GU J Sci, Part C, 7(2): 503-522 (2019)

Gazi Üniversitesi

Fen Bilimleri Dergisi

PART C: TASARIM VE TEKNOLOJİ

http://dergipark.gov.tr/gujsc

A Survey of Hyper-parameter Optimization Methods in Convolutional Neural Networks

Ayla GÜLCÜ[1,*] Zeki KUŞ[2]

[1]*Fatih Sultan Mehmet Vakıf Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 59300, Haliç/İSTANBUL*

[2]*Fatih Sultan Mehmet Vakıf Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Anabilim Dalı, 59300, Haliç/İSTANBUL*

Article Info:

Received: 18/01/2019
Revision 10/06/2019
Accepted: 17/06/2019

Highlights

- *CNN hyper-parameter optimization.*
- *Importance of meta-heuristic methods.*
- *Image classification in deep learning networks*

Keywords

Deep Learning, Convolutional Neural Networks, Heuristic Algorithms, Hyper-parameter optimization

Graphical/Tabular Abstract

In this study, we provide a review on the meta-heuristic methods like Genetic Algorithms, Particle Swarm Optimization, Differential Evolution and Bayes Optimization that have been used extensively to optimize hyper-parameters in Convolutional Neural Networks (CNN). We highlight the hyper-parameters that have been selected to be optimized in those studies along with the value domains of those parameters. These studies reveal that the number of layers, number of kernels and size of those kernels at each layer, learning rate and the batch size are among the hyper-parameters that affect the performance of the CNNs the most.

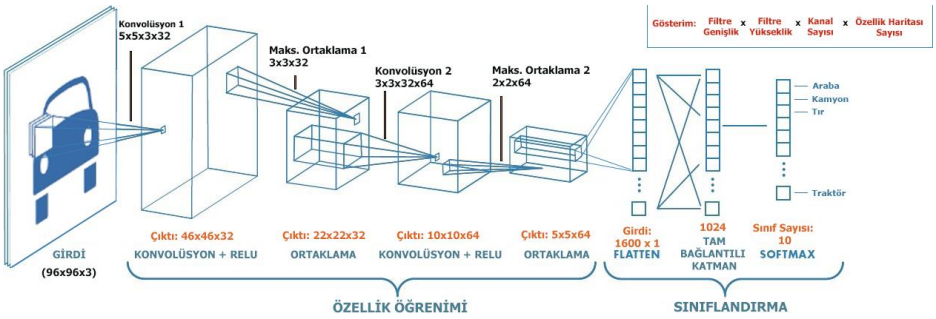


Figure A. structure of convolutional neural networks

Purpose: In this study, meta-heuristic methods that have been used to optimize convolutional neural networks are investigated. A performance comparison of these methods on different image datasets has been presented. The advantages and disadvantages of the optimization approaches have been presented with the aim of providing the user important points that should be considered during hyper-parameter selection process.

Results: The definiton of *"the best"* set of hyper-parameters in convolutional neural networks depends on the problem or in this case, on the dataset. But it is clear from the studies that the selection of some parameters directly affect the performance of the networks. Number of layers, number of filters in each layer and size of each filter, regularization method, learning rate and batch size are among the most important parameters. It is easy to conclude that Genetic Algorithms (GA) are the most widely studied techniques used in hyper-parameter optimizaton. This is due to the fact that they yield successful results in most of the studies. While selecting the optimization method, one should consider the size of the problem, available computational budget and time. In addition, accuracy expectations should also be taken into account. For the problems with small hyper-parameter search space, methods like Grid Search would be sufficient, but for the problems with large search space, meta-heuristic methods would be more convenient.

Conclusion: In this study, the effect of hyper-parameter optimization methods on classification performance is investigated. GA and Particle Swarm Optimization (PSO) methods are the two most-widely used meta-heuristics for hyper-parameter optimization. The computational burden of these methods can be justified with the accuracy improvement achieved with them. If the computational resources are limited, and it is desired to obtain good results in reasonable amount of time, then other methods like TPE and SMAC would be good choices.

*İletişim yazarı, e-mail: agulcu@fsm.edu.tr

DOI: 10.29109/gujsc.514483

GU J Sci, Part C, 7(2): 503-522 (2019)

Gazi Üniversitesi

Fen Bilimleri Dergisi

PART C: TASARIM VE TEKNOLOJİ

http://dergipark.gov.tr/gujsc

Konvolüsyonel Sinir Ağlarında Hiper-Parametre Optimizasyonu Yöntemlerinin İncelenmesi

Ayla GÜLCÜ[1,*] Zeki KUŞ[2]

[1]*Fatih Sultan Mehmet Vakıf Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 59300, Haliç/İSTANBUL*

[2]*Fatih Sultan Mehmet Vakıf Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Anabilim Dalı, 59300, Haliç/İSTANBUL*

Makale Bilgisi

Başvuru: 18/01/2019
Düzeltme: 10/06/2019
Kabul: 17/06/2019

Anahtar Kelimeler

Derin Öğrenme, Konvolüsyonel Sinir Ağları, Sezgisel Algoritmalar, Hiper-Parametre Optimizasyonu

Keywords

Deep Learning, Convolutional Neural Networks, Heuristic Algorithms, Hyper-parameter optimization

Öz

Konvolüsyonel Sinir Ağları (KSA), katmanlarının en az bir tanesinde matris çarpımı yerine konvolüsyon işleminin kullanıldığı çok katmanlı yapay sinir ağlarının bir türüdür. Özellikle bilgisayarlı görü çalışmalarında çok başarılı sonuçlar elde edilse de KSA hala birçok zorluk içermektedir. Daha başarılı sonuçlar elde etmek için geliştirilen mimarilerin giderek daha derinleşmesi ve kullanılan görüntülerin giderek daha yüksek kalitede olmasıyla daha fazla hesaplama maliyetleri ortaya çıkmaktadır. Hem bu hesaplama maliyetlerinin düşürülmesi, hem de başarılı sonuçlar elde edilebilmesi, güçlü donanımların kullanılmasına ve kurulan ağın hiper-parametrelerin optimize edilmesine bağlıdır. Bu çalışmada, *Genetik Algoritma*, *Parçacık Sürü Optimizasyonu*, *Diferansiyel Evrim* ve *Bayes Optimizasyonu gibi* yöntemler ile KSA optimizasyonu gerçekleştirilen çalışmalar incelendi. Bu çalışmalarda optimize edilen hiper-parametreler, tanımlanan değer aralıkları ve elde edilen sonuçlar incelendi. Buna göre, KSA' nın performansında en etkili hiper-parametrelerin filtre sayısı, filtre boyutu, katman sayısı, seyreltme oranı, öğrenme oranı ve yığın boyutu olduğu görülmüştür. Aynı veri kümelerinin kullanıldığı çalışmalar, elde edilen doğruluk değerleri açısından karşılaştırıldığında çoğu veri kümesi için en iyi doğruluk oranlarının popülasyon tabanlı yöntemlerden Genetik Algoritma ve Parçacık Sürü Optimizasyonu kullanılan çalışmalarda elde edildiği görülmüştür. Bu üst-sezgiseller ile elde edilen modellerin performanslarının "state of the art" modellerle yarışabilir durumda hatta bazen daha iyi oldukları görülmüştür. Yine üst-sezgisel kullanılan bazı çalışmalarda üretilen modellerin aşırı büyümesi engellenmiş; basit ve kolay eğitilebilir modeller üretilmiştir. Hesaplama maliyeti açısından çok avantajlı bu basit modeller ile literatürdeki karmaşık modellere çok yakın sonuçlar elde edilebilmiştir.

A Survey of Hyper-parameter Optimization Methods in Convolutional Neural Networks

Abstract

Convolutional neural networks (CNN) are special types of multi-layer artificial neural networks in which convolution method is used instead of matrix multiplication in at least one of its layers. Although satisfactory results have been achieved by CNN especially in computer vision studies, they still have some difficulties. As the proposed network architectures become deeper with the aim of much better accuracy and the resolution of the input images increases, this results in a need for more computational power. Reducing the computational cost while at the same time still having high accuracy rates depend on the use of powerful equipments and the selection of hyper-parameter values in CNN. In this study, we examined methods like Genetic Algorithms, Particle Swarm Optimization, Differential Evolution and Bayes Optimization that has been used extensively to optimize CNN hyper-parameters, and also listed the hyper-parameters selected to be optimized in those studies, ranges of those parameter values and the results obtained by each of those studies. These studies reveal that the number of layers, number and size of the kernels at each layer, learning rate and the batch size parameters are among the hyper-parameters that affect the performance of the CNNs the most. When the studies that use the same datasets are compared in terms of accuracy, Genetic Algorithms and Particle Swarm Optimization which are both population-based methods achieve the best results for the majority of the datasets. It is also shown that the performance of the models found in these studies are competitive or sometimes better than those of the "state of the art" models. In addition, the CNNs produced in these studies are prevented from being overgrown by imposing limits on the hiper-parameter values. Thus simpler and easier to train models have been obtained. These computationally advantageous simpler models were able to achieve competitive results compared to complicated models.

1. GİRİŞ (INTRODUCTION)

Konvolüsyonel sinir ağları (Convolutional Neural Networks - KSA), katmanlarının en az bir tanesinde matris çarpımı işlemi yerine konvolüsyon (filtrelerin girdi üzerinde özellik çıkarımı için dolaşması) işlemi kullanan, yapay sinir ağlarının özelleşmiş bir türüdür [1]. Klasik yapay sinir ağlarından farklı olarak konvolüsyonel sinir ağlarında belirlenen kare boyutta bir matris (filtre/kernel), girdiler üzerinde dolaşarak verilerin işlenmesini, özellik çıkarımını gerçekleştirir. Yapay sinir ağlarında sistem öncelikle örnekler ile eğitilir, ardından test edilerek sistemin hata değeri belirlenir. Öğrenme işlemi, sistemin hata değerinin azaltılması için sistemdeki her bir özellik için kullanılan ağırlıklara verilen değerlerin güncellenmesidir [2].

Yapay sinir ağlarında ilk hesaplama modeli W.S. McCulloch ve W.A. Pitts'in [3], 1943 yılında yayınladığı makale ile ortaya çıkmıştır. Bu çalışmaları takiben 1954 yılında B.G. Farley ve W.A. Clark [4] tarafından uyarılara tepki verebilen bir model oluşturulmuştur. 1970 yılına kadar XOR probleminin sinir ağları ile çözülememesi nedeniyle yavaşlayan çalışmalar bu problemin çözümüyle tekrar hız kazanmış ancak yapay sinir ağları asıl ivmelenmeyi son 10 yıl içerisinde gelişen bilgisayar teknolojisi ve donanımlar ile elde etmiştir.

Günümüzde yapay sinir ağları sınıflandırma, tahmin, örüntü tanıma vb. gibi birçok uygulamada başarılı olarak kullanılmaktadır. Problem tipleri farklılık gösterdiğinden dolayı ortak bir yapay sinir ağı modeli bulunmamaktadır. Problemin girdileri, girdilerin tipleri (tam sayı, metin, reel sayı vb.), beklenen çıktılar, çıktıların şekli (kategorik, ikili) gibi özellikler problemin zorluğunu ve aynı zamanda yapay sinir ağı modelinin yapısını değiştirmektedir. Yapay sinir ağlarında karşılaşılan zorluklardan biri de problem için en başarılı ağ modelinin seçilmesidir. Özellikle konvolüsyonel sinir ağlarında başarılı sonuçlar elde edilebilmesi kullanılacak hiper-parametrelerin doğru seçilmesine bağlıdır. Ancak hiper-parametrelerin optimizasyonu yüksek işlem gücü gerektirdiğinden yapılacak deneylerde hiper-parametre değerleri için belirtilen olası tüm ihtimallerin değerlendirilmesi (grid search) pratik açıdan çok zordur. Bu nedenle hiper-parametre optimizasyonu için üst-sezgisel yöntemler (meta-heuristics) tercih edilmektedir. Bu çalışmada KSA hiper-parametre optimizasyonu çalışmaları incelenmiştir. Araştırma 2012 ve 2019 yılları arasında yapılan çalışmalarla sınırlandırılmıştır. Kullanılan anahtar kelimeler ve operatörler şunlardır: "Deep Learning" OR "Convolutional Neural Networks" AND "Hyper-Parameter Tuning" OR "Parameter Tuning" OR "Parameter Optimization" OR "Hyper-Parameter Optimization". Ardından kullanılan üst-sezgisellerin anahtar kelime olarak tanımlandığı makalelere de ulaşabilmek için "Deep Learning" OR "Convolutional Neural Networks" arama anahtarına teker teker "Meta-Heuristics", "Automatic Configuration", "Evolutionary Algorithms", "Genetic Algorithms" "Particle Swarm Optimization", "Differential Evolution", "Harmony Search" ve "Simulated Annealing" anahtarları da eklenmiştir.

Makalenin ana hatları şu şekildedir: ikinci bölümünde yapay sinir ağlarının yapısı, üçüncü bölümde ise KSA yapısı anlatılmaktadır. Dördüncü bölümde KSA hiper-parametre optimizasyonu çalışmalarında kullanılan yöntemler, veri setleri ve bu çalışmaların performansları sunulmaktadır. Beşinci bölümde ise bu çalışmalarla ilgili değerlendirmeler gösterilmektedir.

2. YAPAY SİNİR AĞLARI YAPISI

Yapay sinir ağları, beyindeki nöronlardan esinlenilerek oluşturulmuş bir yapıdır. Biyolojik sinir sisteminde bulunan soma, yapay sinir ağlarında nörona, dendrit girdiye, akson çıktıya ve sinaps ağırlıklara karşılık gelmektedir. Nöronlar, sinyalleri bir nörondan diğer nörona doğru geçiren ağırlıklandırılmış bağlantılar ile birbirine bağlıdır. Sinir hücreleri arasındaki iletişim sinapslar yardımıyla gerçekleşir. Yapay sinir ağı, sinir hücrelerine gelen bilgileri toplayıp, belirlenen aktivasyon fonksiyonundan geçirerek çıktıyı üretir. Nöron yapısına bakıldığında girdilerin tutulduğu düğümler, bu girdilere ait ağırlıklar, bütün girdi ve ağırlıkların çarpılıp toplanmasıyla elde edilen toplama fonksiyonu, toplama fonksiyonundan gelen değerleri kullanarak sonuç üreten bir aktivasyon fonksiyonundan ve çıktıdan oluşur.

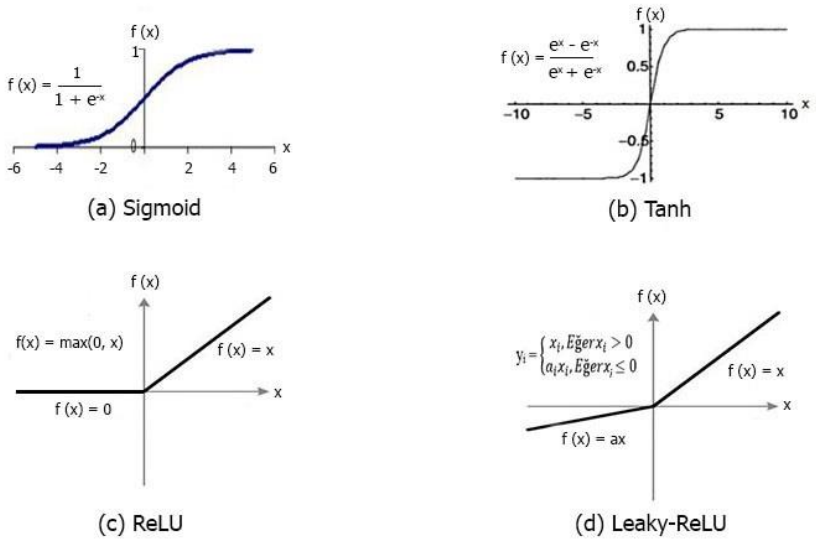
2.1 Aktivasyon Fonksiyonları

Yapay sinir ağlarında gerçekleşen matris işlemleri sonucunda ağ ve ağın elemanları doğrusal yapıdadır [5]. Oluşan doğrusal yapı, aktivasyon fonksiyonları ile doğrusal olmayan bir yapıya dönüştürülür. Bu durum ağın elemanlarının kolay türevlenebilir olmasına katkı sağlar. Ağın hesaplama hızını arttırmak amacıyla kolay

türevlenebilir aktivasyon fonksiyonlarını seçmek çok önemlidir. Sıklıkla kullanılan aktivasyon fonksiyonları Şekil 1' de gösterilmiştir.

- **Sigmoid Aktivasyon Fonksiyonu:** Yapay sinir ağlarında sınıflandırma problemlerinde sıklıkla kullanılan aktivasyon fonksiyonu olan *sigmoid*, gelen girdileri Şekil 1(a)'da görüldüğü gibi 0 ile 1 arasında çıktılara dönüştürür. Bu fonksiyonun en büyük dezavantajı çok fazla katmandan oluşan derin ağlarda belirli bir noktadan sonra türevlenen elemanların sıfıra doğru yaklaşmasıdır [6]; *kaybolan eğim (vanishing gradient)* problemi denen bu durum öğrenme işlemini zorlaştırmaktadır.

- **Tanjant Hiperbolik Aktivasyon Fonksiyonu:** *Tanjant hiperbolik* aktivasyon fonksiyonu gelen girdileri -1 ile 1 arasında çıktılara dönüştürür. Şekil 1(b)'de görüldüğü gibi, [-1, 1] aralığında değer alan çıktılar sıfır merkezlidir ve optimizasyonu kolaylaştırıcı etkisi olan bu durum nedeniyle bu fonksiyon *sigmoid* aktivasyon fonksiyonuna göre daha sık tercih edilmektedir. Fakat bu aktivasyon fonksiyonunda da *kaybolan eğim* problemi devam etmektedir.



Şekil 1. *Aktivasyon Fonksiyonları [7].*

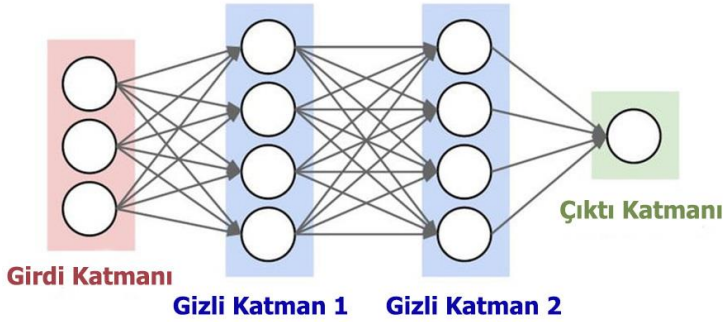
- **Doğrultulmuş Lineer Ünite (Rectified Linear Unit – ReLU):** *ReLU* aktivasyon fonksiyonu, gelen girdileri Şekil 1(c)'de görüldüğü gibi sıfır ile sonsuz arasındaki çıktılara dönüştürür. Bu nedenle *ReLU* doyuma ulaşmayan bir fonksiyon olarak adlandırılır. Bu fonksiyonun en büyük avantajı çok hızlı hesaplanabilmesi ve aynı zamanda *kaybolan eğim* problemine karşı dayanıklı olmasıdır.

- **Sızdırılmış Doğrultulmuş Lineer Ünite (Leaky Rectified Linear Unit - Leaky ReLU):** *ReLU* negatif değerleri direkt olarak sıfıra eşitlediği için ölü *ReLU* olarak adlandırılmaktadır. Bu durum negatif değerlerin fazla olduğu durumlarda katmanlardaki birimlerin aktif hale gelmemesine neden olmaktadır. Bu durumun önüne geçmek amacıyla *Leaky-ReLU* aktivasyon fonksiyonu önerilmiştir. Şekil 1(d)' de görüldüğü gibi negatif değerlerde fonksiyona eğim eklemek amacıyla küçük, sabit bir parametre (a_i) seçilir. Eğer, a_i parametresi 0.01 gibi küçük, sabit bir değer alır ise yöntem *Leaky-ReLU* olarak adlandırılmaktadır. Ancak a_i diğer sinir ağı parametreleri ile beraber adaptif bir şekilde öğreniliyorsa yöntem *PreLU (Parametric ReLU)* olarak adlandırılır [8].

2.2 Çok Katmanlı Yapay Sinir Ağları

Çok katmanlı yapay sinir ağları, girdilerin temsil edildiği *girdi katmanı (input layer),* girdi katmanından gelen bilgilerin işlenerek bir çıktıya dönüştürüldüğü *gizli katmanlar (hidden layers)* ve en son gizli katmandan gelen sonuçların çıktı değerlerine dönüştürüldüğü *çıktı katmanından (output layer)* oluşmaktadır [9]. Şekil 2'de gösterildiği gibi, girdiler gizli katmandaki düğümler ile ağırlıklar aracılığı ile bağlıdır. Gizli katman sayısı ve her bir gizli katmandaki nöron sayısı problemin zorluğuna göre belirlenmektedir. Her bir nöronda girdiler ile

ağırlıkların işlenmesiyle oluşan çıktılar çıktı katmanına gönderilir. Çıktı katmanından hesaplanan çıktı değeri ile beklenen çıktı değeri belirlenen hata hesaplama fonksiyonuna göre hesaplanır. Hesaplanan hata değeri beklenen sonuçtan ne kadar uzak olduğumuzu belirler. Hesaplanan bu değere göre seçilen optimizasyon fonksiyonu ile ağırlıklar güncellenir; yani yapay sinir ağı öğrenmeye başlar.



Şekil 2. *Çok katmanlı YSA yapısı*

2.3 Hata Hesaplama

Yapay sinir ağlarında elde edilen çıktıların, istenilen çıktılara ne kadar yakın olduğunu belirleyebilmek çok önemlidir. Beklenen ve hesaplanan değerler arasındaki fark sinir ağının doğru sınıflandırmalar yapmaya ne kadar uzak olduğunu belirler. Yapay sinir ağlarında hataların hesaplanmasına geçilmeden önce çıktı katmanında elde edilen sonuçların normalizasyonu gerçekleştirilir. Daha sonra bu değerler seçilen hata hesaplama fonksiyonuna verilir. Genel olarak ikili sınıflandırmaların olduğu çalışmalarda *sigmoid* fonksiyonu, daha fazla sınıfa sahip çalışmalarda ise *softmax* fonksiyonu kullanılır. Maliyet fonksiyonu olarak *Ortalama Kareli Hata (Mean Squared Error), Ortalama Mutlak Hata (Mean Absolute Error)* ve *Çapraz Entropi (Cross-Entropy)* yöntemlerinden biri kullanılır.

Softmax yönteminde sınıf sayısı kadar düğüm oluşturulur ve oluşturulan düğümler için olasılık değerleri üretilir [10]. Örneğin: a_i aktivasyon değerlerine sahip düğüm için p_i olasılık değeri denklem 1'de gösterildiği gibi o düğüm için bulunan değerin tüm düğümler için hesaplanan toplam değere bölünmesiyle elde edilir ve en yüksek değere sahip düğümün temsil ettiği sınıf tahmin edilen sınıf olarak belirlenir. Ardından *çapraz entropi* yöntemi ile her bir girdi için hata hesaplanması denklem 2'ye göre gerçekleştirilir ($\hat{y}$: hesaplanan değer, y: gerçek değer) [11]; toplam hata (maliyet) ise denklem 3'e göre hesaplanır (m: örnek sayısı).

$$p_i = exp(a_i)/\sum_j exp(a_j) \quad (1)$$

$$L(\hat{y}, y) = -y^{(i)} log(\hat{y}^{(i)}) - (1 - y^{(i)}) log(1 - \hat{y}^{(i)}) \quad (2)$$

$$J(W, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \quad (3)$$

Yapay sinir ağlarında *aşırı öğrenmeyi* (*overfitting*) engellemek amacıyla en yaygın olarak kullanılan *düzenlileştirme* (*regularization*) yöntemleri şunlardır:

- **L2 ve L1 Düzenlileştirme:** Toplam hata fonksiyonunun sonuna eklenen terimler ile düzenlileştirme yapılır. L2 düzenlileştirme yönteminde her bir ağırlığın karelerinin toplamı eklenirken L1 düzenlileştirme yönteminde ise ağırlıkların mutlak değerlerinin toplamı eklenir. Hesaplanan bu toplamlar, bir düzenlileştirme katsayısı (λ), ile çarpıldıktan sonra toplam hata fonksiyonuna eklenir. Bu λ değeri çok büyük seçilirse *eksik öğrenmeye (underfitting)*; çok küçük seçilirse *aşırı öğrenmeye (overfitting)* neden olabilir.
- **Seyreltme (Dropout) Yöntemi:** En sık kullanılan yöntem olan seyreltme yönteminde amaç, belirlenen bir seyreltme oranına göre gizli katmanda rastgele seçilen bazı düğümlerin hata optimizasyonuna katılmasını engellemektir [12]. Aşırı öğrenmenin engellenmesi için işleme dahil edilmeyen nöronlar, hesaplanan toplam hatanın geriye yayılması sırasında da işleme dahil edilmezler.

2.4 Optimizasyon Metotları

Yapay sinir ağlarında, ağırlıkların doğru bir şekilde güncellenmesi öğrenme işlemi için çok önemlidir. Hata hesaplandıktan sonra elde edilen hataya göre ağırlıklar güncellenir. Ağırlıkların güncellenmesi için *Geri Yayılım Algoritması* (Back-Propagation) kullanılır [5]. *Geri yayılım algoritması* ile sinir ağındaki her bir ağırlığın, hesaplanan hataya olan etkisini hesaplamak için gradyan tabanlı *Stokastik Gradyan Azaltma* [13], *RMS-Prop, Adam* [14] ve *Adadelta* [15] metotları sıklıkla kullanılmaktadır.

- **Gradyan Azaltma (Gradient Descent):** *Gradyan azaltma algoritması* (*Batch Gradient Descent*), tüm veri kümesinde gradyan hesabını gerçekleştirir. Bu durum bir iterasyonun çok uzun sürmesine neden olmakta ve erken yakınsama ile yerel optimumlara yakalanma riskini arttırmaktadır. Bu gibi problemlerden dolayı *mini-batch gradient descent* yöntemi önerilmiştir. Bu yöntemde eğitim veri seti belirlenen sabit değerde yığınlara (*batch*) bölünmüştür ve bölünen her bir yığın için toplam hata hesaplanır ve ağırlıklar güncellenir. Eğer mini-batch değeri "1" olacak şekilde seçilirse *stokastik gradyan azaltma* algoritması uygulanmış olur; bu durumda her defasında bir örnek için hata hesaplanır ve ağırlıklar güncellenir, dolayısıyla vektörizasyonun getirmiş olduğu hız kaybedilir.

- **Momentumlu Gradyan Azaltma (Gradient Descent with Momentum):** Gradyan azaltma yöntemlerinde öğrenme adımlarının en iyi sonuca doğru daha hızlı hareket etmesi ve daha az sapmaların olması istenir. Atılan öğrenme adımları bazen çok fazla olabilir ve atılan bu yanlış adımlar en iyi sonuca ulaşma süresini doğrudan etkileyebilir. Bu problemleri iyileştirmek amacıyla *momentumlu gradyan azaltma* [16, 17] yöntemi önerilmiştir. Geçmişteki öğrenme adımlarının ortalama hızı bir sonraki öğrenme adımının hızını sınırlamak için kullanılabilir. Bu yöntem ile üstel olarak azalan geçmişteki gradyanların dinamik ortalaması tutulur ve bu dinamik ortalamalar hesaba katılarak o yönde bir ilerleme sağlanır [1]. Bu sayede öğrenme adımları en iyi sonuca doğru daha hızlı hareket eder ve daha az sapma olur.

- **RMS-Prop (Root Mean Square Propagation):** *RMS-Prop* yöntemi, Geoffrey Hinton'ın Coursera'da verdiği derste önerdiği bir optimizasyon yöntemidir [16]. Bu yöntem uyarlanabilir öğrenme oranı kullanan algoritmalar alt kümesine girmektedir. Yani öğrenme adımları boyunca her bir parametre için ayrı bir öğrenme oranı kullanılır ve bu öğrenme oranları elde edilen sonuçlara göre uyarlanabilir, güncellenebilir. Bu yöntem üstel olarak azalış gösteren ortalama yaklaşımını kullanarak, uzak geçmişteki noktaları dikkate almaz. Bu sayede yakınsamayı hızlandırabilir [1]. *Adadelta* optimizasyon yönteminde, parametrelerin kendi öğrenme hızları vardır. Bu öğrenme hızları giderek azalmaktadır ve sistem belirli bir noktadan sonra öğrenme işlemini gerçekleştiremez. *RMS-Prop* yöntemi bu sorunu çözmek amacıyla önerilmiştir.

- **Adam (Adaptive Moment Estimation):** Adam, klasik *stokastik gradyan azaltma* yöntemi yerine kullanılabilecek daha verimli, adaptif bir optimizasyon algoritmasıdır. Yani her bir parametre için dinamik bir şekilde öğrenme oranını (learning rate) günceller [14,16]. Hesaplama yükü olarak verimlidir ve düşük bellek gereksinimlerine ihtiyaç duyar. *Adam, RMS-Prop* ve *Momentum* ile birlikte gradyan düşürme yöntemlerine benzer bir parametre güncelleme metodu kullanmaktadır.

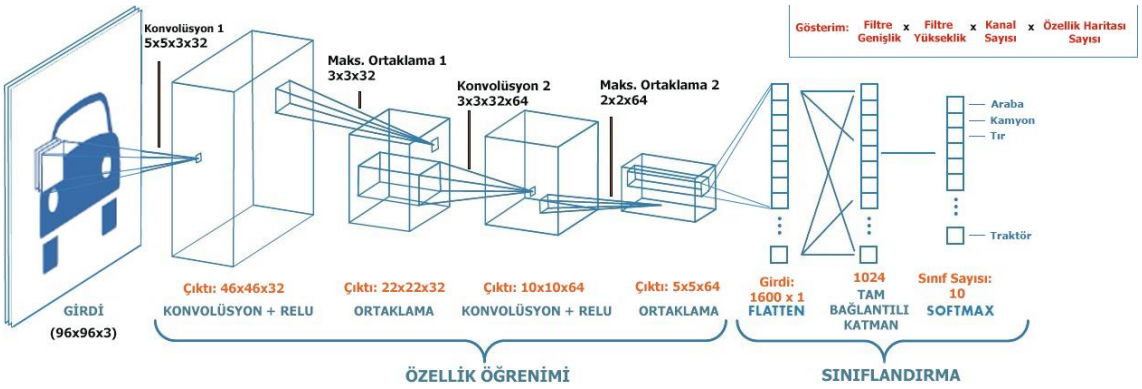
3. KONVOLÜSYONEL SİNİR AĞLARI

Konvolüsyonel sinir ağları (KSA) bilgisayarlı görüntü tanıma çalışmalarında sıklıkla kullanılan ve çok başarılı sonuçlar veren bir derin öğrenme yöntemidir. Çok katmanlı YSA'dan farklı olarak KSA'da, katmanlarının en az bir tanesinde matris çarpımı yerine konvolüsyon (filtrelerin girdi üzerinde özellik çıkarımı için dolaşması) işlemi kullanılmaktadır [1,18,19]. İlk katmanlarda verilen girdi üzerinde, filtreler yoluyla özellik çıkarımı gerçekleştirilir. Aynı zamanda bir yandan hesaplama maliyetini düşürmek amacıyla diğer yandan ise girdiden öğrenilen özelliklerin özet bilgisini diğer katmanlara aktarmak amacıyla boyut düşürme fonksiyonları kullanılır. Daha sonra girdiden elde edilen bu özellikler tek boyutlu bir vektör haline getirilir ve tam bağlantılı katman veya katmanlara girdi olarak verilip, sınıflandırma işlemi gerçekleştirilir. Şekil 3'te gösterildiği gibi

96*96*3 boyutlu (3:kanal) bir resim girdisi, 5*5*3 boyutlu 32 adet filtreden geçirildikten sonra 46*46*32 boyutuna indirilmiştir, ancak 32 filtre ile 32 ayrı özellik çıkarılmıştır.

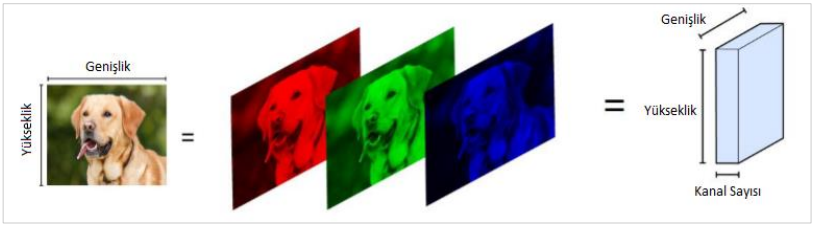
3.1 Konvolüsyonel Sinir Ağları Yapısı

Girdi Katmanı: Resim sayısal ifadelerden oluşan matrise dönüştürüldükten sonra girdi katmanına verilebilir. Resmin özelliklerine göre matris boyutları değişebilmektedir. Örneğin, 28x28 gri tonlamalı bir resim 28x28x1 boyutunda matrisle gösterilebilirken; aynı boyutlardaki renkli resim için RGB (Red, Green, Blue) kanallarından dolayı 28x28x3 boyutunda bir matris gereklidir (Şekil 4).



Şekil 3. *Konvolüsyonel Sinir Ağlarının Yapısı*

Konvolüsyon Katmanı: Bu katmanda önceki katmandan gelen girdiler üzerinde belirlenen filtreler uygulanarak yeni özellik setleri çıkarılmaya çalışılır. Uygulanan her bir filtre belirtilen boyutta matrisler ile temsil edilir. Girdi üzerine uygulanan her filtre farklı bir özelliği ortaya çıkarmaya çalışır. Filtrelerin boyutlarının seçimi çok önemlidir. Zira çok büyük boyutlardaki filtreler girdi üzerinde bazı özelliklerin kaçırılmasına neden olabilir. Filtrelerin katsayıları, yapay sinir ağlarında bulunan ağırlıkları temsil eder ve her bir adımda hata değerine göre bu filtre katsayıları güncellenerek yeni filtreler elde edilir.



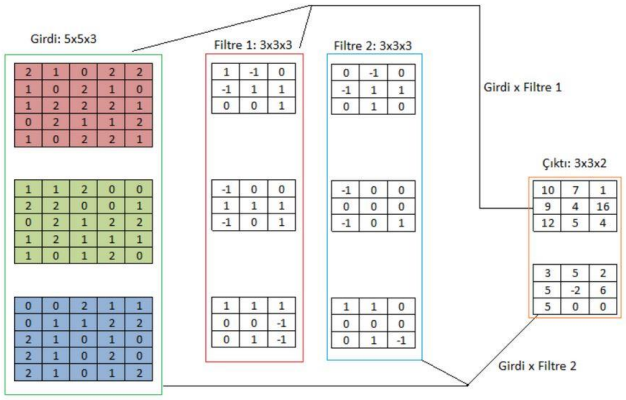
Şekil *[WK1]*[KM2]***4.*** *Renkli (3 kanallı) Resmin Gösterimi*

Girdi üzerinde her bir filtre sıra ile tüm pikselleri dolaşır. Girdi katsayıları ile filtre katsayıları noktasal çarpılır ve bu işlem girdinin her bir kanalı için gerçekleştirilir (Şekil 5). Denklem 4' te gösterildiği gibi girdi matrisi ve ağırlık matrislerinin çarpılmasıyla özellik haritaları oluşmaktadır. $W^{[i-1]}$, i-1. katmanda uygulanan her bir filtrenin değerlerinin tutulduğu matrisi; $Z^{[i]}$ ise ağırlık ve girdi matrislerinin işleme sokulması sonucunda oluşan yeni özellik haritalarının tutulduğu matrisi göstermektedir.

$$Z^{[i]} = X^{[i]}W^{[i-1]} + b^{[i-1]} \quad (4)$$

Ortaklama Katmanı: Ortaklama (pooling) işlemi özellik boyutunu düşürmek için kullanılır. Yükseklik ve genişlik değerlerinin düşürülmesi özellik kaybına neden olsa da sinir ağının eğitilmesini hızlandırır. Ortaklama işleminde öğrenilecek parametre yoktur; sabit ortaklama boyutu ve aralık (stride) hiper-parametreleri vardır. Ortaklama işlemi için yaygın olarak maksimum ortaklama (max pooling) ve ortalama ortaklama (average pooling) yöntemleri kullanılmaktadır. Maksimum ortaklama yönteminde, belirli bir aralık değerine (stride) göre ortaklama filtresi girdi üzerinde dolaşır ve girdi üzerinde filtrenin bulunduğu alana karşılık gelen kısım içerisindeki maksimum değer, çıktıda karşılık gelen alana yerleştirilir.

Tam Bağlantılı Katman: Konvolüsyonel sinir ağlarının son katmanlarını oluşturur. Tam bağlantılı katmana gelene kadar olan katmanlarda girdiler için özellik çıkarımı, boyut düşürme ve normalizasyon işlemleri gerçekleştirilmiş olur. Elde edilen çıktıya göre hata hesaplanır. Hesaplanan hata değerine göre ağırlıklar tekrar güncellenir ve bu çevrim istenen yakınsama değeri elde edilene ya da belirli bir adım sayısı tamamlanana kadar devam eder.



Şekil 5. *Konvolüsyon İşlemin Uygulanması ve Çıktılar (Aralık = 1, Dolgu = 0)[WK3]*

4. KONVOLÜSYONEL SİNİR AĞLARINDA HİPER-PARAMETRE OPTİMİZASYONU

Konvolüsyon katmanı sayısı, her bir katmanda kullanılan filtre sayısı, filtre boyutu, aralık ve dolgu değeri, tam bağlantılı katman sayısı ve her katmandaki nöron sayısı gibi hiper-parametreler KSA'ların başarısını doğrudan etkilemektedirler. Ancak optimize edilmek istenen hiper-parametreler için tanımlanan değerlerden oluşan arama uzayı çok geniş olabileceğinden bütün seçenekleri denemek, hesaplama zamanı açısından mümkün olmamaktadır. Bu nedenle genellikle genetik algoritma, parçacık sürü optimizasyonu ve diferansiyel gelişim algoritmaları gibi üst-sezgisel yöntemlerden yararlanılır.

Literatürde, KSA'nın eğitilmesi ve test edilmesi için kullanılan, boyut, örnek sayısı ve sınıf sayısı açısından farklı zorluklarda veri kümeleri bulunmaktadır. Bu veri kümeleri ve özellikleri Tablo 1'de gösterilmektedir.

Tablo 1. *Veri kümeleri ve özellikleri*

Veri Seti	Örnek Sayısı	Sınıf Sayısı	Görüntü Boyutları
Caltech-101 [20, 21]	9,146	101	300x200
Cancer Genome Atlas – miRNASeq [22]	8,129	29	-
CIFAR10 [23]	60,000	10	32x32
CIFAR100 [23]	60,000	100	32x32
Convex Set [24]	13,000	2	28x28
EMNIST Balanced [25]	131,600	47	28x28
EMNIST Digits [25]	280,000	10	28x28
EMNIST Letters [25]	145,600	26	28x28
Fashion-MNIST [26]	70,000	10	28x28
Human Sketches Dataset [27]	20,000	250	90x250
LIDC-IDRI [28]	1018	2	-
MNIST [29]	70,000	10	28x28
MNIST Random Background (MRB) [24]	17,000	10	28x28
MNIST Rotated Digits (MRD) [24]	17,000	10	28x28
MNIST with Background Images (MBI) [24]	17,000	10	28x28
MNIST RD plus Background Images (MRDBI) [24]	17,000	10	28x28
STL-10 [30]	13,000	10	96x96
UCF-50 [31]	-	50	-

KSA parametreleri, konvolüsyon katmanı parametreleri, ortaklama katmanı parametreleri, tam bağlantılı katman parametreleri ve ortak parametreler olarak gruplandırılabilir. Konvolüsyon katmanında filtre sayısı ve boyutu, dolgu boyutu gibi parametreler; ortaklama katmanında filtre boyutu, aralık değeri ve ortaklama algoritması gibi parametreler; tam bağlantılı katmanda nöron sayısı ve katman sayısı gibi parametreler bulunmaktadır. Öğrenme oranı, optimizasyon yöntemi, ağırlık düşürücü, ağırlık başlatıcı ve seyrekleştirme oranı gibi parametreler ortak parametreler arasındadır. Derin öğrenme literatüründe çoğunlukla konvolüsyon katmanı ile ortaklama katmanı tek bir konvolüsyon katmanı olarak kabul edilmektedir. Ancak her bir katmanda optimize edilebilecek parametreleri açıkça belirtebilmek açısından ortaklama katmanı Tablo 2'de ayrı bir kolonda gösterilmektedir.

Tablo 2. *KSA'da optimize edilebilecek hiper-parametreler*

Konvolüsyon	**Ortaklama**	**Tam Bağlantılı**	**Ortak**
Filtre boyutu – ***k***	*Filtre boyutu* – ***k***	Nöron sayısı - ***n***	*Optimizasyon metodu* - ***o*****
Filtre sayısı - ***c***	*Aralık (stride)* - ***s***	*Katman sayısı* - $\boldsymbol{l_t}$	*Batch boyutu* - ***ba***
Aralık (stride) - ***s***	Ortaklama alg. - ***pa***	*Başlangıç katman sayısı* - $\boldsymbol{l_{ti}}$	*Öğrenme oranı* - $\boldsymbol{\alpha}$
Dolgu (padding) - ***p***	*Katman sayısı* - $\boldsymbol{l_o}$		*Başlangıç öğrenme oranı* - $\boldsymbol{\alpha_i}$
Aktivasyon fonksiyonu - ***a****			*Seyreltme oranı (dropout)* - ***d***
Katman sayısı - $\boldsymbol{l_k}$			*Seyreltme aktif* - ***de***
Başlangıç katman sayısı - $\boldsymbol{l_{ki}}$			*Düzenlileştirme metodu* - ***r***
			L2, L1 düzenlileştirme katsayısı - $\boldsymbol{\lambda_{L2}}$, $\boldsymbol{\lambda_{L1}}$
			Ağırlık başlatıcı (weight initializer) – ***wi***
			Ağırlık düşürücü – ***wd***
			Ağırlık çarpanı – ***wm***
			Ağırlık normalizasyonu – ***wn***
			Ağırlık ceza değeri - ***wp***
			İterasyon sayısı - ***it***
			Momentum – ***m***
			Yanlılık aktif (bias) – ***b***
			Yanlılık başlangıç - $\boldsymbol{b_i}$
			Yanlılık başlangıç katsayısı – $\boldsymbol{b_{ic}}$
			Gauss katsayısı - ***gc***

* $\boldsymbol{a_i}$: *A_1: Trelu, A_2: Elu, A_3: Prelu, A_4: Leaky-relu, A_5: Relu, A_6: Softmax, A_7: Sigmoid, A_8: Lineer, A_9: tanh, A_{10}: selu*

** $\boldsymbol{o_i}$: *O_1: Sgd, O_2: Nesterov, O_3: Momentum, O_4: Adamax, O_5: Adagrad, O_6: Adam, O_7: Adadelta, O_8: Rms-prop*

4.1 Hiper-parametre Optimizasyonu için Kullanılan Üst-sezgiseller

KSA hiper-parametre optimizasyonu çalışmalarında en çok kullanılan üst-sezgiseller, *Genetik Algoritma (Genetic Algorithm–GA), Parçacık Sürü Optimizasyonu (Particle Swarm Optimization – PSO), Diferansiyel Gelişim (Differantial Evolution – DE)* ve *Harmonik Arama*'dır *(Harmonic Search – HA)*. Bu bölümde, bu üst-sezgisel metotların kullanıldığı çalışmalarda optimize edilen hiper-parametreler ve bu hiper-parametreler için tanımlanmış değer aralıkları bilgileri sunulmaktadır. İncelenen çalışmaların bütünü görüntü sınıflandırma problemi için gerçekleştirilen çalışmalardır.

4.1.1 Genetik Algoritmalar

John Holland'ın temellerini geliştirdiği ve "Adaptation in Natural and Artifical Systems" kitabında anlattığı [32] *genetik algoritmalar (GA)* kısaca, en iyinin hayatta kalması ilkesine dayanarak çözüm uzayında en iyi çözümü bulmaya çalışan arama yöntemidir [33, 34]. İlk olarak her biri bir çözümü temsil eden rastgele oluşturulan bireylerden meydana gelen bir başlangıç popülasyonu ile çalışmaya başlanır; her bir iterasyonda mevcut jenerasyondan seçilen bireylerin çaprazlanmasıyla ve mutasyona uğratılmasıyla yeni bir popülasyon elde edilir ve bu yeni jenerasyon öncekinin yerini alır. Her yeni jenerasyonda daha iyi genetik özelliklere sahip bireylerden oluşan bir popülasyonun oluşması umut edilir. Çaprazlama ve mutasyon işlemleri bir çaprazlama ve mutasyon olasılığına göre yapılır (P_{cx} ve P_m). Ayrıca, popülasyon büyüklüğü, çözüm gösterim yöntemi ve iterasyon sayısı gibi parametrelerin doğru seçilmesi önemlidir.

GA ile yapılan hiper-parametre optimizasyonu çalışmalarından bazılarında sabit bir KSA mimarisi kabul edilirken bazı çalışmalarda dinamik olarak genişleyebilen bir mimari kabul edilmiştir. Tablo 3'te optimize

edilen hiper-parametreler ve parametre değer aralıkları gösterilmektedir; Tablo 4' te ise GA parametrelerinin belirtildiği çalışmalar için seçilen parametre değerleri gösterilmektedir.

Tablo 3. *GA çalışmalarında optimize edilen hiper-parametreler ve değer aralıkları*

Referans	Konvolüsyon	Ortaklama	Tam Bağlantılı	Ortak
Dufourq (2017) [38]	***k:*** [1, 6] ***c:*** [10, 100] ***a:*** A_8, A_4, A_3, A_5	***k:*** [1, 6]	***n:*** [10, 100] ***a:*** [A_8, A_7, A_6, A_5]	***d:*** 0, 1
Bochinski (2017) [40]	l_k***:*** [0, 6] ***k:*** [1, 8] ***c:*** [1, 128] l_{ki}***:*** [2, 4]	-	l_t***:*** [0, 4] ***n:*** [16, 2048] l_{ti}***:*** [1, 2]	-
Fujino (2017) [41]	***k:*** 3, 5, 7, 9, 11 ***c:*** 16, 32, 64	***k:*** 3, 5, 7	n_1***:*** 2^9, 2^{10}, 2^{11}, 2^{12} n_2***:*** 2^7, 2^8, 2^9, 2^{10}	***ba:*** 10, 20,30 ***a:*** relu, no relu
Rincon (2018) [37]	***k:*** [4, 64] ***c:*** [2, 256]	***k:*** [4, 64]	-	-
Ma (2018) [43]	***k:*** 3, 5, 7 ***c:*** [16, 512]	***pa:*** yok, max, avg	***n:*** [16, 512]	***a:*** A_1,..,A_6 ***d:*** [0, 0.5] ***o:*** O_1, O_4, O_5,..,O_8
Assunçao (2018) [44]	***b:*** true, false ***k:*** 1, 3, 5 ***c:*** [32, 256] ***p:*** same, valid ***s:*** [1, 3]	***k:*** [1, 5] ***pa:*** max, avg ***s:*** [1, 3]	***b:*** true, false ***n:*** [128, 2048]	***a:*** A_5, A_7, A_8 ***α:*** [10^{-4}, 10^{-1}]
Baldominos (2018) [45]	***a:*** A_5, A_8 ***k:*** [2, 9] ***c:*** 8, 16, 32, 64, 128, 256 l_k***:*** [1, 4]	***k:*** [2, 6]	***a:*** A_5, A_8 ***d:*** 0, 0.5 l_t***:*** 1, 2 ***n:*** 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10} ***r:*** null, L1, L2	***ba:*** 25, 50, 100 ***α:*** 5E-1, 1E-1, 5E-2, 1E-2, 5E-3, 1E-3 ***o:*** O_1,..,O_8

- : *Katman için hiper-parametre optimizasyonu gerçekleştirilmemiş*

Sabit bir KSA topolojisinin kabul edildiği çalışmalarda başlangıç ağırlıkları, her bir katmandaki filtre boyutu ve filtre sayısı gibi parametreler optimize edilmeye çalışılmıştır. İjjina ve arkadaşları [35] tam bağlantılı katmanda başlangıç ağırlıklarının oluşturulacağı "seed" değerini ve konvolüsyon katmanı ağırlıklarının başlangıç değerlerini GA ile optimize etmeye çalışmışlardır. Kromozomlar reel sayı tipinde toplam 64 gen ile temsil edilmiştir; 63 gen konvolüsyon katmanındaki ağırlıkları, 1 gen ise seed değerini temsil etmek için kullanılmıştır. Ağırlıklarının GA ile başlatılarak eğitilen modelin diğer modellerden daha iyi sonuç verdiği gösterilmiştir. Silva ve arkadaşlarının [36] akciğer nodüllerini sınıflandırmak amacıyla yaptıkları çalışmada sabit topolojili bir KSA kullanılarak her katmandaki filtre sayısı ve nöron sayısı parametreleri GA ile optimize edilmiştir. Tam sayı tipinde 3 gen ile gösterilen kromozomlarda her bir gen sırasıyla ilk konvolüsyon katmanı filtre sayısını, ikinci konvolüsyon katmanı filtre sayısını ve gizli katman nöron sayısını temsil etmek için kullanılmıştır. Önerilen modelin diğer çalışmalardan daha iyi duyarlılık değeri verdiği gösterilmiştir. Rincon ve arkadaşları [37] tarafından önerilen çalışmada gene sabit bir KSA mimarisi için filtrelerin boyut ve sayıları optimize edilmeye çalışılmıştır. Medikal veri setleri üzerinde test edilen metot ile diğer makine öğrenmesi algoritmalarından daha iyi sonuçlar elde edilmiştir.

Dinamik olarak genişleyebilen bir KSA mimarisinin kabul edildiği çalışmalarda kromozomlar da değişken boyutlarda seçilmektedir. Bu çalışmalarda ağın kontrolsüzce büyümesini engellemek amacıyla katman sayısı değeri kısıtlı bir aralıktan seçilmektedir (Tablo 3). Dufourq ve arkadaşları [38] her bir konvolüsyon katmanındaki filtre boyutu ve sayısı ile aktivasyon fonksiyonunun yanında konvolüsyon katmanı sayısını da optimize etmeye çalışmışlardır. Elde edilen sonuçlar bazı veri kümeleri için literatürdeki diğer KSA ile benzer olsa da bazı veri kümeleri için daha kötü sonuçlar elde edilmiştir. Ancak üretilen modeller literatürde temel alınan modellere göre daha basit olduğundan çok daha makul bir hesaplama gücü ve süresi içinde eğitilebilmişlerdir. Sun ve arkadaşları [39] GA kullanarak elde ettikleri basit modeller ile karmaşık mimarilere karşı rekabetçi sonuçlar elde etmişlerdir. Bochinski ve arkadaşlarının [40] önerdikleri çalışmada gene sabit olmayan bir topoloji kullanılmıştır. Kullanılan evrimsel algoritma tabanlı yaklaşımda her bir KSA'nın uygunluk değeri o bireyin doğruluk değerinin bir KSA komitesindeki bireylerin değerleriyle kıyaslanmasıyla bulunmaktadır. Bu yaklaşımın çok başarılı sonuçlar verdiği gösterilmiştir. Fujino ve arkadaşlarının [41] el

çizimlerin sınıflandırılması çalışmalarında, *AlexNet* mimarisi [42] temel alınarak GA ile optimize edilen bir KSA önerilmiştir. Ma ve arkadaşları [43] tarafından önerilen çalışma bilinen resim tanıma problemleri ile test edildiğinde "state-of-the-art" mimarilere karşı başarılı sonuçlar vermiştir. Assunçao ve arkadaşları da [44] evrimsel algoritmalar ile otomatik olarak oluşturdukları KSA ile bilinen mimarilerle yarışabilir sonuçlar elde etmişlerdir. Baldominos ve arkadaşlarının [45] önerdiği evrimsel algoritma yönteminde ise üretilen her bireye daha önceki bireylere benzeme derecesine göre bir ceza puanı verilerek sürekli aynı bireylerin elde edilmesi engellenerek genetik çeşitlilik oluşturulmaya çalışılmıştır. Bu yöntemin genetik algoritmanın başarısını artırdığını ortaya koymuşlardır.

Tablo 4. *GA ile KSA optimizasyonu çalışmalarında GA için sabit seçilen parametreler*

Referans	Popülasyon Boyutu	P_{cx}	P_m	Jenerasyon Sayısı
Ijjina (2016) [35]	20	0.8	0.01	5
Silva (2017) (2017) [36]	6	0.7	0.3	100
Dufourq (2017) [38]	100	-	-	10
Sun (2017) [39]	100	0.9	0.1	100
Bochinski (2017) [40]	5	-	0.3	50
Fujino (2017) [41]	20	1.0	-	20
Rincon (2018) [37]	50	-	0.1	-
Ma (2018) [43]	-	-	-	10
Assunçao (2018) [44]	100	0.7	0.3	100
Baldominos (2018) [45]	50	0.7	0.015	100

- : *Parametrenin aldığı değer belirtilmemiş*

4.1.2 Parçacık Sürü Optimizasyonu

Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO), 1995 yılında Dr. Eberhart ve Dr. Kennedy tarafından ortaya atılmış, kuş veya balık sürülerinin sosyal davranışlarına dayanarak geliştirilmiş olan bir optimizasyon yöntemidir [46, 47]. PSO temel olarak, bireylerin en uzun süre hayatta kalması için yaptıkları davranışları taklit eder. Her bir birey birer parçacık olarak düşünülürse, her parçacık kendisinin daha önce verdiği karar ve grubun kararları doğrultusunda hareket eder. Kararının sonucunda oluşan durumlara göre performansını değerlendirir ve bir önceki adımıyla kıyaslayarak kendisi için en iyi kararı seçmeye çalışır. PSO ile KSA optimizasyonu yapan çalışmalarda optimize edilen hiper-parametreler ve tanımlanan değer aralıkları Tablo 5'te özet olarak verilmiştir.

Yamasaki ve arkadaşları [48] sabit *AlexNet* mimarisi üzerinde filtre sayısı, boyutu, ortaklama tipi gibi parametreleri optimize etmeye çalışmışlardır. Elde edilen en iyi sonuçlar CIFAR10, CIFAR100, Subset10, Subset30, Subset50 veri setleri için sırasıyla karşılaştırıldığında önerilen yöntem, orijinal *AlexNet* mimarisinden +2.40%, +3.41%, +2.20%, +5.74% ve +0.72% daha iyi doğruluk değerleri elde etmiştir. Ancak önerilen yöntem, orijinal *AlexNet* mimarisinden 2% - 4% daha fazla hesaplama maliyetine neden olmuştur. Silva ve arkadaşları da [49] sabit mimarili bir KSA'yı PSO tabanlı bir yöntem ile optimize etmeye çalışmışlardır. Elde dilen ağ, tıbbi bir veri seti (LIDC-IDRI) üzerinde test edilmiştir.

Dinamik ağ modelinin kabul edildiği PSO ile KSA hiper-parametre optimizasyonu çalışmalarından, Lorenzo ve arkadaşları [50] tarafından önerilen yöntem SimpleNet-1, SimpleNet-2 ve LeNet-4 topolojisinin optimize edilmesi için kullanılmıştır. PSO algoritmasının diğer hiper-parametre seçim algoritmalarından 0.0026 daha kötü doğruluk değeri vermesine rağmen 94 kat daha hızlı olduğu öne sürülmüştür. Lorenzo ve arkadaşları daha sonra [51] PSO' nun KSA hiper-parametre optimizasyonu problemi için hem teorik hem pratik açıdan yakınsama analizini sunmuşlardır. Sun ve arkadaşlarının [52] önerdikleri dinamik KSA mimarisinde elde edilen ağlar CIFAR10, MNIST, STL-10 ve Caltech-101 veri setleri için değerlendirilmiş ve "state of the art" sonuçlara eş değerler elde edilmiştir. Wang ve arkadaşlarının [53] önerdikleri çalışmada yeni bir çözüm gösterim şekli sunulmuştur; bu yeni gösterimin PSO'nun performansını artırdığı gösterilmiştir.

4.1.3 Diferansiyel Gelişim

Popülasyon tabanlı bir optimizasyon yöntemi olan *diferansiyel gelişim* (DE) algoritması, 1995 yılında Price ve Storn [54] tarafından önerilmiştir. GA ile benzerlik gösteren bu yöntemde rastgele seçilen 3 bireyin çaprazlanması ve mutasyona uğratılmasıyla elde edilen yeni birey hedef birey ile karşılaştırılır. Daha iyi değere sahip birey bir sonraki jenerasyona aktarılır [55].

Tablo 5. *PSO, DE ve diğer yaklaşımlarda optimize edilen hiper-parametreler ve değer aralıkları*

	Referans	Konvolüsyon	Ortaklama	Tam Bağlantılı	Ortak
PSO	Yamasaki (2017) [48]	***k:*** [2, 7] ***c:*** [50, 180] ***p:*** [0, 7]	***k:*** [2, 7] ***p:*** max, avg	-	-
	Lorenzo (2017) [50]	***k:*** [2, -] ***c:*** [2, -]	***k:*** [2, -] ***s:*** [2, -]	-	-
	Sun (2017) [52]	***k:*** [2, 5] ***c:*** [20, 100] λ_{L2}***:*** [10^{-4}, 10^{-2}]	***k:*** [2, 5] ***pa:*** max, avg	-	-
	Silva (2018) [49]	***k:*** 3, 5, 7 ***c:*** [4, 100]	***pa:*** max, avg	l_i***:*** [4, 100]	***ba:*** [10, 100] ***d:*** [0.1, 0.99]
	Lorenzo (2018) [51]	***k:*** [2, -] ***c:*** [1, -]	***k:*** [2, -] ***s:*** [2, -]	-	-
	Wang (2018) [53]	***k:*** [1, 8] ***c:*** [1, 128] ***s:*** [1, 4]	***k:*** [1, 4] ***pa:*** max, avg ***s:*** [1, 4]	***n:*** [1, 2048]	-
DE	Wang (2018) [56]	***k:*** [1, 8] ***c:*** [1, 128] ***s:*** [1, 4]	***k:*** [1, 4] ***pa:*** max, avg ***s:*** [1, 4]	***n:*** [1, 2048]	-
Diğer Yaklaşımlar	Bengio (2012) [61] (Random Search vs Deep Belief Network)	-	-	***n:*** [128, 4000]	***wi:*** uniform, normal ***it:*** [1, 10^4] α_i***:*** [10^{-4}, 1] λ_{L2}***:*** [10^{-7}, 10^{-4}]
	Domhan (2015)[62]	***wm:*** [0.1, 10] ***gc:*** [10^{-6}, 10^{-1}] ***c:*** [16-64, 64-192]	***l:*** 2, 3	***wd:*** [0.9, 1] b_i***:*** 0, sabit b_{ic}***:*** [0, 1] ***n:*** [128, 6144] ***d:*** [0.05, 0.95] ***gc:*** [10^{-6}, 10^{-1}] ***de:*** true, false ***wi:*** gauss, xavier	***l:*** [1, 6] α_i***:*** [10-7, 0.5] ***m:*** [0, 0.99] ***wd:*** [5x10^{-7}, 0.05] ***ba:*** [10, 1000] ***de:*** true, false ***d:*** [0.05, 0.8]
	Neary (2018)[64]	l_k***:*** 2 ***k:*** 2, 3 ***c:*** [32, 72]	-	l_i***:*** [2, 4] ***n:*** [512, 1280]	-
	van Stein (2018)[65]	***k:*** [1, 8] ***c:*** [1, 512] ***s:*** [1, 5]	-	-	***l:*** [1, 6] ***d:*** [10^{-5}, 0.8] λ_{L2}***:*** [10^{-5}, 10-2] α***:*** [10^{-5}, 1] ***a:*** A_2, A_5, A_7, A_9, A_{10}

- : *Katman için hiper-parametre optimizasyonu gerçekleştirilmemiş*

DE kullanılarak KSA optimizasyonu yapılan çalışmaların sayısı oldukça kısıtlıdır. Literatürde bulabildiğimiz kadarıyla sadece Wang ve arkadaşları [56] tarafından sunulan çalışmada, DE kullanılarak dinamik olarak oluşturulan bir KSA için hiper-parametre optimizasyonu gerçekleştirilmiştir (Tablo 5). Bu çalışmada PSO ve DE yöntemleri karşılaştırılmıştır; DE tabanlı yöntemin daha iyi sonuçlar ürettiği gösterilmiştir.

4.1.4 Harmonik Arama

Harmonik arama, caz müzisyenlerinin doğaçlamalarından ilham alınarak geliştirilen, daha az adımda daha iyi sonuçlar elde etmeyi amaçlayan optimizasyon algoritmasıdır. Her müzisyen, çözüm uzayındaki herhangi bir çözümün bir özelliğine karşılık gelir ve her bir cihazın perde aralığı, karar değişkenindeki sınırlara karşılık gelir. Müzisyenler arasındaki uyum, belirli bir zamanda bir aday çözüm olarak alınır. Amaç fonksiyonu değeri

ise seyircilerin değerlendirdiği estetik uyumudur. Müzisyenler, zaman içinde küçük değişiklikler ve doğaçlamalarla daha iyi estetik uyumu aramaktadırlar. [57].

Lee ve arkadaşlarının [58] çalışmalarında veri kümelerine göre farklı boyutlarda KSA seçilmiş ve konvolüsyon katmanında filtre boyutu, filtre sayısı ve aralık değeri; ortaklama katmanında filtre boyutu ve aralık değeri optimize edilmiştir.

4.1.5 Benzetimli Tavlama

Temeli termodinamiğe dayanan *benzetimli tavlama* yönteminde metalin soğutulması sırasında yaşanan enerji dalgalanmaları örnek alınmıştır. Sistemin çok hızlı bir şekilde soğutulmasında veya ısıtılmasında ortaya çıkan yerel enerji minimumuna takılma riski başlangıçta enerji dalgalanmalarına izin verilerek en aza indirilmeye çalışılmaktadır [59].

Rasdi ve arkadaşlarının [60] KSA hiper-parametrelerinin optimizasyonu için önerdikleri çalışmada SA, HA ve DE üst-sezgiselleri karşılaştırılmıştır. Diğer çalışmalardan farklı olarak ağırlıklar ve yanlılık değerleri de hiper-parametre optimizasyonu için seçilmiştir. Önerilen çalışmada *LeNet-5* mimarisinin farklı değerlerden oluşan varyasyonları kullanılmıştır ve çalışma MNIST ve CIFAR-10 veri setlerinde test edilmiştir.

4.2 Diğer Yaklaşımlar

KSA hiper-parametrelerinin optimize edilmesi için üst-sezgisel algoritmalar dışında *Izgara Arama (Grid Search)*, *Rastgele Arama* ve *Bayes Optimizasyonu* gibi yöntemler de kullanılmaktadır.

Bengio ve arkadaşlarının *Sıralı Manuel Optimizasyon (Sequential Manual Optimization)*, *Izgara Arama* ve *Rastgele Arama* yöntemlerinin karşılaştırıldığı çalışmalarında, oluşturulan KSA topolojileri Mnist, Mnist Background Images, Mnist Background Random, Mnist Rotated, Mnist Rotated Background Images, Convex, Rectangles ve Rectangle Images veri kümeleri üzerinde test edilmiştir. Buna göre, 7 hiper-parametrenin optimize edildiği durumda *rastgele arama* yöntemi *ızgara arama* metodundan daha az deneme sonucunda daha iyi doğruluk değerleri vermiştir. 32 hiper-parametrenin optimize edildiği durumda *rastgele arama* yöntemi, *Derin İnanç Ağı (Deep Belief Network)* yöntemiyle benzer sonuçlar vermiştir [61]. Domhan ve arkadaşlarının [62] önerdikleri çalışmada *Sequential Model-based Algorithm Configuration (SMAC), Tree Parzen Estimator (TPE)* ve *Rastgele Arama* yöntemleri, bütün hiper-parametrelerin optimize edildiği büyük bir KSA ve daha küçük bir KSA için karşılaştırılmıştır (Tablo 7). Saranyaraj ve arkadaşlarının [63] önerdikleri çalışmada sabit mimarili bir KSA kabul edilmiş; diğer çalışmalardan farklı olarak girdi katmanında kullanılan girdi boyutları da optimizasyona dahil edilmiştir. Girdi boyutu ilk olarak 200'den 100'e düşürülmüştür ve seçilen veri seti için 90.27% doğruluk oranı elde edilmiştir. Bir sonraki adımda girdi boyutu 200'den 300'e çıkarılmıştır ve 92.45% doğruluk oranı elde edilmiştir (Tablo 6). Oluşturulan topolojilerin test edilmesi için göğüs kanserine ait görüntüleri içeren *"Digital Database for Screening Mammography"* veri seti kullanılmıştır. Bu çalışmalarda optimize edilmek için seçilen hiper-parametreler ve değer aralıkları Tablo 5'te gösterilmektedir.

***Tablo 6**. Çalışmada kullanılan hiper-parametreler ve değer aralıkları (Saranyaraj (2018) [63])*

KSA Blok	Hiper-Parametre	Değer Aralığı	Doğruluk Oranı
Konvolüsyon Katmanı	c_1	128 -> 64	94.34%
	c_1 ve c_2	128 -> 64	88.68%
	k_1 ve k_2	5 -> 3	88.68%
	k_1 ve k_2	5 -> 7	92.45%
Tam Bağlantılı Katman	n	256 -> 128	90.57%

Neary ve arkadaşlarının [64] *Pekiştirmeli Öğrenme (Reinforcement Learning – RL)* yöntemi kullandıkları çalışmada belirli bir problem için, en uygun ağ konfigürasyonu otomatik olarak oluşturulmaya çalışılmıştır. Van Stein ve arkadaşlarının [65] çalışmalarında *Verimli Global Optimizasyon (Efficient Global Optimization - EGO)* yöntemi kullanılmıştır. Her bir iterasyonda sadece tek bir aday konfigürasyonun değerlendirildiği bir yaklaşım olan standart EGO algoritması, yapılan çalışmada her bir iterasyonda paralel olarak birden fazla

konfigürasyonun değerlendirilebildiği şekilde uyarlanmıştır. Hinz ve arkadaşlarının [66] çalışmalarında *Sequential Model-based Algorithm Configuration (SMAC), Tree Parzen Estimator (TPE), Rastgele Arama* ve GA yöntemleri karşılaştırılmış; oluşturulan topolojiler Cohn-Kanade, STL-10 ve Flowers-102 veri kümeleri üzerinde test edilmiştir. Buna göre, hızlı hesaplama süreleri için TPE ve SMAC yöntemlerinin; daha iyi doğruluk değerleri için ise GA ve *rastgele arama* yöntemlerinin daha uygun olacağı öne sürülmüştür.

4.3 Performans Değerlendirmesi

KSA hiper-parametre optimizasyonu için kullanılan yöntemler arasında performans kıyaslaması yapabilmek amacıyla aynı veri seti kullanılarak yapılan tüm çalışmaların doğruluk oranları Tablo 7'de gösterilmiştir.

Literatürde üzerinde en çok çalışılmış veri kümeleri CIFAR-10, CIFAR-100 ve MNIST veri kümeleridir. CIFAR-10 için en yüksek doğruluğu % 93.37 ile GA vermiş; PSO ile elde edilen en iyi değer % 83.5' te kalmıştır. CIFAR-100 veri seti için en yüksek değeri % 78.75 ile GA verirken diğer yöntemler % 58' in üzerine çıkamamışlardır. MNIST veri kümesi için ise GA, PSO ve EGO yöntemlerinin her biriyle % 99.39' un üzerinde doğruluk değerleri elde edilmiştir. Her ikisi de populasyon temelli yöntemlerden olan GA ve PSO en başarılı sonuçlar veren üst-sezgisellerdendir. Bununla birlikte GA % 99.76 ile bir adım önde görülmektedir. Hiper-parametre optimizasyonu yapılan bu çalışmalarda elde edilen doğruluk değerleri, AlexNet, VGGNet ve ResNet gibi "*state-of-the-art"* mimarilerle [42, 67, 68, 69, 70] elde edilen değerler ile karşılaştırıldığında Tablo 7' de görüldüğü gibi bir iyileştirme olduğu görülmektedir. Tablo 7' de diğer veri kümeleri için elde edilen sonuçlar da gösterilmektedir.

Tablo 7. *Kullanılan veri kümeleri ve elde edilen doğruluk oranları*

Veri Seti	**Referans**	**Kullanılan Metot**	**Doğruluk Oranı (%)**
CIFAR-10	Domhan (2015) [62]	Küçük KSA + TPE	81.88
		Küçük KSA + SMAC	82.53
		Büyük KSA+ SMAC	91.19
	Dufourq (2017) [38]	GA	75.5
	Yamasaki (2017) [48]	PSO	80.15
	Lorenzo (2017) [50]	PSO	59.59
	Sun (2017) [52]	PSO	83.5
	Ma (2018) [43]	GA	89.23
	Assunçao (2018) [44]	GA	**93.37***
	Lorenzo (2018) [51]	PSO	58.47
	Lee (2018) [58]	HA	74.76
	Van Stein (2018) [65]	EGO	86.46
	Lee (2018) [58]	CifarNet	73.32
	Ma (2018) [43]	AlexNet	82.53
	Ma (2018) [43]	VGGNet	84.62
	Ma (2018) [43]	ResNet	90.61**
CIFAR-100	Domhan (2015) [62]	Küçük KSA + SMAC	57.79
	Yamasaki (2017) [48]	PSO	53.28
	Ma (2018) [43]	GA	66.77
	Assunçao (2018) [44]	GA	**78.75***
	Ma (2018) [43]	AlexNet	60.53
	Ma (2018) [43]	VGGNet	64.37
	Ma (2018) [43]	ResNet	67.61**
Convex Set	Sun (2017) [39]	GA	**94.62***
	Wang (2018) [53]	PSO	87.94
Fashion-MNIST	Dufourq (2017) [38]	GA	90.6
	Sun (2017) [39]	GA	92.72
	Ma (2018) [43]	GA	94.59
	Assunçao (2018) [44]	GA	**95.26***
	Assunçao (2018) [44]	AlexNet	89.90
	Assunçao (2018) [44]	VGGNet	93.50
	Assunçao (2018) [44]	ResNet	94.90**
LIDC-IDRI	Silva (2017) [36]	GA	94.78
	Silva (2018) [49]	PSO	**97.62***
MNIST	Sun (2017) [39]	GA	98.72

	Bochinski (2017) [40]	GA	**99.76***
	Lorenzo (2017) [50]	PSO	99.45
	Sun (2017) [52]	PSO	99.34
	Ma (2018) [43]	GA	99.64
	Assunçao (2018) [44]	GA	99.70
	Baldominos (2018) [45]	GA	99.63
	Lorenzo (2018) [51]	PSO	98.82
	Wang (2018) [53]	PSO	98.79
	Wang (2018) [56]	DG	98.24
	Lee (2018) [58]	HA	99.25
	Neary (2018) [64]	RL	95.8
	Van Stein (2018) [65]	EGO	99.39
	Lee (2018) [58]	LeNet-5	98.94
	Ma (2018) [43]	AlexNet	98.81
	Ma (2018) [43]	VGGNet	99.32
	Ma (2018) [43]	ResNet	99.37**
MNIST with Background Images	Sun (2017) [39]	GA	**95.38***
	Wang (2018) [56]	DG	91.31
MNIST Random Background	Sun (2017) [39]	GA	**96.41***
	Wang (2018) [56]	DG	94.44
MNIST RD Plus Background Images	Sun (2017) [39]	GA	62.62
	Wang (2018) [53]	PSO	**67***
	Wang (2018) [56]	DG	62.45
MNIST Rotated Digits	Sun (2017) [39]	GA	**94.54***
	Wang (2018) [56]	PSO	94.47

**** :*** *İlgili veri seti için en iyi doğruluk oranını veren çalışma*
***** :*** *İlgili veri seti için en iyi doğruluk oranını veren "state of the art" model*

Hiper-parametre optimizasyonu ile doğruluk değeri açısından iyi sonuçlar veren çalışmalarda elde edilen KSA'ların hiper-parametrelerine bakıldığında ve aynı zamanda "*state-of-the-art"* mimariler incelendiğinde aşağıdaki yorumlar yapılabilir:

1. Filtre boyutu: KSA'nın derinliği arttıkça girdi boyutlarının küçültüldüğü ve özellik haritası sayılarının arttırıldığı görülmektedir. Bunun için başlangıçta büyük filtre boyutları seçilirken, sonraki katmanlarda küçük filtre boyutları seçilmektedir.

2. Seyreltme oranı: Tam bağlantılı katmanlarda kullanılan seyreltme değerleri katmanlardaki nöron sayıları ile doğru orantılı olarak seçilmelidir. Çok fazla nöron sayısına sahip olan katmanlarda daha yüksek seyreltme değerleri belirlenirken, daha düşük nöron sayısına sahip katmanlarda düşük seyreltme değerleri seçilmelidir.

3. Öğrenme oranı: Adaptif yöntemler kullanılmadığı durumda, öğrenme oranı için başlangıçta daha büyük değerler, sonlara doğru ise daha küçük değerler seçilmektedir. Başlangıçta büyük öğrenme oranlarının seçilmesi eğitim adımlarını hızlandırırken, hedefe doğru yaklaştıkça öğrenme oranının küçültülmesi sınıflandırma başarısını arttıracaktır.

4. Yığın boyutu: Bu değerin çok büyük seçilmesi (1024, 2048 vb.) sınıflandırma başarısını düşürürken, çok küçük seçilmesi de eğitim adımlarını çok yavaşlatacaktır. Bu yüzden kullanılan veri setinin büyüklüğü ve elde edilen başarı oranları göz önüne alınarak 32, 64, 128, 256 gibi orta değerler seçilmelidir.

5. SONUÇ VE DEĞERLENDİRME

KSA ile bilgisayarlı görü çalışmalarında çok başarılı sonuçlar elde edilmektedir. Ancak kurulacak ağdaki katman sayısı, her bir katmandaki filtre boyutu ve sayısı, ortaklama katmanı türü, seyreltme oranı, öğrenme oranı ve optimizasyon metodu gibi hiper-parametrelerinin doğru seçilmesi ağın performansı açısından çok

önemlidir. Ancak bu hiper-parametrelerin optimizasyonu yüksek hesaplama zamanı gerektirdiğinden yapılacak deneylerde parametre değerleri için belirtilen olası tüm ihtimallerin değerlendirilmesi, *ızgara arama* gibi yöntemlerin kullanılması, pratik açıdan mümkün değildir. Bu nedenle hiper-parametre optimizasyonu için *üst-sezgisel yöntemler* yaygın olarak kullanılmaktadır. Hiper-parametre optimizasyonu için en çok kullanılan *üst-sezgisel yöntemler* GA ve PSO' dur. Kullanılan veri setlerine göre değişken doğruluk oranları elde edilse de genel olarak GA ve PSO bu alanda en iyi sonuç veren *üst-sezgisel* yöntemler olarak karşımıza çıkmaktadır. Bu çalışmalarda optimize edilmeye çalışılan hiper-parametrelere bakıldığında bir bütünlük görülmemekte; bir çalışmadan diğerine geçildiğinde farklı hiper-parametrelerin optimize edilmeye çalışıldığı görülmektedir. Aynı zamanda, bazı çalışmalarda sabit bir KSA mimarisi kabul edilirken bazı çalışmalarda dinamik olarak genişleyen bir mimari kullanılmıştır. Bunun yanında, seçilen hiper-parametreler için belirlenen değer aralıkları da bir çalışmadan diğerine değişmektedir.

KSA hiper-parametrelerinin optimize edilmesi için bir yöntem seçilmek istenildiğinde optimize edilmek istenen hiper-parametre sayısı yanında bu parametreler için belirlenen değer aralıkları da dikkate alınmalıdır. Seçilen hiper-parametre alt kümesi ve değer aralıkları küçük bir arama uzayı oluşturuyorsa başarılı sonuçlar elde etmek için *ızgara arama* yöntemi kullanılabilir; arama uzayının genişlemeye başladığı durumlarda ise *rastgele arama* yöntemi, *üst-sezgisel algoritmalar* ya da *pekiştirmeli öğrenme* algoritmaları tercih edilebilir. Hızlı hesaplama süreleri için TPE ve SMAC yöntemlerinin, daha iyi doğruluk değerleri için ise *genetik algoritmalar, parçacık sürü optimizasyonu* ve *rastgele arama* gibi yöntemlerin seçilmesinin daha uygun olduğu görülmektedir. *Üst-sezgisel* algoritmalarda daha az parametreli *üst-sezgiseller,* daha çok parametreli *üst-sezgisellere* göre daha avantajlıdır. Bunun yanında implementasyon kolaylığı ve gerektireceği hesaplama maliyeti de *üst-sezgisel* seçiminde dikkate alınmalıdır.

KSA'daki ağırlıkların sayısı, varsa yanlılık değerleriyle birlikte o ağın parametre sayısını, başka bir ifadeyle büyüklüğünü belirtir. Düşük parametre sayısına sahip küçük ağlar büyük ağlara göre daha kısa sürede öğrenir. Bazı çalışmalarda KSA'nın aşırı büyümesi engellenmiş; böylece basit ve kolay eğitilebilir modeller üretilmiştir. Hesaplama maliyeti açısından çok avantajlı bu basit modeller ile literatürdeki karmaşık modellere çok yakın sonuçlar elde edilebilmiştir. Çok uzun sürede eğitilip çok iyi doğruluk oranı veren ağlar yerine kabul edilebilir seviyede doğruluk oranı veren ancak çok daha hızlı eğitilen ağların tercih edilebileceği durumlar için KSA hiper-parametre optimizasyonu problemi çok amaçlı optimizasyon problemi olarak ele alınmaktadır. Yeni çalışılmaya başlanan bu alanda daha çok sayıda araştırmanın yapılacağı düşünülmektedir.

KAYNAKLAR (REFERENCES)

[1] I. Goodfellow, Y. Bengio, A. Courville and Y. Bengio (2016), Deep learning (Vol. 1). Cambridge: MIT press

[2] E. Öztemel (2003), Yapay Sinir Ağları. Istanbul: Papatya Yayincilik.

[3] S. McCulloch Warren, W. Pitts, A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5:4 (1943) 115-133.

[4] B. W. A. C. Farley, W. Clark, Simulation of self-organizing systems by digital computer. Transactions of the IRE Professional Group on Information Theory, 4:4 (1954) 76-84.

[5] E. Ülker, Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri. Gaziosmanpaşa Bilimsel Araştırma Dergisi, 6.3 (2017) 85-104.

[6] X. Glorot, B. Yoshua, Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics, (2010).

[7] B. Karlik, A. V. Olgac, Performance analysis of various activation functions in generalized MLP architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems, 1:4 (2011) 111-122.

[8] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE international conference on computer vision, (2015).

[9] M. Sinecen, B. Kaya, Ö. Yıldız, Artificial Neural Network Based Early Warning System For Aydin Province Towards Air Factors Which Primarily Affect Human Health. Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji, 5:4 (2017) 121-131.

[10] T. Yichuan, Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, (2013).

[11] E. Alpaydin (2009). Introduction to machine learning. MIT press.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15:1 (2014).

[13] L. Bottou, Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010, Physica-Verlag HD, (2010) 177-186.

[14] D. P. Kingma, J. L. Ba, Adam: Amethod for stochastic optimization. Proc. 3rd Int. Conf. Learn. Representations, (2014).

[15] M. D. Zeiler, ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, (2012).

[16] S. Ruder, An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, (2016).

[17] N. Qian, On the momentum term in gradient descent learning algorithms. *Neural networks*, *12* (1999), 145-151.

[18] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems, (1990).

[19] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86:11 (1998) 2278-2324

[20] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer vision and Image understanding, 106:1 (2007) 59-70.

[21] "Caltech-101", son güncelleme 5 Nisan, 2006, http://www.vision.caltech.edu/.

[22] Cancer Genome Atlas - miRNASeq son güncelleme 20 Kasım, 2018, http://cancergenome.nih.gov/.

[23] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images. Technical report, University of Toronto, (2009).

[24] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio, An empirical evaluation of deep architectures on problems with many factors of variation. Proceedings of the 24th international conference on Machine learning, (2007) 473-480.

[25] G. Cohen, S. Afshar, J. Tapson, A. van Schaik, EMNIST: an extension of MNIST to handwritten letters. arXiv preprint arXiv:1702.05373, (2017).

[26] H. Xiao, R. Kashif, V. Roland, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, (2017).

[27] Eitz M., Hays J., Alexa M., How do humans sketch objects?. ACM Trans. Graph, 31:4 (2012) 44-1.

[28] Armato III, G. Samuel, et al, The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans. Medical physics, 38:2 (2011) 915-931.

[29] Y. LeCun, C. Cortes, C. J. Burges, MNIST handwritten digit database. AT&T Labs [Online], Available: http://yann. lecun. com/exdb/mnist, (2010).

[30] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning. Proceedings of the fourteenth international conference on artificial intelligence and statistics, (2011).

[31] K. K. Reddy, M. Shah, Recognizing 50 human action categories of web videos. Machine Vision and Applications, 24:5 (2013) 971-981.

[32] J. H. Holland (1992), Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence. MIT press.

[33] J. H. Holland, Genetic algorithms. Scientific american, 267 (1992) 66-73.

[34] D. E. Goldberg, J. H. Holland, Genetic algorithms and machine learning. Machine learning, 3:2 (1988) 95-99.

[35] E. P. Ijjina, M. C. Krishna, Human action recognition using genetic algorithms and convolutional neural networks. Pattern recognition, 59 (2016) 199-212.

[36] G. L. da Silva, O. P. da Silva Neto, A. C. Silva, A. C. de Paiva, M. Gattass, Lung nodules diagnosis based on evolutionary convolutional neural network. Multimedia Tools and Applications, 76:18 (2017) 19039-19055.

[37] A. Lopez-Rincon, A. Tonda, M. Elati, O. Schwander, B. Piwowarski, P. Gallinari, Evolutionary optimization of convolutional neural networks for cancer miRNA biomarkers classification. Applied Soft Computing, 65 (2018) 91-100.

[38] E. Dufourq, A. B. Bruce, EDEN: Evolutionary deep networks for efficient machine learning. Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), IEEE, (2017).

[39] Y. Sun, X. Bing, M. Zhang, Evolving deep convolutional neural networks for image classification. arXiv preprint arXiv: 1710.10741, (2017).

[40] E. Bochinski, S. Tobias, T. Sikora, Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. Image Processing (ICIP), 2017 IEEE International Conference on. IEEE, (2017).

[41] S. Fujino, M. Naoki, K. Matsumoto, Deep convolutional networks for human sketches by means of the evolutionary deep learning. Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), 2017 Joint 17th World Congress of International. IEEE, (2017).

[42] A. Krizhevsky, S. Ilya, G. E. Hinton, Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, (2012).

[43] B. Ma, Y. Xia, Autonomous Deep Learning: A Genetic DCNN Designer for Image Classification. arXiv preprint arXiv:1807.00284, (2018).

[44] F. Assunçao, N. Lourenço, P. Machado, B. Ribeiro, DENSER: Deep Evolutionary Network Structured Representation. Genetic Programming and Evolvable Machines, (2018) 1-31.

[45] A. Baldominos, S. Yago, P. Isasi, Evolutionary convolutional neural networks: An application to handwriting recognition. Neurocomputing, 283 (2018) 38-52.

[46] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory. In Micro Machine and Human Science, Proceedings of the Sixth International Symposium, (1995, October) 39-43

[47] J. Kennedy (2011), Particle swarm optimization. In Encyclopedia of machine learning, Boston: Springer, 760-766.

[48] T. Yamasaki, H. Takuto, A. Kiyoharu, Efficient Optimization of Convolutional Neural Networks Using Particle Swarm Optimization. Multimedia Big Data (BigMM), 2017 IEEE Third International Conference on IEEE, (2017).

[49] G. L. F. da Silva, T. L. A. Valente, A. C. Silva, A. C. de Paiva, M. Gattass, Convolutional neural network-based PSO for lung nodule false positive reduction on CT images. Computer methods and programs in biomedicine, 162 (2018) 109-118.

[50] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, J. R. Pastor, Particle swarm optimization for hyper-parameter selection in deep neural networks. Proceedings of the Genetic and Evolutionary Computation Conference, ACM, (2017).

[51] J. Nalepa, P. R. Lorenzo, Convergence Analysis of PSO for Hyper-Parameter Selection in Deep Neural Networks. International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, (2018).

[52] Y. Sun, B. Xue, M. Zhang, A Particle Swarm Optimization-based Flexible Convolutional Auto-Encoder for Image Classification. arXiv preprint arXiv:1712.05042, (2017).

[53] B. Wang, Y. Sun, B. Xue, M. Zhang, Evolving Deep Convolutional Neural Networks by Variable-length Particle Swarm Optimization for Image Classification. arXiv preprint arXiv:1803.06492, (2018).

[54] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11 (1997) 341-359.

[55] T. Keskintürk, Diferansiyel gelişim algoritması. İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 5 (2006) 85-99.

[56] B. Wang, Y. Sun, B. Xue, M. Zhang, A Hybrid Differential Evolution Approach to Designing Deep Convolutional Neural Networks for Image Classification, (2018).

[57] Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search. simulation, 76 (2001) 60-68.

[58] W. Y. Lee, S. M. Park, K. B. Sim, Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. Optik, 172 (2018) 359-367.

[59] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. Optimization by simulated annealing. science, 220 (1983) , 671-680.

[60] L. M. Rere, M. I. Fanany, A. M.Arymurthy, Metaheuristic algorithms for convolution neural network. Computational intelligence and neuroscience, (2016).

[61] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13.Feb (2012), (281-305).

[62] T. Domhan, J. T. Springenberg, F. Hutter, Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. IJCAI, 15 (2015).

[63] D. Saranyaraj, M. Manikandan, S. Maheswari, A deep convolutional neural network for the early detection of breast carcinoma with respect to hyper-parameter tuning. Multimedia Tools and Applications, (2018) 1-26.

[64] P. Neary, Automatic Hyperparameter Tuning in Deep Convolutional Neural Networks Using Asynchronous Reinforcement Learning. 2018 IEEE International Conference on Cognitive Computing (ICCC), IEEE, (2018).

[65] B. van Stein, H. Wang, T. Bäck, Automatic Configuration of Deep Neural Networks with EGO. arXiv preprint arXiv:1810.05526, (2018).

[66] T. Hinz, N. Navarro-Guerrero, S. Magg, S. Wermter, Speeding up the Hyperparameter Optimization of Deep Convolutional Neural Networks. International Journal of Computational Intelligence and Applications, (2018)

[67] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[68] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, A. Rabinovich, Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015) 1-9.

[69] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) 770-778.

[70] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) 4700-4708.