T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİ İLE GÖRÜNTÜ HASH KODLARINI OLUŞTURMA

YÜKSEK LİSANS TEZİ

Elif AKKAYA

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Bilim Dalı

HAZİRAN 2024

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİ İLE GÖRÜNTÜ HASH KODLARINI OLUŞTURMA

YÜKSEK LİSANS TEZİ

Elif AKKAYA

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Bilim Dalı

Tez Danışmanı: Dr.Öğr.Üyesi Burhan BARAKLI

HAZİRAN 2024

Elif AKKAYA tarafından hazırlanan "Derin Pekiştirmeli Öğrenme Yöntemi ile Görüntü Hash Kodlarını Oluşturma" adlı tez çalışması 28.06.2024 tarihinde aşağıdaki jüri tarafından oy birliği/oy çokluğu ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı **Elektronik Mühendisliği** Bilim Dalı'nda Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı : **Doç.Dr. Selçuk EMİROĞLU** .....
Sakarya Üniversitesi

Jüri Üyesi : **Dr.Öğr.Üyesi Burhan BARAKLI**(Danışman) .....
Sakarya Üniversitesi

Jüri Üyesi : **Dr.Öğr.Üyesi Muhammed Ali PALA** .....
Sakarya Uygulamalı Bilimler Üniversitesi

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum "DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİ İLE GÖRÜNTÜ HASH KODLARINI OLUŞTURMA" başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(28/06/2024).

Elif AKKAYA

Aileme

TEŞEKKÜR

Tez çalışması sürecinde bilgi birikimi ile destekleyen, yönlendirmeleri ve gösterdiği ilgi ile ilerleme motivasyonumu her zaman yüksek tutan, her adımda yanımda olan değerli danışman hocam Dr.Öğr.Üyesi Burhan BARAKLI ‘ya minnet ve teşekkürlerimi sunarım.

Elif AKKAYA

İÇİNDEKİLER

Sayfa

KISALTMALAR

ADSH : Asymmetric Deep Supervised Hashing
CIFAR : Canadian Institute For Advanced Research
CNN : Convolution Neural Network
CPU : Central Process Unit
CSQ : Central Similarity Quantization
DAGH : Deep Anchor Graph Hashing
DBDH : Balanced Discrete Hashing
DHN : Deep Hashing Network
DPSH : Deep Pairwise Supervised Hashing
DSH : Deep Supervised Hashing
GPU : Graphics Process Unit
ISDH : Instance Similarity Deep Hashing
MNIST : Modified National Institute Of Standards And Technology
RAM : Recurrent Attention Model
ReLU : Rectified Linear Units
RNN : Recurrent Neural Network
SGD : Stochastic Gradient Descent

SİMGELER

s	: Durum
a	: Eylem
r	: Ödül
t	: Zaman adımı
π	: Politika
V	: Değer fonksiyonu
Q	: Eylem-değer fonksiyonu
γ	: Zayıflatma faktörü
$\mathbb{E}$	: Beklenen değer
α	: Öğrenme oranı
θ	: Politika parametreleri
w	: Değer fonksiyonu parametreleri
R	: Ödül fonksiyonu
P	: Geçiş olasılıkları
∇_θ	: Gradyan
r_t	: Zaman t de alınan ödül
G_t	: Belirli bir zamandan sonraki toplam indirilmiş ödül

TABLO LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

DERİN PEKİŞTİRMELİ ÖĞRENME YÖNTEMİ İLE GÖRÜNTÜ HASH KODLARINI OLUŞTURMA

ÖZET

Görüntü hash'leri, dijital görüntülerin özünü temsil eden kısa, benzersiz ve sabit uzunluktaki karakter dizileridir. Hashing yöntemlerinde çoğunlukla ikili sayı sistemindeki kodlar oluşturulur. Her kod bir görüntüyü ya da görüntü ile ilgili bir kurguyu temsil eder. Görüntü hashlerinin çıkarılmasının temel amacı, görüntülerin daha kolay ve verimli bir şekilde işlenmesini ve karşılaştırılmasını sağlamaktır. Genellikle büyük ölçekli veri tabanlarında arama, veri sıkıştırma ve güvenlik ihtiyacı için görüntülerin hash kodu üretilir. Hash kodları hızlı arama, veriyi kolay saklama gibi avantajlar sağlamaktadır. Geleneksel hash yöntemleri genellikle farklı hash işlevlerini bağımsız olarak öğrenir ve bu işlevler arasındaki ilişkileri göz ardı etmektedir. Oysa bu ilişkiler duyarlılık doğruluğunu önemli ölçüde artırabilmektedir. Geleneksel hash yöntemleri arasında, hash kodunu sıralı olarak öğrenen yöntemler bulunmaktadır. Bu yöntemlerin karmaşık optimizasyon gerektirmesi ve derin ağlara doğrudan uygulanamaması gibi bazı kısıtlamaları bulunmaktadır. Bu nedenle, daha etkili ve derin hash yöntemlerine ihtiyaç duyulmaktadır. Bu tür kısıtlamaları aşmak ve daha etkili bir hash yöntemi geliştirmek için derin pekiştirmeli öğrenme prensiplerinden ilham alınmaktadır. Pekiştirmeli öğrenme, karmaşık ve belirsiz ortamlarda etkili kararlar vermeyi sağlayan güçlü bir araçtır. Pekiştirmeli öğrenme genellikle Markov karar süreci (Markov Decision Process, MDP) ve güçlü öğrenme algoritmaları kullanılarak modellenir. Bu algoritmalar, ajanın görevi başarıyla tamamlaması için gerekli olan en iyi eylem stratejilerini keşfetmesine yardımcı olur. Pekiştirmeli öğrenme, bir yapay zeka ve makine öğrenimi alanı olarak karşımıza çıkar. Bu öğrenme şekli, belirsiz ve karmaşık ortamlarda hareket eden bir ajanın nasıl optimal kararlar alacağını öğrenmesini sağlar. Temelde, ajan bir çevre içinde eylemler gerçekleştirir ve bu eylemlerin sonuçlarına göre ödüller alır veya cezalarla karşılaşır. Amaç, toplamda maksimum ödülü elde etmek için hangi eylemlerin yapılması gerektiğini öğrenmektir. Bu çalışmada derin öğrenme ve pekiştirmeli öğrenme kullanarak yenilikçi bir yöntem ile görüntüler için hash kodu üretme amaçlanmıştır. Derin öğrenme yöntemleri makine öğrenme yöntemlerinden farklı olarak görüntü özelliklerini kendi başına çıkarabilmektedir. Derin pekiştirmeli öğrenme, bir ajanın çevresiyle etkileşimler yoluyla davranış öğrenmesi gereken bir problemi temsil etmektedir. Bu çerçevede, her bir hash işlevini bağımsız olarak öğrenmek yerine, hash işlevlerini ardışık bir karar süreci olarak modellemeyi ve önceki işlevler tarafından yapılan hataları düzelterek öğrenme hedeflenmiştir. Bu yaklaşım, farklı hash işlevleri arasındaki ilişkileri göz önünde bulundurarak duyarlılık doğruluğunu artırmayı amaçlamaktadır. Geri beslemeli sinir ağları (Recurrent Neural Networks, RNN), ardışık verilerle çalışabilen ve zaman içindeki bağımlılıkları yakalayabilen özel bir yapay sinir ağı türüdür. RNN'ler, önceki zaman adımlarındaki bilgileri hatırlayarak, bu bilgileri mevcut zaman adımının çıktısını üretmek için kullanabilir. Bu şekilde, geleneksel hash yöntemlerinde olduğu gibi karmaşık optimizasyon işlemleri gerektirmeden, derin ağlar üzerinde etkili bir şekilde uygulanabilir bir hash yaklaşımı

geliştirme amaçlanmıştır. Politika yöntemi olarak Actor-Critic yöntemi kullanılmıştır. Yaygın olarak kullanılan Cifar-10, Nus-wide, Mirflickr veri setleri üzerinde yapılan deneyler yaklaşımın etkinliğini göstermiştir.

GENERATING IMAGE HASH CODES WITH DEEP REINFORCEMENT LEARNING METHOD

SUMMARY

In the digital world, processing and comparing images quickly and efficiently has become a necessity. This need becomes even more evident for search, data compression and security purposes, especially in large-scale databases. Image hashes are short, unique, fixed-length strings of characters that represent the essence of digital images and are an ideal solution to meet these requirements. It enables faster searching, easier storage and security of digital images. The main purpose of image hashes is to enable easier and more efficient processing and comparison of digital images. This process meets the need for fast searching in large databases, data compression and security. Hash codes are fixed-length strings of characters that represent the essence of images. These codes are used to determine the similarity or dissimilarity of digital images and enable rapid searches in large databases. Provides a concise representation of digital images. In this way, searching or comparing a specific image becomes much faster and more efficient. Storing and processing large data sets is often difficult and costly. Hash codes allow these data sets to be represented more compactly, reducing storage and processing costs. Hash codes can be used to verify the authenticity of an image or to detect unauthorized access. Traditional hashing methods usually learn different hash functions independently and ignore the relationships between these functions.This may negatively impact the accuracy and efficiency of hash codes. Among traditional methods, there are also methods that learn the hash code sequentially. However, these methods have limitations such as requiring complex optimization and cannot be directly applied to deep networks. Traditional hashing methods that work with independent hash functions usually learn different hash functions independently. This approach can reduce the accuracy of hash codes because relationships between hash functions are ignored. Traditional methods that require complex optimization processes often require complex optimization processes. This makes hash codes difficult to learn and apply. Deep neural networks are ideal for processing complex and large data sets. However, since traditional hashing methods cannot be integrated into these networks, their effectiveness remains limited. Inspired by deep reinforcement learning principles to overcome the limitations of traditional hashing methods and develop a more effective hashing method. Reinforcement learning is a powerful tool that enables effective decision-making in complex and uncertain environments. This learning method allows an agent to learn how to behave in an environment. Through rewards and punishments given to the agent, the agent adapts to its environment and develops optimal strategies. Deep reinforcement learning is an effective approach to overcome the limitations of traditional hashing methods. Deep reinforcement learning has made significant progress in the field of artificial intelligence and machine learning in recent years and is used effectively in complex decision-making processes. This method enables an agent to learn how to behave within an environment to achieve certain goals, allowing it to develop optimal strategies by taking advantage of environmental conditions and interactions. Deep

reinforcement learning is considered a powerful tool for making effective decisions, especially in environments containing uncertainty. Deep reinforcement learning essentially involves the agent learning how to act in an environment to maximize or minimize a specific goal. This learning process usually occurs through the rewards the agent receives from situations, actions, and the environment. When the agent observes a situation (observation), he chooses an action depending on the environmental situation and receives a reward or punishment as a result of this action. Deep reinforcement learning aims for the agent to optimize this observation-action-reward cycle to maximize long-term reward. Deep reinforcement learning has been used successfully in various application areas. For example, it has potential for use in game theory and strategy games (e.g. AlphaGo), robotic systems, optimizing financial algorithms, automation processes, and even medical decision support systems. These methods can successfully perform various tasks without human intervention by processing complex and large data sets. The deep reinforcement learning process generally requires powerful computational resources and evolves in direct proportion to the ability to learn from large data sets. As the agent explores its environment, it learns to choose the most appropriate actions through trial and error, and over time this process becomes more efficient. The optimization process involves various algorithms and approaches used to improve the behavior of the agent.This learning method is used to obtain effective results on complex and large data sets. Deep reinforcement learning enables learning hash functions more accurately and effectively. It also manages complex optimization processes more effectively. In this way, the accuracy of hash codes is increased and the learning process becomes more efficient. Deep reinforcement learning can be easily integrated with deep neural networks. This integration increases the efficiency of hashing methods and makes processing large data sets easier. Deep reinforcement learning provides many advantages in hashing methods. These advantages enable more accurate, faster and efficient processing and comparison of digital images. To ensure high accuracy, deep reinforcement learning learns and applies hash functions more accurately. In this way, the accuracy of hash codes is increased. In terms of efficiency, deep reinforcement learning manages complex optimization processes more efficiently. This makes hash codes easier to learn and apply. In processing large data sets, deep reinforcement learning makes processing large data sets easier. Thanks to integration with deep neural networks, the efficiency of hashing methods is increased and large data sets are processed faster. While traditional hashing methods have some limitations, new generation hashing methods inspired by deep reinforcement learning principles overcome these limitations. Deep reinforcement learning offers great advantages in processing and comparing digital images by enabling learning hash functions more accurately and effectively. The high accuracy, efficiency and security advantages provided by deep reinforcement learning lead to revolutionary developments in the field of digital image processing. These advances enable faster, safer and more efficient processing of digital images. In this way, users and institutions can manage and use digital image data more effectively.One of the main components of the model is the deep learning network. This network consists of a feature representation network to extract features of images and a policy network to convert images into binary codes. The policy network includes the Actor-critic method along with RNN (recurrent neural network) and acts like an agent. Feedback neural networks (RNN) are a special type of artificial neural network that can work with sequential data and capture dependencies over time. By remembering information from previous time steps, RNNs can use this information to produce the output of the current time step. Thanks to these features, they are widely

used in many application areas such as language modelling, time series analysis and speech recognition. The architecture of RNNs includes feedback connections in their hidden layers, and these connections process information through weight matrices and activation functions. However, RNNs have difficulties in learning long-term dependencies, which can lead to problems such as vanishing gradients or exploding gradients. To overcome these difficulties, applications such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are used. Improved RNN architectures have been developed. LSTMs learn long-term dependencies more effectively by using cell state and gate mechanisms, thus greatly reducing the problem of vanishing gradients. GRUs, on the other hand, work similarly to LSTMs, but have a simpler structure and control the flow of information through update gates. These improved architectures increase the performance of RNNs, enabling them to deal with data that is more complex and has long-term dependencies. RNN and its derivatives are now successfully applied in many fields such as natural language processing, financial forecasting and speech recognition. The actor-critic method is an approach frequently used in the field of reinforcement learning and is an extension of policy gradient methods. This method consists of two main components: "actor" and "critic". The Actor component learns and implements the policy that determines which action to choose in a given situation. The Critic component learns a value function that evaluates how good the actions chosen by the actor are. This evaluation provides a feedback mechanism used to improve the actor's policy. The biggest advantage of the actor-critic method is that it provides more efficient and stable learning processes by combining the strengths of both value-based and policy-based methods. It is known for its effective performance, especially in continuous action spaces and large state spaces. This method forms the basis of many deep reinforcement learning algorithms and has been successfully applied in various fields such as robotics, gaming, financial modelling. Another main component of the approach is the reward system used to shape future decisions based on the results of previous decisions. A reward function is used to evaluate the agreement between the generated hash codes and the actual tags. In the future, with the development of artificial intelligence and machine learning technologies, image hashing methods are expected to be further optimized and expanded. The role of these technologies in the management and analysis of more complex and larger data sets will increase and new application areas will be discovered. The entire network is trained by optimizing the reward function, and these reward functions evaluate the actions taken by the agent, encouraging correct actions and punishing incorrect actions. In this way, the performance and accuracy of the model are increased. The proposed deep reinforcement learning-based hashing approach works with a sequential learning strategy, which continuously optimizes the overall accuracy by correcting errors produced by past actions. The method is optimized with reward functions and sequential learning strategies, ensuring efficient and accurate conversion of images into binary codes. The contribution of deep learning models to hashing performance heralds important future developments in this field. In this study, various hashing methods were examined on important data sets such as CIFAR10, NUS-WIDE and MIRFlickr. The CIFAR10 dataset contains a total of 60,000 color images consisting of 10 different classes, and each image is 32x32 pixels in size. A query set of 1000 randomly selected images was used in the experiments. The NUS-WIDE dataset consists of approximately 270,000 images, and in the study, experiments were carried out using 1000 images selected from 21 concepts. The MIRFlickr dataset contains 25,000 images, and 1000 random images from this dataset were determined as the query set. The experiments were carried out using the PyTorch

framework and the initial parameters of the network were initialized with the VGG-19 model trained on the ImageNet dataset. The results of this study show that deep learning and deep reinforcement learning techniques provide significant advantages compared to traditional methods in hashing operations.

.

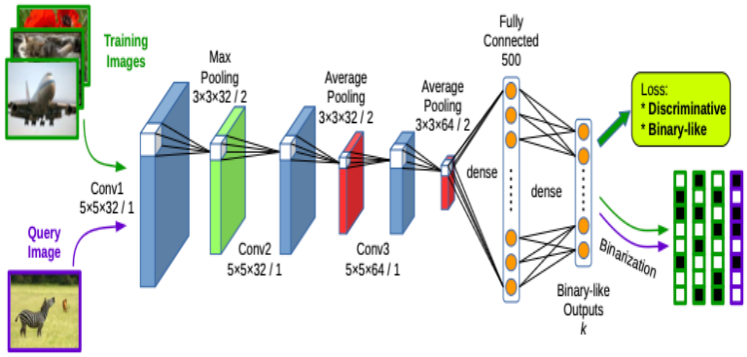
1. GİRİŞ

Görüntü hashleri, görüntü işleme alanında önemli bir role sahip olan benzersiz tanımlayıcılardır. Hash değerleri, bir görüntünün özgün içeriğini temsil eder ve birçok uygulamada kullanılmaktadır. Veri tabanında benzer görüntüleri hızlı bir şekilde bulmak için kullanılan hash değerleri, benzer içeriğe sahip olan görüntüleri gruplamak ve sıralamak için kullanılmaktadırlar. Ayrıca görüntülerin hash değerleri, orijinal görüntü verisini daha az miktarda bilgiyle temsil ederek depolama ve iletim açısından verimliliği artırmaktadır [1]. Bir görüntünün hash değeri, orijinalinde herhangi bir değişiklik yapılmadığı sürece sabit kalır, bu sayede görüntünün orijinalliğini doğrulamak için kullanılabilmektedir. Bunların yanı sıra görüntü hashleri, hızlı nesne tanıma, yüz algılama ve telif hakkı ihlali tespiti gibi görevlerde kullanılabilmektedir. Benzer görüntülerin hash değerleri hızlı bir şekilde eşleştirilebilir ve tanınabilir, bu da çeşitli görüntü işleme problemlerini çözmek için önemli bir araç sağlamaktadır [1,2]. Pekiştirmeli öğrenme, bir ajanın çevresiyle etkileşimde bulunarak deneyimlerden öğrenmesini sağlayan bir öğrenme yaklaşımıdır. Ajan, çevreden gelen bilgileri gözlemleyerek, çevredeki durumu anlamaya çalışır ve bu duruma bağlı olarak eylemler seçmektedir. Seçilen eylemler, çevreyi değiştirir ve ajan bir ödül sinyali alır, bu da seçilen eylemin kalitesini veya başarısını yansıtmaktadır. Ödül sinyalinden alınan geri bildirim, ajanın en iyi davranışlarını belirlemede stratejilerini ayarlamak amacıyla kullanılabilir. Bu şekilde, pekiştirmeli öğrenme, karmaşık ve belirsiz ortamlarda etkili kararlar verme yeteneği kazandırmak için güçlü bir araç olabilmektedir [3]. Derin pekiştirmeli öğrenme modelinde, bu süreç genellikle derin sinir ağları kullanılarak gerçekleştirilmektedir. Derin ağlar, geniş bir veri setinden öğrenilen karmaşık ilişkileri temsil etmek için kullanılmaktadır. Ajan, ağın çıktılarını kullanarak çevreyle etkileşime girer ve en iyi eylemi seçmek için ağın öğrenilmiş temsillerini kullanmaktadır [3,4]. Bu öğrenme yaklaşımı, birçok alanda başarıyla uygulanmıştır. Özellikle, atari oyunlarında insan düzeyinde performans elde edilmesi ve Go gibi karmaşık strateji oyunlarında ustalık düzeyinde performans sergilenmesi gibi etkileyici sonuçlar elde edilmiştir. Ayrıca, pekiştirmeli öğrenme, robotikte, otonom sürüşte, finansal tahminlerde ve doğal dil işlemede de kullanılmıştır [3]. Son

zamanlardaki, görüntü sınıflandırma ve nesne tespiti gibi birçok bilgisayarlı görme görevinde derin ağların başarısından ilham alarak, derin hash yöntemleri önerilmiştir. Derin hash yöntemleri, son zamanlarda umut verici sonuçlar elde etmiştir. Görüntü hash yöntemleri, benzer görüntüleri benzer hash kodlarına eşlemek için kullanılmaktadır ve görüntüler Hamming uzayına yansıtılarak etkili bir görüntü eşleme sağlanmaktadır. Bu yöntemler veri bağımsız ve veri bağımlı olarak iki kategoriye ayrılmaktadır. Veri bağımsız yöntemler, LSH gibi yöntemler kullanarak görüntüleri ikili kodlara eşlerken, veri bağımlı yöntemler etiket bilgisini kullanarak hash işlevlerini öğrenmektedir. Denetimsiz yöntemler, etiket bilgisine ihtiyaç duymadan hash işlevlerini öğrenirken, denetimli yöntemler etiket bilgisini kullanmaktadır. Bu yöntemler, farklı hash yaklaşımlarını ve performanslarını değerlendirerek görüntü eşleme alanında önemli bir yer bulmaktadır. Ancak, bu yöntemler farklı hash işlevlerini bağımsız olarak öğrenir ve duyarlılık doğruluğunu artırabilen aradaki ilişkileri göz ardı eder. Derin ağlar üzerinde etkili bir şekilde uygulanabilir, karmaşık optimizasyon işlemleri gerektirmeyen bir hash yaklaşımı geliştirme hedeflenmektedir. Politika yöntemi olarak Actor-Critic yöntemi kullanılmıştır, 3 veri seti üzerinde gerçekleştirilen uygulamalar önerilen yaklaşımın etkinliğini göstermiştir.

2. LİTERATÜR ÖZETİ

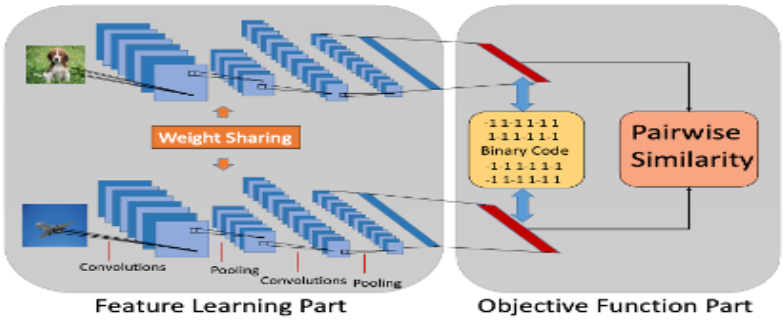
Liu ve arkadaşları [5], yaptıkları çalışmada büyük veri kümelerinde etkili görüntü alımını hedefleyen Derin Denetimli Hash (DSH) yöntemini sunmuşlardır. DSH, Evrişimli Sinir Ağları'nın (CNN'ler) güçlü görüntü temsilini kullanarak benzerlikleri koruyan ikili kodları öğrenmeyi amaçlamaktadır. DSH, eğitimde benzer ve benzersiz görüntü çiftlerini kullanarak her görüntünün çıktısını ayrık değerlere yaklaşmaya teşvik eden bir CNN mimarisi benimsemektedir. Özel olarak tasarlanan kayıp fonksiyonu, görüntü çiftlerinden elde edilen bilgiyi kodlarken aynı zamanda çıkış alanını düzenleyerek, ikili kodlama ile görüntüler arasındaki benzerliği korumayı hedeflemektedir. CIFAR-10 ve NUS-WIDE veri setleri üzerinde yapılan deneyler, DSH'nin umut verici bir performans sergilediğini ortaya koymaktadır. Yöntemin hızlı kodlama yeteneği ve genel kullanılabilirliği, gelecekteki araştırmalar için önemli bir potansiyel sunmaktadır.



Şekil 2.1. DSH modeli [5].

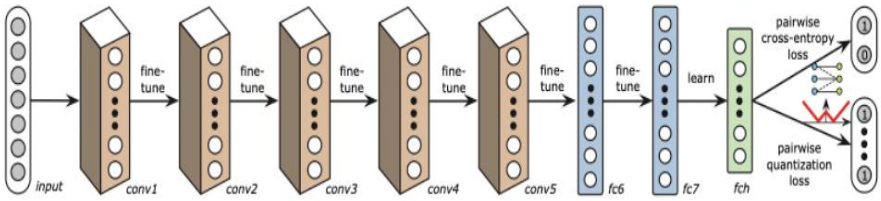
Wu-Jun Li ve arkadaşları [6], büyük ölçekli görüntü alımı için geliştirilen derin hash yöntemlerini ele alan bir çalışma sunmuşlardır. Geleneksel yöntemlerden farklı olarak, önerilen derin ikili denetimli hash yöntemi (DPSH), ikili etiketlere sahip uygulamalarda eş zamanlı özellik öğrenme ve hash kod öğrenme yetenekleri sunmaktadır.

Yapılan deneyler, DPSH'nin diğer yöntemlere kıyasla daha iyi performans göstererek, görüntü alma uygulamalarında son teknolojiye uygun çözümler sunabileceğini ortaya koymaktadır.



Şekil 2.2. DPSH modeli [6].

Zhu ve arkadaşları [7], büyük ölçekli multimedya alımı için kullanılan yaygın bir yaklaşım olan hash kodlama üzerine odaklanmışlardır. Denetimli hash yöntemlerinde, görüntüler önceden belirlenmiş veya makine öğrenimi ile elde edilmiş özellik vektörleri olarak temsil edilmektedir.



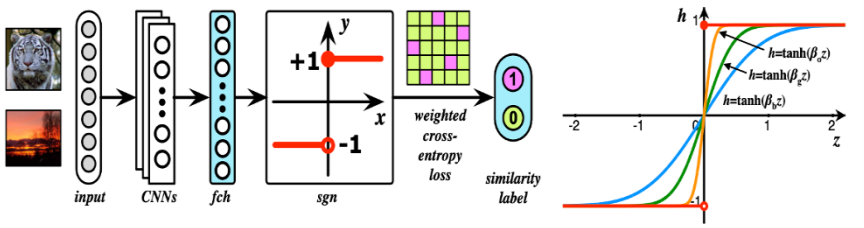
Şekil 2.3. DHN modeli [7].

Ancak mevcut yöntemlerde niceleme hatası ve özellik gösterimi arasındaki uyumsuzluk nedeniyle optimal olmayan hash kodlamalar üretilebilmektedir. Bu sorunu çözmek için, yeni bir Derin Hashing Ağı (DHN) mimarisi önermişlerdir. DHN modeli, görüntü temsilini yakalamak için evrişim havuzu katmanları, ikili hash kodları oluşturmak için hash katmanı, benzerliği korumak için çift yönlü çapraz entropi kaybı katmanı ve hash kalitesini kontrol etmek için ikili niceleme kaybı içermektedir. Yapılan kapsamlı deneyler, DHN modelinin diğer hash yöntemlerine göre önemli artışlar sağladığını göstermektedir.

Cao ve arkadaşlarının [8] çalışmalarında sunduklarında HashNet, büyük ölçekli multimedya erişiminde kullanılan en yakın komşu aramasında etkili olan bir derin

öğrenme mimarisidir. Bu mimari, dengesiz benzerlik verilerinden tam ikili hash kodları öğrenmektedir ve yakınsama garantili devam yöntemi kullanmaktadır. HashNet'in ana fikri, düzgünleştirilmiş aktivasyon fonksiyonu ile başlayarak gradyan problemi ile başa çıkarak, süreç boyunca gelişen bir eğitim stratejisi kullanmaktır.

HashNet, tam olarak ikili hash kodları üretebilme yeteneğine sahiptir ve kapsamlı ampirik kanıtlar, standart kıyaslamalarda son teknoloji ürünü multimedya alma performansı sağlayabildiğini göstermektedir.



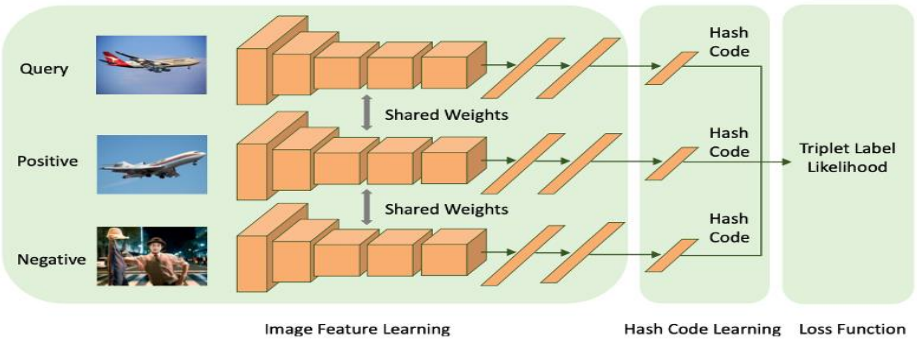
Şekil 2.4. HashNet modeli [8].

Su ve arkadaşları [9] çalışmalarında, büyük ölçekli görüntü kümelerinde yaklaşık en yakın komşu aramasında kullanılan hash algoritmasının derin hash ile birleştirilmesini ele almaktadır. Derin hash işlemindeki zorluk, ağ çıkışına uygulanan ayrık kısıtlamalardan kaynaklanmaktadır. Bu zorluğu aşmak için, açgözlü bir prensibi benimseyen ve her yinelemede ağı olası optimal ayrık çözüme doğru güncelleyen bir yöntem önerilmektedir. Ayrıca, ayrık kısıtlamaları korumak için işaret fonksiyonunu sıkı bir şekilde kullanan bir hash kodlama katmanı tasarlanmıştır. Deneyler, önerilen yöntemin hem denetlenen hem de denetlenmeyen görevlerde diğer hash yöntemlerden daha iyi performans gösterdiğini göstermektedir.

Zhenan ve arkadaşları [10] çalışmalarında, görsel ve video verilerinin artan büyüklüğüyle birlikte hash işlemlerin önemini vurgulamışlardır. Derin öğrenme tekniklerinin son gelişmelerinden faydalanarak geliştirilen derin hash yöntemleri, görüntü alımında umut verici sonuçlar elde etmiştir. Ancak, önceki yöntemlerin özellikle anlamsal bilgilerin tam olarak kullanılmaması gibi bazı sınırlamaları bulunmaktadır. Çalışma, derin denetimli ayrık hash algoritması geliştirmekte ve öğrenilen ikili kodların sınıflandırma için uygun olduğu varsayımına dayanmaktadır. Hem ikili etiket bilgisi hem de sınıflandırma bilgisi, hash kodların tek bir akış çerçevesinde öğrenilmesi için kullanılmıştır. Ayrıca, hash kodların ayrık doğasından kaynaklanan zorlukları ele almak için alternatif bir minimizasyon yöntemi

benimsenmiştir. Yapılan deneyler, bu yöntemin karşılaştırmalı görüntü alma veri kümelerinde mevcut en gelişmiş yöntemlerden daha iyi performans sergilediğini göstermektedir**.**

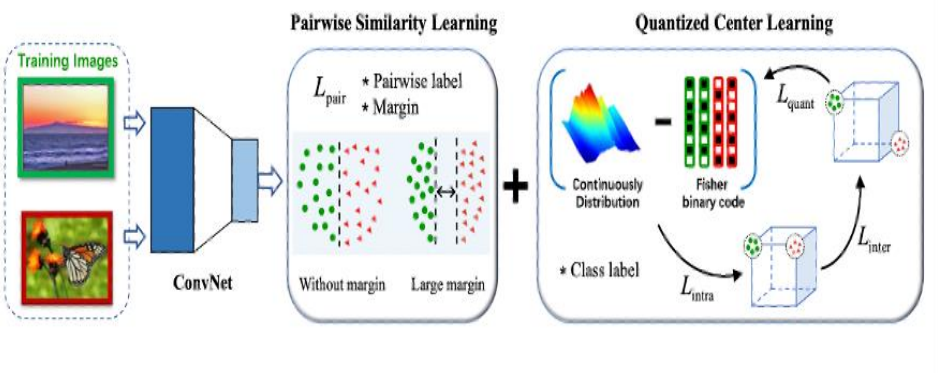
Wang ve arkadaşları [11] çalışmalarında önerdikleri derin hash yöntemi, üçlü etiketlerin denetiminde görüntü özelliklerini ve hash kodlarını eş zamanlı olarak öğrenmektedir. Yöntem, öğrenilen hash kodlar altında verilen üçlü etiket olasılığını maksimize ederek yüksek kaliteli hash kodları elde etmektedir. Yapılan kapsamlı deneyler, yöntemin, DPSH yöntemi ve diğer üçlü etiket tabanlı derin hash yöntemlerine kıyasla daha üstün performans sergilediğini göstermektedir. Bu başarı, CIFAR-10 ve NUS-WIDE veri kümelerindeki tüm temel yöntemlerle karşılaştırılmıştır.



Şekil 2.5. DPSH with triplet labels modeli [11].

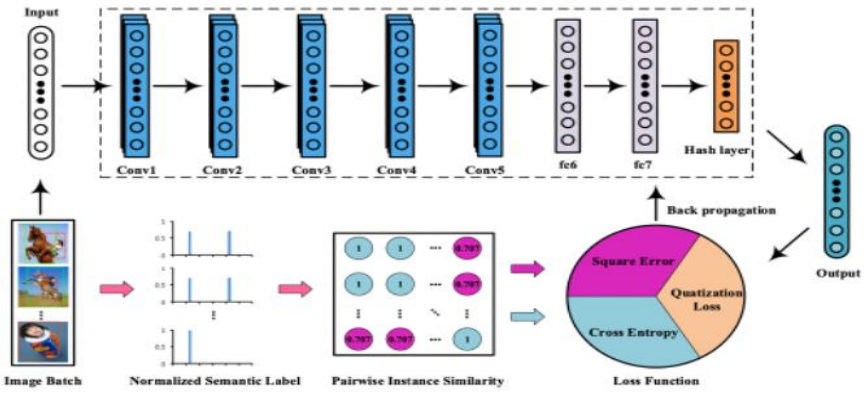
Nqiang ve arkadaşları [12] çalışmalarında, kolay erişim gerektiren büyük veri kümelerinin analizi için ayrık hash yöntemlerinin önemini vurgulamışlardır. Varolan yöntemler, yüksek boyutlu verileri depolamak ve işlemek için verimli olan kompakt ikili kodları kullanarak anlamsal benzerliği korumaktadır. Ancak, ikili kodların sürekli türevleri olmaması nedeniyle gradyan tabanlı yöntemlerin kullanımı zor olmaktadır. Çalışmada, ikili uzayda maksimum sınıf ayrılabilirliğini optimize etmeye odaklanan bir denetimli derin ikili hash yöntemi tanıtılmıştır.

Yöntem, sınıf içi mesafeleri azaltırken sınıflar arası mesafeleri artırmak için Fisher'in doğrusal diskriminant analizinden ilham almaktadır. Yapılan deneyler, özellikle az sayıda eğitim verisi içeren büyük veri kümelerinde yöntemin diğerlerine üstünlüğünü göstermektedir.



Şekil 2.6. Deep fisher hashing modeli [12].

Zhang ve arkadaşları [13] çalışmalarında, büyük ölçekli görüntü alımı için hash kodlama yöntemlerinin geliştirilmesi üzerine odaklanmışlardır. Geleneksel ikili benzerlik tanımlarının sınırlamalarını aşmak amacıyla, çok etiketli görüntüler için örnek benzerliği temel alan yeni bir derin hash yöntemi olan ISDH önermişlerdir. ISDH, ikili benzerlik kaybının çok etiketli görüntülerin sıralama bilgilerini yetersiz şekilde kodlamasını önlemektedir ve örnek benzerliğine dayalı bir yaklaşımla öğrenme süreçlerini geliştirmektedir.Yapılan deneyler, ISDH'nin diğer yöntemlere kıyasla daha iyi performans sergilediğini ve çok etiketli görüntü alımında öncü bir konumda olduğunu göstermektedir.

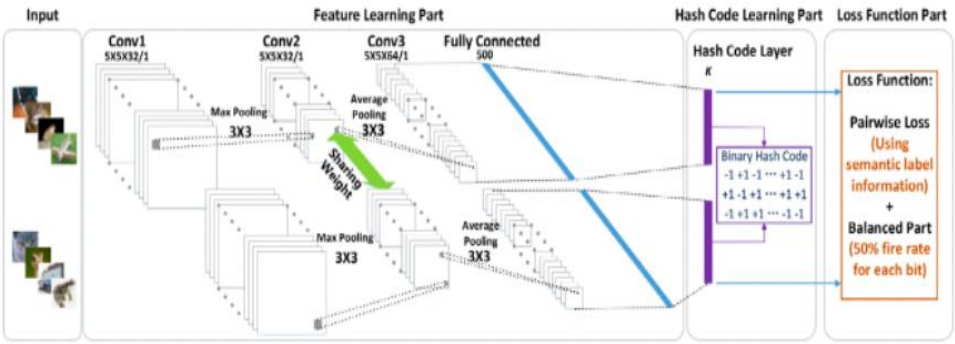


Şekil 2.7. ISDH modeli [13].

Zhenga ve arkadaşları [14] çalışmalarında sundukları Derin Dengeli Ayrık Hashing (DBDH) yöntemi, büyük ölçekli multimedya alımlarında yaygın olarak kullanılan hash yöntemlerdeki sürekli gevşeme stratejisinin neden olduğu nicelendirme hatalarını azaltarak, ayrık optimizasyonun kolaylığını sağlamaktadır.

Bu yöntem, düz tahminci ve ayrık gradyan yayılımını kullanarak hash kodlarını öğrenmektedir ve ayrıca ikili etiket ve sınıflandırma bilgilerini birleştirerek benzerlik

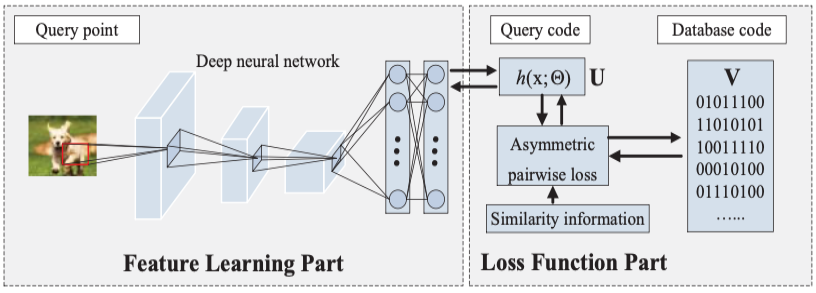
ve dengeyi koruyan bir kayıp fonksiyonu sunmaktadır. Yapılan kapsamlı deneyler, DBDH yönteminin, dört görüntü alım kıyaslama veri kümesinde diğer gelişmiş hash yöntemlere göre daha üstün performans sergilediğini doğrulamıştır.



Şekil 2.8. DBDH modeli [14].

Jiang ve arkadaşları [15] çalışmalarında, büyük ölçekli en yakın komşu aramasında yaygın olarak kullanılan ve depolama ile arama verimliliği açısından etkili olan en yakın komşu arama yöntemlerinden biri olan asimetrik derin denetimli hash işlemini tanıtmışlardır. Mevcut derin denetimli hash yöntemlerinden farklı olarak, ADSH (Asimetrik Derin Denetimli Karma), sorgu noktalarını simetrik olmayan bir şekilde ele alarak, yalnızca sorgu noktaları için derin bir hash işlevi öğrenirken, veri tabanı noktalarının hash kodlarını doğrudan öğrenmektedir.

Yapılan deneyler, ADSH'nin büyük ölçekli veri tabanlarıyla etkili bir şekilde başa çıkabilen ve gerçek uygulamalarda gelişmiş performans gösteren bir yöntem olduğunu ortaya koymaktadır.

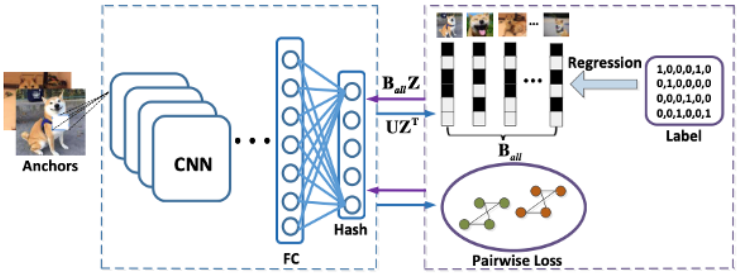


Şekil 2.9. ADSH modeli [15].

Chen ve arkadaşları [16] çalışmalarında, ikili kod öğrenimi için derin denetimli hash yöntemlerin yüksek hesaplama maliyeti ve sınırlı donanım hafızası sorunlarına çözüm

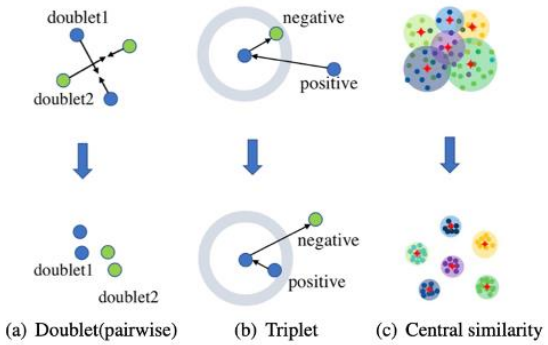
olarak, bir bağlantı grafiği tabanlı öneri sunmuşlardır. Bu öneri, eğitim setinden bir alt küme seçme ve mini toplu veri ile ağı güncelleme stratejisi kullanarak verimli ikili kod öğrenmeyi sağlamaktadır. DAGH çerçevesi, derin özelliklerle ikili kodlar arasındaki benzerlikleri bağlantı grafiği üzerinden karakterize ederek, daha ayırt edici geri bildirimler elde etmeyi amaçlamaktadır.

Yapılan deneyler, önerilen yöntemin diğer derin hash yöntemlere kıyasla daha iyi geri getirme performansı sağladığını ve hesaplama süresinin daha düşük olduğunu göstermektedir.



Şekil 2.10. DAGH modeli [16].

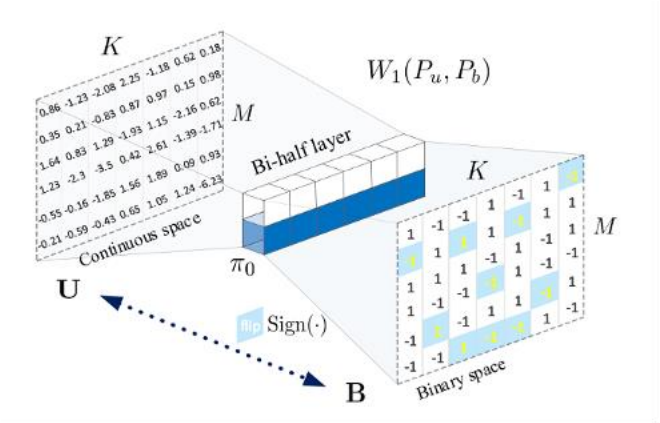
Yuan ve arkadaşları [17] çalışmalarında, hash öğrenmeyi geliştirmek amacıyla yeni bir küresel benzerlik metriği olan "Merkezi Benzerlik Nicelemesi"ni (CSQ) önermişlerdir. CSQ, hash kodlarının belirli merkezlere yaklaşmasını teşvik ederek, benzer veri çiftleri için birleşik hash kodları ve benzer olmayan çiftler için dağınık hash kodları üretebilmektedir. Yapılan deneyler, CSQ'nun büyük ölçekli görüntü ve video alma görevlerinde alma performansını önemli ölçüde artırabildiğini göstermektedir. Önerilen yöntem, derin hash öğrenmenin merkezi benzerliğini formüle ederek yüksek kaliteli hash kodları oluşturmayı hedeflemektedir.



Şekil 2.11. CSQ modeli [17].

Li ve arkadaşlarının [18] çalışmalarında denetimsiz hash için önerdikleri Bi-half Net, büyük resim veya video koleksiyonlarını kolay bir şekilde dizine eklemeyi hedefleyen bir yaklaşımdır. Bu yöntem, ikili kodların entropisini maksimize etmek amacıyla yeni bir parametresiz derin hash katmanı olan Bİ-half Net'i içermektedir. Bu katman, bit entropisini optimize etmek için sürekli görüntü özelliklerini optimum yarım-yarım bit dağılımına yaklaştırmaya odaklanmaktadır.

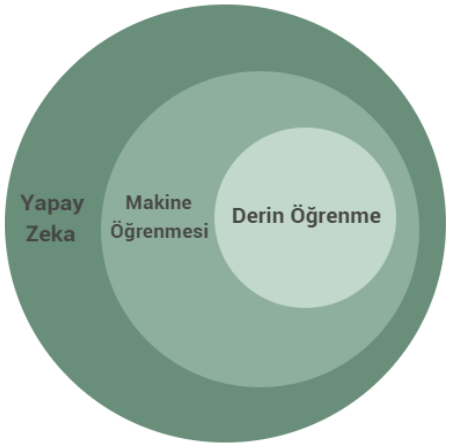
Yaklaşım, çeşitli görüntü ve video veri kümeleri üzerinde yapılan deneylerde, kompakt kodlar elde etme konusunda olumlu sonuçlar vermiştir. Bi-half Net'in avantajları, parametresiz olması ve diğer hash yöntemlere göre daha iyi performans göstermesidir.



Şekil 2.12. Bi-halfNet modeli [18].

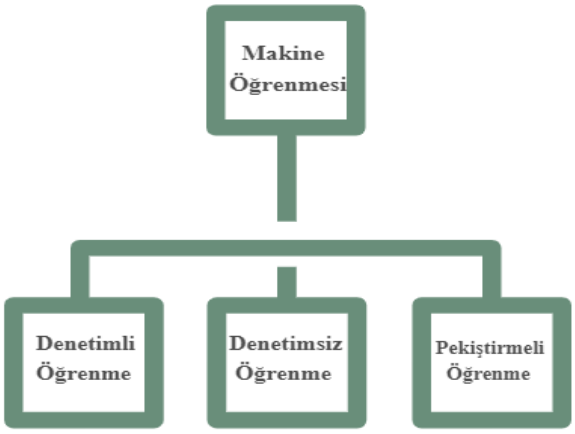
3. PEKİŞTİRMELİ ÖĞRENME

Tecrübe ile kazanılmış insan eylemlerinin bilgisayarlar ile gerçekleştirilebilme becerisi yapay zeka kavramını oluşturmuştur. Yapay zeka, makine öğrenimi ve derin öğrenme dallarını kapsamaktadır.



Şekil 3.1. Yapay zeka ve alt alanları.

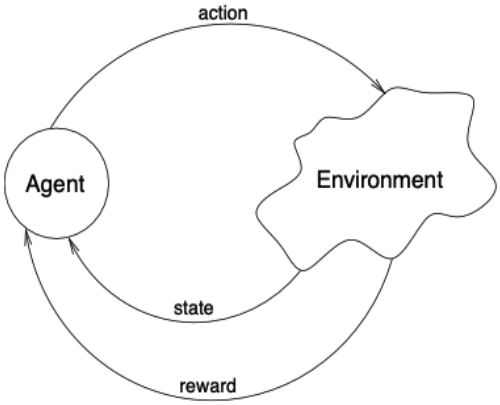
Makine öğrenmesi algoritmalar aracılığı ile verileri kullanarak öğrenme ve tanıma yapar. Bu öğrenme şekli denetimli, denetimsiz ve pekiştirmeli (reinforcement) olmak üzere üç algoritmayı içermektedir. Denetimli öğrenmede girdi etiketleri ile eğitim yapılır ve bununla gelecek girdi etiketlerinin tahmini gerçekleştirilir.



Şekil 3.2. Makine öğrenmesi algoritmaları

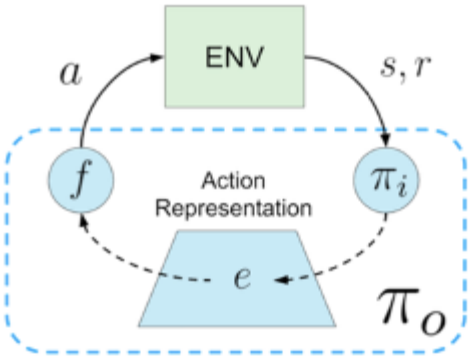
Denetimli öğrenme sınıflandırma ve regresyon problemlerini ele alır. Sınıflandırma problemlerin de kategorilere göre sınıflandırma yapılır ve SVM, Naive Bayes, Lojistik regresyon gibi algoritmalar ile çözülür. Çıkış değişkeninin gerçek ya da sürekli değer olduğu problemeler ise regresyon problemleridir ve bu problemler bayesian doğrusal regresyon, doğrusal regresyon veya doğrusal olmayan regresyon algoritmaları ile çözülür [19].

Denetimsiz öğrenmede ise model, etiket olmadan sadece girdiler ile eğitilir ve bu girdiler benzer özellikler üzerinden sınıflandırılır. Sonraki girdi etiketleri özelliklerinin sınıflar ile olan benzerliği üzerinden tahmin edilir [19].



Şekil 3.3. Pekiştirmeli öğrenme modeli [20].

Pekiştirmeli öğrenme deneme-yanılma ile en iyi eylemi öğrenir ve bu sayede en iyi eylem ile ilgili önceden bir bilgi ihtiyacı yoktur bu durum pekiştirmeli öğrenmenin verilen örneklerle sınırlanmamasını sağlar.



Şekil 3.4. Genel politikanın yapısı [21].

Çevreyi keşfederek öğrenen ajan, başlangıçta bilinmeyen bir ortama yerleştirilir ve çevreden aldığı geri bildirimler olan ödül (takviye sinyali) ile eylemlerini şekillendirir.

Ajan plan yapmak için geçmiş deneyimlerinden edindiği bilgileri kullanır ve bu yönde bir eylem seçer. Ajanın kontrol politikasını durum, eylem ve ödül ilişkisi oluşturur. Ajan, durumu gözlemleyerek kararlarını verir.

Belirli bir t zamanda çevre, ajanına bir durum s_t sağlar, ajan bu duruma bağlı olarak bir eylem a_t gerçekleştirir. Ardından çevre güncellenir ve ajanına yeni bir durum s_{t+1} ile sayısal bir ödül r_{t+1} sağlamaktadır. Model, ajanın politikası π tarafından belirlenen duruma bağlı olarak yapılan eylemi içermektedir [20,22].

Ajanın optimum eylemi seçmeye çalışması toplam ödülü maksimum yapmaya yönelik politika olan π* 'yi bulma sürecini oluşturur. İteratif olarak uygulanan politikalarla elde edilen ödüllerin değerlendirilmesi yapılarak eylemler tekrar ayarlanır bunun sonucunda toplam ödülü maksimum yapmaya yönelik politika olan π* keşfedilir [23,24,25].

Ajan bazen anlık ödüllerden vazgeçerek daha büyük uzun vadeli ödüller elde etmelidir. Ancak keşfetme ve sömürme arasındaki denge önemlidir. Bu süreçte finite-horizon, infinite-horizon discounted ve average-reward modelleri kullanılabilir [26,27].

Finite-horizon modeli, sınırlı bir zaman aralığında ajanın beklenen ödülü optimize etmesi gerektiğini belirtir [27].

Average-reward modelinde amaç maksimum uzun vadeli ortalama elde etmektir [27].

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_{t+n} = \sum_{i=0}^{n} r_{t+i} \tag{3.1}$$

Sonsuz-horizon indirimli modelinde vurgulanan uzun vadeli ödüldür, bu model ajan aktivitesine bir zaman kısıtlaması koymaz [27].

$$R_t = \lim_{n\to\infty} \left(\frac{1}{n} \sum_{i=0}^{n} r_{t+i} \right) \tag{3.2}$$

Model, ayrık zaman adımları kullanır ve durum kümesi S ile tanımlanır, her bir durum için ayrı bir eylem kümesi A(s) bulunur. Ajan, politika π'ye göre eylem seçer.

Ardışık karar alma sürecinde ajan, çevrenin başlangıç keşfi ile olasılıklar kümesinden bir aksiyon seçer.

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \gamma^3 \cdot r_{t+4} \cdots = \sum_{i=1}^{\infty} \gamma^{i-1} \cdot r_{t+i} \tag{3.3}$$

Çevre, bu seçime cevap olarak ajanın performansını gösteren bir ödül sinyali üretir. Bu süreç, ajanın çevreden edindiği gözlemlerden aksiyon seçimleri yapması ve çevrenin ödül sinyalleri ile yanıt vermesiyle tekrarlanır [28].

Markov karar süreçleri algı, eylem ve hedef unsurlarını en sade halleriyle barındıracak şekilde tasarlanmıştır.

Pekiştirmeli öğrenmede, dinamik programlamadaki politika değerlendirme ve politika iyileştirme süreçlerine benzer olarak öğrenme tahmini ve öğrenme kontrolü süreçleri vardır [25].

Öğrenme kontrolü ile markov karar verme sürecinin optimal politikasını veya değer fonksiyonunu tahmin etmeyi model bilgisine gerek duymadan gerçekleştirme amaçlanır. Öğrenme tahmini ise, önceki model bilgisi olmadan markov karar verme sürecinin politika değerlendirme sorununu çözmeye yöneliktir. Pekiştirmeli öğrenmede öğrenme tahmini, tahmin problemlerinin çok adımlı olması ve çözüm için, gelecekteki bir dizi karara bağlı olarak sonuçları tahmin etme zorunluluğuyla denetimli öğrenmedekinden farklıdır [25].

3.1. Markov Karar Süreci

Markov özelliği, bir durumun gelecekteki durumunun sadece mevcut durumuna bağlı olduğunu; geçmiş durumlar, eylemler veya ödüllerle bağlantılı olmamasını ifade eder. Markov özelliği, sistemin mevcut bilgisiyle geleceği tahmin etme yeteneği açısından önemli bir kısıtlamadır [20].

Markov özelliğini taşıyan pekiştirmeli öğrenme görevi olan markov karar sürecinde, mevcut durumun bilgisiyle gelecekteki durumları tahmin etme yeteneği ön plandadır. Markov karar süreci, bir ajanın çevresiyle etkileşimde bulunduğu, mevcut durumunun markov özelliğine sahip olduğu ve ajanın belirli eylemler alarak ödüller kazandığı bir öğrenme çerçevesidir. Bu süreç, ajanın mevcut durumunu temel alarak gelecekteki durumları tahmin etme ve en uygun eylemleri seçme yeteneği üzerine odaklanır [20].

$$P(s' \mid s, a) = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \quad (3.4)$$

$$R(s', s, a) = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\} \quad (3.5)$$

3.2. Markov Özelliği

Pekiştirmeli öğrenme modeli markov özelliği kullanılarak tanımlanabilir. Ajan farklı eylemlerin sonuçlarını, t zaman adımında bulunduğu durum (s_t) hakkındaki bilgiye dayanarak tahmin eder. Markov durumu, eylemin sonucunun önceki durumlar, eylemler veya ödüllerle bağlantılı değil sadece mevcut durum s_t 'ye bağlı olmasıdır. Markov çevresi ise çevredeki tüm durumların markov özelliğinde olmasıdır [20].

Markov özelliği iki olasılık dağılımının eşitliği kullanılarak formüle edilebilir:

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} = Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t \dots s_0, a_t \dots a_0, r_t \dots r_0\} \quad (3.6)$$

Markov özelliği, pekiştirmeli öğrenme için önemli bir prensiptir. Ajanın yalnızca mevcut durumu bilmesine dayanır ve ajan eylem seçmelerini bu temelde gerçekleştirir. Ancak gerçek hayattaki çoğu ortam markov özelliklere yaklaşsada tam anlamıyla sahip olmayabilir. Çevre markov ortamına iyi bir yaklaşım sunmuyorsa, ajanın sorunlara etkili çözümler bulma da zorlukları oluşabilir [20].

3.3. Ödül Fonksiyonu

Kısa vadeli ödül, bir sonraki zaman adımındaki ödülü ifade eder. Sadece kısa vadeli ödülü maksimize etmeye odaklanan bir politika, anlık ödülü optimize etmelidir. Uzun vadeli ödülü optimize etmeye yönelik bir politika ise tüm gelecekteki ödüllerin toplamını optimize etmelidir. Politika hem kısa vadeli hem de uzun vadeli ödülleri optimize etmeye çalışıyorsa, gelecekteki ödülleri bir indirim oranı (γ) ile ağırlıklandırarak bunu başarabilir. İndirim oranı γ, 0 ile 1 arasında bir değer alır; γ sıfır olduğunda beklenen indirimli gelecekteki ödül kısa vadeli ödülle, 1 olduğunda ise uzun vadeli ödülle aynı olmaktadır [20].

$$R_t = r_{t+1} \quad (3.7)$$

$$R_t = \sum_{k=0}^{T} r_t \; + k + 1 \quad (3.8)$$

3.4. Değer Fonksiyonu

Optimal politika π^*, herhangi bir durum için optimum toplam ödülü sağlayan politikadır. Gelecekteki ödül $V^{\pi}(s)$, tüm s durumları için optimize edilmelidir [20].

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{\Sigma_{k=0}^{\infty}\gamma^k r_t + k + 1 \mid s_t = s\} \quad (3.9)$$

Ajanın belirli bir durumda (s) belirli bir eylem (a) seçmesi durumunda beklenen gelecekteki ödülünü ifade eden bu fonksiyon, durum-değer fonksiyonu olarak adlandırılır ve $Q^{\pi}(s,a)$ olarak ifade edilir [20].

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\{\Sigma_{k=0}^{\infty}\gamma^k r_t + k + 1 \mid s_t = s, a_t = a\} \quad (3.10)$$

İki değer fonksiyonunun da optimal bir alt yapıya sahip olması yerel hesaplamaların birleştirilerek genel bir optimal çözüm oluşturulabilmesine imkan tanır.

$$Q^{\pi}(s,a) = E_{\pi}\{\Sigma_{k=0}^{\infty}\ \gamma^k r_t + k + 1 \mid s_t = s, a_t = a\} \quad (3.11)$$

$$E_{\pi} * \{r_{t+1}\ \gamma\Sigma_{k=0}^{\infty}\gamma^k r_t + k + 2 \mid s_t = s, a_t = a \quad (3.12)$$

3.5. Optimal Politikanın (π*) Tanımı

Tüm s ∈ S durumlardaki optimal eylem-değer fonksiyonu denklemde verildiği gibi tanımlanabilir:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad (3.13)$$

$$(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad (3.14)$$

Durum-değer fonksiyonu, eylem-değer fonksiyonu ile ilişkili bir şekilde tanımlanabilir:

$$V^*(s) = \max_{a \in A(s)} Q^*(s,a) \quad (3.15)$$

Q*(s,a) fonksiyonuna bağlı olarak π*(s) tanımlanabilir:

$$\pi^*(s) = arg\max_{a \in A(s)} Q^*(s,a) \tag{3.16}$$

π*(s)'i hesaplamak için yapılan tanımlama , tek opsiyonun bütün olası değerleri π için denemeyi gerektiren Q*(s,a)'yı içermesinden dolayı daha kolay hesaplanabilen yeni bir Q*(s,a) tanımına ihtiyaç duyulmaktadır [20].

$$V^*(s) \ = V^{\pi^*}(s) \tag{3.17}$$

$$V^*(s) \ = \max_{a \in A(s)} \Sigma_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma V^*(s') \right] \tag{3.18}$$

$$Q^{\pi^*}(s,a) = E_{\pi^*}\{\Sigma^{\infty}_{k=0} \ \gamma^k r_t + k + 1 \mid s_t = s, a_t = a\} \tag{3.19}$$

$$Q^{\pi^*}(s,a) = \ \pi^*(s) \ = \ \underset{a \in A(s)}{argmax} \ \Sigma_{s' \in S} P^a_{ss'} \ [\ R^a_{ss'} + \gamma \max_{a' \in A(s')} Q^*(s',a') \ ] \tag{3.20}$$

3.6. $P^a_{ss'}$ ve $R^a_{ss'}$ 'dan Optimal Politikayı Bulmak

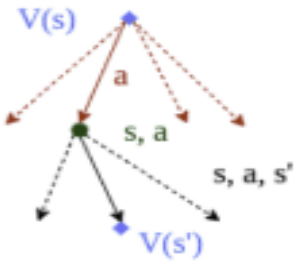
Optimal politika π*(s)'i bulma aşamasında Q*(s) veya V*(s)'i bilmeyi ve bütün olası π değerlerini ile hesaplama yapmayı gerektirdiğinden (3.17) ve (3.18) denklemleri kullanışlı olmamaktadır. Politika hesaplamalarında kullanmak için Bellman'ın eşitlik denklemi $P^a_{ss'}$ve $R^a_{ss'}$ ile optimal değer iterasyonu ve politika iterasyonu barındıran dinamik programlama algoritmaları geliştirilmiştir [20].

3.7. Değer İterasyonu

Optimal politika π*(s)'i bulmak için değer iterasyonunda öncelikle optimal değer fonksiyonu V*(s) hesaplanır [20,29]. Bu süreçte rastgele değer fonksiyonuyla başlanır ve $V(s)$ her adımda güncellenir.

$$V(s) = \max_a \ \sum_{s',r'} p(s',r|s,a)[r + \gamma V(s')] \tag{3.21}$$

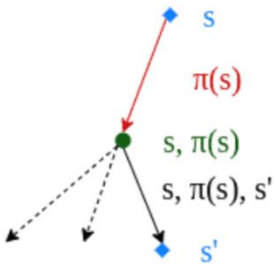
Değer iterasyonu algoritmasında olası tüm eylemlerden maksimumu alınır. Bu iterasyon algoritmasında değerlendirme ve sonrasında iyileştirme yapmak yerine, durum değeri fonksiyonu tek bir adımda güncellenir.



Şekil 3.5. Değer iterasyonu [20].

3.8. Politika İterasyonu

Değer iterasyonu, V* bulunduktan sonra politikayı bir kez günceller, politika iterasyonu ise her iterasyonda politikayı günceller ve modifiye edilmiş politikayı bir sonraki iterasyonda kullanır. Her iki algoritmanın ana zorluğu, Pa′ ve Ra′ değerlerinin önceden bilinmesini gerektirmesidir [20,30].



Şekil 3.6. Politika iterasyonu yapısı [20].

Politika iterasyonunda bir politika (π) seçerek başlanır ve yakınsamaya kadar politika yinelemeli olarak değerlendirilir ve geliştirilir.

$$V(s) = \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')] \tag{3.22}$$

$$\pi(s) = \underset{a}{argmax} \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')] \tag{3.23}$$

(r) eylemin gerçekleştirilmesiyle elde edilen ödüldür. (α) gelecekteki ödüller, (γ) bir indirim faktörü, (p) geçiş olasılığıdır.

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_* \tag{3.24}$$

Politikanın iyileşme göstermediği son duruma kadar politika değerlendirme ve politika güncelleştirme adımlarına devam edilir.

3.9. Değer İterasyonu ve Politika İterasyonu Farkları

Politika iterasyonuna sabit bir politika ile değer iterasyonuna ise değer fonksiyonu seçilerek başlanır ve yakınsamaya ulaşılana kadar iteratif olarak iyileştirilir [29,30].

Politika iterasyonunda politika güncellenirken değer iterasyonu değer işlevi üzerinden yenilenir [29 ,30].

Her iterasyonda politika iterasyonu, politikanın değerlendirildiği ve geliştirildiği iki aşamadan geçer. Buna karşılık değer iterasyonunda iki aşamayıda kapsayacak şekilde tüm olası eylemler için fayda fonksiyonunun maksimumu alınır. Bu durum değer iterasyonunu politika iterasyonuna göre daha basit kılsada, politika iterasyonundaki iki aşamasının birleştirilmesi ve tüm olası eylemlerin aynı anda çalıştırması değer iterasyonun hesap yükünü ağırlaştırır [29].

Politika iterasyonu daha az yineleme ile optimal politikaya yakınsama yapar ve daha hızlı sonuçlanır [30].

3.10. Temporal-Difference Öğrenmesi

Temporal-Difference Learning (TD), gecikmeli ödüller yaşandığında öğrenmeyi tahmin etme yoluyla yapılabilecek eylemlerin değerlendirilmesi durumudur [27]. Temporal-Difference öğrenme yönteminde tahmin değeri ile gelecekteki geçici tahmin arasındaki fark baz alınır yani tahmin değişikliklerine dayanır. Bu yönü ile tahmin hatası ile öğrenen geleneksel öğrenme yöntemlerinden ayrışmaktadır. Bu öğrenme yöntemi için geçmiş tahminlerin saklanması gerekmez sadece önceki tahmin değerini kullanır [27].

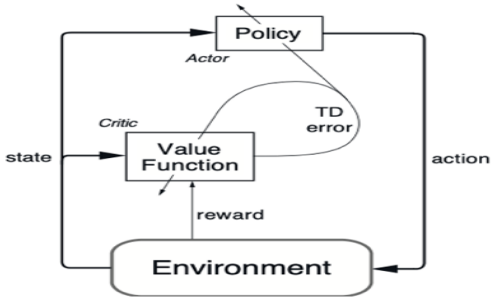
Temporal-Difference (TD) öğrenme geleneksel öğrenme yöntemlerine kıyasla hesaplama kolaylığı ve hızlı yakınsama sağlamaktadır [27]. Temporal-Difference yönteminde tahmin problemleri, deneyimler kullanılarak çözülür.

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + 1 + \gamma V(s_t + 1) - V(s_t)] \tag{3.25}$$

$$r_t + 1 \ + \gamma V(s_t + 1) \tag{3.26}$$

3.11. Actor-Critic Metodu

Actor-critic bir temporal-difference yöntemidir ve politika temsili için değer fonksiyonundan bağımsız olarak ayrı bir bellek yapısı kullanarak politika temsili yapar. Actor eylem seçimlerinde kullanılan politika olarak bilinir. Critic ise eylemleri eleştirir ve tahmini değer fonksiyonu olarak bilinmektedir. Bu eleştiri skaler bir sinyaldir ve temporal-difference hatası olarak alınır. Actor-critic ilişkisi ile öğrenme yönlendirilir [25].



Şekil 3.7. Actor-Critic mimarisi [31].

Critic, bir durum değer fonksiyonudur. Her eylem seçiminden sonra, critic yeni durumu değerlendirir [25].

$$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V(S_t) \tag{3.27}$$

Denklemde verilen ifadedeki V_t critic tarafından t zaman da uygulanan değer fonksiyonudur. Temporal-difference hatası pozitif ise bu A_t nin gelecekte seçme eğiliminin güçlendirilmesi gerektiğini, negatif ise bu eğilimin zayıflatılması gerektiğini önerir [25].

3.12. Çevre Modelleri Olmadan Öğrenme

Pekiştirmeli öğrenme de model tabanlı ve model tabansız olmak üzere iki farklı yaklaşım vardır. Model tabanlı yaklaşımdaki çevre modeli çevreyi simüle eder bu sayede ajanın en iyi politikayı bulmasına katkıda bulunur. Model-tabansız yaklaşım ise model olmadan en iyi politikayı bulabilen ancak veriyi verimli kullanamayan ve

fazla deneyim gerektiren bir yaklaşımdır [27]. Model-tabansız yaklaşım, daima optimum ödülü seçmeye çalışarak olabildiğince çok ödül elde etmeyi amaçlar. Elde edilen bilgi hemen ödül elde amacıyla kullanılarak aç gözlü bir yaklaşım sergilenebilir ama bu durum keşfi kısıtlayarak ajanın daha iyi çözümleri gözardı etmesine sebep olabilir [20].

3.13. Q-Öğrenmesi (Q-Learning)

Q-Öğrenme yöntemi, model-tabansız bir yaklaşım ile en iyi politikayı öğrenmeye odaklanan bir pekiştirmeli öğrenme algoritmasıdır. Q-öğrenmesi algoritmasındaki ana düşünce, gerçekleştirilecek eylemlerin değerlendirilip ödülü maksimum yapan hareket yönünde eylem seçiminin yapılmasıdır.

$$\pi^*(s) = argmax_a \, Q^*(s,a) \tag{3.28}$$

$$Q(s,a) \leftarrow (1 - \alpha)\, Q(s,a) + \alpha\, (\mathrm{r} + \gamma max_{a'}\, Q(s',a')) \tag{3.29}$$

Eylem-değer fonksiyonu Q(s, a), durum (s) ve eylem (a) çiftinin değerini depolar. Q*(s, a), optimal davranışı elde etmeyi sağlayan en iyi politikaya karşılık gelir. Optimal politika, eylem-değer fonksiyonundan her durum için en yüksek değere sahip olan eylem seçimi ile elde edilebilir [27]**.**

Denklem 3.28 de verilen ifadedeki α değeri 0 ile 1 arasında değerler alır ve öğrenme hızı faktörüdür. Bu öğrenme hızı faktörü, Q-değerlerinin optimal değerlere yaklaşım hızını etkiler. Q-Öğrenmede öğrenmenin off-policy olması ajanın eğitimi için izlenen politika ve eylem için gerekli politikanın farklı olmasındandır. Q değerlerinin güncellenmesi, bir zaman farkı yöntemi ile gerçekleşir ve Q (s', a')'deki (tahmini) değere dayanır [27].

$$Q_{a_i}(t+1) \leftarrow Q_{a_i}(t) + \alpha[r_i(t+1) + \gamma \max_j Q_{a_i}(t) - Q_{a_i}(t)] \tag{3.30}$$

Standart Q-öğrenme formülü, eylem-değer güncelleme fonksiyonu ile Q'yu düzenler. Formülde verilen γ indirim faktörü, α adım büyüklüğü parametresi, a_i ise t zamanındaki alınan eylemdir. Eylem seçme mekanizması ile eylem-değer fonksiyonu olasılık dağılımına dönüştürülerek her Q güncellemesinden sonra yeni optimal politika elde edilir. Sık kullanılan eylem seçme mekanizmaları arasında ϙ-Greedy ve

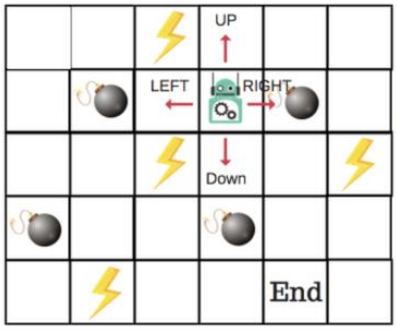
Boltzmann keşfi mekanizmaları vardır. Boltzmann keşif mekanizması, keşif ve sömürü dengesini kontrol eden bir parametreyi kullanırken, ϱ-Greedy mekanizması ise optimum eylemi belirli bir olasılıkla seçer [32].

3.14. Q-Tablosu

Ajanın en iyi eylemi seçmesi için karar verme sürecini yönlendirmek ve davranışını optimize etmek amacıyla kullanılan Q-Tablosu, durum ve eylem sayısına göre çok boyutlu olabilir. Tablodaki her bir hücre, o durumda o eylemin gerçekleştirilmesi ile elde edilecek maksimum ödülü temsil eder. Tablodaki değerler zamanla güncellenir ve mevcut politika sürekli olarak iyileştirilir [33].

Q-Tablosunu labirent içerisindeki bir robot senaryosu üzerinden örneklendirirsek;

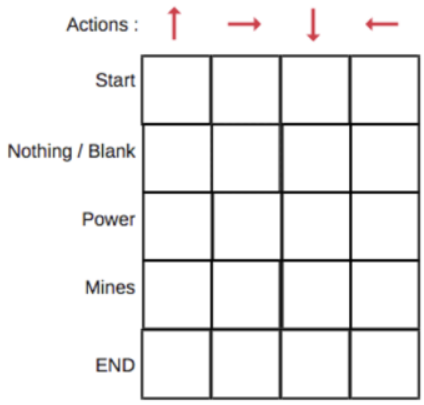
Bu senaryoya göre robot, labirentteki mayınlardan kaçarak en kısa sürede bitiş noktasına ulaşmalıdır, ancak aynı anda yalnızca bir taşı hareket ettirebilir ve her adımda 1 puan kaybeder güç alırsa bir puan kazanır, hedefe ulaşması durumunda 100 puan alır, mayına basması durumunda ise ölür.



Şekil 3.8. Robot-Labirent senaryosu [34].

Robot senaryosunda dört eylem (a=4) ve beş durum (s=5) var. Böylece dört sütun ve beş satırdan oluşan bir tablo oluşturulur.

Algoritma, Bellman denklemini kullanarak Q fonksiyonunu günceller durum (s) ve eylem (a) olmak üzere iki girdi alır ve her bir durum-eylem çifti için bu değerleri hesaplar.



Şekil 3.9. Robot-Labirent senaryosu eylem-durum tablosu [34].

Başlangıçta tecrübesiz olduğu için Q-tablosundaki tüm değerler sıfırdır ve robot çevreyi keşfetmeye başladığında, Q fonksiyonu tablodaki değerleri sürekli olarak günceller, daha iyi stratejiler geliştirir ve optimal eylemler için tahminler sağlar. Epsilon oranı yüksek iken, robot çevreyi keşfeder ve eylemleri rastgele seçer; bu, robotun çevre hakkında hiçbir önceden bilgiye sahip olmadığı anlamına gelir. Ancak, epsilon oranı azaldıkça, robot çevreyi daha iyi kullanmaya başlar. Her aksiyon alındığında, sonucu ve ödülü gözlemleyerek Q fonksiyonunu günceller.

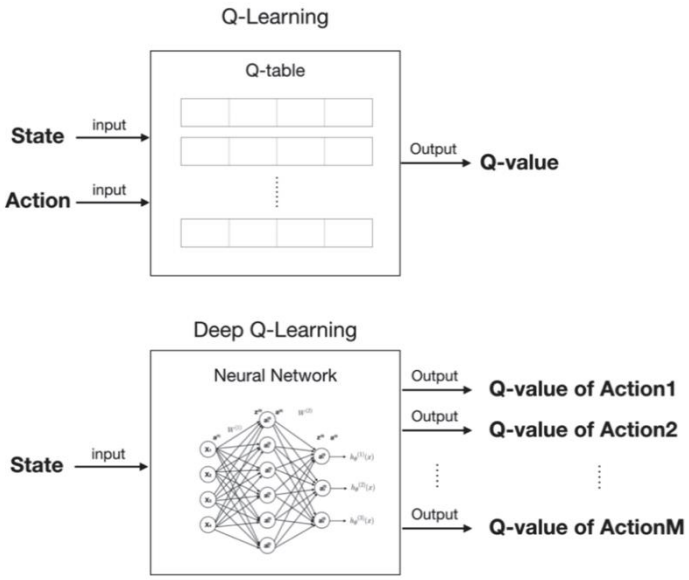
$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots [s_t, a_t] \tag{3.31}$$

3.15. Derin Q-Öğrenmesi (DQN)

Q-Learning algoritması, Q (s, a) fonksiyonunun tablo ile temsil edilebilir olduğu daha basit ortamlarda kullanılır ancak çok daha büyük tabloları kullanmayı gerektiren durum ve eylem içeren ortamlar da kullanımı verimli olmamaktadır. Bu durumlar için kullanılan Deep Q Network (DQN) algoritması, derin sinir ağlarının ilkelerini Q-öğrenmeyle birleştirerek karmaşık ortamlarda en uygun politikaların öğrenilmesini sağlar. Derin sinir ağları Q fonksiyonuna yaklaşmak için kullanılması ile Q değerlerinin temsili için tablo kullanılmasını gerektirmez [35].

DQN, sadece durum bilgisini alarak ve her aksiyon için Q-değerlerini hesaplayarak optimal politikaya yakınsamaya çalışır.

En yüksek Q-değerine sahip olan aksiyon seçilir. Bu yöntemde, modelin ağırlık parametreleri güncellenerek öğrenme gerçekleştirilir.



Şekil 3.10. Q-öğrenmesi ve derin Q-öğrenmesi yapıları [36].

Deep Q Network (DQN) algoritması, öğrenme sürecini dengelemek için ana sinir ağı ve hedef sinir ağı olmak üzere iki DNN kullanır. Ana sinir ağı, θ ağırlık vektörü ile temsil edilir ve mevcut durum s ve eylem a olmak üzere Q (s, a; θ) değerlerini tahmin etmek için kullanılır. İkincisi, hedef sinir ağıdır ve aynı mimariye sahip olmasına rağmen, gelecekteki tahminler için kullanılır. Öğrenme süreci ana ağda gerçekleşir ve hedef ağın ağırlıkları periyodik olarak kopyalanarak bilgi aktarımı sağlanır. Bu, hedef ağın tahminlerinin daha doğru olmasını sağlar.

Sinir ağını eğitmek için DQN algoritmasında bellman denkleminin iki tarafı arasındaki kare farkı olarak tanımlanan bir kayıp (veya maliyet) fonksiyonuna gerekir.

$$Q(s,a;\theta) = r + \gamma \max_{a'} Q(s',a';\theta') \tag{3.32}$$

DQN, deneyim tekrarı ve hedef ağ yapısını içermektedir. Bu yapılardan deneyim tekrarı ajanın aksiyonlarını, durumlarını ödüllerini ve yeni durumları ile ilgili bilgilerin hafızaya alınmasıdır. Sinir ağı modelinin eğitimi sırasında bu bilgiler kullanılır [35].

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta))^2] \tag{3.33}$$

Algoritmanın hedef değerle tahmin değerini aynı sinir ağında hesaplaması, istikrar sorunlarına yol açabilir. Bu sorunu çözmek için hedef ağ algoritmada kullanılır. Ajanın çevreden aldığı durum bilgisine göre tahmin ağı tarafından Q değeri üretilir. Optimum

Q değerindeki aksiyon gerçekleştirildikten sonra çevreden ödül ve bir sonraki durum bilgisi alınır ve bu deneyim hafızasına kaydedilir. Tahmin Ağı belirtilen kayıp fonksiyonuna göre kendi ağırlık parametrelerini günceller, hedef ağı ise belirli periyotlarla güncellenir [35].

3.16. Çevre Modelleri ile Öğrenme

Model-tabansız algoritmalarının iyi bir politika yaklaşımı sağlamak için tüm durumlarda çok fazla deneyim gerektirmektedir. Model-tabanlı algoritmalar, bu sorununu aşabilmek amacıyla ajanın durumları daha fazla deneyimlemesine olanak tanıyarak politikaların daha hızlı yaklaşmasını sağlar [27]. Model-tabanlı algoritmalar, geçiş modeli (T) ve ödül modeli (R) kullanarak optimal politikayı elde etmeye çalışır, geçiş modeli durumlar arasındaki geçişleri, ödül modeli ise her durum ve eylem kombinasyonuna ilişkin ödülleri temsil eder [27]. Ajan başlangıçta bilgisizdir ve modelleri, gerçek ve simülasyon deneyimleriyle günceller. Model-tabanlı algoritmalar, kaynakları eksik olmayan ajanlar için ve hesaplama maliyeti gerçek dünya deneyimlerine göre düşük olduğunda uygun bir seçenektir [27].

3.17. Politika-Gradyan Yöntemleri

Toplam ödülü maksimuma çıkaran eylem seçimini belirleyen optimal politika, eğitim ile sağlanabilmektedir. Optimal politika için doğrudan ve dolaylı olmak üzere iki yaklaşım vardır [37]. Politika tabanlı yöntemler ajanın seçmesi gereken eylemi doğrudan öğretmektedir. Değer tabanlı yöntemler ise değerli olan durumu ajana öğreten, ajanın eylemlerini daha değerli seçimler üzerine yönlendirmesini sağlayan dolaylı yöntemlerdir. Politika tabanlı yöntemlerden biri olan politika gradyan yöntemleri, optimum prensibin ağırlık tahminini gradyan artışı ile gerçekleştirmektedir [38].

Politika gradyanı yöntemlerinde, politika ağı ağrılıkları yenilenerek optimum politikaya ulaşana kadar ödülü arttıran eylemler sıklaştırılır.

Şekil 3.11. Değer tabanlı yöntemler ve politika tabanlı yöntemler

3.18. Reinforce Algoritması

Reinforce algoritması pekiştirmeli öğrenme politika gradyan yöntemlerinden biridir. Bu algoritma için ortamın modeli gerekli değildir bu yönü ile ortamın modellenmesinin zor olduğu ya da bilinmediği problemlerde kullanıma uygundur. Bu algoritma artan ödüller doğrultusunda politika parametrelerini optimize etmektedir.

$$\Delta_{wij} = a_{ij}(r - b_{ij})e_{ij} \tag{3.34}$$

Denklem 3.34 de ağın ağırlık değeri wij, öğrenme faktörü aij (t ye bağlı ve negatif değildir) baseline (pekiştirme taban çizgisi) bij, karakteristik uygunluk eij ile ifade edilmektedir. Durum, aksiyon ve ödül den oluşan yörünge, bir durumda alınan eylemlerin sırasını belirtmektedir.

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_2, \dots, a_H, r_{H+1}, s_{H+1}) \tag{3.35}$$

$$R(\tau) = (G_0, G_1, \dots G_H) \tag{3.36}$$

Denklem 3.35 de τ yörüngeyi, H uzunluğu ifade etmektedir. Yörüngelerdeki ödülün en yüksek değerde olması için politikanın belirlenmesi reinforce yöntemi ile sağlanmaktadır [38]. Denklem 3.36 da τ yörünge ödülü R(τ) ifadesi verilmiştir. $G(k)$, yörünge boyunca beklenen toplam ödüldür ve denklem 3.36 da verilmiştir.

$$G_k \leftarrow \sum_{t=k+1}^{H+1} \gamma^{t-k-1} R_k \tag{3.37}$$

Beklenen getiriyi en iyi duruma ulaştıran θ ağırlıkları $U(\theta)$ fonksiyonu ile gösterilir ve denklem 3.36 da verilmiştir. Gradyan yükselişi, $U(\theta)$ fonksiyonunu belirlemek için kullanılan bir yöntemdir [38]. Denklem 3.38 de gradyan yükselişinde kullanılan yenileme adımı verilmiştir:

$$\cup(\theta) = \sum_T P(\tau, \theta) R(\tau) \tag{3.38}$$

Denklemde yer alan α zamanla azalan adım boyutudur. $\nabla\theta$ gradyanın değerini hesaplamak yerine yörüngeleri örnekleyerek bunlar üzerinden gradyan tahmini yapmak daha uygun olmaktadır [38]. Yörünge örnekleme Monte carlo yöntemi ile gerçekleştirildiği için Reinforce yöntemine, Monte carlo politika gradyanları da denilmektedir. Şekil 3.12 de yöntemin pseudo koduna yer verilmiştir.

Input:a differentiable policy parameterization $\pi_\theta(a_t|s_t, \theta)$

Algorithm parameter: step size $a > 0$

Initiliaze the policy parameter θ at random

(1) Use the policy π_θ to collect a trajectory $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, s_{2,\dots} \alpha_H, r_{H+1}, s_{H+1})$

(2) Estimate the Return for trajectory τ: $R(\tau) = (G_0, G_1, \dots G_H)$

where G_k is the expected return for transition k:

$$G_k \leftarrow \sum_{t=k+1}^{H+1} \gamma^{t-k-1} R_k$$

(3) Use the trajectory τ to estimate the gradient $\nabla_\theta\, U(\theta)$

$$\nabla_\theta\, U(\theta) \leftarrow \sum_{t=0}^{H} \nabla_\theta log \pi_\theta(a_t|s_t) G_t$$

(4) Update the weights θ of the policy

$$\theta \leftarrow \theta + \alpha \nabla_\theta\, U(\theta)$$

(5) Loop over steps 1-5 until not converged

Şekil 3.12. Reinforce algoritması pseudo kodu [38].

$$\pi_\theta(a_t|s_t) \tag{3.39}$$

Denklem 3.39 da durum s_t de aksiyon a_t 'in belirli bir politika altında alınma olasılığını göstermektedir.

$$\nabla_\theta \, log\pi_\theta(a_t|S_t) \tag{3.40}$$

Denklem 3. 40 da verilen ifade, belirli bir durum s_t'te belirli bir aksiyon a_t'in log olasılığının θ parametresine göre gradyanını ifade eder. Bu ifade pekiştirmeli öğrenme algoritmalarında politika parametrelerinin güncellenmesinde kullanılır, politikanın performansını iyileştirmek için gradyan iniş yöntemi kullanılır ve bu gradyan politikanın belirli bir durumda belirli bir aksiyonu seçme olasılığını artırmaya veya azaltmaya yönelik bir kılavuz oluşturur [38].

3.19. Baseline (Taban Çizgisi)

Taban çizgisi, pekiştirmeli öğrenme algoritmasında kullanılan bir referans noktasıdır. Ödül tahminlerinin düzeltilmesine ve daha doğru tahminler elde edilmesine yardımcı olabilmektedir. Algoritma tahmin edilen değerlerin bu taban çizgisinden ne kadar farklı olduğunu dikkate alarak değerlerini güncellemektedir. Genellikle, bir taban çizgisi kullanarak, ajanın tahmin etmesi gereken değerlerin bir kısmını daha doğru bir şekilde tahmin edebilir ve bu da öğrenme sürecini hızlandırabilir veya daha istikrarlı hale getirebilmektedir [39,40].

3.20. Monte Carlo Yöntemi

Monte carlo yöntemi, ajanın davranışını optimize etmek için değerlidir. Pekiştirmeli öğrenmede politika değerlerinin tahmini ve politika iyileştirmesi olmak üzere iki ana kullanım alanına sahiptir. Politika değerlerinin tahmininde, ajanın belirli bir politikayı izleyerek aldığı dönüşümlerin ortalamasını hesaplayarak politika değerlerini tahmin etmek için kullanılabilir. Ajan, çevresiyle etkileşim halindeyken çeşitli durumlara maruz kalır ve bu durumlarla ilgili dönüşler (ödüller) toplanır. Monte carlo yöntemi, bu dönüşlerin ortalamasını alarak her bir durumun politika değerlerini tahmin eder. Politika değerlerinin tahmin edilmesinden sonra ajanın davranışını iyileştirmek için politikasını güncellemek mümkündür. Politika iyileştirme, politikadaki hataları düzelterek veya daha iyi performans elde etmek için politikayı güncelleyerek gerçekleştirilir [41].

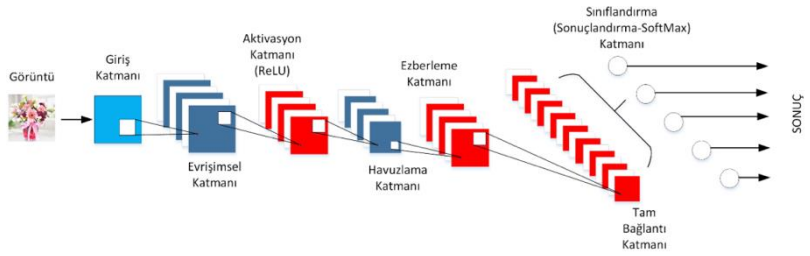
4. DERİN ÖĞRENME

Programlanabilir bilgisayarların icadıyla makinelerin düşünme yeteneği fikri 1940'larda ciddi şekilde araştırılmaya başlanmıştır. Makine öğrenimi, bilgisayarın verilerden öğrenme ve tahmin yapma yeteneğini sağlayan algoritmaları içermektedir [42]. 1980'lerin sonlarında geri yayılım algoritmasının icadı ile yapay sinir ağlarında kullanılması, modelin birçok eğitim verisinden desenleri öğrenmesi ve desenlere dayalı tahminler yapmasını sağlamıştır. 1990'larda, lojistik regresyon, destek vektör makineleri ve boosting gibi çeşitli sığ makine öğrenimi modelleri ortaya çıkmıştır [43]. İnternet'in gelişimi ile 2000'lerde, büyük veri setlerinin tahmin ve akıllı analizinde yapay sinir ağlarınının kullanımı popüler hale gelmiştir [42].

Derin öğrenme, çok katmanlı işlem katmanlarından oluşan hesaplama modelleriyle verilerin çok katmanlı soyutlamalarını öğrenme olanağı sağlar. Bu yöntemler, konuşma tanıma, görsel nesne tanıma ve nesne tespiti gibi çeşitli alanlardaki teknolojik gelişmeleri büyük ölçüde arttırmıştır [44]. 2006'da Geoffrey Hinton , daha fazla gizli katmana sahip yapay sinir ağlarının daha iyi öğrenme yeteneğine sahip olduğunu ve derin sinir ağlarının eğitim zorluğunun katman katman önceden eğitim ile aşılabileceğini belirttirmiştir. İnsan sinir hücresinin yapısından esinlenen yapay sinir ağları, geniş bir alanda kullanılmakta olup, 2011'den itibaren Google ve Microsoft gibi büyük şirketler derin sinir ağlarını kullanarak özellikle konuşma ve görüntü tanıma alanlarında önemli başarılar elde etmişlerdir [42].

Derin öğrenme, yapay sinir ağlarının evrim geçirerek günümüzdeki son haline gelmesidir. Gelişim süreci, Perceptron modeliyle başlayıp Tek Katmanlı Yapay Sinir Ağı ve Çok Katmanlı Yapay Sinir Ağlarına doğru ilerlemiştir [45]. Derin öğrenme, sığ makine öğreniminden farklı olarak katmanlar halinde olmayan işlem birimlerinin kaskadını kullanarak verilerden hiyerarşik özellikler çıkarır ve daha yüksek seviye özellikler daha düşük seviye özelliklerden türetilir [42].

Veri artışıyla birlikte, anlamlı bilgilerin çıkarılması zorlaştığı için esnek hesaplama yöntemlerinin, özellikle yapay sinir ağları gibi, önemi giderek artmaktadır. Derin öğrenme, çok katmanlı bir ağ yapısı kullanarak, doğrusal olmayan pek çok problemi çözmek için esnek bir hesaplama yöntemi olarak kullanılır. Bu algoritmalar, girdi katmanından başlayarak sınıflandırma işlemlerini gerçekleştiren son katmana kadar çeşitli katmanlardan oluşur ve veri girişi bu katmanlardan biri olan girdi katmanından yapılır ve ardından normalizasyon işlemi gerçekleştirilir [45].



Şekil 4.1. Derin öğrenme katman yapısı [42].

4.1. Derin Öğrenme Katmanları

4.1.1. Giriş (input) katmanı

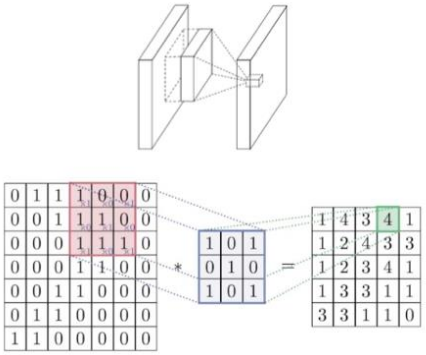
Giriş (Input) katmanı; Input katmanı ham veriyi alır ve giriş görüntü boyutunun belirlenmesinde kritik bir rol oynar. Yüksek boyutlar daha fazla bellek ve eğitim süresi gerektirebilirken, düşük boyutlar ağın derinliğini ve performansını olumsuz etkileyebilir. Bu nedenle, uygun bir giriş boyutu seçerken, ağın derinliği, donanımsal hesaplama maliyeti ve başarı dengelenmelidir [46].

4.1.2. Konvolüsyon (convolution) katmanı

Dönüşüm katmanı, bir filtrenin tüm görüntü üzerinde hareket ettirilmesine dayanır. Bu filtreler, bir önceki katmandan gelen görüntülere konvolüsyon işlemi uygulayarak aktivasyon haritasını oluşturur, bu da her filtrenin özelliklerin keşfedildiği bölgeleri gösterir. Filtrasyon süreci boyunca, filtre katsayıları her öğrenme döngüsünde değişir, böylece ağ, verinin önemli bölgelerini belirlemek için filtrelerin ağırlıklarını ayarlar. Bu işlem sırasında, giriş ve çıkış boyutları arasında aynı yoğunluk aralığını sağlamak için aktivasyon haritasındaki değerler normalize edilir [46].

Derin öğrenme algoritmalarında, konvolüsyon işlemi için farklı boyutlarda filtreler kullanılmıştır. Örneğin, AlexNet 11x11, ZfNet 7x7 filtreler kullanırken, GoogleNet,

VggNet, ResNet gibi mimarilerde ise 5x5, 3x3, 2x2, 1x1 filtrelemeler görülmüştür. Bu filtreler, giriş verisine uygulanarak özellik haritalarının oluşturulmasını sağlar [47].



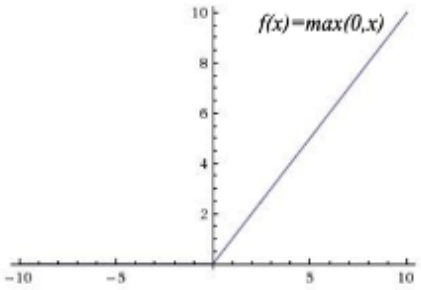
Şekil 4.2. Konvolüsyon işlemi [46].

4.1.3. Aktivasyon (relu) katmanı

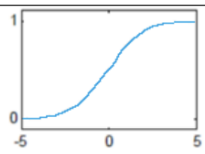
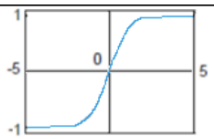
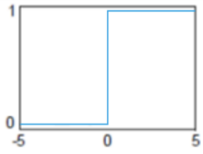
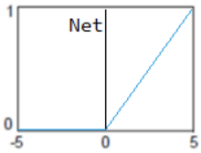
Relu katmanı, genellikle konvolusyon katmanının ardından kullanılan aktifleştirme katmanıdır.

$$f(x) = \begin{cases} 0 & eğer\ \ x < 0 \\ x & eğer\ \ x \geq 0 \end{cases} \tag{4.1}$$

Hiperbolik tanjant, step, sigmoid, sinüs, step, eşik değer fonksiyonları kullanılmaktadır. Konvolusyon katmanında doğrusal yapıda olan ağ, bu katmanda doğrusal olmayan bir yapıya dönüştürülür. Bu katmanın kullanılması ile değerler negatif ise sıfıra çekilir. Denklem 4.1 de matematiksel ifadesi verilmiştir [46].



Şekil 4.3. Relu aktivasyon fonksiyonu [46].

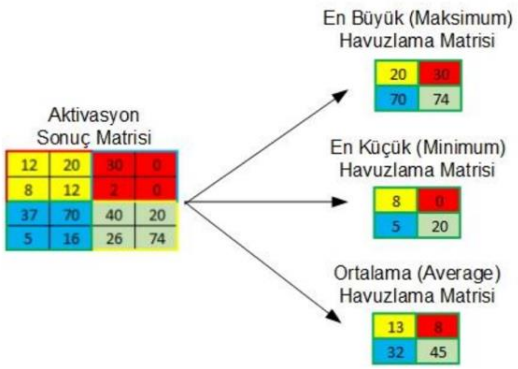
Sigmoid	$y = \frac{1}{1 + e^{-Net}}$	
Tanjant Hiperbolik	$y = \frac{e^{Net} - e^{-Net}}{e^{Net} + e^{-Net}}$	
Adım Fonksiyonu	$y = \begin{cases} 1 & Net \geq 0 \\ 0 & Net < 0 \end{cases}$	
ReLU	$y = \max(0, Net)$	

Şekil 4.4. Aktivasyon fonksiyonu [45].

4.1.4. Havuzlama (pooling) katmanı

Relu katmanından sonra yer alan havuzlama katmanı, bir sonraki katman için verideki derinlik boyutunu etkilemeyecek şekilde giriş boyutunu azaltma işlemi yapmaktadır. Bu boyut azaltma işlemi sonucu oluşan bilgi kaybı ile hesaplama miktarında azalma sağlanır ve ezberlemenin önüne geçilir.

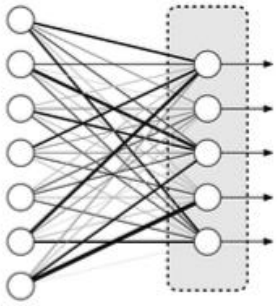
Havuzlama katmanında NxN boyutundaki filtre, görüntü üzerinde gezdirilerek piksellerin ortalaması (average pooling) veya max değerleri alınarak (maximum pooling) işlemi gerçekleştirilir [46].



Şekil 4.5. Havuzlama katmanı [46].

4.1.5. Tam bağlı (full-connected) katmanı

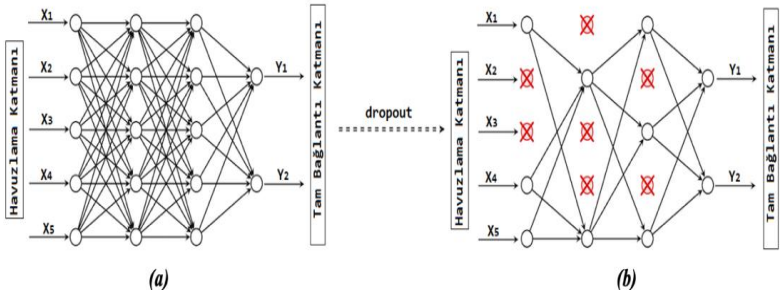
Tam bağlı katman da kendinden önce gelen katmandaki tüm bağlantılar birleşir, her nöron bir sonraki ile bağlanır ve veriler tek boyutlu matris halinde sonraki katmana aktarılır [47].



Şekil 4.6. Tam bağlı katmanı [46].

4.1.6. DropOut katmanı

Hinton ve arkadaşları tarafından önerilmiş olan DropOut katmanı [48,49], eğitimde az sayıda veri kullanılmasından kaynaklı ağın aşırı öğrenme yaparak eğitim setini ezberlemesi durumunun önüne geçmek için kullanılır, bazı düğümler ortadan kaldırılarak bu işlem gerçekleştirilir. DropOut katmanı ile ağın performansı artırılmış olur [50].



Şekil 4.7. (a)Ezberleme yapmış (b)Ezberleme katmanı kullanılmış [47].

4.1.7. Sınıflandırma (classification) katmanı

Sınıflandırma katmanında, tam bağlı katmandan iletilen verilerin değerlendirilmesi ve sınıflandırılması yapılır. Katmanın çıkışında sınıflandırılması yapılacak nesne sayısınca sonuç oluşur. Oluşan her bir sonuç bir sınıftır. Sınıflandırma katmanı çıkış değeri, sınıflandırılması yapılacak farklı nesne sayısında olur. Derin öğrenme uygulamalarında genellikle SoftMax sınıflandırıcısı kullanılır.

Bu sınıflandırıcı, olasılıksal hesaplamalar yaparak 0-1 aralığında değerler üretir ve en yüksek değere sahip olan sınıfı tahmin eder. Ancak bazı durumlarda, öznitelikler farklı bir sınıflandırma algoritmasına yönlendirilerek alternatif sınıf tahmin bilgisi elde edilebilir [45].

4.1.8. Yumuşatma (softmax) katmanı

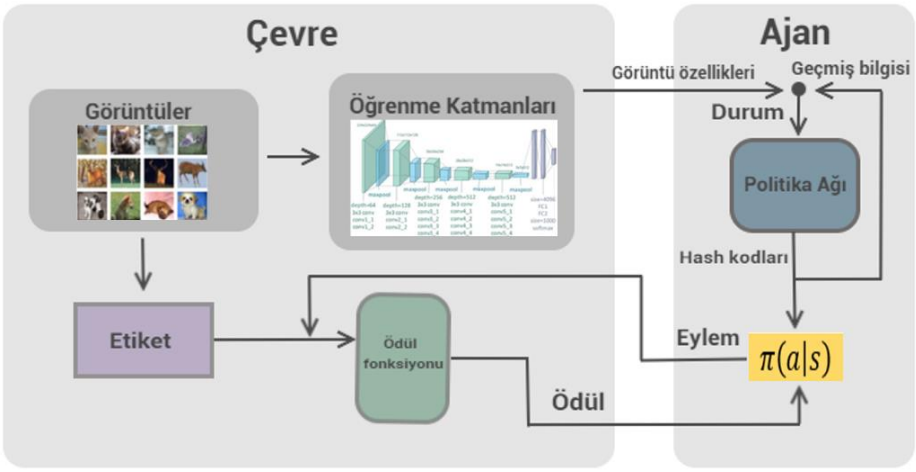
SoftMax katmanı, tam bağlı katmandan gelen veriyi alarak sınıflandırma işlemi yapar ve belirli bir sınıfa ait olma olasılığını belirler. Bu katman, her sınıf için olasılık değerlerini hesaplamak için çapraz entropi kullanır. Bu sayede, girdi verisini hangi sınıfa daha yakın olduğuna dair bir değer üretir [51].

4.1.9. Normalizasyon (normalization) katmanı

Derin konvolüsyonel sinir ağlarının eğitimi genellikle uzun bir süreç alır ve aktivasyonları normalize etmek, eğitim süresini azaltmanın bir yolu olarak kullanılır. Normalizasyon katmanları, gizli katmanlardaki durumları stabilize ederek geri beslemeli ağların performansını artırır. Normalizasyon Relu katmanı sonrasında gerçekleştirilir ve bu işlem, ağın performansını etkiler. Girdi verilerinin normalize edilmesi, ağın verimli çalışması için önemlidir, çünkü fazla ya da yetersiz değerler eğitim ve süreç açısından sorunlara yol açabilir. Bu nedenle, girdi verilerinin belirli bir aralıkta normalize edilmesi ve temsil edilmesi sağlanmalıdır [52,53].

5. PEKİŞTİRMELİ HASH ÖĞRENME MODELİ

Derin pekiştirmeli öğrenme alanındaki ilerlemelerle birlikte hash kodu üretiminin ardışık bir görev olarak ele alınabilmesi, hashing problemini derin pekiştirmeli öğrenme ile modellemeyi mümkün hale getirmiştir. Hashing, büyük boyuttaki veri kümelerini daha küçük boyutlu kodla temsil etme tekniğidir. Benzer öğelerin aynı hash kodlarına sahip olması beklenir bu nedenle hashing de benzer örnekler arasındaki ilişkileri öğrenmek önemlidir. Geleneksel hash yöntemlerinden farklı olarak hash işlemini ardışık karar alma sureci olarak ele alan yaklaşımımızdaki ana bileşenlerden biri derin öğrenme ağıdır. Bu ağ görsellerin özelliklerini çıkarmak için bir özellik temsili ağından ve görselleri ikili kodlara dönüştürmek için bir politika ağından oluşur.

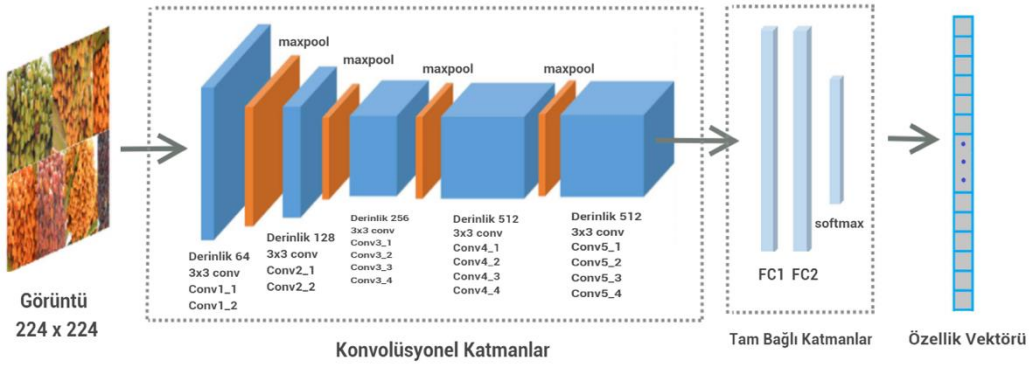


Şekil 5.1. Önerilen derin pekiştimeli öğrenme hash kodlama ağının genel yapısı.

Şekil 5.1 de verildiği gibi model ilk adım olarak görüntülerin özelliklerini çıkartmak için bir dizi evrişimli ve tam bağlı katmandan oluşan bir ağı kullanır. Kullanılan VGG-19 derin öğrenme ağ modelinde, genellikle son katmanlar olan tam bağlı (fully-connected) katmanlar çıkarılarak özellik çıkartma işlemi yapılır.

Bu tam bağlı katmanlar, evrişimli katmanlardan gelen özellikleri alır ve sınıflandırma yapmak için kullanılır. Özellik çıkartma amacıyla kullanım için, bu tam bağlı katmanlar çıkarılarak evrişimli katmanlardan çıkan özellikler elde edilir. Katmanlar görüntülerdeki desenleri algılamak ve önemli özellikleri çıkarmak için kullanılırlar. İlk aşamada görüntüyü boyutlandırma veya normalleştirme gibi ön işlemeler

yapılabilir. Şekil 5.2 de gösterildiği gibi görüntü, ağın evrişim katmanlarından geçirilir. Her bir katman, görüntünün farklı özelliklerini çıkarır. Daha derin katmanlar daha soyut özellikler (yüksek seviyeli desenler, nesne parçaları) çıkarır. Evrişim katmanlarının çıktıları, tam bağlantılı katmanlara beslenir, görüntü özelliklerini çıkarma işlemi için son aşama tam bağlantılı katmanlardır. Bu işlemler her görüntüyü bir dizi özellik vektörüne dönüştürür. Özellik vektörleri modelin daha spesifik görevlerde daha etkili olmasını sağlar.

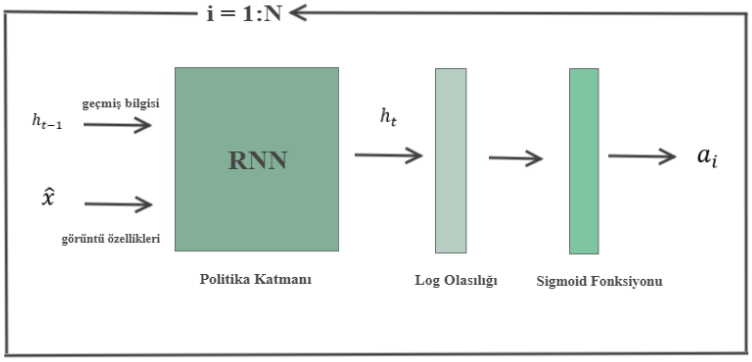


Şekil 5.2. VGG-19 ağı ile görüntü özelliği çıkarma mimarisi

Çıkan özellik vektörleri politika ağını besler. Politika ağı, RNN (tekrarlayan sinir ağı) ile birlikte Actor-critic yöntemini içermektedir ve bir ajan gibi hareket etmektedir. Her bir görsel için RNN ağı sırasıyla her bir pikseli işleyerek görseli bir dizi ikili koda dönüştürmek için karar alır. Bu süreç, görselin hash kodunu üretmek için bir dizi ardışık kararların birleşimidir. RNN, politika ağının hafızası olarak işlev görür ve karar verme sürecine geçmiş deneyimlerin entegrasyonunu sağlar.

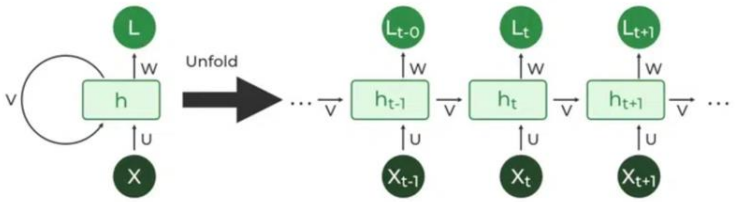
Actor-critic yöntemi ise politika ağının öğrenme mekanizmasını temsil eder. Bu yöntem, doğrusal bir model kullanarak eylem tahminleri yapar. Yani, bir durumda hangi eylemin alınması gerektiğini doğrudan tahmin eder. Bu tahminler, doğrusal bir model kullanılarak elde edilir. RNN, geçmiş deneyimleri hatırlayarak eylem tahminlerinde bulunurken, Actor-critic bileşeni bu tahminlerin doğruluğunu değerlendirir ve politika güncellemeleri yapar. Şekil 5.3 de verilen modelde önceki zaman adımında h_{t-1} elde edilen geçmiş bilgileri, $\hat{x}$ ise görüntüyü temsil eden görüntü vektörünü ifade eder.RNN önceki zaman adımlarından gelen bilgiyi dikkate alarak ve görüntü özelliklerini işleyerek bir çıktı h_t üretir. Bu çıktı log olasılık değerine girer, daha sonra sigmoid fonksiyonu ile çıktı [0,1] aralığına sıkıştırılarak olasılık değerlerini

temsil etmek için kullanılır. Bu şekilde politika ağı, geçmiş deneyimleri hatırlayabilen bir yapıyla birlikte doğrudan eylem tahminleri yapabilen bir yapıyı birleştirir. Bu modelin esnekliğini artırır ve çeşitli hash problemlerine daha iyi adapte olmasını sağlar.



Şekil 5.3. Bit çıkarım modeli.

Politika ağı, önceki kararların bir geçmişini korur ve verilen durum için (görüntü özellikleri ve geçmiş) bir eylem (hash kodu) üretir. Hash kodlarının geçmişi ise bir sonraki aşamada kullanılmak üzere saklanır. Bu, modelin önceki kararları hatırlayabilmesini sağlar.

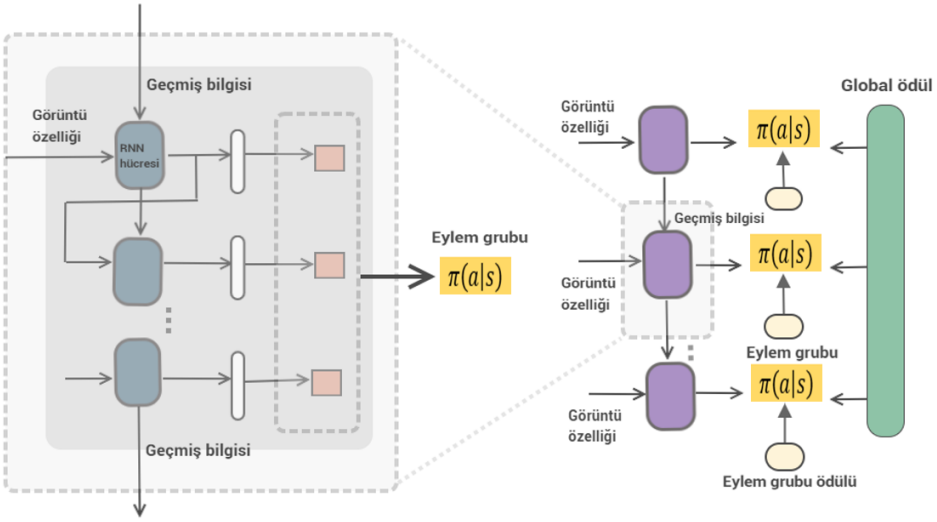


Şekil 5.4. RNN mimarisi [48].

Yaklaşımın ana bileşenlerinden bir diğeri ise önceki kararların sonuçlarına dayalı olarak gelecekteki kararları şekillendirmek için kullanılan ödül sistemidir. Üretilen hash kodları ve gerçek etiketler arasındaki uyumu değerlendirmek için bir ödül fonksiyonu kullanılır. Politika ağının çıktısı olan eylemler ve görüntü etiketleri bu ödül fonksiyonuna girerek bir ödül (reward) elde edilir. Bu fonksiyon modelin ürettiği hash kodlarının kalitesini ölçer. Politika ağının verilen bir durum ve eylem kombinasyonunu ne kadar iyi bulduğunu veya ne kadar doğru bir şekilde tahmin

ettiğini yansıtır. Bu RNN ağının içsel durumu ve görsel özelliklerini içeren bir dizi durumu(state) kullanarak gerçekleştirilir. Stateler ajanın önceki kararları ve bu kararların sonuçlarının hatırlanmasına olanak tanır.

Böylece sadece bir görsel işlemekle kalmayıp aynı zamanda bir grup görseli işleyerek ikili kodların tutarlılığının artmasına olanak tanıyan bir eylem grubu (action) kavramı oluşur. Ödül, politika ağına geri beslenir. Politika ağı, geriye doğru(backpropgation) algoritması kullanılarak güncellenir. Bu politika ağını verilen bir durumda daha iyi kararlar alacak sekilde güncellemesine olanak tanır. Güncelleme süreci, politika ağının öğrenmesini ve daha iyi kararlar almasını sağlar. Bu adımlar, ajanın etkileşime girdiği ortamdan aldığı geri bildirimi kullanarak politika ağını güncellediği bir döngü oluşturur.



Şekil 5.5. Politika ağının detayları.

Reinforce algoritması, politika ağının parametrelerini doğrudan optimize ederek politika ağının performansını artırmayı hedefler.Bu süreç ajanın belirli bir görevi daha iyi öğrenmesini ve daha iyi performans göstermesini sağlar. Reinforce algoritması politika gradient yöntemlerinden biridir, bu algoritmada kullanılan triplet loss yaklaşımı, hashingde benzer örneklerin aynı veya benzer hash kodlarına sahip olmasını sağlamak için kullanılabilir.

İki benzer örnek arasındaki hash kodları arasındaki farkın negatif örnekler arasındaki farktan daha büyük olması beklenir. Bu şekilde, model benzer örnekleri birbirinden ayırt etmeyi öğrenir. Bu, derin pekiştirmeli öğrenme ortamında kullanılarak ağın

eğitimi sırasında bir ödül olarak kullanılabilir. Temel fikir, ajanın aldığı ödüllerin beklentisini maksimize eden politika parametrelerini bulmaktır. Politika ağının belirli bir durumda verilen bir eylemi seçmesiyle bir politika oluşturulur. Bu politika, belirli bir başlangıç durumundan başlayarak belirli bir hedefe kadar olan durumlar ve alınan eylemleri içerir. Politika oluşturulurken, ajanın politika ağından aldığı örnekler kullanılır. Politika boyunca toplanan ödülllerin toplamı hesaplanır ve politikanın ne kadar iyi veya kötü olduğu belirlenir. Toplanan ödüllerin toplamını maksimize etmek için politika parametrelerinin gradyanı hesaplanır. Bu, politika ağının parametrelerini güncellemek için kullanılacak gradyanı belirler. Gradyanı belirlemek için kullanılan bir diğer algoritma ise stokastik gradyan inişi (SGD) algoritmasıdır. SGD'nin önemli avantajlarından biri, büyük veri setleri üzerinde etkili bir şekilde çalışabilmesidir. Bunun nedeni, her bir iterasyonda sadece tek bir örneğin kullanılmasıdır, bu da bellek kullanımını azaltır ve eğitim sürecini hızlandırır. SDG ile hesaplanan gradyan öğrenme oranı (learning rate) parametresi ile çarpılarak mevcut parametrelere eklenir veya çıkarılır ve politika parametreleri güncellenir yani politika parametrelerinin hatayı en aza indirecek veya hedef fonksiyonu optimize edecek şekilde güncellenmesi sağlanır böylece politika ağının gelecekte daha iyi ödüller elde etmek için kendisini optimize etmesini sağlar. Politika ağı, daha yüksek ödül alınmasını sağlayan eylemleri öğrenir ve verilen bir görüntü için en uygun hash kodlarını üretecek şekilde adapte olmuş olur. Bu süreç, bir döngü içinde tekrarlanır. Her iterasyonda model, görüntülerin özelliklerini öğrenir, politika ağı kullanarak hash kodları üretir, ödül fonksiyonuyla değerlendirilir ve politika ağı güncellenir. Bu şekilde model, verilen bir görüntü için en uygun hash kodlarını üretebilmek için sürekli olarak kendini geliştirir ve daha iyi hash sonuçları sağlanır.

5.1. Notasyonlar

Hash yöntemleri, D boyutunda n adet görsel $X \in \mathbb{R}^D$ kümesindeki görsellerin anlamsal benzerliğini korurken her bir görseli Hamming uzayında q boyutlu ikili kod H(x) olarak kodlayan hashing fonksiyonunu elde etmeye çalışmaktadır. Birçok derin hashing yöntemi farklı hash fonksiyonları arasındaki korelasyonu göz ardı ederek doğrudan öğrenirken önerilen yöntem ile ardışık öğrenme amaçlanmaktadır.

5.1.1. Pekiştirmeli hash öğrenmenin tanımı

Durum; önceki hash kodlarının geçmiş bilgisini yansıtmaktadır ve (h, i) olarak tanımlanır, h oluşturulmuş hash kodlarının geçmiş eylem vektörünü, i ise görüntü özellik vektörünü ifade etmektedir. Geçmiş eylem vektörleri politika ağında oluşturulmaktadır. Görüntü özellik vektörü eğitilmiş CNN modeli aracılığıyla orijinal görüntülerden elde edilmektedir.

Ajan mevcut durum $s = (h, i)$ için eylemlerin olasılığını tahmin eder. Hash kodlamada sadece iki olası eylem (1 ve 0) vardır ve eylem olasılıklarının toplamı 1 dir. Bu yöntemde eylem olasılık dağılımı tahminini değil hash kod 1 in olasılığının politika ağ çıktısı kullanılmaktadır. Genel olasılık dağılımı ifadesi denklem 5.1 de verilmiştir.

$$P(a|s,\theta) = \begin{cases} 1 - policy(s,0) & a = 0 \\ policy(s,0) & a = 1 \end{cases} \tag{5.1}$$

Policy (s, θ), giriş durumu s, parametreleri θ olan politika ağının çıktısını ifade etmektedir. Geçmiş sıralama hatalarını bir bitlik hash kod ile düzeltmek mümkün olmadığı için k bitişik hash işlevinden oluşan eylem grubu kullanılmaktadır. Grubun her bir eylemi, sıralama hatalarını düzeltme yeteneğini artırmak için tasarlanmış aynı ödülü paylaşmaktadır. Eylem olasılığı $\pi_\theta(s_i, A_i)$ şu şekilde formüle edilir:

$$A_i = \left[a_{t_i+1}, a_{t_i+2}, .. a_{t_i+k}\right] \tag{5.2}$$

$$\pi_\theta(s_i, A_i) = \prod_{J=1}^{k} P\left(a_{t_i+j} \middle| \hat{s}_{i,j}, \theta\right) \tag{5.3}$$

Burada A_i i. eylem grubudur, $a_{t_i+1..k}$ eylem grubunun elemanıdır, $\hat{s}_{i,j}$ ise a_{t_i+1}'in giriş durumudur. Ödül; Ödül fonksiyonu üretilen hash kodlarının kalitesini ölçmektedir. Eylemin görüntüleri hash kodu 1'e yansıtma olasılığının yüksek olması, olasılık değerinin 1'e daha yakın olması demektir. Dolasıyla, ödülü hesaplamak için görüntüleri hash kodu 1'e yansıtmak için bu olasılık sıralaması kullanılmaktadır. (X, Y) olarak belirtilen eğitim görüntüleri için, Y karşılık gelen etiketleri göstermektedir. x_i ve x_i^+ aynı etiketlere sahip iki benzer görüntü, x_i ve x_i^- farklı etiketlere sahip iki farklı görüntüyü temsil etmektedir, t örneklenmiş üçlü demet sayısı olmak üzere etiketlere dayalı bir dizi örnekleme, T = $\{(x_i, x_i^+, x_i^-)\}$ t_i=1 olarak ifade edilmektedir.

Üçlü demetler $(x_i,\ x_i^+,\ x_i^-),\ i = 1 \ldots t$ için, ödül fonksiyonu şu şekilde tanımlanmaktadır:

$$J\big(h(x_i), h(x_i^+), h(x_i^-)\big) = max(0, m_t + \|h(x_i) - h(x_i^+)\|^2 - \|h(x_i) - h(x_i^-)\|^2) \quad (5.4)$$

Denkleme göre farklı (x_i, x_i^-) çiftinin benzer (x_i, x_i^+) çiftinden en az m_t marjıyla daha büyük olması beklenmektedir. Üçlü sıralama kaybına dayalı ödül, anlamsal sıralama bilgisini yansıtabilmektedir bu da önceki üretilen hash kodlarının kalitesini iyi değerlendirmektedir. Üçlü sıralama kaybı anlamsal etiketlere dayanır ama yaklaşımımız gözetimli hashing yöntemlerinden farklıdır. Çünkü üçlü sıralama kaybı öğrenilen hash işlevlerinin mevcut ortamda ne kadar iyi performans gösterdiğini değerlendirmek için kullanılmaktadır, ardından bir sonraki hash işlevi ödülü hesaplanmış şekilde hashler üretmek için kararlar alabilmektedir, bu nedenle yaklaşımız pekiştirmeli öğrenme paradigmasına uymaktadır.

Doğru hash kodlarını bulup ajanı teşvik etmek için iki hiyerarşik ödül bulunmaktadır. İlk ödül, başlıca eylem grubu ödülüdür ve genellikle grup düzeyinde hash kod kalitesine odaklanmaktadır. İkinci ödül, global eylem ödülüdür ve tüm hash kodun kalitesine odaklanmaktadır. Hiyerarşik ödüller şu şekilde tanımlanmaktadır:

$$R_{i,j}^{g} = -J\left(h_j(x_i), h_j(x_i^+), h_j(x_i^-)\right) \quad (5.5)$$

Burada R, j. eylem grubundaki i. görüntünün grup eylem ödülünü, h_j j. eylem grubunun olasılık dizisini, $R_{i,j}^{G}$ ise i- inci görüntünün global eylem ödülünü ifade etmektedir.

$$R_{i,j}^{G} = -J\big(h(x_i), h(x_i^+), h(x_i^-)\big) \quad (5.6)$$

5.1.2. Derin pekiştirmeli öğrenme hash kodlama ağı

Önerilen derin pekiştirmeli öğrenme ağının genel çerçevesi Şekil 5.1 de gösterilmektedir ve iki bölümden oluşmaktadır. İlk bölüm, ajan için ödül ve durum sağlayan bir temsil ağı ve bir ödül fonksiyonunu içeren bir ortamı içerir. İkinci bölüm, ajan olarak hizmet eden bir politika ağıdır, bu ağ ortamdan durumu alıp hash kodları oluşturmaktadır.

Temsil ağı: Temsil ağı, bir özellik çıkarıcı olarak hizmet etmektedir, birkaç evrişim katmanı ve tam bağlı katman içeren derin evrişimli bir ağdır. Temsil ağı, durum demetindeki (h, i) görüntü özelliklerini sağlamaktadır. Temsil ağı olarak VGG-19 ağı benimsenmiştir. İlk 18 katman, VGG-19 ağında tamamen aynı ayarları takip etmektedir.

Politika ağı: Politika ağı, bir RNN katmanı ve bir politika katmanından oluşmaktadır. RNN katmanı, görüntü özelliklerini bir iç duruma dönüştürürken, politika katmanı iç durumu bir politika olasılığına eşlemektedir. RNN modelinin temel fikri, önceki adımlarda bilgi depolayan ve geçmiş eylemleri kullanan yerleşik bir bellek hücresi olmasıdır. Şekil 5.2, durum demetini *(h,i)* bir eylem grubu olasılığına eşleyen politika ağının detaylarını göstermektedir. Bellek hücresi, önceki hücre belleği birimi ve mevcut adımdaki giriş vektörü olmak üzere iki kaynaktan bilgiyi bir araya getirmektedir.

Bir durum demeti *(h,i)* için RNN katmanı, ağındaki birimlerin aktivasyonlarını aşağıdaki denklemi tekrarlayarak hesaplar ve girişi bir çıkış dizisine eşlemektedir:

$$c_t = tanh\left(\boldsymbol{W}_{x_i}\boldsymbol{x}_t + \boldsymbol{b}_{xi} + \boldsymbol{W}_{x_i}c_{t-1} + \boldsymbol{b}_{h_i}\right) \tag{5.7}$$

Burada $\boldsymbol{x}_t$ giriş, c_t gizli vektördür, t ise t inci adımı belirtmektedir, $\boldsymbol{W}_{x_i}$ ve $\boldsymbol{W}_{h_i}$ giriş $\boldsymbol{x}_t$ ve gizli vektörlerden yeni gizli duruma ağırlık matrisi, $\boldsymbol{b}_{xi}$ ve $\boldsymbol{b}_{h_i}$ bias(sapma) terimleridir. RNN katmanı c_0 =*h* ve x_0 = *i* ile başlatılmaktadır. Ancak sonraki adımda, bir eylem grubundaki geçmiş eylem bilgisine vurgu yapmak için $x_i = c_{i-1}$ olarak ayarlanır.

Görüntü özelliği yalnızca ilk adımın girişi olmasına rağmen, görüntü özelliğinin bilgisi, değişen geçmiş eylem bilgisiyle birlikte gizli durumda kalmaktadır. Bu nedenle, hash kodlar geçmiş bilgilerden etkilenmektedir ve önceki eylemlerden kaynaklanan hataları düzeltmek için ayarlanmaktadır. Son gizli durum c_k, bir sonraki adımın durumunu sentezlemek için geçmiş bilgi h olarak kabul edilmektedir. Tabaka (layer) politikası, tam bağlı bir katman olarak tanımlanır:

$$h_t(x) = sigmoid(W_n^T c_t + v) \tag{5.8}$$

Adım *t*'de RNN katmanından çıkarılan çıktı c_t, $\boldsymbol{W}_h$ ağırlıklarını ifade etmektedir ve *v* sapma parametresidir.

Politika katmanı aracılığıyla, adım *t*'deki RNN çıktısı [0,1] aralığına eşlenmektedir. Son ikili kodları politika olasılığından elde etmek için eşik fonksiyonuna uygulanır.

$$b_k(x) = g\big(h(x)\big) = sgn(h_k(x) - 0.5),\ k = 1, \cdots, q \tag{5.9}$$

5.1.3. Ajan eğitim stratejisi

Şekil 5.2 de politika ağının detayları gösterilmektedir. Eğitim aşamasında global ödül, üretilen kodların anlamsal sıralama bilgisini korumasını sağlarken, eylem grubu ödülü ağı, önceki üretilen hash kodları tarafından oluşturulan hataları düzeltmeyi sağlamaktadır. Şekil 5.2 de gösterildiği gibi, politika ağının eğitimini yönlendirmek için tanımlanan iki ödül fonksiyonu kullanılmaktadır. Eylemin beklenen toplam ödülünü maksimize etmek, parametreleri güncelleme için Monte Carlo Politika Gradyanı kullanılmaktadır.

$$L_g(\theta) = \sum\nolimits_i \sum\nolimits_k log\left[P\big(a_{i,k} \,|\hat{s}_{i,k}; \theta\big)\right] R_i^g \tag{5.10}$$

Burada, $L_g(\theta)$ eylem grubunun beklenen toplam ödülünü temsil etmektedir. Global eylem ödülü öncelikle tüm hash kodları setinin kalitesine odaklanır, global eylem ödülünü optimize etmek için gradyan inişi yöntemi benimsenmektedir. Her üçlü demet $(x_i,\ x_i^+,\ x_i^-)$ için $h(x_i)$, $h(x_i^-)$açısından alt gradyanı denklem (5.13) deki gibi hesaplanmaktadır:

$$\frac{\partial R_i^G}{\partial h(x_i)} = 2\big(h(x_i^-) - h(x_i^+)\big) \times I_c \tag{5.11}$$

$$\frac{\partial R_i^G}{\partial h(x_i^+)} = 2\big(h(x_i^+) - h(x_i)\big) \times I_c \tag{5.12}$$

$$\frac{\partial R_i^G}{\partial h(x_i^-)} = 2\big(h(x_i) - h(x_i^-)\big) \times I_c \tag{5.13}$$

Burada I_c bir gösterge fonksiyonudur, c doğruysa I_c*=1,* aksi halde I_c*=0*'dır.Bu nedenle, global ödül oluşturulan kodların anlamsal sıralama bilgisini korumasını sağlayabilmektedir. RNN gradyan ayrıntıları aracılığıyla politika ağı geçmiş sıralama

hatalarını düzeltme yeteneğine sahiptir. RNN ağının gradyanı şu şekilde formüle edilmektedir.

$$\delta_h^t = \theta'(c_t)(\delta_k^t w_h + \delta_h^{t+1} w_{hi}) \tag{5.14}$$

$$\delta_h^t = \frac{\partial J_L}{\partial c_t} \tag{5.15}$$

$$\delta_k^t = \frac{\partial J_L}{\partial h_t} \tag{5.16}$$

Burada, c_t ve h_t sırasıyla RNN'in gizli durumu ve çıktısını temsil etmektedir. c_t'yi h_t 'ye eşleyen fonksiyon $\theta(c_t)$ olarak ifade edilmektedir. Adım t'deki gizli katmanının gradyanı, iki bölümden oluşmaktadır: t adımındaki çıktıdan gelen gradyan ve $t+1$ adımındaki gizli katmandan gelen gradyan, ikincisi önceki oluşturulan hash kodlarından kaynaklanan hataları düzeltebilen ardışık ödül olarak kabul edilmektedir. Eğitim aşamasında önceki hash fonksiyonları, mevcut hash fonksiyonundan gradyan bilgisi alır ve duyarlılık doğruluğunu artırmak için parametreleri güncellemektedir.

6. DENEY VE SONUÇLAR

6.1. Deney Parametreleri ve Değerlendirme ölçütleri

Çalışmada, CIFAR10, NUS-WIDE ve MIRFlickr gibi popüler veri setleri üzerinde gerçekleştirilen deneylerde, çeşitli hashleme yöntemleri incelenmiştir. Denetimsiz yöntemler arasında LSH, SH ve ITQ gibi geleneksel yöntemler ile SDH gibi biraz daha güncel yöntemler bulunmaktadır. Bu yöntemler, derin ağlar içermeyen ve genellikle daha basit matematiksel işlemlere dayanan hashleme algoritmalarıdır. Diğer yandan, CNNH, NINH, DSH ve HashNet gibi güncel yöntemler ise ham görüntü piksellerini doğrudan girdi olarak alarak, derin öğrenme modeli kullanarak hash fonksiyonu öğrenme işlemini gerçekleştiren derin hash yöntemleridir. Bu yöntemler, verinin daha karmaşık özelliklerini dikkate alarak daha etkili ve özgün hash kodları üretebilmektedirler.

CIFAR10 veri kümesi, 10 farklı sınıftan toplamda 60,000 renkli görüntü içermektedir. Bu görüntülerin her biri 32x32 piksel boyutlarındadır [54],[55]. Çalışmaları takiben, bu veri kümesi üzerinde yapılan deneylerde 1000 görüntüden oluşan bir sorgu kümesi rastgele seçilmiştir.

NUS-WIDE veri kümesi, yaklaşık 270,000 görüntü içeren geniş bir veri setidir [55]. çalışmasını takiben, deneyler için sıkça kullanılan 21 kavramdan oluşan bir alt küme seçilmiştir. Her bir kavram için sorgu kümesi olarak 100 görüntü kullanılarak toplamda 2,100 görüntü seçilmiştir. MIRFlickr veri kümesi, Flickr web sitesinden alınan 25,000 görüntüyü ve bu görüntülerle ilişkili etiketleri içermektedir. Benzer şekilde, deneylerde 1000 görüntüden oluşan bir sorgu kümesi rastgele seçilmiştir. Çalışma açık kaynaklı framework PyTorch üzerinde uygulanmıştır. Ağın ilk 18 katmanının parametreleri, ImageNet veri kümesi üzerinde eğitilmiş bir derin öğrenme modeli olan VGG-19 ağı [56] ile başlatılmıştır. Politika ağındaki RNN in gizli katmanının boyutu 4096 olarak ayarlanmıştır. Tüm deneylerde ağlar 0.001 başlangıç öğrenme hızıyla eğitilmiştir. Eğitim süreci ilerledikçe daha küçük adımlarla ilerlemesini sağlayan bir öğrenme hızı azaltma stratejisi olarak öğrenme hızı her 10000 adımda bir 10 kat azaltılmıştır.

Eğitim sırasında her bir adımda kullanılan örnek sayısı olan mini grup boyutu 16, ağırlık bozulma parametresi 0.0005 olarak belirlenmiştir. Önerilen kayıp fonksiyonundaki parametreler için, tüm deneylerde m_t=1 olarak ayarlanmıştır. Her bir eylem grubunda üretilecek hash biti sayısı olan eylem grubunun uzunluğu, deney boyunca sabit 12 olarak belirlenmiştir. Yöntem ile CNNH, NINH, DSH ve HashNet yöntemlerinin karşılaştırılması için ham görüntü pikselleri giriş olarak kullanılmıştır. Bu, her bir yöntemin aynı giriş verisi üzerinde değerlendirilmesini sağlamaktadır ve sonuçların karşılaştırılabilirliğini artırmaktadır.

CNNH, NINH, DSH ve HashNet in temsili öğrenme katmanları birbirinden farklı olduğundan, doğru bir karşılaştırma yapabilmek için tüm derin hash yöntemlerinin ilgili çalışmalardan aynı VGG-19 modeli ile başlatılmış sonuç değerleri alınmıştır. Bu sayede, her bir yöntem için ağın ilk 18 katmanının aynı önceden eğitilmiş ağı kullanarak başlatılmış ve sonuçların giriş verisine olan bağımlılığı azalmış bir şekilde değerlendirme sağlanmaktadır.

CNNH, NINH, DSH ve HashNet in alınan sonuçları sırasıyla CNNH*, DSH*, NINH* ve HashNet* olarak adlandırılmıştır.

Geleneksel yöntemler ile derin hash yöntemleri arasında doğru bir karşılaştırma yapmak için derin özelliklerle geleneksel yöntemler üzerinde yapılmış deney sonuçlarına yer verilmiştir. Bu deneylerde, her bir görüntü için aynı önceden eğitilmiş VGG-19 ağından 4096 boyutlu derin özellik çıkartılmıştır. Derin özelliklerle geleneksel yöntemlerin sonuçları LSH-VGG19, SH-VGG19, IT-VGG19 ve SDH-VGG19 olarak adlandırılmıştır. SDH, SH, LSH ve ITQ ‘nun sonuçları ilgili çalışmalardan alınmıştır.

Önerilen yaklaşım ve karşılaştırılan yöntemlerin algoritma başarısını nesnel ve kapsamlı bir şekilde değerlendirmek için Ortalama Hassaslık Puanı (MAP), en iyi k dönüş sonucunda hassasiyet, precision-recall (Hassaslık-Duyarlılık) eğrileri ve ikili hash kullanarak Hamming yarıçapı 2 içinde hassasiyet olmak üzere dört değerlendirme ölçütü kullanılmıştır.

En İyi k Dönüş Sonucunda Hassasiyet (topK-precision): Bu ölçüt, belirli bir k değeri için en iyi sonuçları sağlayan hassasiyet değerini ifade eder. Algoritmanın belirli bir k değeri için ne kadar doğru tahminler yaptığını belirlemek için kullanılmaktadır.

Hassasiyet-Duyarlılık Eğrileri: Hassasiyet-duyarlılık eğrileri, çeşitli kesme eşikleri altında hassasiyet ve duyarlılığın değişimini gösteren bir grafiktir. Daha yüksek bir kesme eşiği, daha yüksek bir hassasiyet ve daha düşük bir duyarlılık sağlamaktadır.

İkili Hash Kullanarak Hamming Yarıçapı 2 İçinde Hassasiyet: Bu ölçüt, ikili hashleme yöntemlerinin belirli bir Hamming yarıçapı içinde ne kadar doğru sonuçlar ürettiğini değerlendirir. Hamming yarıçapı, iki kod arasındaki farklı bit sayısını ifade eder. Bu ölçüt, algoritmanın belirli bir hata toleransı içinde ne kadar doğru çalıştığını gösterir. İkili hashleme kullanılarak elde edilen kodlar arasında Hamming mesafesi 2 olan kodları döndüren bir arama yapılır ve bu kodlar gerçek pozitiflerle karşılaştırılır. Hassasiyet, doğru sonuçların bu kodlar içindeki oranını ifade etmektedir.

MAP puanları, tüm sorgular için ortalama hassasiyetin (AP) ortalaması olarak hesaplanmaktadır ve AP (Average Precision) şu şekilde hesaplanır:

$$AP = \frac{1}{R}\sum_{k=1}^{n} \frac{k}{R_k} \text{ x } rel_k \tag{6.1}$$

n veri tabanının boyutunu, R veri tabanındaki ilgili görüntülerin sayısını, R_k ise ilk k dönüşteki ilgili görüntülerin sayısını ifade eder ve rel_k, k. sıradaki görüntünün ilgili olup olmadığını belirtmektedir. Bu değer ilgiliyse 1, değilse 0'dır. AP, her bir sorgu için hesaplanır ve ardından tüm sorguların AP değerlerinin ortalaması alınarak MAP puanı hesaplanmaktadır.

6.2. Deney Sonuçları

CIFAR10, NUS-WIDE ve MIRFlickr veri setleri üzerinde gerçekleştirilen deneyler ve çeşitli hashleme yöntemlerinin inceleme sonuçları tablolarda verilmiştir.

Tablo 6.1. Derin hash yöntemleri ile MAP değerleri

Yöntem	CIFAR10				NUS-WIDE				MIRFlickr			
	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit
DRLIH	0.815	0.842	0.854	0.852	0.822	0.845	0.844	0.852	0.795	0.810	0.809	0.813
HashNet*	0.765	0.823	0.840	0.843	0.812	0.833	0.830	0.840	0.777	0.782	0.785	0.785
DSH*	0.708	0.712	0.751	0.720	0.793	0.804	0.815	0.800	0.651	0.681	0.684	0.686
NINH*	0.792	0.818	0.832	0.830	0.808	0.827	0.827	0.827	0.772	0.756	0.760	0.778
CNNH*	0.683	0.692	0.667	0.623	0.768	0.784	0.790	0.740	0.763	0.757	0.758	0.755

Tablo 6.2. Geleneksel yöntemler-VGG19 ile MAP değerleri

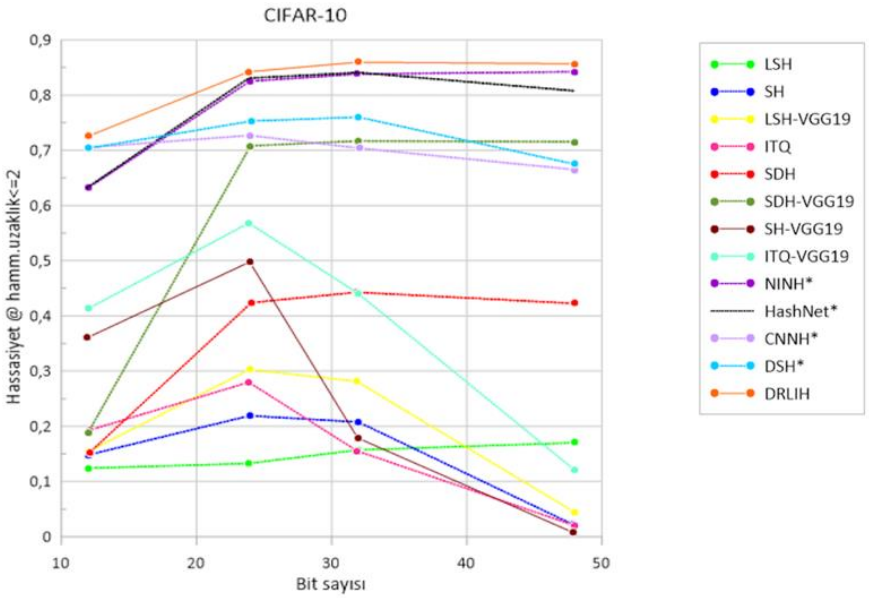
Yöntem	CIFAR10				NUS-WIDE				MIRFlickr			
	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit
DRLIH	0.815	0.842	0.854	0.852	0.822	0.845	0.844	0.852	0.795	0.810	0.809	0.813
SDH-VGG19	0.430	0.652	0.653	0.665	0.730	0.797	0.819	0.30	0.732	0.739	0.737	0.747
ITQ-VGG19	0.339	0.361	0.368	0.375	0.777	0.800	0.806	0.817	0.686	0.685	0.687	0.689
SH-VGG19	0.244	0.213	0.213	0.209	0.712	0.697	0.689	0.682	0.618	0.604	0.598	0.595
LSH-VGG19	0.133	0.171	0.178	0.198	0.518	0.567	0.618	0.651	0.575	0.584	0.604	0.614

Tablo 6.3. Geleneksel yöntemler ile MAP değerleri

Yöntem	CIFAR10				NUS-WIDE				MIRFlickr			
	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit
DRLIH	0.815	0.842	0.854	0.852	0.822	0.845	0.844	0.852	0.795	0.810	0.809	0.813
SDH	0.255	0.330	0.344	0.360	0.460	0.510	0.519	0.525	0.595	0.601	0.608	0.605
ITQ	0.158	0.163	0.168	0.169	0.472	0.478	0.483	0.476	0.576	0.579	0.579	0.580
SH	0.124	0.125	0.125	0.129	0.452	0.445	0.443	0.437	0.561	0.562	0.563	0.562
LSH	0.116	0.121	0.124	0.131	0.436	0.414	0.432	0.442	0.557	0.564	0.562	0.569

6.2.1. CIFAR-10 veri seti sonuçları

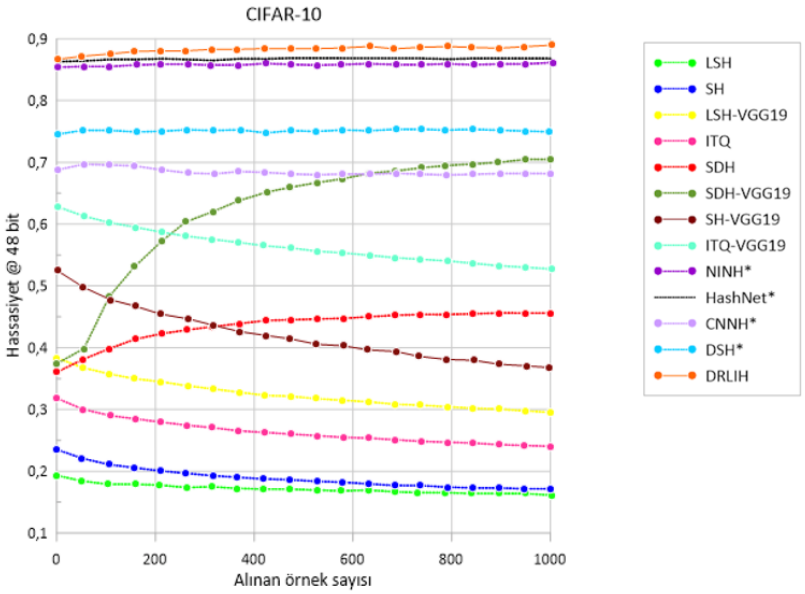
Tablo da CIFAR10 veri kümesinde farklı uzunluklardaki ikili hash kodları için MAP puanları gösterilmektedir. Genel olarak, yöntem ortalama 0.840 MAP puanı ile en yüksek performansı göstermiştir ve tüm ikili hash kodu uzunluklarında sürekli olarak en iyi sonuçları elde etmiştir. Daha spesifik olarak, derin hash yöntemleri, derin özelliklerle geleneksel yöntemler ve geleneksel yöntemler olarak sonuç tablosu üç gruba ayrılmıştır. HashNet* ile karşılaştırıldığında, ortalama 0.818 MAP puanı elde eden HashNet*'e göre yöntemin mutlak iyileştirmesi 0.022'dir.



Şekil 6.1. Cifar-10 hamming yarıçapı 2 hassasiyet grafiği.

Derin özellikler kullanarak geleneksel yöntemlerden SDH-VGG19 ‘nin ortalama 0.600 MAP puanı ile karşılaştırıldığında elde eden önerilen yöntemin mutlak iyileştirmesi 0.240'dır. Geleneksel yöntemlerden SDH'nin ortalama 0.322 MAP puanı ile karşılaştırıldığında önerilen yaklaşımın mutlak iyileştirmesi 0.518'dir.

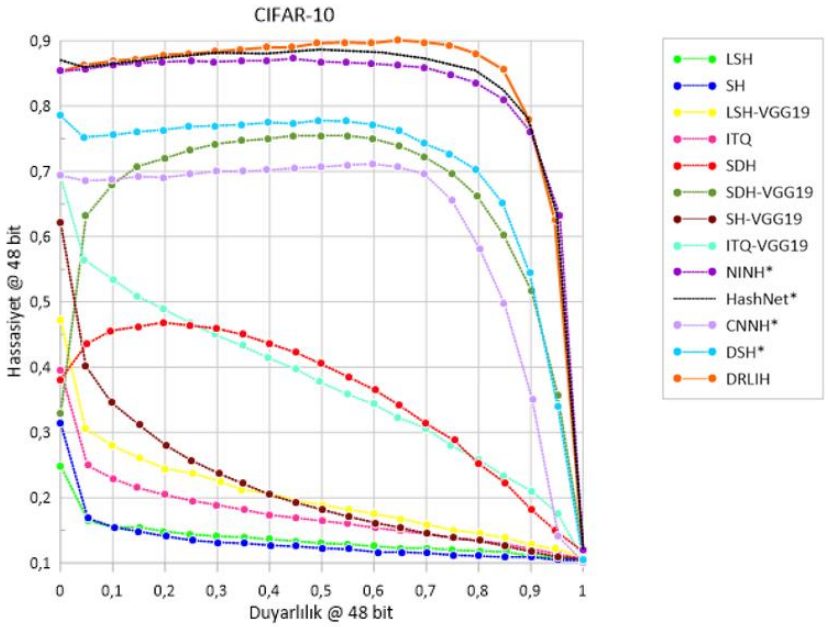
Şekil 6.1, ikili hash kullanarak Hamming yarıçapı 2 de hassasiyeti göstermektedir. Yöntemin hassasiyeti, tüm ikili hash kodu uzunluklarında en iyi sonuçları elde etmiştir ve 48 bit kod uzunluğunda en yüksek hassasiyeti elde etmiştir, bu da önerilen yöntemin uzun hash kodları üzerindeki sağlamlığını göstermektedir.



Şekil 6.2. Cifar-10 top k hassaiyet grafiği.

Şekil 6.2 de, yöntem en iyi hassasiyet ile en iyi sonuçları elde etmektedir. Şekil 6.3, 48 bit kodlarla Hamming sıralaması kullanarak hassasiyet-duyarlılık eğrilerini göstermektedir.

Yöntem tüm duyarlılık seviyelerinde en iyi doğruluğu elde etmiştir, bu da önerilen yaklaşımın etkililiğini daha net göstermiştir.

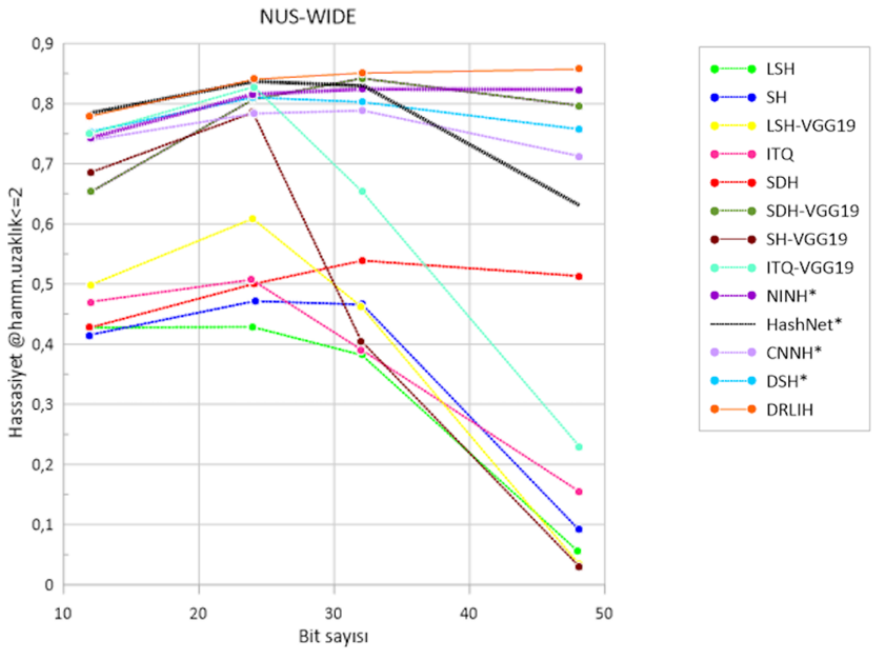


Şekil 6.3. Cifar-10 48 bit hamming hassasiyet-duyarlılık grafiği.

6.2.2. NUS-WIDE veri seti sonuçları

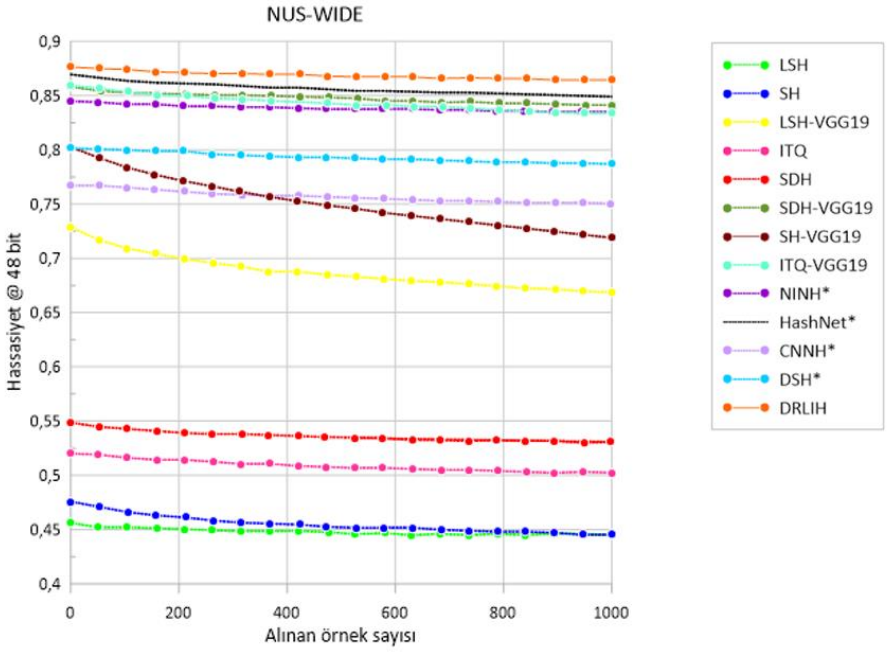
NUS-WIDE veri kümesinde farklı uzunluklardaki ikili hash kodları için MAP puanları tabloda gösterilmektedir. Yöntem, en iyi MAP puanlarını elde etmiştir (ortalama 0.840). HashNet* 'e göre (ortalama 0.829) ortalama MAP'te 0.011 mutlak bir iyileştirme elde etmiştir. Derin özellikler kullanarak geleneksel yöntem olan ITQ-VGG19 ile karşılaştırıldığında, ortalama 0.800 MAP puanı elde eden ITQ-VGG19, yöntemin ortalama MAP'te 0.040 mutlak bir iyileştirmesi vardır.

Şekil 6.4 'de önerilen yöntemin ikili hash kullanarak Hamming yarıçapı 2 içinde en iyi hassasiyeti elde ettiği gözlemlenmiştir.



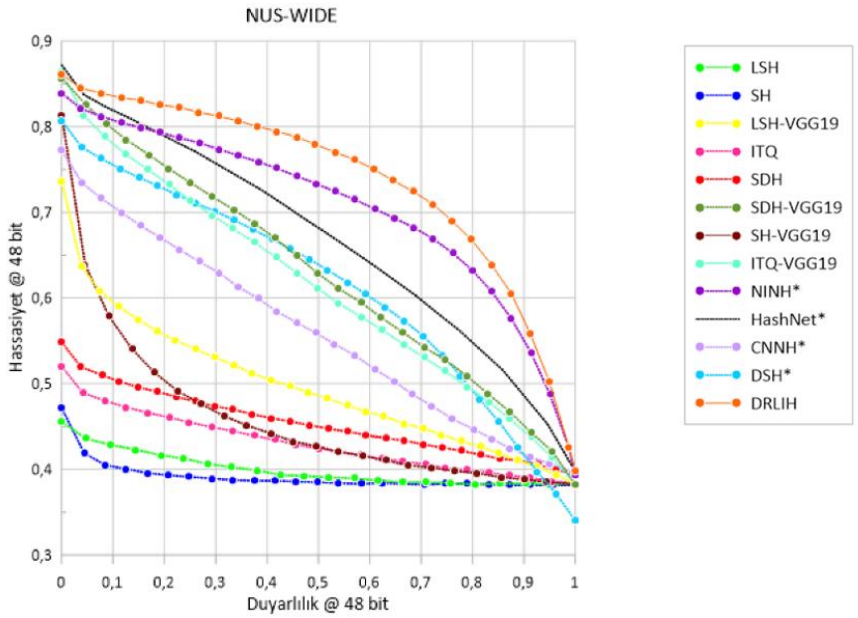
Şekil 6.4. Nus-wide hamming yarıçapı 2 hassasiyet grafiği.

Şekil 6.5, en iyi k dönüş sonuçları hassasiyetini göstermektedir. Yöntem, diğer yöntemlerle karşılaştırıldığında en iyi hassasiyeti elde etmiştir.



Şekil 6.5. Nus-wide top k hassasiyet grafiği.

Şekil 6.6, 48 bitlik Hamming sıralaması kullanarak hassasiyet-duyarlılık eğrilerini göstermektedir; yöntemin tüm duyarlılık seviyelerinde diğer yöntemlere göre en iyi hassasiyeti elde ettiği görülebilmektedir.

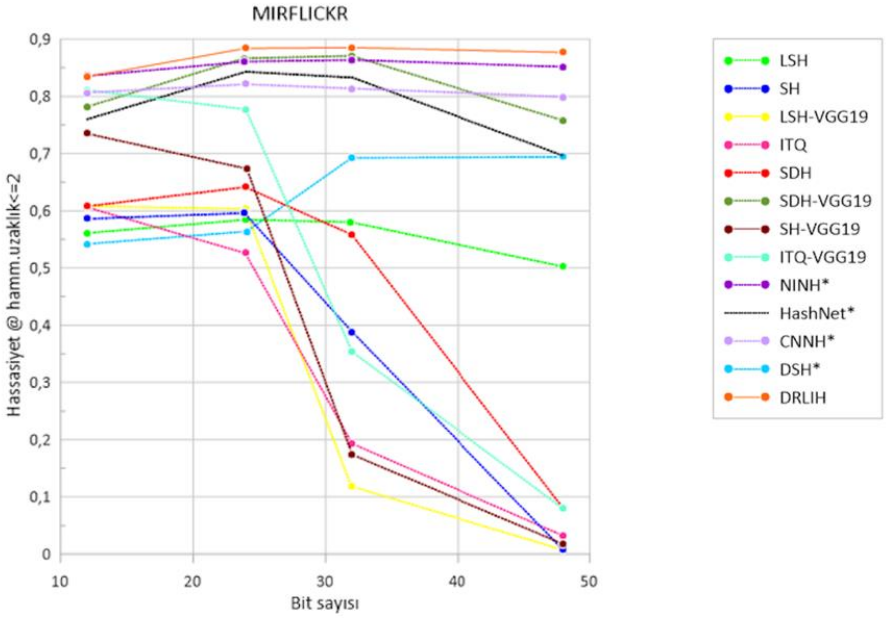


Şekil 6.6. Nus-wide 48 bit hamming hassasiyet-duyarlılık grafiği.

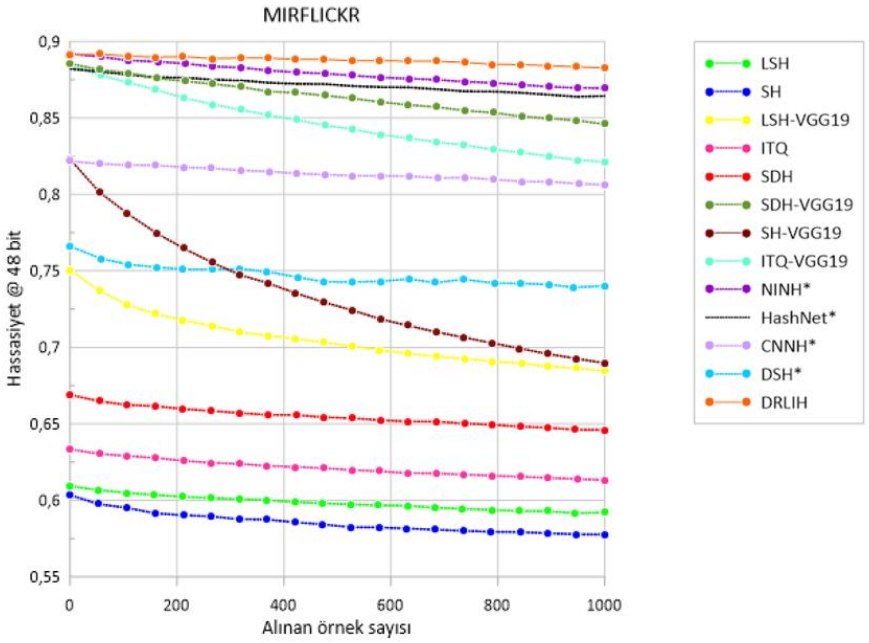
6.2.3. MIRFLICKR veri seti sonuçları

Tabloda MIRFlickr veri kümesinde farklı uzunluklardaki ikili hash kodları için MAP puanları gösterilmektedir, önerilen yöntem en iyi MAP puanlarını elde etmiştir (ortalama 0.808). Yöntem, HashNet* 'e göre (ortalama 0.782) ortalama MAP'te mutlak bir iyileştirme olan 0.026 elde etmiştir. Derin özellikler kullanarak geleneksel yöntem olan SDH-VGG19 ile karşılaştırıldığında, ortalama 0.739 MAP puanı elde eden SDH-VGG19'e göre yöntemin ortalama MAP'te 0.069 mutlak bir iyileştirmesi vardır.

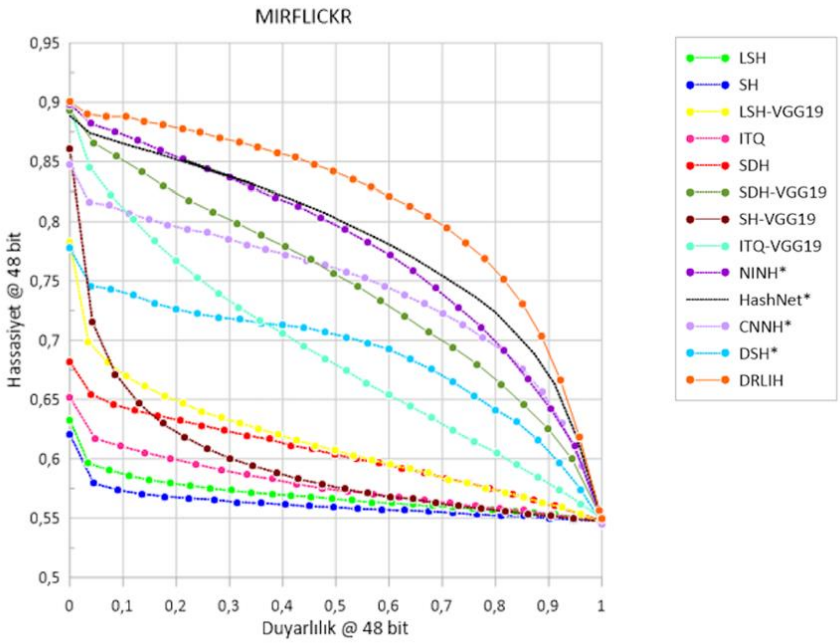
En yüksek puanlı geleneksel yöntem olan SDH ile karşılaştırıldığında, ortalama 0.602 MAP puanı elde eden SDH'e göre yöntemin MAP'te 0.206 mutlak bir iyileştirmesi vardır.



Şekil 6.7. Mırflıckr hamming yarıçapı 2 hassasiyet grafiği.



Şekil 6.8. Mırflıckr top k hassasiyet grafiği.



Şekil 6.9. Mırflıkr 48 bit hamming hassasiyet-duyarlılık grafiği.

7. SONUÇ

Görüntü hash işlemi modern dijital dünyada veri doğrulama, güvenlik ve analiz süreçlerinde kritik bir araç olarak öne çıkmaktadır. Finansal kurumlar, e-ticaret platformları ve güvenlik sistemleri, kimlik tespiti için hash algoritmalarından yararlanılmaktadır. Özellikle yüz tanıma sistemlerinde ve biyometrik doğrulamada, hash işlemi ile hızlı ve güvenilir sonuçlar elde edilir. Görüntü hash teknolojisi, aynı zamanda büyük veri analitiği ve makine öğrenimi uygulamalarında da kullanılmaktadır. Büyük miktarda görüntü verisinin işlenmesi ve sınıflandırılması gereken durumlarda, hash algoritmaları verimliliği artırarak performansı optimize etmektedir.

Çalışmada görüntü hash yöntemi olarak önerilen derin pekiştirmeli öğrenme yaklaşımı, özellikle büyük ve karmaşık veri setleri ile çalışırken yüksek performans sağlayan bir tekniktir. Bu yöntemde, hash işlevlerini modellemek için politika tabanlı derin pekiştirme öğrenme ağı kullanılmıştır. Bu ağ, görüntüleri ikili kodlara sıralı olarak yansıtmak için tekrarlayan sinir ağları (Recurrent Neural Networks - RNN) ile desteklenmektedir. Bu yaklaşımda, hash işlevleri ajanlar olarak ele alınır ve bu ajanlar, görüntüleri belirli bir sırayla hash kodlarına dönüştürmektedir.

Tüm ağ, ödül işlevini optimize ederek eğitilir ve bu ödül işlevleri, ajanın yaptığı eylemleri değerlendirerek, doğru eylemleri teşvik eder ve hatalı eylemleri cezalandırmaktadır.Bu sayede, modelin performansı ve doğruluğu artırılır. Önerilen derin pekiştirmeli öğrenme tabanlı hash yaklaşımı, ardışık öğrenme stratejisi ile çalışır ve bu strateji, geçmiş eylemler tarafından üretilen hataları düzelterek genel doğruluğu sürekli olarak optimize etmektedir. Yöntem, ödül işlevleri ve ardışık öğrenme stratejileri ile optimize edilerek, görüntülerin ikili kodlara etkin ve doğru bir şekilde dönüştürülmesini sağlamaktadır.

Bu yaklaşımın etkililiği, yaygın olarak kullanılan CIFAR-10, NUS-WIDE ve MIRFLICKR veri setleri üzerinde yapılan deneylerle gösterilmiştir. Deney sonuçları, yaklaşımın geleneksel yöntemlere kıyasla daha yüksek doğruluk ve verimlilik sağladığını ortaya koymuştur.

KAYNAKLAR

[1] Swaminathan, A., Mao, Y., & Wu, M. (2006). Robust and secure image hashing. *IEEE Transactions on Information Forensics and Security*, *1*(2), 215–230. https://doi.org/10.1109/TIFS.2006.873601

[2] Venkatesan', R., Koon2, S.-M., Jakubowskil, M. H., & Moulin2, P. (n.d.). ROBUST IMAGE HASHING.

[3] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (n.d.). Deep Reinforcement Learning that Matters. www.aaai.org

[4] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. http://arxiv.org/abs/1509.02971

[5] H. Liu, R. Wang, S. Shan, ve X. Chen, "Deep Supervised Hashing for Fast Image Retrieval".

[6] W.-J. Li, S. Wang, ve W.-C. Kang, "Feature Learning based Deep Supervised Hashing with Pairwise Labels".

[7] H. Zhu, M. Long, J. Wang, ve Y. Cao, "Deep Hashing Network for Efficient Similarity Retrieval *". [Çevrimiçi]. Erişim adresi: www.aaai.org

[8] Z. Cao, M. Long, J. Wang, ve P. S. Yu, "HashNet: Deep Learning to Hash by Continuation *".

[9] S. Su, C. Zhang, K. Han, ve Y. Tian, "Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN".

[10] Q. Li, Z. Sun, R. He, ve T. Tan, "Deep Supervised Discrete Hashing".

[11] X. Wang, Y. Shi, ve K. M. Kitani, "Deep Supervised Hashing with Triplet Labels", Ara. 2016, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1612.03900

[12] Y. Li, W. Pei, Y. zha, ve J. van Gemert, "Push for Quantization: Deep Fisher Hashing", Ağu. 2019, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1909.00206

[13] Z. Zhang, Q. Zou, Y. Lin, L. Chen, ve S. Wang, "Improved Deep Hashing with Soft Pairwise Similarity for Multi-label Image Retrieval", Mar. 2018, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1803.02987

[14] X. Zheng, Y. Zhang, ve X. Lu, "Deep balanced discrete hashing for image retrieval", *Neurocomputing*, c. 403, ss. 224-236, Ağu. 2020, doi: 10.1016/j.neucom.2020.04.037.

[15] Q.-Y. Jiang ve W.-J. Li, "Asymmetric Deep Supervised Hashing". [Çevrimiçi]. Erişim adresi: www.aaai.org

[16] Y. Chen, Z. Lai, Y. Ding, K. Lin, ve W. K. Wong, "Deep Supervised Hashing with Anchor Graph".

[17] L. Yuan *vd.*, "Central Similarity Quantization for Efficient Image and Video Retrieval". [Çevrimiçi]. Erişim adresi: https://github.com/yuanli2333/

[18] Y. Li ve J. van Gemert, "Deep Unsupervised Image Hashing by Maximizing Bit Entropy", Ara. 2020, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/2012.12334

[19] Kumar, S. (2020, 29 Ocak). Supervised vs Unsupervised vs Reinforcement https://www.aitude.com/supervised-vs-unsupervised-vs-reinforcement/ 19 Mart 2024 tarihinde alınmıştır.

[20] M. S. Thesis ve S. Nissen, "Large Scale Reinforcement Learning using Q- SARSA(λ) and Cascading Neural Networks", 2007.

[21] Y. Chandak, G. Theocharous, J. Kostas, S. Jordan, ve P. S. Thomas, "Learning Action Representations for Reinforcement Learning", Oca. 2019, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1902.00183

[22] H. Altay Güvenir, F. Polat, K. Leblebicio, I. Hakkı Toroslu, ve A. Ahmet Co, "Abstraction In Reinforcement Learning in Partially Observable Environments".

[23] R. S. Sutton ve A. G. Barto, "Reinforcement Learning: An Introduction Second edition, in progress".

[24] D. J. Rezende *vd.*, "Causally Correct Partial Models for Reinforcement Learning", Şub. 2020, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/2002.02836

[25] R. S. Sutton ve A. G. Barto, *Reinforcement learning : an introduction*.

[26] "Artificial Intelligence A Modern Approach Third Edition".

[27] P. Fernando ve P. Faustino, "INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA Dynamic Equilibrium through Reinforcement Learning", 2011.

[28] S. Ivanov ve A. D'yakonov, "Modern Deep Reinforcement Learning Algorithms", Haz. 2019, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1906.10025

[29] Cheng, Y., Chen, L., Chen, C. L. P., & Wang, X. (2021). Off-Policy Deep Reinforcement Learning Based on Steffensen Value Iteration. IEEE Transactions on Cognitive and Developmental Systems, 13(4), 1023–1032. https://doi.org/10.1109/TCDS.2020.3034452

[30] Lee, J., & Sutton, R. S. (2021). Policy iterations for reinforcement learning problems in continuous time and space — Fundamental theory and methods. Automatica, 126. https://doi.org/10.1016/j.automatica.2020.109421

[31] Lee, M. (2005, 4 Ocak). Actor-Critic Methods reinforcement learning http://incompleteideas.net/book/ebook/node66.html 27 Nisan 2024 tarihinde alınmıştır.

[32] D. Bloembergen, "analyzing Reinforcement Learning algorithms using Evolutionary Game Theory".

[33] 2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES). (n.d.).

[34] ADL. (2018, 3 Eylül). An introduction to Q-Learning reinforcement learning https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/ 21 Mart 2024 tarihinde alınmıştır.

[35] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

[36] Amber. (2019, 9 Nisan). (Deep)Q-lerning,Part:1 basic introdution and implementation https://medium.com/@qempsil0914/zero-to-one-deep-q-learning-part1-basic-introduction-and-implementation-bb7602b55a2c 3 Nisan 2024 tarihinde alınmıştır.

[37] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning", 1992.

[38] Torres, J. (2020, 10 Eylül). Policy-Gradient Methods. https://towardsdatascience.com/policy-gradient-methods-104c783251e0 adresinden 16 Mart 2024 tarihinde alınmıştır.

[39] Williams, R. J. (1992). *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning* (Vol. 8).

[40] Torres, J. (2020, 10 Eylül). Policy-Gradient Methods Reinforce algorithm https://towardsdatascience.com/policy-gradient-methods-104c783251e0 12 Nisan 2024 tarihinde alınmıştır.

[41] Torres, J. (2020, 22 Temmuz). Monte Carlo Yöntemleri Deep Reinforcement https://towardsdatascience.com/monte-carlo-methods-9b289f030c2e 12 Nisan 2024 tarihinde alınmıştır.

[42] X. Hao, G. Zhang, ve S. Ma, "Deep Learning", içinde *International Journal of Semantic Computing*, World Scientific Publishing Co. Pte Ltd, Eyl. 2016, ss. 417-439. doi: 10.1142/S1793351X16500045.

[43] Y. Lecun, Y. Bengio, ve G. Hinton, "Deep learning", *Nature*, c. 521, sy 7553. Nature Publishing Group, ss. 436-444, 27 Mayıs 2015. doi: 10.1038/nature14539.

[44] Lecun Y, Bengio Y ve Hilton G. Nature, "Deep Learning".

[45] G. Gündüz ve İ. H. Cedimoğlu, "Derin Öğrenme Algoritmalarını Kullanarak Görüntüden Cinsiyet Tahmini", 2019. [Çevrimiçi]. Erişim adresi: https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

[46] Ö. İnik *vd.*, "GAZİOSMANPAŞA BİLİMSEL ARAŞTIRMA DERGİSİ (GBAD) Gaziosmanpasa Journal of Scientific Research Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri", [Çevrimiçi]. Erişim adresi: http://dergipark.gov.tr/gbad

[47] F. DOĞAN ve İ. TÜRKOĞLU, "Derin Öğrenme Modelleri ve Uygulama Alanlarına İlişkin Bir Derleme", *DÜMF Mühendislik Dergisi*, c. 10, sy 2, ss. 409-445, Haz. 2019, doi: 10.24012/dumf.411130.

[48] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. http://arxiv.org/abs/1207.0580

[49] Xiao, T., Li, H., Ouyang, W., & Wang, X. (2016). Learning Deep Feature Representations with Domain Guided Dropout for Person Re-identification. http://arxiv.org/abs/1604.07528

[50] D. Öğrenme Algoritmalarının Yaprak Sınıflandırma Başarımlarının Karşılaştırılması Ferdi DOĞAN, "The Comparison Of Leaf Classification Performance Of Deep Learning Algorithms", 2018.

[51] S. Dong, P. Wang, ve K. Abbas, "A survey on deep learning and its applications", *Computer Science Review*, c. 40. Elsevier Ireland Ltd, 01 Mayıs 2021. doi: 10.1016/j.cosrev.2021.100379.

[52] Aishwary. (2023, 4 Aralık). Introduction to Recurrent Neural Network https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ 25 Nisan 2024 tarihinde alınmıştır.

[53] S. Ioffe ve C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", Şub. 2015, [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1502.03167

[54] Xia, R., Pan, Y., Lai, H., Liu, C., & Yan, S. (n.d.). Supervised Hashing for Image Retrieval via Image Representation Learning. www.aaai.org

[55] Lai, H., Pan, Y., Liu, Y., & Yan, S. (n.d.). *Simultaneous Feature Learning and Hash Coding with Deep Neural Networks*.

[56] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. http://arxiv.org/abs/1409.1556

ÖZGEÇMİŞ

Ad-Soyad : Elif AKKAYA

ÖĞRENİM DURUMU:

- **Yükseklisans :** Sakarya Üniversitesi, Elektronik, Elektrik Elektronik Mühendisliği
- **Lisans :** 2020, Sakarya Üniversitesi, Mühendislik Fakültesi, Elektrik Elektronik Mühendisliği