$

Benzer şekilde 2. sorgu örneği için ortalama kesinlik değeri 0,7 olarak bulunmuştur. Bu sorgu için formülde belirtilen *S* ifadesi ilişkili iki örnek bulunduğundan 2 olmuştur. I_r fonksiyonu 2., 3. ve 4. sıradaki örnekler sorgu örneğiyle ilgisiz olduğundan 0 olmuş ve bu sorgu için bu sıradaki örnekler AP hesaplamasına katılmamıştır.

$AP_2 = [\ P(1) + P(5)\ ] / 2 = (1 + 0{,}40)\ /\ 2 = 0{,}70$



Şekil 5.1. Her sorgu için ortalama kesinlik bulunması

İki sorgu örneği için MAP değeri ise her iki sorgunun AP değerlerinin ortalaması olan 0,78 değeri olmuştur.

$MAP = (AP_1 + AP_2\ ) / 2 = (0{,}86 + 0,\ 70) / 2 = 0{,}78$

MAP değerinin hesaplanması veri kümesi çok büyük olduğunda fazla zaman alan bir işlem olabilmektedir. Bu yüzden çoğu çalışmada bu yöntem ilk sıralarda bulunan belli sayıda örneğe bakılarak hızlı bir şekilde hesaplama yapılması şeklinde kullanılmıştır. Literatürde genellikle ilk 100, 500 veya 1000 örneğe bakılarak elde edilen sonuçlar MAP@100, MAP@500, MAP@1000 şeklinde ifade edilmektedir [16, 37-39].

5.4. Hızlanma

Hızlanma (speedup) aynı iş için daha fazla sayıda düğüm kullanıldığında sistemin etkinliğini ölçen bir performans ölçütüdür [38]. *n* düğüm sayısı olmak üzere hızlanma değerinin hesaplanması Eş. 5.6'da gösterilmiştir.

$$\text{Hızlanma}(n)=\frac{\text{Bir düğüm için eğitim süresi}}{n \text{ düğüm kullanıldığında eğitim süresi}} \tag{5.6}$$

Sistemin hızlanma oranı tek düğüm için elde edilen eğitim süresinin *n* adet düğüm kullanıldığında elde edilen eğitim süresine oranlanmasıyla bulunmaktadır. Hızlanma performansının artan düğüm sayısıyla doğrusal olarak artması, düğüm sayısının artırılmasının hesaplama işlemini dengeleyebileceğini ve toplam eğitim süresini azaltılabileceğini göstermektedir [38].

5.5. Genişleme

Genişleme (sizeup) *m* boyutlu bir veri kümesi *k* kat daha büyüdüğünde eğitim süresinin ne kadar süreceğini ölçen bir performans ölçütüdür [38].

$$\text{Genişleme}(k)=\frac{m \times k \text{ eğitim süresi}}{m \text{ örnek için eğitim süresi}} \tag{5.7}$$

Sistemin genişleme oranı veri boyutu *k* kat artırıldığında elde edilen eğitim süresinin veri boyutu artırılmadan elde edilen eğitim süresine oranlanmasıyla bulunmaktadır. Artan eğitim örnek sayısıyla beraber eğitim süresinin doğrusal olarak artması beklenmektedir [38].

6. DENEYSEL ÇALIŞMALAR

Bu tez çalışmasında önerilen yöntemin başarısını değerlendirebilmek için farklı veri kümeleri üzerinde çok sayıda deneysel çalışma yapılmıştır. Önerilen RDH yöntemi için birden fazla sayıda düğüm kullanılmış ve her bir durumda veri düğümlere eşit ve rastgele olarak dağıtılmıştır. Deneysel çalışmalar için Corel-10K, SIFT-1M ve GIST-1M veri kümeleri kullanılmıştır. Önerilen yöntem Java 1.8 versiyonu kullanılarak uygulanmıştır. Dağıtık ağ yapısı Apache Ignite 2.7.0 kullanılarak oluşturulmuştur. Intel Core i7 işlemciye ve 32 GB ana belleğe sahip olan bir bilgisayar üzerinde çalışılmıştır. LSH yönteminin yanı sıra SH, PCAH ve ITQ yöntemleri de aynı geliştirme ortamında Java ile yeniden kodlanmış ve sonuçlar elde edilmiştir. Sistemin testi için Corel-10K ve SIFT-1M veri kümeleri kullanıldığında rastgele 100 adet test örneği seçilmiş, GIST-1M kullanıldığında ise rastgele 1000 adet test örneği seçilmiştir.

Özet tablo arama yaklaşımı benimsenerek özetleme yapılmıştır. SH, PCAH ve ITQ yöntemleri veri bağımlı yöntemler olduklarından bu yöntemler için tek bir özet tablosu kullanılmıştır. Deneysel çalışmalarda 8, 16, 24, 32 and 64-bit özet kodları kullanılmıştır. Önerilen yöntemin performans değerlendirmesinde Corel-10K ve SIFT-1M veri kümeleri için en yakın 100 komşu için hesaplanan MAP@100 değeri, GIST-1M veri kümesi için ise en yakın 1000 komşu için hesaplanan MAP@1000 kriteri kullanılmıştır. Her bir sorgu örneği için Öklid uzayındaki en yakın komşuları gerçek referans olarak kullanılmıştır. Literatürde referans olarak kullanılacak en yakın komşuların belirlenmesinde Öklid uzayı yaygın bir şekilde kullanılmaktadır [37, 39, 64].

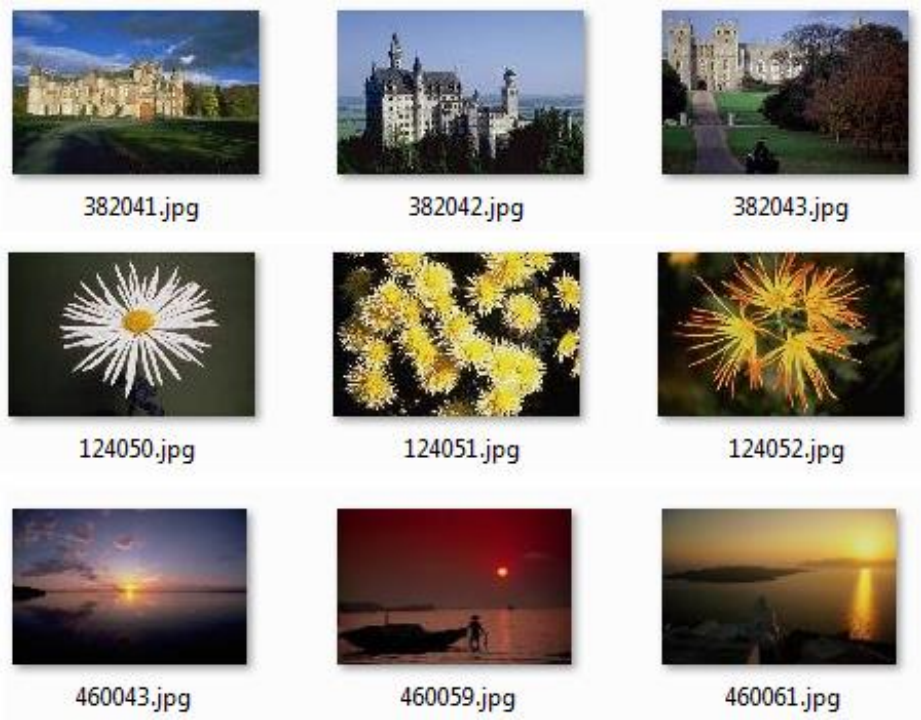
Özetleme işleminden önce veri ilk olarak her bir düğüm eşit sayıda ve rastgele örnek içerecek şekilde düğümlere dağıtılmaktadır. Bu çalışmada veriyi indekslemek için tüm düğümlerde aynı özet fonksiyon kümesinin kullanılmasının yanı sıra tüm düğümlerde birbirinden farklı özet fonksiyon kümesi de kullanılmıştır. Ayrıca düğümlerde veriyi yaklaşık olarak ikiye ayıran özet fonksiyonları kullanılarak ve her bir özet kovası içerisinde bir özetleme işlemi daha gerçekleştiren hiyerarşik bir şekilde özetleme yapılarak deneysel sonuçlar elde edilmiştir.

6.1. Görüntü Veri Tabanları

Görüntü alma ve arama üzerine yapılan çalışmalarda çok sayıda veri tabanı kullanılmaktadır. Bu veri tabanları benzer şekilde özetleme yöntemlerinin performansının test edilmesi için de kullanılmaktadır. Veri tabanı içindeki örneklerin büyük bir kısmı sistemin eğitimi için kullanılırken bir kısmı da sistemin testi için kullanılmaktadır. Özetleme yöntemlerinin test edilebilmesi için genellikle sınama örneklerinin Öklid uzaklığına göre en yakın komşuları referans olarak kullanılmaktadır. Bir çok çalışmada sınama örneklerinin Öklid uzaklığına göre en yakın %2 oranındaki komşularına bakılarak referans kümesi oluşturulmaktadır [37, 39, 65]. Bazı çalışmalarda ise bu değer veri kümesinin boyutuna bağlı olarak 100, 500 ve 10.000 gibi sabit bir değer olarak seçilmektedir [38].

Corel-10K veri kümesi sınıf çeşitliliğinin çok olması ve fazla sayıda örnek içermesinden dolayı literatürde yoğun kullanılan bir veri kümesidir. Corel-10K veri kümesi 9902 görüntü içermektedir ve bu veri kümesinde 80 sınıf bulunmaktadır [5, 7, 12, 26, 32, 41]. Bu veri kümesinde bazı benzer görüntüler farklı sınıflarda da yer alabilmektedir. Ayrıca bazı sınıfların çok soyut kalması da problemli bir durum olmakla birlikte genellikle görüntü alma işlemlerinden önce ön işleme adımının uygulanması gereklidir [5]. Şekil 6.1'de Corel veri kümesinde farklı sınıflarda bulunan görüntü örnekleri gösterilmiştir.

CIFAR-10 veri kümesi 32×32 boyutlu 10 sınıfa ait 60 bin görüntü içermektedir [1, 3, 21, 37, 39]. Sınıflarda bulunan görüntüler birbirleriyle ilişkili değildir. Wang veri kümesi 10 sınıf ve 1000 görüntü [62], PascalVoc2006 veri kümesi 10 sınıf ve 5304 görüntü ve Caltech101 veri kümesi 101 sınıf ve 9143 görüntü içermektedir [12, 15]. Caltech-256 veri kümesi 29.780 görüntü içermektedir. Bu veri kümesinde 256 sınıf bulunmaktadır ve her bir görüntü bir sınıf altındadır. Ayrıca her bir sınıfta en az 80 görüntü bulunmaktadır [65]. ILSVRC2012 veri kümesi ImageNet veri kümesi kullanılarak oluşturulmuştur ve 1000 kategoriye ait 1,2 milyon görüntü içermektedir [1, 3, 4, 22, 26, 31]. MNIST veri kümesi 70 bin 28×28 boyutlu el yazısı rakam içermektedir [19, 22, 65]. Bu veri kümesini bazı çalışmalarda eğitim kümesi 69 bin, test kümesi bin eleman içerecek şekilde kullanılmıştır [22, 65]. NUS-WIDE veri kümesi Flickr üzerinden toplanmıştır ve 270 bin görüntüden oluşmaktadır. Görüntüler 81 farklı konsepte göre etiketlenmiştir [19, 21, 22, 39, 55, 65].



Şekil 6.1. Corel-10K veri kümesindeki görüntü örnekleri

SIFT-1M veri kümesi 128 boyutlu 1 milyon yerel öznitelik vektörü içermektedir. Bu veri kümesi görüntülerden oluşturulmuş yerel SIFT tanımlayıcılarından oluşmaktadır. Ayrıca bu veri kümesi 10 bin adet sorgu örneği içermektedir [3, 21, 37, 38, 65]. Veri kümesindeki vektörler yönelim histogramlarını temsil etmektedir [65].

GIST-1M, SIFT-10M, SIFT-20M, SIFT-100M, Tiny-80M ve MNIST-8M literatürde yaygın kullanılan büyük hacimli veri tabanlarıdır. GIST-1M veri kümesi rastgele seçilmiş 1 milyon görüntü kullanılarak oluşturulmuştur. Bu veri kümesinde her bir görüntü bir vektörle temsil edilmekte ve her bir vektör 960 boyutlu GIST özniteliklerinden oluşmaktadır [19, 37-39, 68]. SIFT-10M veri kümesi rastgele görüntüler kullanılarak oluşturulan 10 milyon, SIFT-20M veri kümesi ise oluşturulan 20 milyon 128 boyutlu SIFT tanımlayıcılarından oluşmaktadır. SIFT-100M veri kümesi 100 milyon SIFT tanımlayıcısından oluşan büyük bir veri kümesidir [37, 38, 68]. Tiny-80M 80 milyon 364 boyutlu GIST özniteliğinden oluşmaktadır [37, 38]. MNIST-8M veri kümesi 8 milyon 784 boyutunda öznitelik içeren örneklerden oluşmaktadır [37].

Bu tez çalışmasında Corel-10K, SIFT-1M ve GIST-1M veri kümeleri kullanılmıştır. Corel-10K veri kümesi kullanılarak 220.565 adet SIFT tanımlayıcısı oluşturulmuştur. Oluşturulan

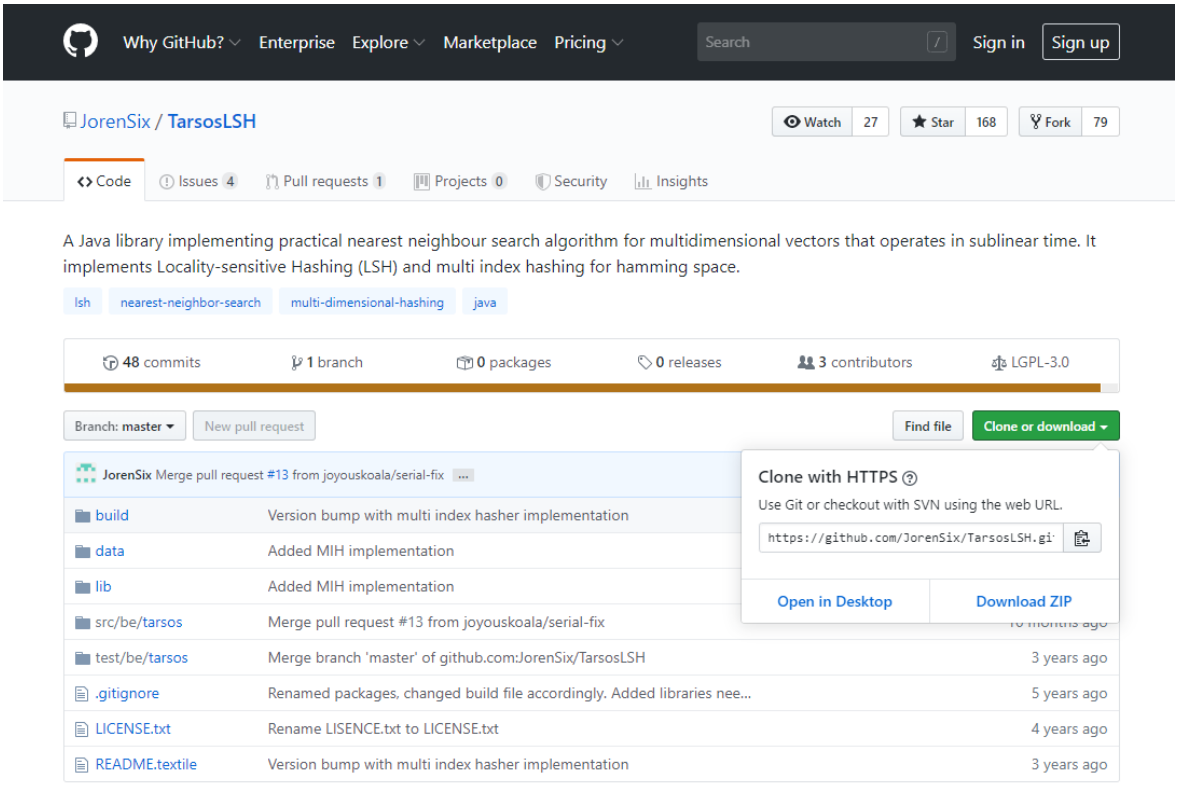
özniteliklerin %90'ı sistemin eğitiminde, kalan %10'u ise sistemin testinde kullanılmıştır. SIFT-1M ve GIST-1M veri kümeleri kullanılmadan önce herhangi bir işlem yapılmamıştır.

6.2. Uygulamada Kullanılan Yazılım Kütüphaneleri

Bu tez çalışmasında uygulanan tüm yöntemler Java dilinde uygulanmıştır. Dağıtık ağ yapısı için Apache Ignite kütüphanesi kullanılmıştır. Apache Ignite hafıza tabanlı dağıtık veri tabanı sunan ve bununla birlikte büyük veriler üzerinde bilgisayar hafızası hızında işlem yapmayı sağlayan bir yazılım kütüphanesidir. Ignite kütüphanesinde kullanılan hafıza tabanlı veri tabanları yatay genişleme ve veri tutarlığı konularında çözümler sunmaktadır. Ayrıca kümeleme, veri sürekliliği, hesaplama ortamı gibi konularda da çözümler sunmaktadır. Bunun yanında Ignite kütüphanesi *K*-ortalamalar, karar ağaçları, *k* en yakın komşuluk arama, genetik algoritma, destek vektör makinesi, yaklaşık en yakın komşu bulma gibi pek çok makine öğrenme algoritmasını da içerisinde bulundurmaktadır. Ignite kütüphanesinin sağladığı hesaplama sistemiyle birden fazla düğümde dağıtık olarak veri işlenebilmektedir.

Şekil 6.2. Kullanılan yazılım kütüphaneleri

LSH yönteminin uygulanması için açık kaynak koduna sahip Java dilinde kodlanmış TarsosLSH yazılım kütüphanesi kullanılmıştır. Bu yazılım kütüphanesinde alt doğrusal arama algoritmalarının uygulaması mevcuttur. Bu kütüphanede LSH yöntemi merkezi olarak kodlanmıştır. Ayrıca Öklid özet ailesi yanında, city-block ve kosinüs özet aileleri de desteklenmektedir. Bunun yanında karşılaştırmalar için kullanılan SH, PCAH, ITQ gibi yöntemlerde Java dilinde aynı ortamda kodlanmıştır. Bu yöntemler kodlanırken matris işlemlerini yapabilmek için EJML (Efficient Java Matrix Library) yazılım kütüphanesinden faydalanılmıştır. EJML kütüphanesi matris işlemlerini etkili bir şekilde gerçekleştirmek için tasarlanmış lineer cebir kütüphanesidir.



Şekil 6.3. TarsosLSH açık kaynak kodlu kütüphanesi

TarsosLSH yazılım kütüphanesi Apache Ignite kütüphanesinin sağladığı dağıtık alt yapıyla birlikte kullanılarak RDH yönteminin kodlanmasında kullanılmıştır.

6.3. Düğümlerde Aynı Özet Fonksiyon Kümesinin Kullanılması

Tüm düğümlerde aynı özet fonksiyon kümesinin kullanıldığı durumda tüm veri aynı özet fonksiyon kümesi ile indekslenmiştir. Sorgu örneği dağıtık düğümlere gönderilmiş ve sonrasında her bir düğüm bu örneği kendi yerel indeksine göre sorgulamıştır. Sonrasında her bir düğüm bulduğu yaklaşık komşuları merkezi düğüme göndermektedir. Bu düğümde yerel sonuçlar birleştirilmekte ve nihai en yakın komşular elde edilmektedir. 8, 16, 32 ve 64 bit için MAP sonuçları elde edilmiştir. LSH ve RDH yöntemleri için 50, 100, 150 ve 200 olarak değişken sayıda özet tabloları kullanılmıştır. MAP sonuçlarının yanında sorgu süreleri de incelenmiştir.

6.3.1. Corel-10K veri kümesi kullanılarak elde edilen sonuçlar

Corel-10K veri kümesi için elde edilen MAP sonuçları Çizelge 6.1'de gösterilmiştir. Elde edilen sorgu süreleri ise Çizelge 6.2'de gösterilmiştir. Corel-10K veri kümesi üzerinde KNN ile arama yapıldığında sorgu süresi 158,30 ms olarak ölçülmüştür. Yaklaşık en yakın komşu arama yöntemleri kullanıldığında artan bit sırasına göre KNN yönteminden daha iyi sorgu süreleri elde edilmiştir. Bu veri kümesi üzerinde LSH ve RDH yöntemleri için benzer MAP sonuçları elde edilmekle birlikte RDH yöntemin LSH yönteminden daha iyi sorgu performansı göstermiştir. Test örneklerinin sorgulanmasında 8 ve 16 bit özet kodları kullanıldığında yaklaşık 6 kat hızlanma elde edilmiştir.

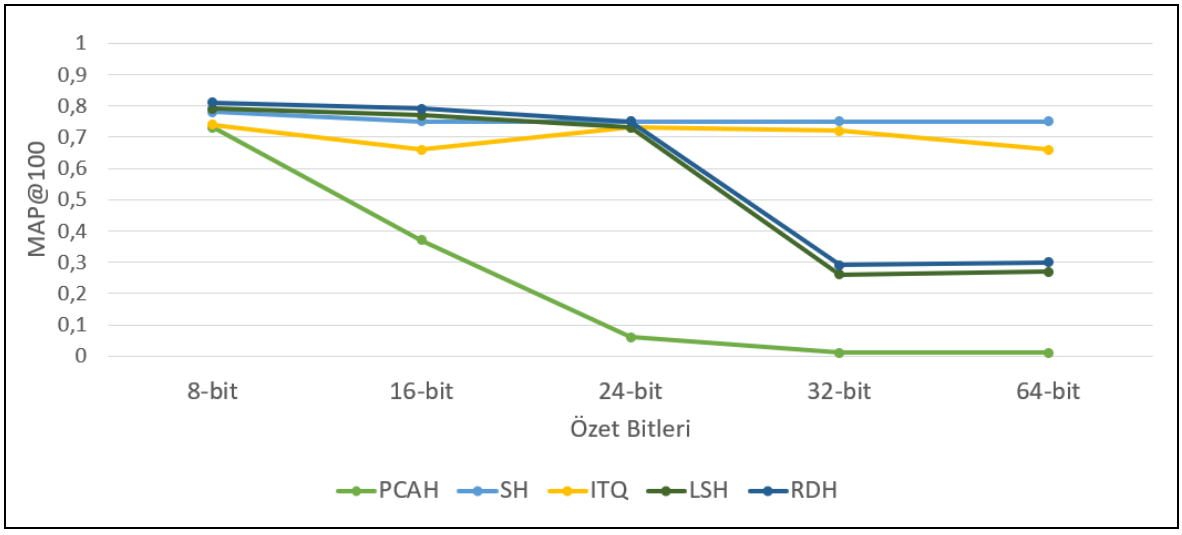
Çizelge 6.1. Corel-10K veri kümesi için MAP@100 sonuçları

Yöntem	Özet Tablosu Sayısı	8-bit	16-bit	24-bit	32-bit	64-bit
PCAH	1	0,73	0,37	0,06	0,01	0,01
SH	1	0,78	0,75	0,75	0,75	0,75
ITQ	1	0,74	0,66	0,73	0,72	0,66
LSH	50	0,81	0,80	0,65	0,22	0,26
LSH	100	0,81	0,79	0,75	0,29	0,30
LSH	150	0,81	0,79	0,78	0,36	0,40
LSH	200	0,81	0,79	0,80	0,47	0,43
RDH	50	0,81	0,80	0,65	0,22	0,26
RDH	100	0,81	0,79	0,75	0,29	0,30
RDH	150	0,81	0,79	0,78	0,36	0,40
RDH	200	0,81	0,79	0,80	0,47	0,43

Çizelge 6.2. Corel-10K veri kümesi için sorgu süreleri (ms)

Yöntem	Özet Tablosu Sayısı	8-bit	16-bit	24-bit	32-bit	64-bit
LSH	50	488,57	5,44	0,37	0,23	0,44
LSH	100	901,22	11,29	1,22	0,48	1,72
LSH	150	1224,79	20,82	1,47	0,80	2,73
LSH	200	1857,07	28,68	1,65	1,25	2,71
RDH	50	39,53	2,53	0,31	0,22	0,41
RDH	100	71,16	3,73	0,59	0,45	0,97
RDH	150	88,66	4,61	0,85	0,69	1,72
RDH	200	107,64	5,45	1,25	1,05	2,52

Corel-10K veri kümesi için RDH yöntemi LSH yönteminde MAP ve sorgu süresi bakımından daha iyi sonuçlar vermiştir. Bit sayısı artırıldığında her iki yöntemde sorgu süresini azalmakla birlikte RDH yöntemi daha iyi sorgu performansı göstermiştir. Bu veri kümesi için MAP sonuçlarının karşılaştırılması Şekil 6.4'de gösterilmiştir. RDH ve LSH yöntemlerinde farklı sayıda özet kod uzunlukları kullanıldığında benzer sonuçlar elde elde edilmiştir.



Şekil 6.4. Corel-10K veri kümesi kullanıldığında elde edilen MAP@100 sonuçları (LSH ve RDH yöntemleri için 100 özet tablosu kullanılmıştır)

6.3.2. SIFT-1M veri kümesi kullanılarak elde edilen sonuçlar

SIFT-1M veri kümesi üzerinde alınan MAP sonuçları Çizelge 6.3'de ve sorgu zamanı sonuçları Çizelge 6.4'de gösterilmiştir. Bu veri kümesinde KNN arama yapıldığında ortalama sorgu süresi 765,58 ms olmuştur. RDH yöntemi kullanıldığında LSH yönteminden daha iyi MAP sonuçları elde edilmiştir. Bununla birlikte ilgili olan örneklerin bulunması ise yaklaşık 10 kat daha hızlı yapılmıştır.

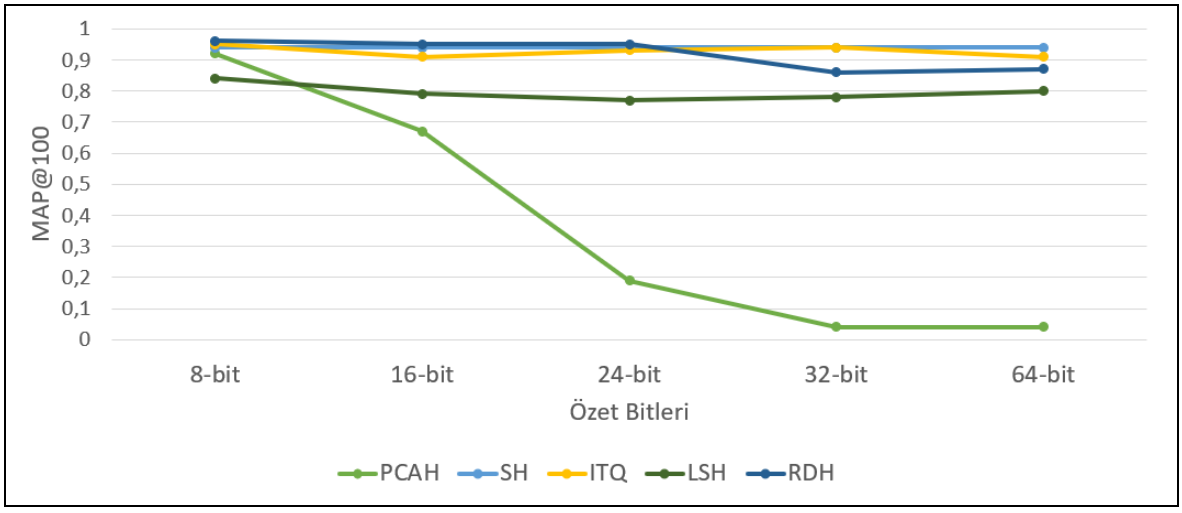
Çizelge 6.3. SIFT-1M veri kümesi için MAP@100 sonuçları

Yöntem	Özet Tablosu Sayısı	8-bit	16-bit	24-bit	32-bit	64-bit
PCAH	1	0,92	0,67	0,19	0,04	0,04
SH	1	0,94	0,94	0,94	0,94	0,94
ITQ	1	0,95	0,91	0,93	0,94	0,91
LSH	50	0,84	0,75	0,79	0,72	0,67
LSH	100	0,84	0,79	0,77	0,78	0,80
LSH	150	0,84	0,82	0,79	0,81	0,79
LSH	200	0,84	0,83	0,79	0,81	0,78
RDH	50	0,96	0,94	0,95	0,77	0,71
RDH	100	0,96	0,95	0,95	0,86	0,87
RDH	150	0,96	0,96	0,95	0,93	0,91
RDH	200	0,96	0,96	0,94	0,94	0,91

Çizelge 6.4. SIFT-1M veri kümesi için sorgu süreleri (ms)

Yöntem	Özet Tablosu Sayısı	8-bit	16-bit	24-bit	32-bit	64-bit
LSH	50	3725,56	237,30	13,30	1,97	1,55
LSH	100	6751,53	477,73	33,78	3,69	3,62
LSH	150	6679,04	500,13	37,37	5,13	6,69
LSH	200	8677,25	1064,10	60,62	10,71	9,25
RDH	50	291,92	19,80	2,88	0,75	0,86
RDH	100	414,99	30,11	4,58	1,33	1,86
RDH	150	491,18	40,07	5,93	1,82	3,06
RDH	200	550,31	55,01	6,56	2,54	3,91

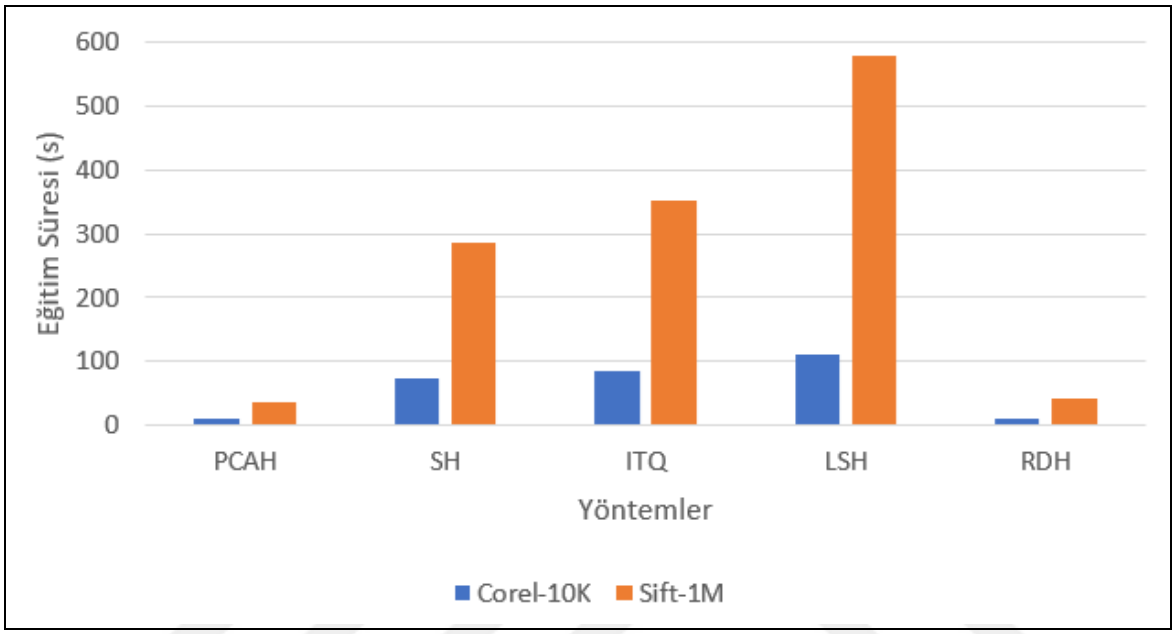
SIFT-1M veri kümesi için RDH yöntemi LSH yönteminden MAP ve sorgu süresi bakımından daha iyi sonuçlar vermiştir. Bit sayısı artırıldığında her iki yöntemde sorgu süresini azalmakla birlikte RDH yöntemi daha iyi sorgu performansı göstermiştir. Bu veri kümesi için MAP sonuçlarının karşılaştırılması Şekil 6.5'de gösterilmiştir. Corel-10K veri kümesi üzerinde RDH ve LSH yöntemleri farklı sayıda özet kodu uzunlukları kullanıldığında benzer sonuçlar elde elde edilmiştir. SIFT-1M veri kümesi üzerinde ise RDH yöntemi daha iyi sonuçlar vermiştir.



Şekil 6.5. SIFT-1M veri kümesi kullanıldığında elde edilen MAP@100 sonuçları (LSH ve RDH yöntemleri için 100 özet tablosu kullanılmıştır)

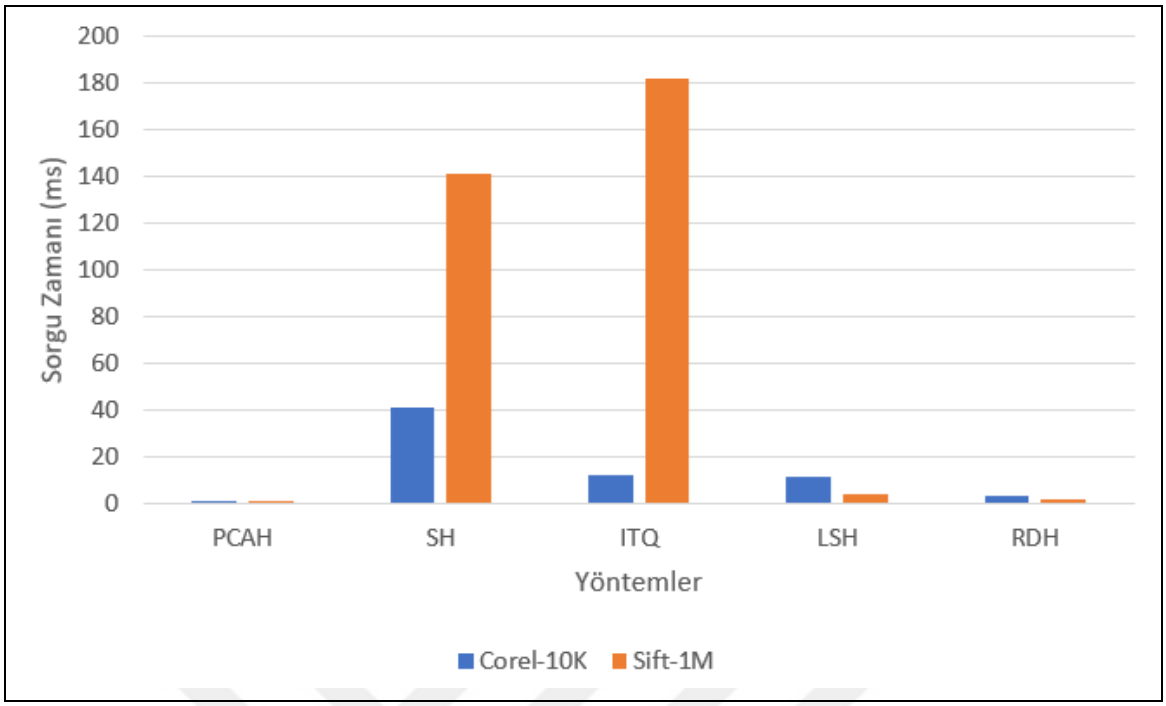
6.3.3. Corel-10K ve SIFT-1M veri kümeleri eğitim ve sorgu sürelerinin karşılaştırılması

Corel-10K ve SIFT-1M veri kümeleri için farklı özetleme yöntemleri kullanıldığında elde edilen eğitim sürelerinin karşılaştırılması Şekil 6.6'da gösterilmiştir.



Şekil 6.6. Eğitim sürelerinin karşılaştırılması (RDH ve LSH yöntemleri için 32-bit özet kodu ve 100 özet tablosu kullanılmıştır)

Corel-10K ve SIFT-1M veri kümeleri için farklı özetleme yöntemleri kullanıldığında elde edilen sorgu sürelerinin karşılaştırması Şekil 6.7'de gösterilmiştir.



Şekil 6.7. Sorgu sürelerinin karşılaştırılması (RDH ve LSH yöntemleri için 32-bit özet kodu ve 100 özet tablosu kullanılmıştır)

6.3.4. GIST-1M veri kümesi kullanılarak elde edilen sonuçlar

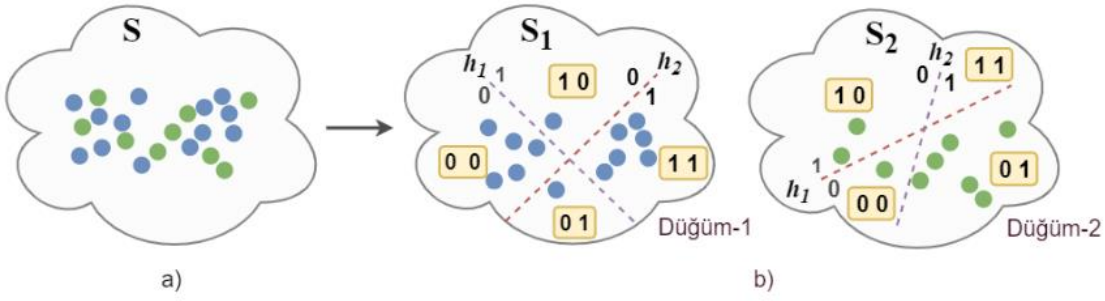
Tez çalışmasında RDH yöntemi literatürde son zamanlarda yapılan bir çalışmayı [39] referans alarak yakın zamanda önerilen dağıtık özetleme yöntemleriyle karşılaştırılmıştır. Bu çalışmada dağıtık veri kümeleri üzerinde SDH (Sequential Distributed Hashing) ve PDH (Parallel Distributed Hashing) yöntemleri önerilmiştir. Bunun yanında önerilen yöntemler yakın zamanda farklı çalışmalarda önerilen DisH (Distributed Hashing) [37], ABQ (Adaptive Binary Quantization) [38] ve SGH (Scalable Graph Hashing) [69] gibi öğrenme tabanlı dağıtık özetleme yöntemleriyle karşılaştırılmıştır. Referans olarak kullanılan çalışmada oluşturulan deneysel ortama benzer bir ortam oluşturulmuştur. RDH yöntemi bu ortam kullanılarak test edilmiştir. GIST-1M veri kümesi kullanılmış ve tüm veri düğümlere rastgele ve eşit olarak dağıtılmıştır. Önerdiğimiz sistem 1000 sınama sorgusuyla test edilmiştir. Alınan sonuçlar Çizelge 6.5'de gösterilmiştir. RDH yöntemi 64-bit özet kodu kullanılarak uygulandığında sorgu süresi 65,59 ms olmuştur.

Çizelge 6.5. GIST-1M veri kümesi üzerinde RDH yöntemiyle elde edilen MAP@1000 sonuçlarının referans çalışmayla karşılaştırılması

Yöntem	64-bit	96-bit
PCAH	0,2957	0,2777
AGH	0,2733	0,2872
ITQ	0,4918	0,5232
ABQ	0,5812	0,6457
SGH	0,5026	0,5524
DisH	0,4908	0,5117
DABQ	0,5801	0,6446
SDH	0,4929	0,5472
PDH	0,4934	0,5421
RDH	0,5080	0,4970

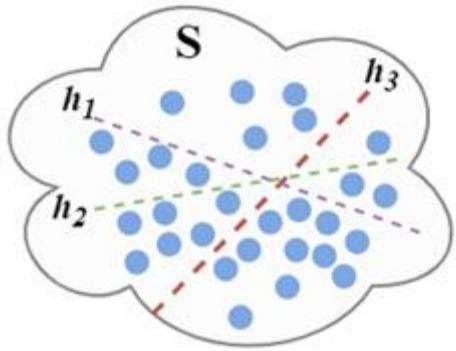
6.4. Düğümlerde Birbirinden Farklı Özet Fonksiyon Kümesinin Kullanılması

Düğümlerde aynı özet fonksiyon kümesinin kullanılmasının yanında düğümlerde farklı özet kümelerinin kullanılması ve veriyi yaklaşık olarak eşit bölen özet fonksiyonlarının kullanılması da incelenmiştir. LSH yöntemi tek düğümde veya aynı özet fonksiyonları kullanılarak birden fazla düğümde uygulandığında öğrenme örnekleri için oluşan özet kodları değişmemektedir. Bu yüzden RDH yöntemiyle MAP sonuçları LSH yöntemiyle benzer olmakla birlikte sorgu süresinde ciddi bir artış kazanılmaktadır. Bunun yanında LSH yöntemi veri bağımsız bir yöntem olduğu için dağıtık düğümlerde birbirinden farklı özet kodlarının kullanılması örneklerin bulunacağı özet kovalarını değiştirse de benzer örneklerin aynı kovalara toplanma davranışını değiştirmeyecektir.



Şekil 6.8. Veri kümesinin düğümlere rastgele dağıtılması ve düğümlerde özetleme yapılması

Her bir düğüm sınama örneğini kendi indeksine göre sorgulamakta ve sonuçlar her düğümde aynı özetleme fonksiyon kümesi kullanıldığı duruma benzer olmaktadır. Rastgele özet fonksiyonları veri özellikleri kullanılmadan oluşturulması bu durumun temel sebebidir. Bunlara ek olarak özet fonksiyonlarının veriyi bölme özelliğine göre de seçilmesiyle de çalışmalar yapılmıştır. Veriyi yaklaşık olarak eşit sayıda örnek içeren özet fonksiyonları seçilmiştir. Örneğin SIFT-1M veri kümesi 1 milyon örnek içermekte ve bu veri kümesi düğümlere dağıtıldığında her bir düğümde 100.000 örnek olmaktadır. Rastgele özet fonksiyonları veriyi 20.000 ve 80.000 eleman içeren iki parçaya veya 45.000 ve 55.000 içeren iki parçaya bölebilmektedir. Verinin bölünmesiyle elde edilen veri parçalarının yaklaşık olarak benzer sayıda örnek içermesi indeksleme işleminden önce bir ön işlem olarak uygulanmıştır. Rastgele üretilen özet fonksiyonlar Şekil 6.9'da gösterilmiştir. Bu şekilde h_3 özet fonksiyonu h_1 ve h_2 özet fonksiyonundan daha dengeli bir bölümleme sağlamaktadır. Bu yöntemle ilgili MAP ve sorgu sürelerini içeren sonuçlar Çizelge 6.6'da gösterilmiştir. MAP sonuçları birbirine yakın olmakla birlikte seçilmiş özet fonksiyonları kullanıldığında daha iyi sorgu süresi elde edilmiştir.



Şekil 6.9. Veri kümesini farklı oranlarda bölen 3 farklı özet fonksiyonu

Çizelge 6.6. SIFT-1M veri kümesi üzerinde farklı özet fonksiyonlarının seçilmesiyle elde edilen MAP@100 ve sorgu süresi değerleri

	Aynı Özet Fonksiyon Kümesi		Farklı Özet Fonksiyon Kümesi		Seçilen Özet Fonksiyon Kümesi	
Özet Tablo	MAP	Sorgu Süresi	MAP	Sorgu Süresi	MAP	Sorgu Süresi
50	0,77	0,75	0,85	0,72	0,71	0,50
100	0,86	1,33	0,88	1,39	0,84	0,93
150	0,93	1,82	0,92	1,92	0,83	1,39
200	0,94	2,54	0,87	2,51	0,90	1,86

6.5. Çapraz Doğrulama Yöntemi ile Elde Edilen Sonuçlar

CV (Cross Validation - Çapraz Doğrulama) sınırlı sayıda örnek içeren bir veri kümesi üzerinde uygulanan öğrenme algoritmasının değerlendirilmesinde öngörme hatasını elimine etmeyi sağlayan bir yöntemdir [70]. Literatürde genellikle k-fold (k-katlamalı) çapraz doğrulama tekniği yaygın olarak kullanılmaktadır ve genellikle k değeri 10 olarak seçilmektedir [71-75]. Bu yöntemde veri kümesi kesişimleri boş küme olacak şekilde k adet gruba ayrılmaktadır [70, 73]. Veri kümesinin boyutu N olmak üzere her bir grubun boyutu ise N / k olmaktadır. 10-katlamalı çapraz doğrulama yönteminde 10 parçadan 9 tanesi eğitim için kalan bir parça ise sistemin testi için kullanılmaktadır. k değeri 10 olarak seçildiğinde 10 iterasyon yapılarak elde edilen 10 sonuç değerinin ortalaması sonucu vermektedir [74].

Bu tez çalışmasında önerilen yöntem 10-katlamalı çapraz doğrulama yaklaşımıyla SIFT-1M kullanılarak test edilmiştir. SIFT-1M veri kümesi 1 milyon öznitelik bulunduğundan k değerinin 10 seçilmesiyle her biri 100 bin eleman içeren 10 farklı küme elde edilmiştir. Sonrasında sırayla 10 gruptan birisi sistemin testi, geriye kalan 9 tanesi sistemin eğitimi için kullanılmıştır. Bu şekilde oluşturulan eğitim kümesi üzerinde LSH yönteminin tek düğümde uygulanmasıyla elde edilen sonuçlar Çizelge 6.7'de gösterilmiştir.

Çizelge 6.7. LSH yöntemi *k*-katlamalı çapraz doğrulama yöntemiyle tek düğüm üzerinde uygulandığında elde edilen sonuçlar

Özet Tablo	Özet Kod	MAP@100 Ortalama	MAP@100 Standart Sapma	Sorgu Sürelerinin Ortalaması	Sorgu Sürelerinin Standart Sapması
100	16	0,96	0,00278	454,513	48,461
150	16	0,96	0,00286	638,031	56,442
100	32	0,92	0,02279	4,346	0,980
150	32	0,94	0,01574	6,178	1,259
100	64	0,91	0,01534	5,052	0,759
150	64	0,94	0,01067	8,749	1,485

Benzer şekilde aynı veri kümesi üzerinde LSH yöntemi 10 düğüm kullanıldığında elde edilen sonuçlar Çizelge 6.8'de gösterilmiştir.

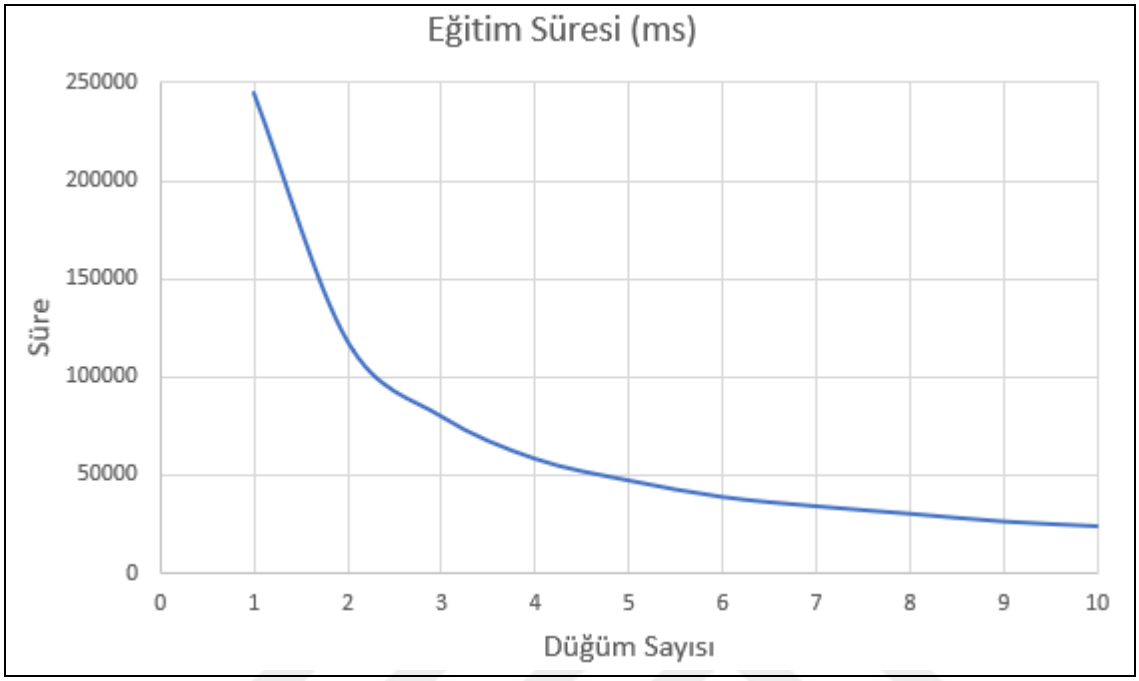
Çizelge 6.8. LSH yöntemi k-katlamalı çapraz doğrulama yöntemiyle 10 düğüm üzerinde uygulandığında elde edilen sonuçlar

Özet Tablo	Özet Kod	MAP@100 Ortalama	MAP@100 Standart Sapma	Sorgu Sürelerinin Ortalaması	Sorgu Sürelerinin Standart Sapması
100	16	0,96	0,0027	34,355	2,040
150	16	0,97	0,0028	46,545	2,210
100	32	0,93	0,0227	1,628	0,149
150	32	0,94	0,0157	2,138	0,139
100	64	0,92	0,0153	2,159	0,123
150	64	0,94	0,0106	3,619	0,444

6.6. Hızlanma

Hızlanma (speedup) aynı iş için daha fazla sayıda düğüm kullanıldığında sistemin etkinliğini ölçen bir performans ölçütüdür [38]. Önerilen sisteminde 16-bit ve 32-bit özet kodu kullanıldığında artan düğüm sayısına göre eğitim süresindeki değişim Şekil 6.10 ve Şekil 6.11'de gösterilmiştir. Her iki durum içinde artan düğüm sayısıyla beraber eğitim süresi ciddi oranda düşmüştür. 16-bit özet kodu kullanılarak tek düğüm üzerinde özetleme

yapıldığında eğitim süresi 244.993 ms olurken, 10 düğüm üzerinde özetleme yapıldığında eğitim süresi 23.765 ms olmuştur.

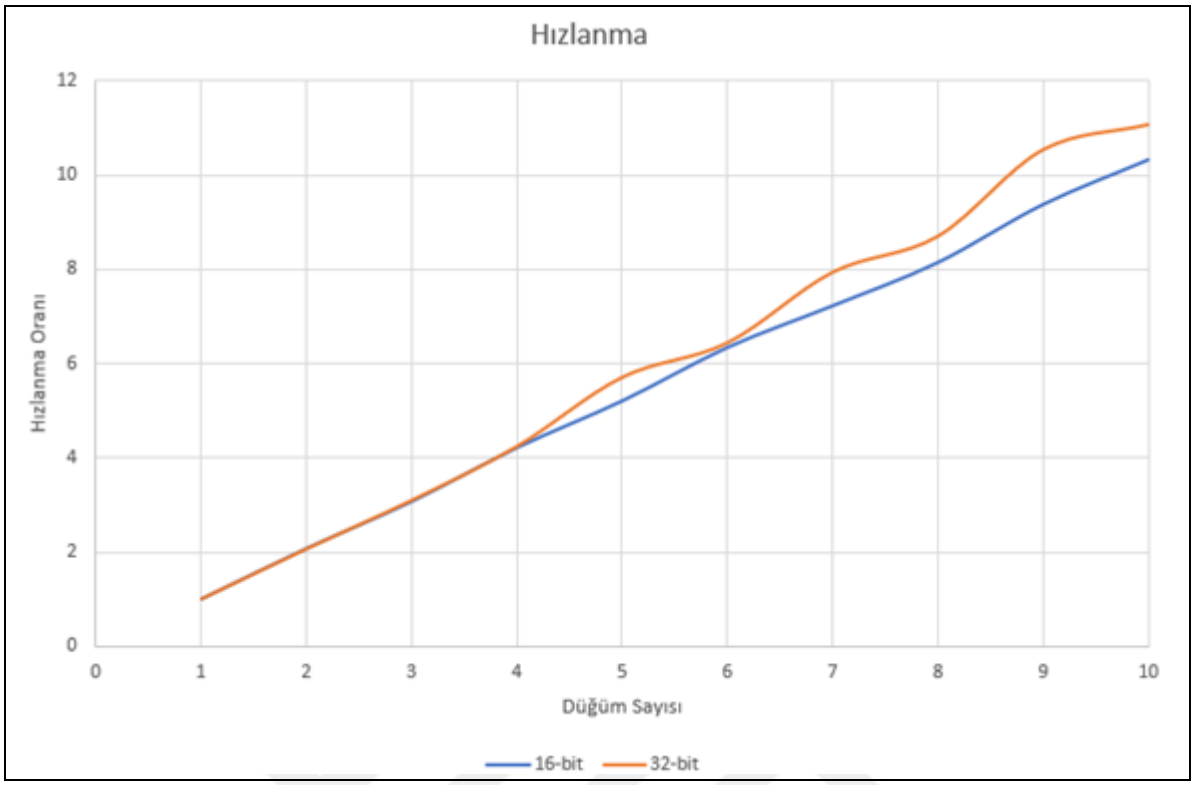


Şekil 6.10. 16-bit kullanıldığında eğitim süresindeki değişim

32-bit özet kodu kullanılarak tek düğüm üzerinde özetleme yapıldığında eğitim süresi 530.132 ms olurken, 10 düğüm üzerinde özetleme yapıldığında eğitim süresi 47.807 ms olmuştur.

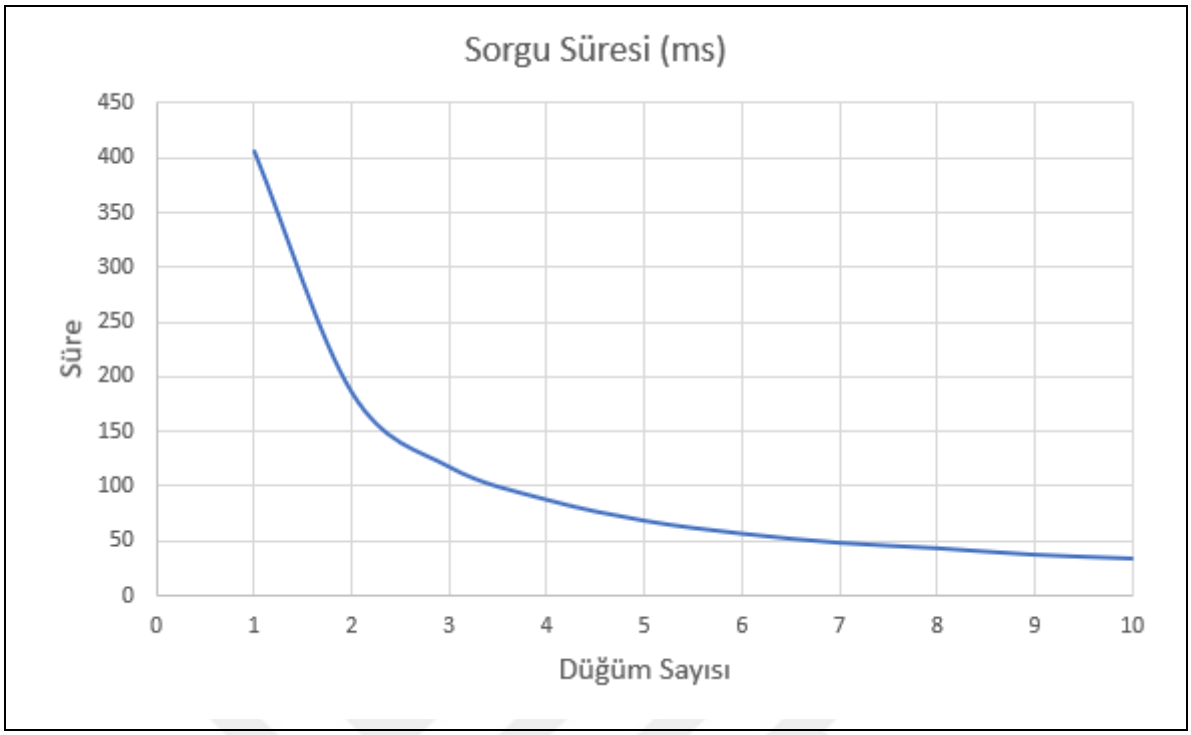
Şekil 6.11. 32-bit kullanıldığında eğitim süresindeki değişim

16-bit ve 32-bit için elde edilen hızlanma oranları Şekil 6.12'de gösterilmiştir. 16-bit ve 32-bit özet kodu kullanılarak 10 düğüm üzerinde özetleme yapıldığında her iki durum içinde 10 kattan fazla performans artışı olmuştur. Bununla birlikte artan düğüm sayısıyla beraber sistemin hızlanması doğrusal bir şekilde artmıştır.



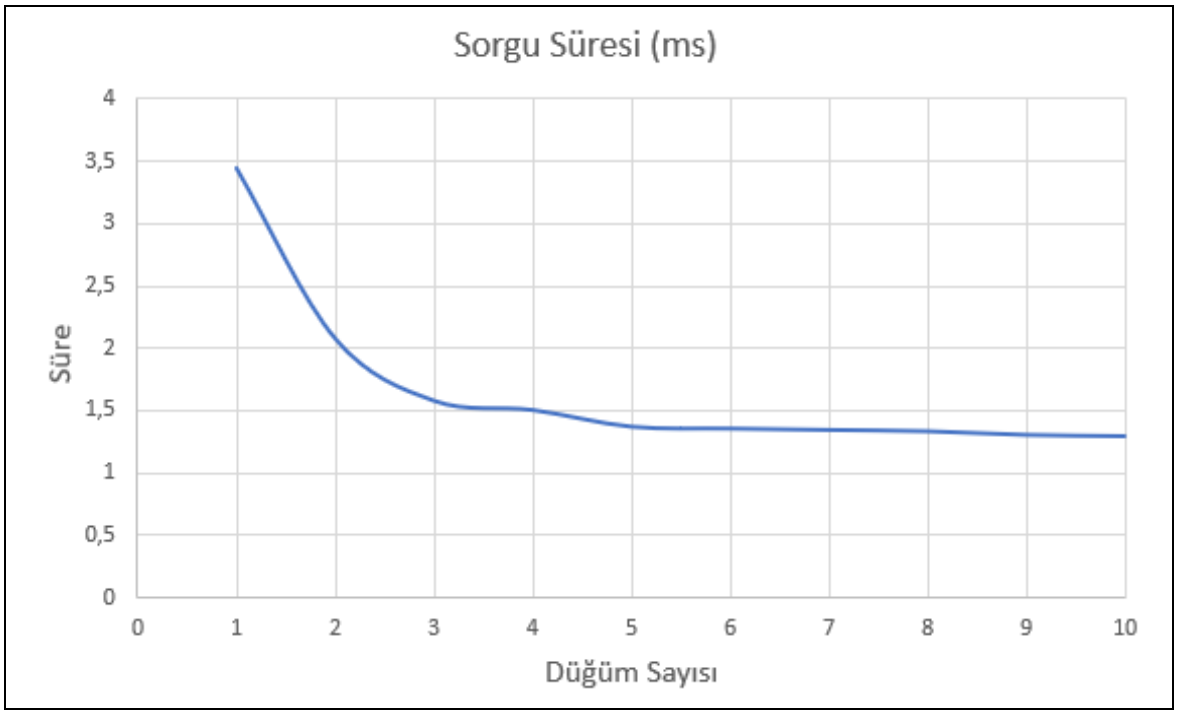
Şekil 6.12. 16-bit ve 32-bit özet kodu kullanıldığında eğitim süresindeki hızlanma oranları

Bu tez çalışması kapsamında eğitim sürelerinde elde edilen hızlanma oranlarının yanı sıra sorgu sürelerinde elde edilen hızlanma oranları da incelenmiştir. Önerilen yöntem için 16-bit ve 32-bit özet kodu kullanıldığında artan düğüm sayısına göre sorgu süresindeki değişim Şekil 6.13 ve Şekil 6.14'de gösterilmiştir. 16-bit özet kodu kullanıldığında sorgu süresinde ciddi oranda azalma gerçekleşmiştir. 32-bit özet kodu kullanıldığında 3 düğüme kadar ciddi bir azalma sonrasıda ise daha yumuşak bir azalma gerçekleşmiştir. 16-bit özet kodu kullanılarak tek düğüm üzerinde özetleme yapıldığında sorgu süresi 405 ms olurken, 10 düğüm üzerinde özetleme yapıldığında sorgu süresi 34 ms olmuştur.



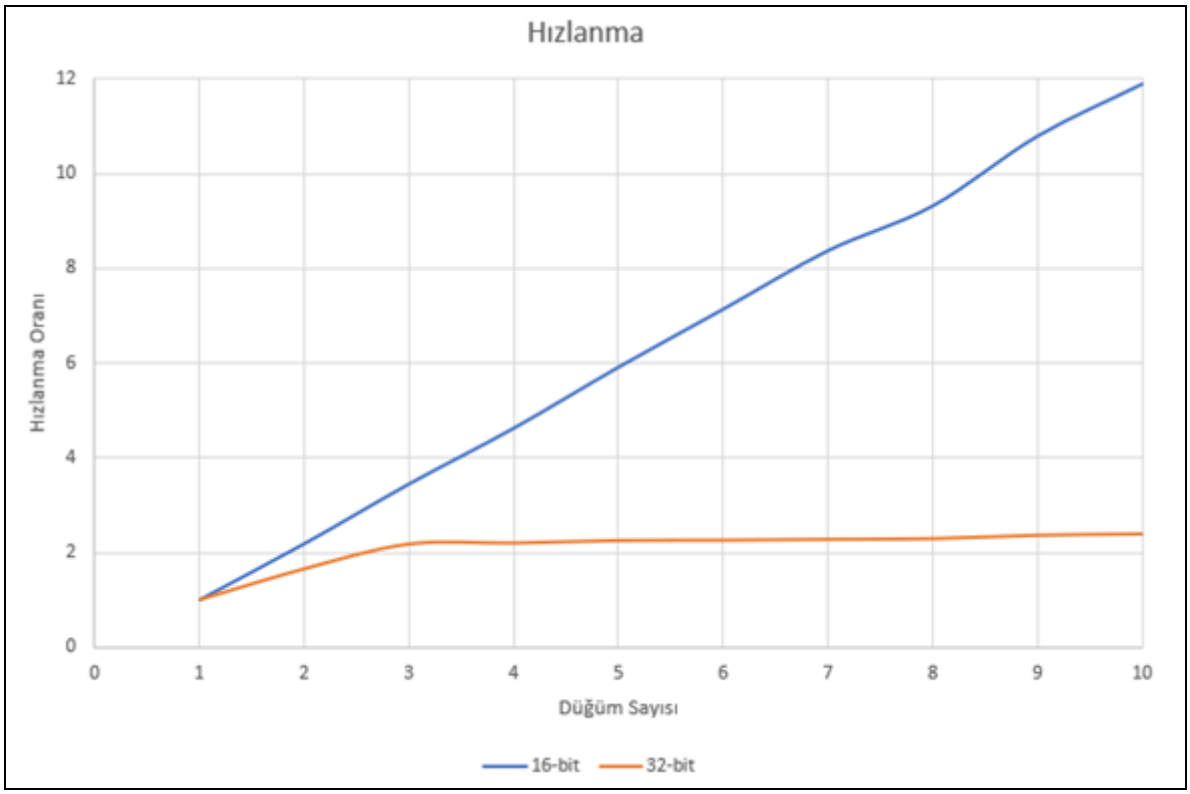
Şekil 6.13. 16-bit kullanıldığında sorgu süresindeki değişim

32-bit özet kodu kullanılarak tek düğüm üzerinde özetleme yapıldığında sorgu süresi 3,44 ms olurken, 10 düğüm üzerinde özetleme yapıldığında sorgu süresi 1,29 ms olmuştur.



Şekil 6.14. 32-bit kullanıldığında sorgu süresindeki değişim

Her iki durum elde edilen hızlanma oranları Şekil 6.15'de gösterilmiştir. 16-bit özet kodu kullanıldığında sorgu süresi doğrusal olarak artmıştır. 32-bit özet kodu kullanıldığında 3 düğüme kadar ciddi bir artma sonrasında ise daha yumuşak bir artma gerçekleşmiştir.



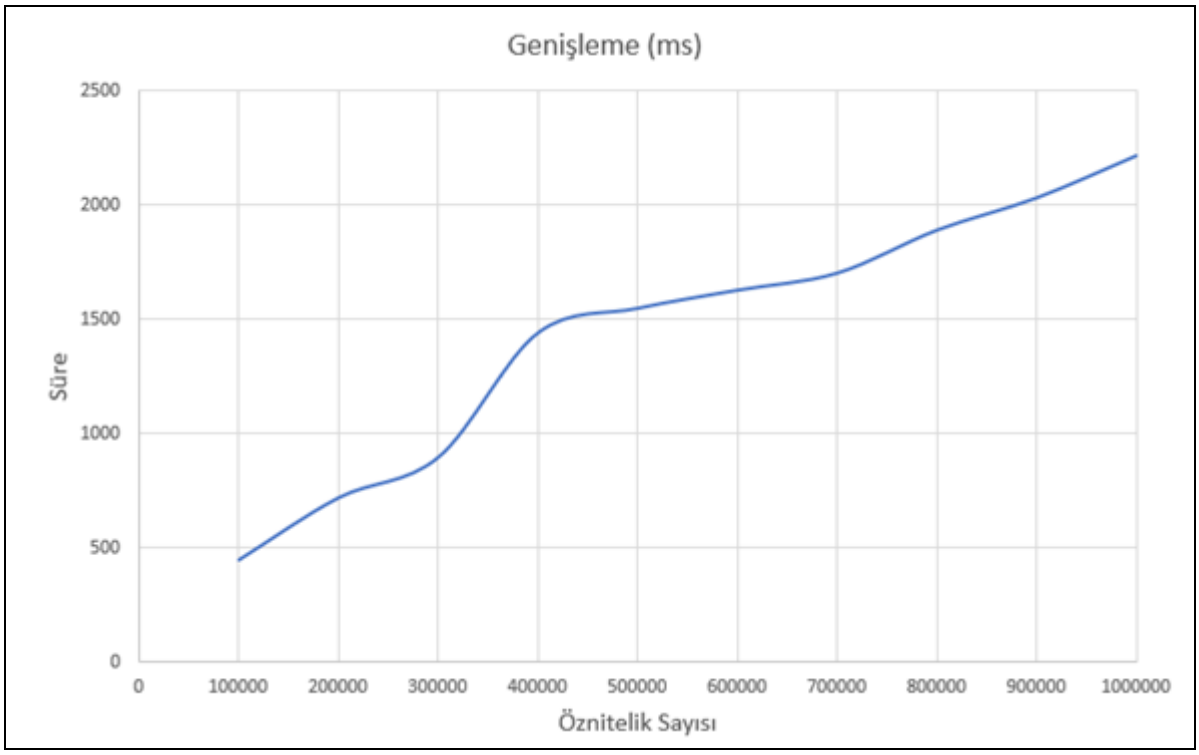
Şekil 6.15. 16-bit ve 32-bit özet kodu kullanıldığında sorgu süresindeki hızlanma oranları

RDH yöntemi artan düğüm sayısıyla uygulandığında eğitim süresini farklı özet bit sayıları için doğrusal olarak artırmıştır. 10 düğüm ve 16-bit özet kodu kullanıldığında eğitim süresi hızlanma oranı 10,30 kat olurken, 32-bit özet kodu kullanıldığında hızlanma oranı 11,08 olmuştur. RDH yöntemi sorgu süresindeki hızlanmayı da artan düğüm sayısıyla benzer bir şekilde artırmıştır. 10 düğüm ve 16-bit özet kodu kullanıldığında sorgu süresi hızlanma oranı 11,89 kat olurken, 32-bit özet kodu kullanıldığında hızlanma oranı 2,40 kat olmuştur.

6.7. Genişleme

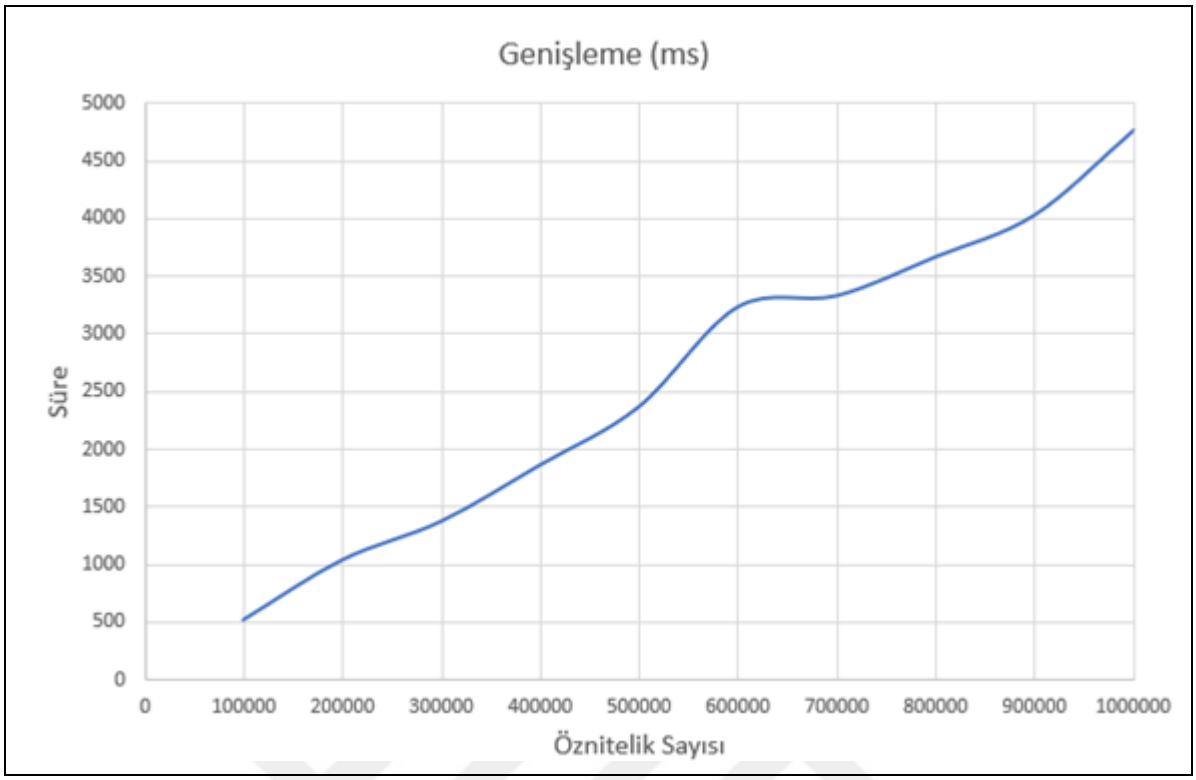
Genişleme (sizeup) *m* boyutlu bir veri kümesi *k* kat daha büyüdüğünde yapılan işin ne kadar süreceğini ölçen bir performans ölçütüdür [38]. Önerilen yöntemde 16-bit ve 32-bit özet kodu kullanıldığında artan veri sayısına göre eğitim süresindeki değişim Şekil 6.16 ve Şekil 6.17'de gösterilmiştir. Her iki durum içinde artan veri sayısıyla beraber eğitim süresi

doğrusal olarak artmıştır. RDH yöntemi 16-bit özet kodu kullanılarak 100.000 öznitelik üzerinde uygulandığında eğitim süresi 446,89 ms olurken, 1.000.000 öznitelik üzerinde uygulandığında 2215,02 ms olmuştur.



Şekil 6.16. 16-bit kullanıldığında eğitim süresindeki genişleme

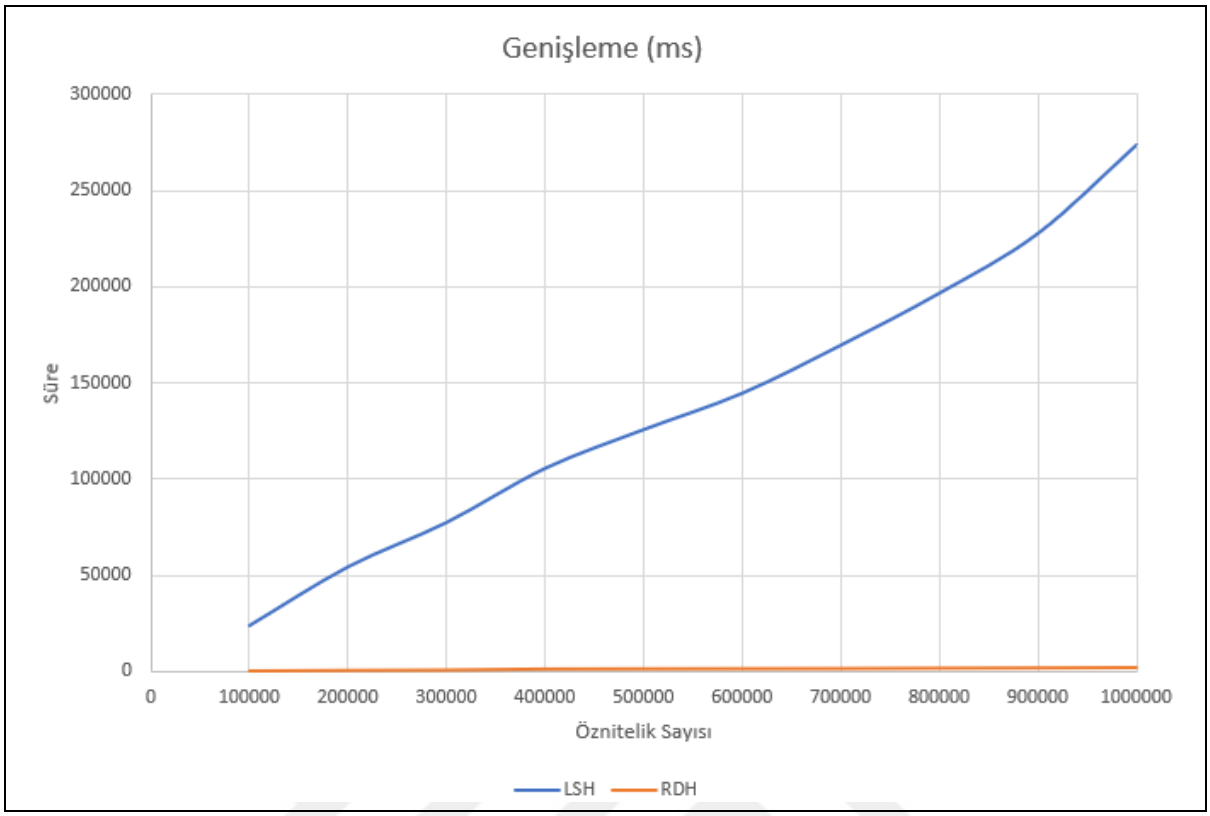
RDH yöntemi 32-bit özet kodu kullanılarak 100.000 öznitelik üzerinde uygulandığında eğitim süresi 517,11 ms olurken, 1.000.000 öznitelik üzerinde uygulandığında 4772,34 ms olmuştur.



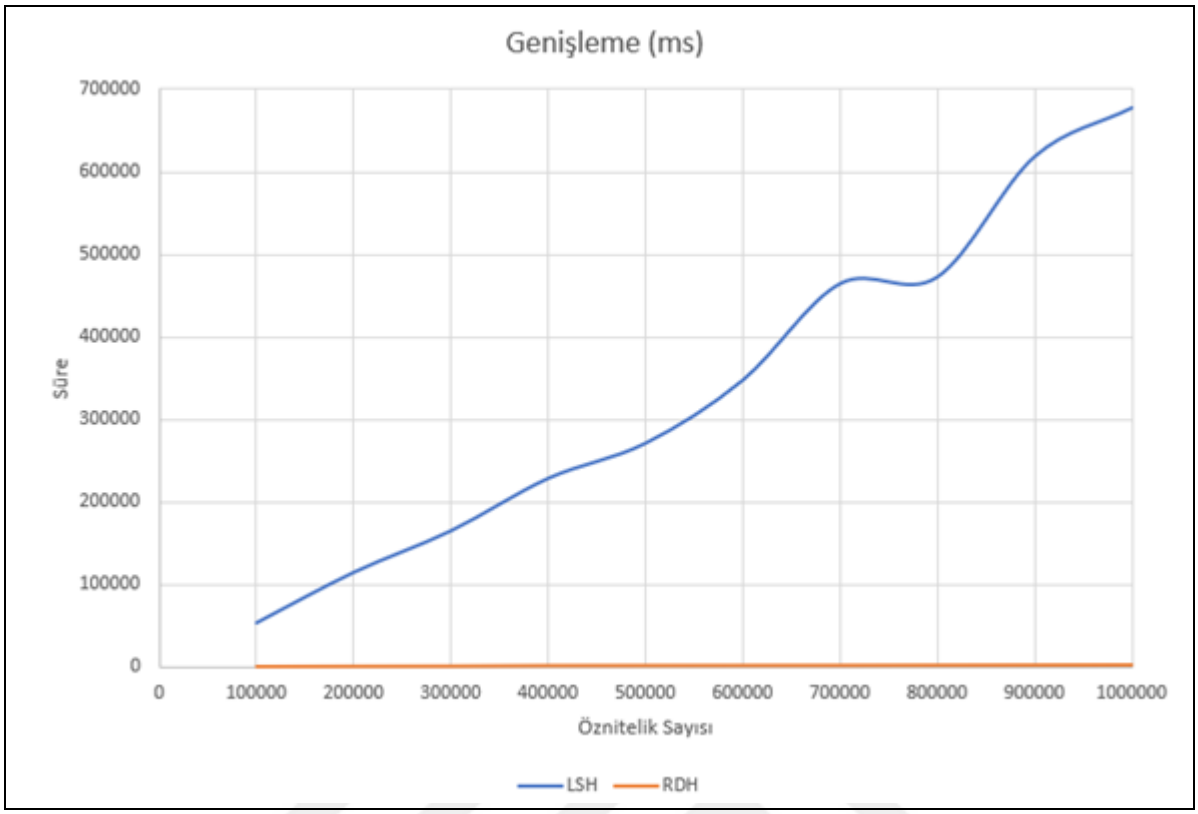
Şekil 6.17. 32-bit kullanıldığında eğitim süresindeki genişleme

Bununla birlikte RDH ve LSH yöntemlerinin artan veri sayısına göre eğitim sürelerindeki değişimlerinin karşılaştırılması Şekil 6.18 ve Şekil 6.19'da gösterilmiştir. LSH yöntemi 16-bit özet kodu kullanılarak 100.000 öznitelik üzerinde uygulandığında eğitim süresi 23494,31 ms olurken, 1.000.000 öznitelik üzerinde uygulandığında 274.027,23 ms olmuştur. LSH yöntemi 32-bit özet kodu kullanılarak 100.000 öznitelik üzerinde uygulandığında eğitim süresi 53.990,83 ms olurken, 1.000.000 öznitelik üzerinde uygulandığında 677.388,49 ms olmuştur.

16-bit ve 32-bit özet kodu kullanıldığında RDH yöntemi LSH yöntemine oranla oldukça iyi genişleme performansı sergilemiştir. 16-bit kullanıldığında RDH yöntemi için veri sayısı 10 kat artırıldığında genişleme oranı 4,96 olurken, LSH yöntemi için bu oran 11,66 olmuştur. 32-bit kullanıldığında RDH yöntemi için veri sayısı 10 kat arttırıldığında genişleme oranı 9,22 olurken, LSH yönteminde bu oran 12,54 olmuştur.



Şekil 6.18. 16-bit kullanıldığında LSH ve RDH yöntemleri için eğitim süresindeki genişleme



Şekil 6.19. 32-bit kullanıldığında LSH ve RDH yöntemleri için eğitim süresindeki genişleme.

6.8. Tez Çalışmasında Yapılan Görüntü Alma Çalışmaları

Bu tez çalışmasında önerilen RDH yöntemin uygulanmasından önce hızlı görüntü alma üzerine farklı çalışmalar yapılmıştır. Yapılan ilk çalışmalardan biri görüntünün sahip olduğu özniteliklerin ilişkisel veri tabanları üzerinde tutularak görüntülere veri tabanı sorguları yapılarak ulaşmaktır. Yapılan çalışmada Corel-10K veri kümesi üzerinde renk ve doku özniteliği ile SIFT tanımlayıcıları kullanılmıştır. Corel-10K veri kümesi kullanılarak 3 ayrı deney yapılmıştır. 1. deneyde veri kümesindeki 3 sınıf kullanılmış, 2. deneyde 11 sınıf kullanılmış ve 3. deneyde 80 sınıf kullanılarak çalışmalar yapılmıştır.

Her bir deney için veri kümesindeki sınıflarda bulunan görüntülerin %10'u test için, %90'ı eğitim için kullanılmıştır. Elde edilen öznitelikler ilişkisel veri tabanlarında tablolar üzerinde saklanmıştır. Yöntem Intel i7 2.4 GHz işlemciye sahip ve 32 GB ana bellek içeren bir sistem üzerinde uygulanmıştır. Veri tabanı yönetim sistemi olarak Microsoft Sql Server kullanılmıştır. Renk, doku ve SIFT tanımlayıcıları için öznitelikler farklı veri tabanı tablolarında tutulmuştur. İlgili özniteliklere göre görüntülerin benzerlikleri veri tabanı

yönetim sisteminin özellikleri kullanılarak veri tabanı sorguları ile bulunmuştur. Performansı artırmak için tablolardaki veriler indekslenmiştir. LSH yöntemi eğitim örneklerinin öznitelikleri üzerinde uygulanmış ve farklı parametrelere göre oluşturulan indeks yapısıyla sistem test edilmiştir. LSH yöntemi en yakın komşu aramasını daha az görüntü üzerinde yapabilmek amacıyla kullanılmıştır. İzlenilen yöntem şu şekildedir;

• Eğitim ve test görüntülerinin renk, doku öznitelikleri ve SIFT tanımlayıcıları çıkartılarak ilgili veri tabanı tablolarında saklanmıştır.

• Veri tabanındaki eğitim görüntülerinin öznitelikleri ve LSH yöntemi kullanılarak her özniteliğe özel özet kodları üretilmiş ve ilgili veri tabanı tablosunda saklanmıştır.

• Test görüntüleri için eğitim kümesinin indekslendiği yapı kullanılarak özet kodlar üretilmiştir.

• Test görüntülerine benzeyen eğitim görüntüleri özet kodlar kullanılarak SQL sorguları ile bulunmuştur.

• *k*-en yakın komşuluk yöntemine göre test görüntüsüne en çok benzeyen *k* eğitim görüntüsü listelenmiştir.

Oluşturulan sistemde benzer görüntüler SQL sorguları ile bulunmuştur. Öznitelikleri ve indeks yapısını veri tabanı tablolarında saklayarak LSH yöntemi ile oluşturulan indeks yapısının ana bellekte olma gereksinimi ortadan kaldırılmıştır. Bu çalışmada beklenen performans artışı elde edilememiştir. Yapılan çalışmalar SIU (Sinyal İşleme ve İletişim Uygulamaları Kurultayı) 2017 konferansında sunulmuştur [76] .

Sonraki çalışmada büyük boyutlu görüntü veri kümeleri üzerinde hızlı ve doğru arama yapabilme amacıyla LSH yönteminin dağıtık uygulanması önerilmiştir. Dağıtık LSH yöntemi ile tüm verinin tek bir noktadan indekslenmesi yerine küme üzerindeki düğümler üzerinde ayrık olarak indekslenmesi sağlanmıştır. Ayrıca LSH yönteminde kullanılan rastgele fonksiyonlar yerine veri kümesi üzerine PCA yönteminin uygulanmasıyla elde edilen temel bileşenler kullanılarak özet tablo sayısı azaltılmış ve bu şekilde de sınıflandırma yapılmıştır. Deneysel çalışmalar Corel-10K veri kümesinden elde edilen SIFT öznitelikleri kullanılarak yapılmıştır. Önceki çalışmaya benzer şekilde veri kümesindeki görüntülerin %90'ı eğitim geri kalanı test için kullanılmıştır.

Eğitim görüntüleri için SIFT yöntemi uygulandığında 220.565 öznitelik elde edilmiştir. Yapılan uygulamada büyük boyutlu görüntüler üzerinde çalışabilmek için hafıza tabanlı veri tabanlarından biri olan Apache Ignite kullanılmıştır. 1, 2 ve 3 düğüm kullanılarak önerilen yöntemler dağıtık olarak uygulanmıştır. Her düğümde LSH yöntemi uygulanarak düğüm bazında ayrık indeksler elde edilmiştir. LSH yöntemi için 50 özet tablosu ve 16 bit özet kodu kullanılmıştır. Yapılan bu çalışma RDH yönteminin temel çalışması olmuştur. Bu çalışmada LSH yöntemiyle özet fonksiyonları rastgele oluşturmanın yanında PCA yöntemini düğümlerdeki verinin üzerine uygulayarak elde edilen temel bileşenlerle de özetleme yapılmıştır.

Bu yöntem LSH yönteminde çok sayıda tablo ve bit kodu gereksinimini ortadan kaldırmak amacıyla yapılmıştır. Tek tabloda 8 adet temel bileşen kullanılarak özetleme yapılmıştır. Bu çalışmada veriyi yaklaşık olarak eşit iki parçaya bölebilen rastgele özet fonksiyonları seçilerek indeksleme yapılmış ve bu indekse göre arama yapılmıştır. Önerilen yöntemle başarı korunmuş bunun yanında arama süresi büyük oranda azalmıştır. Yapılan çalışmalar SIU (Sinyal İşleme ve İletişim Uygulamaları Kurultayı) 2018 konferansında sunulmuştur [77].

Yapılan başka bir çalışmada LSH yöntemindeki rastgele özet fonksiyonların veriyi bölebilme özellikliğine göre belirlenmesi önerilmiştir. Her bir özet fonksiyonu bir biti temsil etmektedir ve veriyi iki parçaya ayırmaktadır. Rastgele özet fonksiyonları kullanıldığında verinin hangi oranda bölüneceği bilinememektedir. Bu çalışmada veriyi yaklaşık olarak eşit iki parçaya bölebilen rastgele özet fonksiyonları seçilerek indeksleme yapılmış ve bu indekse göre arama yapılmıştır.

Önerilen yöntemle başarı korunmuş bunun yanında arama süresi büyük oranda azalmıştır. Bu çalışmada literatürde yaygın bir şekilde kullanılan SIFT-1M ve GIST-1M veri kümeleri kullanılmıştır. LSH yöntemi uygulanırken özet kod tablosu yöntemi benimsenmiş ve SIFT-1M için 100 adet özet tablosu kullanılmış, GIST-1M için ise 300 adet özet tablosu kullanılmıştır. Özet kodları SIFT-1M için 16 bit, GIST-1M için 64 bit uzunluğunda seçilmiştir. Bu çalışmada özet fonksiyonları rastgele üretilmiş ve geleneksel LSH yönteminden farklı olarak veri kümesini yaklaşık olarak ikiye bölen özet fonksiyonları kabul edilmiştir. Örneğin 1 milyon örnek içeren veri kümesini rastgele özet fonksiyonu 100.000 ve 900.000 şeklinde iki parçaya bölebilmektedir. Benzer şekilde 495.000 ve 505.000 olarak

da bölebilmektedir. Her iki parça arasındaki farkın 10.000'den az olduğu durumlar kabul edilmiştir. Performans ölçütü olarak bir grup sorgunun ortalama kesinlik değerlerinin hesaplandığı MAP (Mean Average Precision) ölçütü kullanılmıştır. Bu değer hesaplanırken özniteliklerin Öklid uzayındaki komşuları referans (ground-truth) olarak kabul edilmiştir. Sistem 100 adet test örneği ile sınanmıştır. SIFT-1M veri kümesi için önerilen yöntem kullanıldığında MAP@100 değeri 0,82 ve sorgu süresi 269 ms olmuştur. Özet fonksiyonları rastgele oluşturulduğunda ise MAP@100 değeri 0,83 ve sorgu süresi 507 ms olmuştur.

Ön tanımlı özet fonksiyonları kullanıldığında her bir özet tablosunda ortalama 20,24 öznitelik bulunurken, rastgele özet fonksiyonları kullanıldığında ortalama 22,37 öznitelik bulunmuştur. Ön tanımlı özet fonksiyonları kullanıldığında 100 adet test örneği için ortalama muhtemel en yakın komşu sayısı 21.029 olurken, rastgele özet fonksiyonları kullanıldığında bu değer 42.816 olmuştur. Sorgu örneklerinin muhtemel komşu uzayı azaldığı için sorgu hızı neredeyse yarıya düşmüştür. Yapılan çalışmalar SIU (Sinyal İşleme ve İletişim Uygulamaları Kurultayı) 2019 konferansında poster olarak sunulmuştur [78].

7. SONUÇ VE ÖNERİLER

Bu tez çalışmasında büyük hacimli görüntü veri tabanları üzerinde görüntü benzerlik arama problemine çözüm sunmak amacıyla LSH yönteminde kullanılan rastgele özet fonksiyonlar dağıtık bir şekilde uygulanarak RDH yöntemi önerilmiştir. Önerilen yöntemde büyük hacimli görüntü verisinin tek bir kaynaktan indekslenmesi yerine, rastgele olarak bir küme içindeki düğümlere dağıtılarak her bir düğümün rastgele özet fonksiyonlarını kendi içinde kullanmasıyla birbirinden ayrık bir şekilde verinin indekslenmiştir. Bu şekilde sorgu ve eğitim süresinde önemli bir performans artışı sağlanmıştır.

Düğümlerde birbirinin aynı rastgele özet fonksiyonlar kullanıldığında elde edilen sonucun tek düğüm kullanıldığında elde edilen sonuçtan farklı olmadığı gösterilmiştir. Bununla birlikte rastgele özet fonksiyonların dağıtık kullanımı detaylı bir şekilde irdelenerek bu yaklaşımın getirdiği avantajlar incelenmiştir. Kullanılan yaklaşım paralel sorgu işlemeyi kullanılabilir hale getirmiştir. Yapılan deneysel çalışmalar 10 düğüm kullanıldığında sorgu süresinin LSH yönteminin klasik kullanımına oranla yaklaşık 10 kat arttığı görülmüştür. Sorgu süresindeki performans artışının yanı sıra indeksleme süresi de önemli oranda azalmıştır. İndeksleme sırasında düğümler arasında herhangi bir bilgi alış verişi söz konusu olmadığından düğümler arasında iletişim neredeyse sıfırdır. Önerilen RDH yönteminde rastgele özet fonksiyonlar kullanıldığı için özet fonksiyonların öğrenme aşaması bulunmamaktadır.

Bu çalışmada düğümlerde birbirinden farkı özellikteki özet fonksiyonlarının kullanımı detaylı bir şekilde irdelenmiştir. Her bir düğümde birbirinin aynı özet fonksiyonlar kullanıldığında, birbirinden farklı özet fonksiyonlar kullanıldığında ve veriyi yaklaşık olarak iki eşit parçaya bölebilen özet fonksiyonlar kullanıldığında oluşan durumlar incelenmiştir. Bununla birlikte dağıtık sistemin performansı aynı iş için daha fazla düğüm kullanıldığında sistemin etkinliğini ifade eden hızlanma ve veri kümesi k kat büyüdüğünde eğitim süresinin ne kadar artacağını ölçen genişleme ölçütleriyle incelenmiştir. Hızlanma için 1 düğümden 10 düğüme kadar artan sayıda düğüm kullanılarak sistemin performansı değerlendirilmiştir. Genişleme için 100 bin örnek içeren bir veri kümesinin boyutu 10 kat artırıldığında sistemin tepkisi incelenmiştir.

Öngörme hatasını elimine etmeyi sağlayan ve literatürde yoğun olarak kullanılan çapraz doğrulama tekniği kullanılarak elde edilen sonuçlar doğrulanmıştır. Bu yöntemde *k* değeri 10 olarak seçilerek işlem yapılmış ve elde edilen sonucun ortalama ve standart sapma değerleri paylaşılmıştır.

Önerilen yöntemde rastgele özet fonksiyonların veri bağımsız özelliğinden ve dağıtık sistemin avantajlarından faydalanılmıştır. Bundan dolayı önerdiğimiz yöntem büyük hacimli veri kümeleri üzerinde kullanılabilir olmuştur. Çok sayıda farklı test senaryosu üç popüler veri kümesi üzerinde uygulanmıştır.

Önerilen yöntem MAP sonuçları, hızlanma, genişleme ve çalışma zamanı ölçütlerine göre karşılaştırmalı bir şekilde incelenmiştir. Bu şekilde RDH yöntemi için en uygun şartlar ortaya konmuştur. Ayrıca bu tez çalışmasında elde edilen sonuçlar bu alanda farklı yaklaşımlardan elde edilen sonuçları bir arada sunan ve yakın zamanda dağıtık özetleme üzerine yayınlanan bir çalışmayla benzer bir deney ortam hazırlanarak çalışmada belirtilen sonuçlarla karşılaştırılmıştır.

Bu tez çalışmasında uygulanan tüm yöntemler Java programlama dili kullanılarak uygulanmıştır. Dağıtık ağ yapısı için Apache Ignite kütüphanesi kullanılmıştır. Apache Ignite kütüphanesi hafıza tabanlı dağıtık veri tabanı sunmakta ve bununla birlikte büyük veriler üzerinde bilgisayar hafızası hızında işlem yapmayı sağlamaktadır. Ayrıca Apache Ignite kütüphanesi kümeleme alt yapısı sunarak birden fazla düğümün birbirleriyle haberleşmesini sağlamaktadır.

Hızlı görüntü arama ve özetleme yöntemleri üzerinde yoğun olarak çalışılan konulardır. Son zamanlarda dağıtık veri üzerinde bir çok özetleme çalışması yapılmış ve büyük bir çoğunluğu özet fonksiyonların dağıtık veri özelliklerinin kullanımıyla öğrenilmesi üzerine olmuştur. Ayrıca yapılan çalışmaların bir çoğu düşük seviyeli öznitelikler kullanılarak yapılmıştır.

Sonraki çalışmalarda dağıtık ortamdaki görüntü veri kümeleri üzerinde insanların algıları ile makinelerin görüntüyü değerlendirmesi arasındaki uyumsuzluğu ifade eden anlamsal boşluğu giderebilecek, aynı zamanda özet fonksiyonlarını da buna göre öğrenebilecek sistemlerin geliştirilmesi hedeflenmektedir. Bu bağlamda görüntü özniteliklerini öğrenirken

aynı zamanda özet fonksiyonlarını da öğrenebilen derin ağlar üzerinde çalışmalar yapılacaktır.

KAYNAKLAR

1. Yang, B., Shang, X., Pang, S. (2017). Isometric hashing for image retrieval. *Signal Processing: Image Communication*, 59, 117–130.

2. Wang, J., Liu, W., Kumar, S., Chang, S. F. (2016). Learning to Hash for Indexing Big Data-A Survey. *Proceedings of the IEEE*, 104(1), 34–57.

3. Bai, X., Yan, C., Yang, H., Bai, L., Zhou, J., Hancock, E. R. (2018). Adaptive hash retrieval with kernel based similarity. *Pattern Recognition*, 75, 136–148.

4. Gallas, A., Kacem, N., Zagrouba, E., Barhoumi, W. (2015). Locality-sensitive hashing for region-based large-scale image indexing. *IET Image Processing*, 9(9), 804–810.

5. Liu, Y., Zhang, D., Lu, G., Ma, W. Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1), 262–282.

6. Celik, C., Bilge, H. S. (2017). Content based image retrieval with sparse representations and local feature descriptors: A comparative study. *Pattern Recognition*, 68, 1–13.

7. Alsmadi, M. K. (2017). An efficient similarity measure for content based image retrieval using memetic algorithm. *Egyptian Journal of Basic and Applied Sciences*, 4(2), 112–122.

8. Wang, X. Y., Yu, Y. J., Yang, H. Y. (2011). An effective image retrieval scheme using color, texture and shape features. *Computer Standards & Interfaces*, 33(1), 59–68.

9. Liu, G. H., Yang, J. Y., Li, Z. (2015). Content-based image retrieval using computational visual attention model. *Pattern Recognition*, 48(8), 2554–2566.

10. Salehian, H., Zamani, F., Jamzad, M. (2011). Fast Content Based Color Image Retrieval System Based on Texture Analysis of Edge Map. *Advanced Materials Research*, 341-342, 168–172.

11. Lai, H. P., Visani, M., Boucher, A., Ogier, J. M. (2014). A new interactive semi-supervised clustering model for large image database indexing. *Pattern Recognition Letters*, 37, 94–106.

12. Elalami, M. (2011). A novel image retrieval model based on the most relevant features. *Knowledge-Based Systems*, 24(1), 23–32.

13. Rajkumar, K., Sudheer, D. (2016). A review of visual information retrieval on massive image data using hadoop. *International Journal of Control Theory and Applications*, 9, 425-430.

14. Liu, G. H., Yang, J. Y. (2013). Content-based image retrieval using color difference histogram. *Pattern Recognition*, 46(1), 188–198.

15. Sreedevi, S., Sebastian, S. (2013). *Fast image retrieval with feature levels*. 2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy.

16. Xu, Y., Shen, F., Xu, X., Gao, L., Wang, Y., Tan, X. (2017). Large-scale image retrieval with supervised sparse hashing. *Neurocomputing*, 229, 45–53.

17. Wang, H., Xiao, B., Wang, L., Zhu, F., Jiang, Y. G., Wu, J. (2015). CHCF: A Cloud-Based Heterogeneous Computing Framework for Large-Scale Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12), 1900–1913.

18. Wang, J., Shen, H. T., Song, J. and Ji, J. (2014). Hashing for similarity search: A survey. *arXiv preprint*, arXiv:1408.2927.

19. Zhu, X., Li, X., Zhang, S., Xu, Z., Yu, L. and Wang, C. (2017). Graph PCA hashing for similarity search. *IEEE Transactions on Multimedia*, 19(9), 2033-2044.

20. Falchi, F., Lucchese, C., Orlando, S., Perego, R., Rabitti, F. (2012). Similarity caching in large-scale image retrieval. *Information Processing & Management*, 48(5), 803–818.

21. Gu, X., Zhang, Y., Zhang, L., Zhang, D., Li, J. (2013). An improved method of locality sensitive hashing for indexing large-scale and high-dimensional features. *Signal Processing*, 93(8), 2244–2255.

22. Liu, W., Wang, J., Kumar, S. and Chang, S. F. (2011). *Hashing with graphs*. Proceedings of the 28th International Conference on International Conference on Machine Learning, 1-8.

23. Liong, V. E., Lu, J., Duan, L. Y., Tan, Y. (2018). Deep Variational and Structural Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

24. Lu, J., Liong, V. E., Zhou, J. (2017). Deep Hashing for Scalable Image Search. *IEEE Transactions on Image Processing*, 26(5), 2352–2367.

25. Kulis, B., Grauman, K. (2009). *Kernelized locality-sensitive hashing for scalable image search.* 2009 IEEE 12th International Conference on Computer Vision.

26. Haghani, P., Michel, S., Aberer, K. (2009). *Distributed similarity search in high dimensions using locality sensitive hashing*. Proceedings of the 12th International Conference on Extending Database Technology Advances in Database Technology - EDBT 09.

27. Wang, J., Zhang, T., Sebe, N., Shen, H. T. (2017). A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 769-790.

28. Patel, F. S., Kasat, D. (2017). *Hashing based indexing techniques for content based image retrieval: A survey.* 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA).

29. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V. S. (2004). *Locality-sensitive hashing scheme based on p-stable distributions.* Proceedings of the Twentieth Annual Symposium on Computational Geometry - SCG 04.

30. Tu, N. A., Dinh, D. L., Rasel, M. K., Lee, Y. K. (2016). Topic modeling and improvement of image representation for large-scale image retrieval. *Information Sciences*, 366, 99–120.

31. Liu, L., Yu, M., Shao, L. (2017). Learning Short Binary Codes for Large-scale Image Retrieval. *IEEE Transactions on Image Processing*, 26(3), 1289–1299.

32. Raju, U., George, S., Praneeth, V. S., Deo, R., Jain, P. (2015). *Content Based Image Retrieval on Hadoop Framework.* 2015 IEEE International Congress on Big Data.

33. Kong, W., Li, W. J. (2012). Isotropic hashing. *In Advances in neural information processing systems*, 1646-1654.

34. Indyk, P., Motwani, R. (1998). *Approximate nearest neighbors*. Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing - STOC 98.

35. Ding, G., Zhou, J., Guo, Y., Lin, Z., Zhao, S., Han, J. (2017). Large-scale image retrieval with Sparse Embedded Hashing. *Neurocomputing*, 257, 24–36.

36. Cao, Y., Qi, H., Zhou, W., Kato, J., Li, K., Liu, X., Gui, J. (2018). Binary Hashing for Approximate Nearest Neighbor Search on Big Data: A Survey. *IEEE Access*, 6, 2039–2054.

37. Leng, C., Wu, J., Cheng, J., Zhang, X., Lu, H. (2015). *Hashing for Distributed Data*. Proceedings of the 32nd International Conference on Machine Learning, in PMLR 37:1642-1650.

38. Liu, X., Li, Z., Deng, C., Tao, D. (2017). *Distributed Adaptive Binary Quantization for Fast Nearest Neighbor Search*. IEEE Transactions on Image Processing, 26(11), 5324–5336.

39. Wang, S., Li, C., Shen, H. L. (2018). Distributed graph hashing. *IEEE transactions on cybernetics*, 49(5), 1896-1908.

40. Li, Z., Zhang, X., Müller, H., Zhang, S. (2018). Large-scale retrieval for medical image analytics: A comprehensive review. *Medical Image Analysis*, 43, 66–84.

41. Pavithra, L., Sharmila, T. S. (2018). An efficient framework for image retrieval using color, texture and edge features. *Computers & Electrical Engineering*, 70, 580–593.

42. Zhang, S., Metaxas, D. (2016). Large-Scale medical image analytics: Recent methodologies, applications and Future directions. *Medical Image Analysis*, 33, 98–101.

43. Li, L., Yan, C. C., Ji, W., Chen, B. W., Jiang, S., Huang, Q. (2015). LSH-based semantic dictionary learning for large scale image understanding. *Journal of Visual Communication and Image Representation*, 31, 231–236.

44. Elleuch, Z., Marzouki, K. (2016). Multi-index structure based on SIFT and color features for large scale image retrieval. *Multimedia Tools and Applications*, 76(12), 13929–13951.

45. Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 91–110.

46. Subrahmanyam, M., Wu, Q. J., Maheshwari, R., Balasubramanian, R. (2013). Modified color motif co-occurrence matrix for image indexing and retrieval. *Computers & Electrical Engineering*, 39(3), 762–774.

47. Yang, J., Jiang, B., Li, B., Tian, K., Lv, Z. (2017). A Fast Image Retrieval Method Designed for Network Big Data. *IEEE Transactions on Industrial Informatics*, 13(5), 2350–2359.

48. He, J., Feng, J., Liu, X., Cheng, T., Lin, T. H., Chung, H., Chang, S. F. (2012). *Mobile product search with Bag of Hash Bits and boundary reranking*. 2012 IEEE Conference on Computer Vision and Pattern Recognition.

49. Shakhnarovich, G., Viola, P., Darrell, T. (2003). *Fast pose estimation with parameter sensitive hashing*. Proceedings Ninth IEEE International Conference on Computer Vision.

50. Gong, Y., Kumar, S., Rowley, H. A., Lazebnik, S. (2013). *Learning Binary Codes for High-Dimensional Data Using Bilinear Projections*. 2013 IEEE Conference on Computer Vision and Pattern Recognition.

51. Shi, Q., Li, H., Shen, C. (2010). *Rapid face recognition using hashing*. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

52. Li, X., Shen, C., Dick, A., Hengel, A. V. D. (2013). *Learning Compact Binary Codes for Visual Tracking*. 2013 IEEE Conference on Computer Vision and Pattern Recognition.

53. Chum, O., Philbin, J., Zisserman, A. (2008). *Near Duplicate Image Detection: min-Hash and tf-idf Weighting*. Procedings of the British Machine Vision Conference 2008.

54. Viswanathan, K., Kim, M., Li, J., Gonzalez, M. (2016). A Memory-Driven Computing Approach to High-Dimensional Similarity Search. *Hewlett Packard Labs*.

55. Liu, H., Wang, R., Shan, S., Chen, X. (2016). *Deep Supervised Hashing for Fast Image Retrieval*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

56. Wang, J., Kumar, S., Chang, S.-F. (2012). Semi-Supervised Hashing for Large-Scale Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12), 2393–2406.

57. Qi, L., Zhang, X., Dou, W., Ni, Q. (2017). A Distributed Locality-Sensitive Hashing-Based Approach for Cloud Service Recommendation From Multi-Source Data. *IEEE Journal on Selected Areas in Communications*, 35(11), 2616–2624.

58. Bahmani, B., Goel, A., Shinde, R. (2012). *Efficient distributed locality sensitive hashing*. Proceedings of the 21st ACM International Conference on Information and Knowledge Management - CIKM 12.

59. Wang, J., Kumar, S., Chang, S.-F. (2010). *Semi-supervised hashing for scalable image retrieval*. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

60. Mao, M., Zheng, Z., Chen, Z., Liu, H., He, X., Ye, R. (2016). *Two-dimensional PCA hashing and its extension*. 2016 23rd International Conference on Pattern Recognition (ICPR).

61. Weiss, Y., Torralba, A., Fergus, R. (2009). Spectral hashing. *In Advances in neural information processing systems*. 1753-1760.

62. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F. (2012). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12), 2916-2929.

63. Kulis, B., Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. *In Advances in neural information processing systems*, 1042-1050.

64. Lu, J., Hu, J., Zhou, J. (2017). Deep Metric Learning for Visual Understanding: An Overview of Recent Advances. *IEEE Signal Processing Magazine*, 34(6), 76–84.

65. Lu, X., Zheng, X., Li, X. (2017). Latent Semantic Minimal Hashing for Image Retrieval. *IEEE Transactions on Image Processing*, 26(1), 355–368.

66. Chi, L., Zhu, X. (2017). Hashing Techniques. ACM Computing Surveys, 50(1), 1–36.

67. Slaney, M., Casey, M. (2008). Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes]. *IEEE Signal Processing Magazine*, 25(2), 128–131.

68. Tian, D., Tao, D. (2017). Global Hashing System for Fast Image Search. *IEEE Transactions on Image Processing*, 26(1), 79–89.

69. Jiang, Q. Y., Li, W. J. (2015). *Scalable graph hashing with feature transformation*. In Twenty-Fourth International Joint Conference on Artificial Intelligence.

70. Jiang, G., Wang, W. (2017). Error estimation based on variance analysis of k -fold cross-validation. *Pattern Recognition*, 69, 94–106.

71. Qayyum, A., Anwar, S. M., Awais, M., Majid, M. (2017). Medical image retrieval using deep convolutional neural network. *Neurocomputing*, 266, 8–20.

72. Deng, C., Liu, X., Mu, Y., Li, J. (2015). Large-scale multi-task image labeling with adaptive relevance discovery and feature hashing. *Signal Processing*, 112, 137–145.

73. Wang, D., Otto, C., Jain, A. K. (2015). Face search at scale: 80 million gallery. *arXiv preprint*, arXiv:1507.07242.

74. Chang, L., Pérez-Suárez, A., Hernández-Palancar, J., Arias-Estrada, M., Sucar, L. E. (2017). Improving visual vocabularies: a more discriminative, representative and compact bag of visual words. *Informatica*, 41(3).

75. Kafai, M., Eshghi, K., Bhanu, B. (2014). Discrete Cosine Transform Locality-Sensitive Hashes for Face Retrieval. *IEEE Transactions on Multimedia*, 16(4), 1090–1103.

76. Durmaz, O., Bilge, H. S. (2017). *Fast image searching in large scale image database*. 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Türkiye.

77. Durmaz, O., Bilge, H. S. (2018). *Fast image search with distributed hashing*. 26th Signal Processing and Communications Applications Conference (SIU), İzmir, Türkiye.

78. Durmaz, O., Bilge, H. S. (2019). *Image search with Predefined Randomized Hash Functions*. 27th Signal Processing and Communications Applications Conference (SIU), Sivas, Türkiye.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: DURMAZ, Osman
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 08.06.1985, Sivas
Medeni hali	: Evli
Telefon	: 0 (538) 664 83 66
e-mail	: osmandurmaz85@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Doktora	Gazi Üniversitesi / Bilgisayar Mühendisliği	Devam ediyor
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	2011
Lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	2008
Lise	Bornova Anadolu Lisesi	2004

İş Deneyimi

Yıl	Yer	Görev
2017-Halen	Ericsson	Yazılım Mühendisi
2012-2017	ÖSYM	Bilişim Uzmanı
2009-2012	SGK	Yazılım Mühendisi
2008-2009	LST Yazılım	Yazılım Mühendisi

Yabancı Dil

İngilizce

Yayınlar

Durmaz, O., Bilge, H. S. (2019). Fast image similarity search by distributed locality sensitive hashing. *Pattern Recognition Letters*, 128, 361–369.

Durmaz, O., Bilge, H. S. (2019). *Image Search with Predefined Randomized Hash Functions.* 27th Signal Processing and Communications Applications Conference (SIU) (Poster), Sivas, Türkiye.

Durmaz, O., Bilge, H. S. (2018). *Fast image search with distributed hashing*. 26th Signal Processing and Communications Applications Conference (SIU), İzmir, Türkiye.

Durmaz, O., Bilge, H. S. (2017). *Fast image searching in large scale image database*. 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Türkiye.

Durmaz, O., Bilge, H. S. (2011*). Effects of dimensionality reduction and feature selection in text classification*. 19th Signal Processing and Communications Applications Conference (SIU), Antalya, Türkiye.

Hobiler

Yüzme, Masa tenisi

GAZİ GELECEKTİR..