FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MİKROKANONİKAL OPTİMİZASYON ALGORİTMASI İLE KONVOLÜSYONEL SİNİR AĞLARINDA HİPER PARAMETRELERİN OPTİMİZE EDİLMESİ

YÜKSEK LİSANS TEZİ

Zeki KUŞ

Anabilim Dalı: Bilgisayar Mühendisliği

HAZİRAN 2019

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MİKROKANONİKAL OPTİMİZASYON ALGORİTMASI İLE KONVOLÜSYONEL SİNİR AĞLARINDA HİPER PARAMETRELERİN OPTİMİZE EDİLMESİ

YÜKSEK LİSANS TEZİ

Zeki KUŞ

(170221002)

Anabilim Dalı: Bilgisayar Mühendisliği

Tez Danışmanı:

Dr. Öğr. Üyesi Ayla GÜLCÜ

Teslim Tarihi: 23.05.2019

FSMVÜ, Lisansüstü Eğitim Enstitüsü'nün Bilgisayar Mühendisliği Anabilim dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans öğrencisi 170221002 numaralı Zeki KUŞ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "Mikrokanonikal Optimizasyon Algoritması ile Konvolüsyonel Sinir Ağlarında Hiper Parametrelerin Optimize Edilmesi" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Dr. Öğr. Üyesi Ayla Gülcü** .....

Fatih Sultan Mehmet Vakıf Üniversitesi

Jüri Üyeleri : **Prof. Dr. A. Yılmaz** .....

Fatih Sultan Mehmet Vakıf Üniversitesi

Doç. Dr. A. Şima Uyar .....

İstanbul Teknik Üniversitesi

Dr. Öğr. Üyesi Ayla Gülcü .....

Fatih Sultan Mehmet Vakıf Üniversitesi

Teslim Tarihi : 23 Mayıs 2019

Savunma Tarihi : 21 Haziran 2019

ÖNSÖZ

Tez çalışmam süresince, dünyada nispeten yeni olan bu tez konusu için beni destekleyen, tezin her aşamasında yardımlarını esirgemeyen, daha iyisini yapmak için beni cesaretlendiren, tecrübesiyle beni her zaman doğru bir şekilde yönlendiren değerli danışman hocam Dr. Öğr. Üyesi Ayla GÜLCÜ'ye teşekkürü borç bilirim.

Aynı zamanda eğitim hayatım boyunca manevi ve maddi desteklerini esirgemeyip, beni cesaretlendiren aileme de teşekkür ederim. Bu tezin, ülkemizde ve dünyada yapılacak yeni çalışmalara katkı sağlayabilmesini temenni ederim.

Zeki KUŞ

Bilgisayar Mühendisi

İÇİNDEKİLER

KISALTMALAR LİSTESİ

Kısaltma	Açıklama
BT	Benzetimli Tavlama
DE	Diferansiyel Evrim
GA	Genetik Algoritma
HA	Harmonik Arama
KSA	Konvolüsyonel Sinir Ağları
μO	Mikrokanonikal Optimizasyon
MT	Mikrokanonikal Tavlama
PSO	Parçacık Sürü Optimizasyonu
SGD	Stokastik Gradyan Azaltma
TPE	Tree Parzen Estimator
YA	Yerel Arama (Local Search)
YSA	Yapay Sinir Ağı

ÇİZELGE LİSTESİ

ŞEKİL LİSTESİ

MİKROKANONİKAL OPTİMİZASYON ALGORİTMASI İLE KONVOLÜSYONEL SİNİR AĞLARINDA HİPER PARAMETRELERİN OPTİMİZE EDİLMESİ

ÖZET

Bilgisayarlı görü çalışmaları, günümüzde en çok ilgi duyulan ve üzerinde çalışma yapılan yapay zeka alanlarından biridir. Bilgisayarların insanlar gibi görüntüleri algılamasını, sınıflandırabilmesini ve yorumlayabilmesini sağlamak amacıyla geliştirilen özel derin öğrenme mimarileri bulunmaktadır. Bunlardan en çok kullanılan ve bu çalışmada da bahsedilecek olan mimari konvolüsyonel sinir ağları mimarisidir. Konvolüsyonel sinir ağları, bilgisayarlı görü çalışmalarında popüler olarak kullanılan ve başarılı sonuçlar elde edilebilen özelleşmiş bir derin öğrenme yöntemidir. Derin öğrenme yöntemleri karşılaşılan problemlerin zorluğu nedeniyle yüksek hesaplama maliyetlerine neden olabilmektedir. Hesaplama maliyetinin düşürülmesi güçlü donanımların kullanılmasına, oluşturulan konvolüsyonel sinir ağı topolojilerindeki toplam parametre sayısının azaltılmasına ve konvolüsyonel sinir ağlarındaki hiper-parametreler için seçilen değerlere bağlıdır. Bu yüzden konvolüsyonel sinir ağlarında hiper-parametre optimizasyonu çalışmaları, ağın başarısını arttırmaya çalışırken, hesaplama maliyetini de düşük tutmaya çalışmaktadırlar. Bu tez çalışmasında ilk olarak daha önce konvolüsyonel sinir ağlarının optimize edilmesi için gerçekleştirilen optimizasyon çalışmaları incelendi. İncelenen çalışmalarda, konvolüsyonel sinir ağlarında hiper-parametrelerin optimizasyonu için sıklıkla üst-sezgisel algoritmaların ve istatistik tabanlı model bazlı algoritmaların kullanıldığı gözlemlendi. Özellikle Genetik Algoritma, Parçacık Sürü Optimizasyonu, Diferansiyel Evrim, Rastgele Arama ve Bayes Optimizasyonu gibi yöntemlerin, incelenen çalışmalarda sıklıkla kullanıldığı gözlemlendi. Bu çalışmalar başarı açısından incelendiklerinde genetik algoritma ve parçacık sürü optimizasyonu yöntemlerinin genel olarak hiper-parametre optimizasyonu gerçekleştirmeyen çalışmalara göre başarılı ve rekabetçi sonuçlar verdiği görüldü. Yapılan tez çalışmasında kullanılacak veri setleri, seçilecek optimizasyon yöntemi, hiper-parametreler ve değer aralıklarının belirlenmesi için incelenen çalışmalarda kullanılan veri setleri, hiper-parametreler ve bu hiper-parametreler için seçilen değer aralıkları göz önünde bulunduruldu. Yapılan çalışmalarda, farklı çalışmaları karşılaştırmak için elimizde parametre sayısı ve hesaplama zamanı bilgileri bulunmadığından sadece doğruluk oranı bilgisi performans karşılaştırması için kullanıldı. Daha önce yapılan bu çalışmalardan farklı olarak bu tez çalışmasında "Mikrokanonikal Optimizasyon" olarak adlandırılan bir yöntem kullanıldı. Seçilen optimizasyon yöntemi kullanılarak farklı boyutlarda konvolüsyonel sinir ağları oluşturuldu ve oluşturulan konvolüsyonel sinir ağlarının hiper-parametreleri optimize edilmeye çalışıldı. Seçilen optimizasyon algoritmasının çalışması sırasında üretilen konvolüsyonel sinir ağları, bilgisayarlı görü çalışmalarında sıklıkla kullanılan MNIST, FashionMNIST, EMNIST (Balanced, Digits, Letters) ve

CIFAR10 veri setleri üzerinde test edildi. Elde edilen sonuçlar, hiç hiper-parametre optimizasyonu gerçekleştirmeyen ve state-of-the-art olarak adlandırılan çalışmalar ile doğruluk oranı ve parametre sayısı gibi değerler üzerinden karşılaştırıldı. Ek olarak, önerilen sezgisel yöntemin performansı, Bayesçi model tabanlı bir optimizasyon yöntemi olan Tree Parzen Estimator yöntemiyle karşılaştırılmıştır. Elde edilen sonuçlara bakıldığında, Konvolüsyonel sinir ağları için belirlenmesi gereken birçok hiper-parametre olmasına rağmen seyreltme oranı, filtre sayısı, öğrenme oranı ve yığın boyutu gibi hiper-parametrelerin oluşturulan modellerin başarısında önemli bir katkısı olduğu çıkarımına ulaşıldı.

OPTIMIZATION OF HYPER PARAMETERS IN CONVOLUTIONAL NEURAL NETWORKS BY MICROCANONICAL OPTIMIZATION ALGORITHM

SUMMARY

Computer vision is probably the most widely studied sub-area of artifical intelligence which has been drawing considerable interest of many researchers for years. There are special deep learning architectures developed to enable computers to perceive, classify and interpret images as humans. Convolutional neural networks are the most popular deep learning methods that can be used successfully in computer vision studies. Deep learning methods may result in high computational costs due to the difficulty of the problems encountered. This computational cost can only be reduce by careful selection of hyperparameters of the convolutional neural networks and the computational time can also be reduce by the use of powerful equipment. Therefore, in the studies that try to optimize hyperparameters in convolutional neural networks, the researchers try to increase the success rate of the network while at the same time try to keep the computational cost as low as possible. In this thesis, firstly a detailed literature review on the studies that perform hyperparameter optimization has been given. It has been observed that heuristic algorithms and statistics model based algorithms are among the most widely used methods for hyper-parameter optimization in convolutional neural networks. In particular, Genetic Algorithms, Particle Swarm Optimization, Differential Evolution, Random Search and Bayes Optimization methods are the most frequently used approaches. When we compare these methods in terms of their success rates, we see that the studies in which genetic algorithms and particle swarm optimization methods are used were able to achieve greater results than the studies that did not perform hyper-parameter optimization in general. In order to determine the optimization method to be used in the study along with the hyper-parameters and their value ranges, we benefited the studies in the literature. Moreover, the datasets used in this study are selected among the most widely used datasets in the literature. Most of the studies in the literature do not provide sufficient information about the number of parameters of the network and the computational time, therefore we took in the account accuracy as the performance measure. In this study, Microcanonical Optimization which is previously known in different areas was but not used in this concepts has been applied fort he hyperparameter optimization of convolutional neural networks. By this method, different network architectures has been created and the hyper-parameters of the network is optimized. The convolutional neural networks generated during the optimization process are trained on the MNIST, FashionMNIST, EMNIST (Balanced, Digits, Letters) and CIFAR10 datasets, which are the most frequently used datasets in computer vision studies. The accuracy results are compared to the state-of-the-art

architectures in which no hyper-parameter optimization has been performed. In addition, the performance of proposed heuristic method has been compared to Tree Parzen Estimator method which is a Bayesian model based optimization method. The results suggest that among the many hyperparameters dropout rate, feature map count, learning rate and batch size are among the most important parameters that directly affect the success of the networks.

1. GİRİŞ

İnsan gibi düşünen, insana benzer, sorunlara insanların bakış açısıyla çözüm üretebilen genel bir zeka oluşturmak bilgisayar bilimleri için her zaman önemli bir çalışma konusu olmuştur. Bunların temelinde daha zeki sistemler tasarlamak, sadece belirli bir iş için uzmanlaşmış sistemler yerine insan gibi öğrenebilen, çözüm üretebilen, bulunduğu ortama ve şartlara uyum sağlayabilen sistemleri oluşturma isteği yatmaktadır. Bu sistemleri geliştirebilmek için en başta insan beyninin modellenmesi gerekmektedir. İnsan beynini modelleyebilmek için ise insan beyninin biyolojik olarak nasıl çalıştığını anlamak gerekmektedir.

1.1 Sinir Ağlarının Tarihçesi

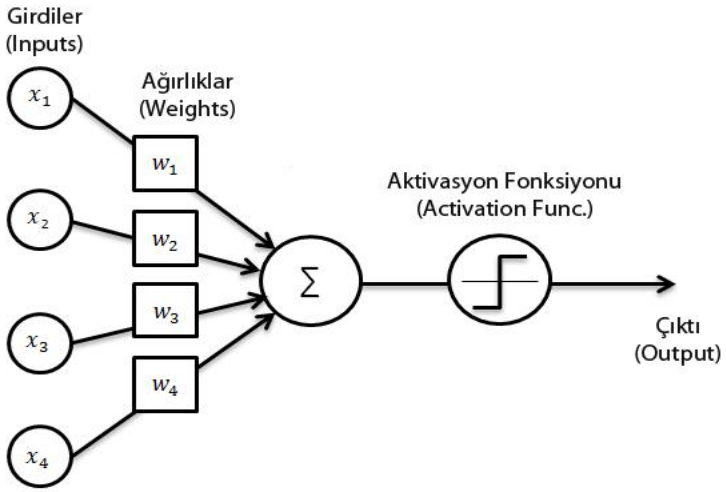
1940'lı yıllardan beri daha zeki sistemler geliştirebilmek adına yapılan çalışmalar mevcuttur. İlk olarak 1943 yılında S. McCulloch ve Walter H. Pitts tarafından yayınlanan "A Logical Calculus of The Ideas Immanent In Nervous Activity" [1] isimli makale ile bu çalışmalar başlamıştır. Bu çalışmada insan beyninde öğrenmeyi sağlayan sinir ağlarından esinlenilmiştir. Beyin fonksiyonları, sinir aktiviteleri için mantıksal bir analiz yapılmış ve matematiksel modeller ile açıklanmıştır [2]. Sinirsel aktiviteler ve bunlar arasındaki ilişkiler (ağırlıklar), önermeli mantık yolu ile iyileştirilebilir denilmiş ve günümüzde kullanılana benzer bir yapay sinir ağı modeli gerçekleştirilmiştir. Bu modelde nöronlar ve nöronlar arasındaki ilişkileri ifade etmek için ağırlıklar kullanılmıştır. Fakat günümüzdeki yaklaşımdan farklı olarak ağırlıkların öğrenilemeyeceği fikri savunulmuştur [1].

1958 yılında F. RosenBlatt tarafından "The perceptron: a probabilistic model for information storage and organızatıon ın the brain" çalışması yayınlanmıştır. Bu çalışma temel olarak 3 temel soruya yanıt aramaktadır [3]:

- Biyolojik sistem tarafından fiziksel dünyadaki olaylar nasıl algılanır veya tespit edilir?
- Elde edilen bilgiler hangi formda saklanır veya hatırlanır?
- Depolamada veya bellekte yer alan bilgiler tanıma ve davranışları nasıl etkiler?

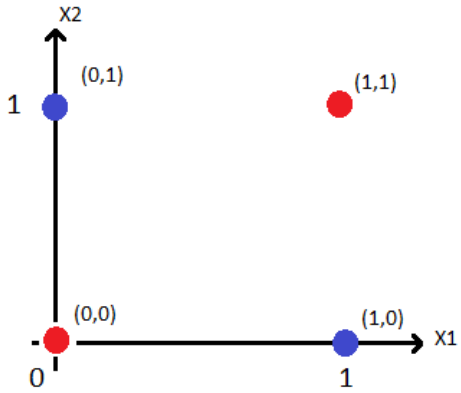
Bu yaklaşım da insanda bulunan sinir ağlarının çalışma yapısından esinlenilerek geliştirilmiştir. Aktivasyon fonksiyonu ve eşik değeri (threshold) kavramları bu çalışmayı önemli kılan noktalardandır [3]. Belirli bir eşik değerine göre nöron ve

çıktıların güncellenmesini ve karar vermeyi sağlayan bir yapı oluşturulmuştur. Bu yapı ile ilk defa öğrenme kavramı da konuşulmaya başlanmıştır. Basit mantıksal operatörler için bu yapının başarılı sonuçlar vermesi insanların bu modelin hatasız olduğu fikrine kapılmalarına neden olmuştur. Ta ki XOR problemi ile karşılaşılıncaya kadar. Yapılan bu çalışma günümüz yapay sinir ağlarının oluşumu için önemli bir temel oluşturmuştur.



Şekil 1.1: Perceptron yapısı

1960'lı yılların sonuna doğru gelindiğinde, 1969 yılında M. Minsky ve S. Papert tarafından yayınlanan "Perceptrons" isimli çalışma ile F.Rosenblatt tarafından ortaya koyulan *perceptron* yapısının (şekil 1.1) doğrusal olmayan problemler için başarısız olduğu ortaya konulmuştur. *Perceptron* yapısının doğrusal ayrılabilen problemler için başarılı olduğu bilinmekteydi. Fakat *Perceptron* yapısı XOR problemi (şekil 1.2) gibi doğrusal olmayan bir problem için uygulandığında, bu yapının başarısız olduğu görülmüştür [4]. Yani *perceptron* yapısı ile doğrusal olmayan problemlerin sonuçlarının iki boyutlu uzayda tek bir doğrusal çizgi ile ayrılamayacağı anlaşılmıştır. Bu durum yapay zeka çalışmalarındaki 9 yıllık (1960 - 1969) altın çağı sonlandırmıştır. Aynı zamanda bu problem 1986 yılına kadar çözülememiş ve yapay zeka için karanlık çağın (1969 – 1986) başlangıcına neden olmuştur.

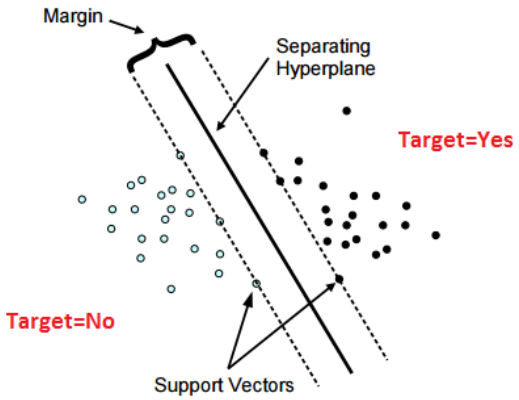


Şekil 1.2: XOR problemi

1986 yılına gelindiğinde bugün ki derin öğrenme modellerinin de temellerini oluşturacak bir yaklaşım ortaya atıldı. 1986 yılında G. Hinton, D. Rumelhart ve R. Williams tarafından yayınlanan "Learning representations by back-propagations errors" [5] isimli çalışma yapay sinir ağları için yeni bir öğrenme metodu getirdi. Yeni metod *geri yayılım* olarak isimlendirildi. Daha önce kullanılan *perceptron* yapısında kullanılan katman sayısı arttırılarak *Çok Katmanlı Perceptron (Multi Layer Perceptron)* yapısı elde edildi. Bu yapının eğitilmesi, ağırlıkların doğru bir şekilde güncellenmesi için *geri yayılım* algoritması kullanıldı. *Geri yayılım* algoritması, ağda bulunan nöronlar arasındaki bağlantıları, ağırlıkları gerçek çıktı vektörü ile hesaplanan çıktı vektörü arasındaki farkın minimize edilmesini sağlayacak şekilde günceller. Girdi ve çıktı katmanları arasındaki katmanlar *gizli katman (hidden layer)* olarak isimlendirilir. Öğrenme prosedürü, verilen girdi için istenen çıktıyı elde edebilmek adına gizli birimlerin hangi durumlarda aktif olacağına karar vermelidir [5]. Hatanın minimuma indirgenmesi için katmanlar arası ağırlıkların doğru bir şekilde güncellenmesi gerekir. *Geri yayılım* algoritması ağırlıkların güncellenmesini, seçilen optimizasyon yöntemine bağlı olarak her bir örnekten sonra ya da belirli örnek topluluklarından (batch) sonra hesaplanan hataya göre yineleyerek yapar. Ağırlıkların güncellenmesi hataların türevi alınacak şekilde yapıldığından ve katman sayısı ya da katmanlardaki nöron sayılarının fazla olmasından dolayı yüksek işlem gücü gerektirir. Fakat *geri yayılım* algoritması ve çok katmanlı perceptron yapısı doğrusal olmayan, lineer olarak ayrılamayan problemlerin çözümü için başarılı sonuçlar elde etmiştir ve yapay zeka çalışmalarını tekrar ivmelendirmiştir.

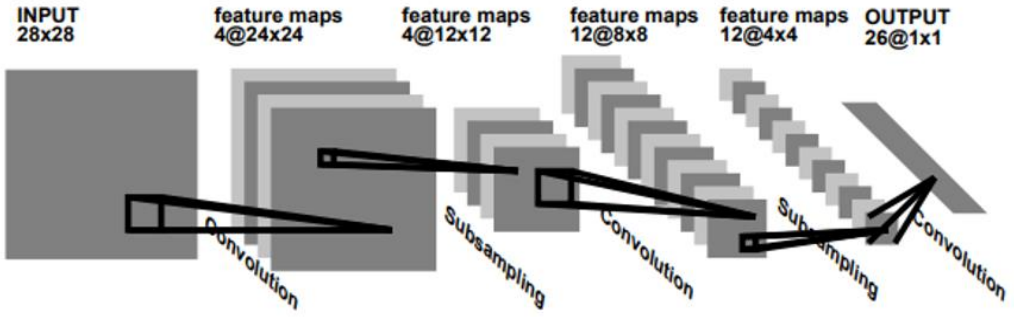
Çok katmanlı perceptron yapısının ortaya çıktığı dönemde yeterli işlem gücünün olmaması yapay sinir ağlarının gelişimini olumsuz yönde etkilemiştir. Bu dönemde daha az işlem gücü gerektiren ve analitik yöntemlere dayanan yeni bir öğrenme

yaklaşımı ortaya çıkmıştır. 1995 yılında V. Vapnik ve C. Cortes tarafından "Support Vector Networks" isimli çalışma yayınlanmıştır. Bu çalışma, iki gruplu sınıflandırma problemleri için yeni bir öğrenme makinesi olarak tanıtılmıştır [6]. Sınıflandırma işlemi vektörel işlemler sonucu elde edilen karar destek çizgileri ile yapılmaktadır. Şekil 1.3 'te karar destek vektör makinesi gösterilmiştir.



Şekil 1.3: SVM [7]

1990'lı yıllarda Yann LeCun tarafından AT & T Bell Labs'ta yürütülen el yazısı tanıma projesi [8], bugün kullandığımız anlamda konvolüsyonel sinir ağlarının temelini oluşturmuştur. Konvolüsyonel Sinir Ağları (KSA), katmanlarının en az bir tanesinde matris çarpımı yerine konvolüsyon işleminin kullanıldığı çok katmanlı yapay sinir ağlarının bir türüdür. Aynı zamanda derin öğrenme ağlarının bilgisayarlı görü çalışmaları için özelleşmiş bir mimarisidir. KSA'lar 3. bölümde daha detaylı anlatılacaktır. Yann LeCun, ağın eğitilmesi için kendisinin kurduğu Le-Net 5 ağını kullanmıştır. Bu çalışma başarılı sonuçlar elde etse de yeterli hesaplama gücünün elde edilememesi ve ağların eğitimi için yeterli miktarda eğitim verisinin olmaması bu çalışmaların bir süre daha köşede beklemesine neden olmuştur. Şekil 1.4 'te Yann LeCun tarafından yapılan çalışmada kullanılan Le-Net mimarisi gösterilmiştir.



Şekil 1.4: El yazısı tanıma veri seti için kullanılan konvolüsyonel sinir ağı mimarisi [8]

2000'li yıllarda Geoffrey Hinton tekrar sahneye çıktı ve sinir ağlarını, çok fazla gizli katman içeren derin öğrenme ağlarına dönüştürdü. Derin öğrenme ağları etkili sonuçlar vermesine rağmen gerektirdiği yüksek işlem gücü nedeniyle hala etkin olarak kullanılamamaktaydı. 2006 yılında Geoffrey Hinton, ağırlıkların, parametrelerin ve katman sayısının çok fazla olduğu derin öğrenme ağlarının açgözlü katmanlı ön eğitim metodu kullanılarak başarılı bir şekilde eğitildiğini kanıtlamıştır.

Gelişen teknoloji ile birlikte derin öğrenme ağlarının eğitilmesi için gerekli olan işlem gücüne nispeten ulaşılmaya başlandı. Bulut hizmetlerinin artması, paralel işlem gücü, ekran kartlarının çekirdek sayıları ve hızları, mobil cihazların ve sosyal medya kullanımının artması ile elde edilen büyük veri setleri derin öğrenme çalışmalarını hızlandırdı. 2010 yılında Stanford Üniversitesi'nde Fei-Fei Li'nin başında bulunduğu grup "ImageNet" adı verilen milyonlarca etiketlenmiş resim içeren veri setini oluşturdu ve paylaştı. Büyük ölçekli resimlerin tanınması için geliştirilen mimari ve algoritmaların yarıştığı LSVRC yarışması düzenlenmeye başlandı. İlki 2010 yılında düzenlenen yarışmanın ilk yıllarında %28 ve %26 doğruluk oranları (accuracy) elde edildi. 2012 yılında ise Alex Krizhevsky, Ilya Sutskever ve Geoffrey E. Hinton tarafından geliştirilen mimari Top-1 için 37.5% ve Top-5 için 17% doğruluk değerleri elde etti. Bu ifadelerde Top-1, tahmin olasılığı en yüksek olan sonucun gerçek sonuç ile doğru olarak eşleşme olasılığını, Top-5 ise tahmin olasılığı en yüksek olan 5 sonuç içerisinde gerçek sonucun geçme olasılığını temsil etmektedir (Yani tahmin ettiğim 5 sonuç içerisinde gerçek sonuç ile eşleşen bir sonuç var mı?) . Elde ettiği bu başarı ve getirmiş olduğu yenilikçi yaklaşımlar derin öğrenme ağları ile yapılan çalışmaların tekrar hız kazanmasını sağlamıştır. Bu çalışmanın ayrıntıları "ImageNet Classification with Deep Convolutional Neural Networks" [9] adıyla yayınlanmıştır. Bu çalışmanın en önemli noktalarından biri de çok daha fazla çekirdek sayısına sahip olan ve verilerin paralel bir şekilde hesaplanabilmesine imkan sağlayan GPU'lardır. GPU'lar kullanılarak derin öğrenme modelleri çok daha hızlı bir şekilde eğitilebilmektedir. Bu durum da daha büyük modellerin oluşturulmasına, eğitilmesine ve daha başarılı sonuçların elde edilmesine olanak sağlamaktadır.

Günümüzde derin öğrenme çalışmaları bilgisayar bilimlerinin en sıcak konularından biri haline gelmiştir. Özellikle bilgisayarlı görü tarafında yapılan çalışmalar ve geliştirilen mimariler sonucunda elde edilen başarı oranları 2015 yılında insan hata

oranının altına inmiştir. Yani eğitilen modeller insanlardan daha iyi bir doğrulukla görüntüleri sınıflayabilmektedir.

Bu tez çalışmasında yapay sinir ağlarının yapısı, sinir ağlarının öğrenme süreci, konvolüsyonel sinir ağlarının yapısı, konvolüsyonel sinir ağlarında kullanılan hiper-parametreler ve bu hiper-parametrelerin optimizasyonu için kullanılan yöntemler anlatılmıştır. Aynı zamanda optimizasyon çalışması yapılan konvolüsyonel sinir ağları ile optimizasyon çalışması yapılmayan konvolüsyonel sinir ağları karşılaştırılmıştır. Karşılaştırma için kullanılan yöntemler anlatıldı ve daha önce bu alanda yapılan çalışmalar ile performans karşılaştırması gerçekleştirildi. Sonuç olarak konvolüsyonel sinir ağlarını oluştururken hiper-parametrelerin seçiminde dikkat edilmesi gereken noktalara, günümüzde sıklıkla kullanılan yöntemlere ve ileride kullanılabilecek metodolojilere değinildi.

2. YAPAY SİNİR AĞLARI

İnsan beyninin nasıl çalıştığı yıllarca bu alanda çalışan sinirbilimciler için önemli bir araştırma konusu olmuştur. Yapılan bu araştırmalar boyunca insan beynini oluşturan sinir ağlarının birçok modeli oluşturulup, insan beyninin nasıl öğrendiği sorusuna cevap aranmaya çalışılmıştır. İnsan beyninin nasıl anladığını öğrenmek için sinir ağlarının çalışma yapısının da öğrenilmesi gerekmektedir. Yapılan bu çalışmalardaki amaç yalnızca insan beyninin veya sinir ağlarının nasıl çalıştığını anlamak değil aynı zamanda insan beyni gibi öğrenebilen yapay sistemler, sinir ağları tasarlayabilmektir [10].

İnsan beyninin öğrenme özelliğini makinelere kazandırmak için geçmişten günümüze birçok çalışma yapılmıştır. Bu alanda bilgisayar bilimciler sinir ağlarının oluşturulan modellerini göze alarak yapay sinir ağlarını oluşturmuşlardır. Sinir ağının temel bileşenleri olan nöron, sinaps, dentrit ve aksonlar yapay sinir ağlarında karşılık gelecek şekilde nöron, girdi, çıktı ve ağırlıklar olarak adlandırılmış ve ilk yapay sinir ağı modelleri oluşturulmuştur. İlk başlarda sadece girdi, girdilere ait ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve tek bir çıktıdan oluşan bu daha basit sistem algılayıcı (*perceptron*) olarak adlandırılmıştır.

Bu algılayıcı, farklı nöronlardan gelen sinyalleri, girdi olarak alır. Alınan bu girdiler ve ağırlıklar (sinaps) çarpılıp toplanarak toplama fonksiyonu elde edilir. Toplama fonksiyonunda elde edilen değer, belirlenen aktivasyon fonksiyonundaki eşik değerine

göre aktif hale getirilir veya getirilmez. Aktivasyon işleminden sonra elde edilen sonuç çıktı olarak adlandırılır ve bu çıktı diğer nöronlara aktarılır. Yani bir nöron için elde edilen çıktı diğer nöron için girdi olarak kullanılır.

Yapay sinir ağlarının çalışma yapısı incelendiğinde, girdinin özelliklerine göre değişken boyutta girdi düğümleri oluşturulur. Bu girdi düğümlerinin her biri girdinin farklı bir özelliğini temsil etmektedir. Temsil edilen bu farklı özelliklerin önem derecesini, çıktıya olan ağırlığını belirten ağırlıklar vardır. Bu ağırlıklar hangi özelliklerin çıktı üzerinde daha etkili olduğunu belirler. Ağırlıkların değerleri sınıflandırma işlemi sonucunda elde edilen sonuç ile beklenen sonuç arasındaki farka göre her çevrimde istenen sonuç elde edilene kadar güncellenir.

2.1 Aktivasyon Fonksiyonları

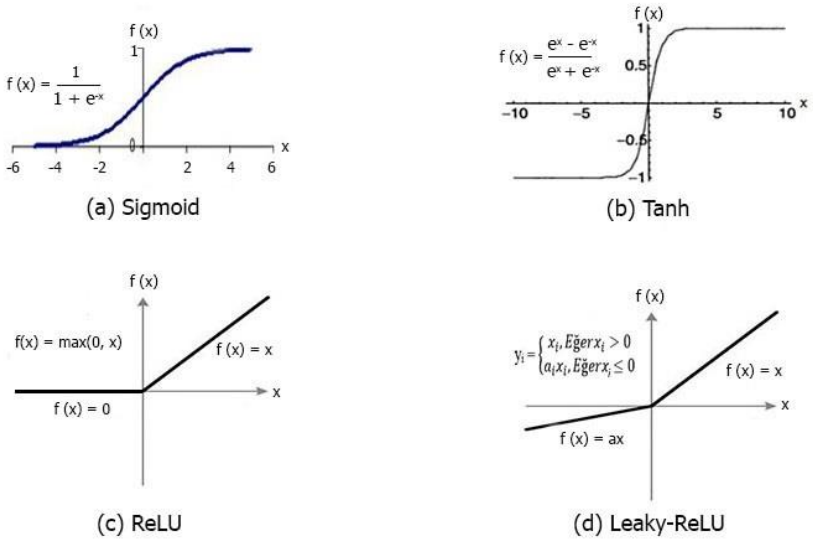
Aktivasyon fonksiyonları, gelen girdilere karşılık oluşacak olan çıktıları belirler. Seçilen aktivasyon fonksiyonu, yapay sinir ağını oluşturan elemanların doğrusal yapısının, doğrulsal olmayan yapıya dönüştürülmesini sağlar. Yapılan dönüşüm sonucunda oluşan doğrusal olmayan yapı, sinir ağının türevlenmesi kolay hale getirilmesini sağlar [11]. Türev alma işlemi ile girdiler için tanımlanan ağırlıkların hata değerine etkisi hesaplanır, sonuca göre ağırlıklar hata değerini azaltacak şekilde güncellenir [12]. Kolay türevlenebilir fonksiyonlar, sinir ağlarının daha hızlı hesaplamalar yapması için gereken avantajı sağlar. Bu nedenle, sinir ağlarında hesaplama zamanının azaltılması için türevlenmesi kolay olan aktivasyon fonksiyonları seçilmelidir. Sıklıkla kullanılan aktivasyon fonksiyonları şunlardır: Sigmoid, tanjant hiperbolik, doğrultulmuş lineer ünite ve sızdırılmış doğrultulmuş lineer ünite aktivasyon fonksiyonları (Şekil 2.1).

2.1.1 Sigmoid aktivasyon fonksiyonu

Sinir ağları ile sınıflandırma çalışmalarında yoğun olarak kullanılan *sigmoid* aktivasyon fonksiyonuna verilen girdi elemanları, denklem 2.1'de gösterildiği üzere 0 ile 1 arasındaki değerlere yani çıktılara dönüştürülür. İkili sınıflandırma problemlerinde sıklıkla kullanılmaktadır. Kullanılmasındaki en büyük dezavantaj ise *vanishing gradient (kaybolan eğim)* sorununun ortaya çıkabilmesidir. Derin sinir ağlarında, çevrim sayısı arttıkça türevi alınan elemanlar giderek sıfıra yakınlaşır [13], *vanishing gradient* problemi denen bu durumda değerlerin çok fazla sıfıra yaklaşması öğrenme işlemini zorlaştırır. Şekil 2.1(a)' da görüldüğü gibi, her iki uca doğru

bakıldığında *x* eksenindeki değişimlere, *y* ekseninde daha sınırlı bir aralıkta yanıt verilmektedir. Yani belirli aralıklarda gelen *x* değerleri için birbirlerine yakın *y* değerleri çıktı olarak üretilebilir. Sinir ağlarında katman sayısı arttıkça *y* değerlerinin doyuma ulaştığı, yani sürekli kısıtlı bir alanda çıktıların elde edildiği görülebilmektedir.

$$f(x) = \frac{1}{1+e^{-x}} \quad \textbf{(2.1)}$$



Şekil 2.1: Aktivasyon fonksiyonları [14]

2.1.2 Tanjant Hiperbolik Aktivasyon Fonksiyonu

Bu aktivasyon fonksiyonunda, girdiler -1 ile 1 aralığında değerlere, çıktılara dönüştürülür. Bu durum şekil 2.1(b)' de gösterilmiştir. Verilen girdiler için [-1, 1] arasında değer alan çıktılar sıfır merkezli olarak adlandırılır. Bu özelliğinden dolayı tanjant hiperbolik aktivasyon fonksiyonu en iyileme işleminin daha kolay hale gelmesine katkı sağlar. Bu aktivasyon fonksiyonu, sigmoid aktivasyon fonksiyonuna oranla daha sık kullanılsada hala *kaybolan eğim* sorunu bu aktivasyon fonksiyonunu da etkilemektedir. Çıktıların nasıl hesaplandığı denklem 2.2 'de gösterilmiştir.

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \textbf{(2.2)}$$

2.1.3 Doğrultulmuş Lineer Ünite (Rectified Linear Unit – ReLU)

ReLU aktivasyon fonksiyonu 2012 yılında [9] yapılan bir çalışma ile popülerliğini arttıran bir aktivasyon fonksiyonudur. Verilen girdiler, şekil 2.1(c)' de gösterildiği üzere [0, ∞] aralığında değerlere, çıktılara dönüştürülür. Girdiler sıfır ile sonsuz

arasına yerleştiğinden bu aktivasyon fonksiyonu doyuma ulaşmayan (non-saturating) aktivasyon fonksiyonu şeklinde isimlendirilir. Doyuma ulaşmayan fonksiyonlar doyuma ulaşan fonksiyonlara göre çok daha hızlı çalışırlar. Derin öğrenme ağlarında *ReLU*, tanjant hiperbolik aktivasyon fonksiyonuna göre birkaç kat daha hızlı eğitim süresi sunar. *ReLU* ile beraber diğer aktivasyon fonksiyonlarında soruna sebep olan *vanishing gradient* problemi çözülmüştür. Çıktıların nasıl hesaplandığı denklem 2.3'de gösterilmiştir.

$$f(x) = max(0, x) \quad \textbf{(2.3)}$$

2.1.4 Sızdırılmış Doğrultulmuş Lineer Ünite (Leaky Rectified Linear Unit - Leaky ReLU)

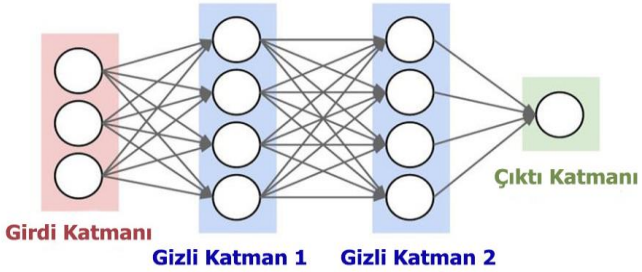
ReLU aktivasyon fonksiyonu bazı durumlarda ölü *ReLU* olarak adlandırılmıştır. Bunun nedeni *ReLU* aktivasyon fonksiyonunun negatif değerleri direkt olarak sıfıra eşitlemesidir. Çok sayıda negatif değer içeren durumlarda, sinir ağının katmanlarındaki düğümler bu sebepten dolayı aktif hale gelemez. Hata hesaplandıktan sonra, hesaplanan hataya göre ağırlıkların güncellenmesi amacıyla *geri yayılım* algoritması başlatılır. Fakat *geri yayılım* algoritması ile geriye doğru yayılan bu hata değeri sıfır ile çarpılır. Bu durum hata değerinin diğer katmanlara geçememesine yani ağırlıkların yanlış güncellenmesine neden olur. Böylece *ReLU* aktivasyon fonksiyonu işlevini kaybeder, ölür. Ortaya çıkan bu problemden dolayı *Leaky-ReLU* önerilmiştir. Bu aktivasyon fonksiyonunda, ReLU'dan farklı olarak, negatif değerler için aktivasyon fonksiyonuna eğim eklenilmesini sağlayan bir değer seçilir (Şekil 2.1(d)). Seçilecek bu değer (a_i), sabit, küçük (0.001) bir değer olursa aktivasyon fonksiyonu Leaky-ReLU şeklinde isimlendirilir. Eğer seçilecek değer uyarlanabilir olarak öğreniliyorsa aktivasyon fonksiyonu PreLU olarak isimlendirilir [15]. Çıktıların nasıl hesaplandığı denklem 2.4'de gösterilmiştir.

$$y_i = \begin{cases} x_i, Eğer\ x_i > 0 \\ a_i x_i, Eğer\ x_i \leq 0 \end{cases} \quad \textbf{(2.4)}$$

2.2 Çok Katmanlı Yapay Sinir Ağları Yapısı

Çok katmanlı yapay sinir ağları temel olarak 3 katmandan meydana gelir. Sıralı bir yapay sinir ağı modelini oluşturan bu katmanlar: Girdi Katmanı (Input Layer), Gizli Katmanlar (Hidden Layers) ve Çıktı Katmanıdır (Output Layer) [16].

Girdi Katmanı (Input Layer): Yapay sinir ağında gelen bilgiler girdi katmanında temsil edilir. Veri setine göre değişen özelliklerin her biri farklı bir düğüm olarak girdi katmanında temsil edilir. Her bir girdinin ağırlık değeri vardır. Bu girdiler gizli katmandaki düğümler ile bu ağırlıklar aracılığı ile bağlıdır. Girdiler için belirlenen ağırlık değerleri yapay sinir ağında o özelliğin önemini, ağırlığını belirtmektedir.



Şekil 2.2: Çok katmanlı YSA yapısı

Gizli Katmanlar (Hidden Layers): Gizli katmanlar, girdi katmanından gelen bilgilerin işlenip bir çıktıya dönüştürüldüğü katmanlardır. Çıktıya dönüştürme işlemi yapay sinir ağının ağırlık değerleri kullanılarak gerçekleştirilir. Gizli katman sayısı problemin zorluğuna göre çeşitlilik gösterebilmektedir. Her bir gizli katman belirlenen sayıda nörona (unit) sahiptir. Nöronlar öğrenilen özelliklerin tutulduğu birimler olarak düşünülebilir. Her bir nöron sonuca olan etkisine göre belirlenen ağırlıklara sahiptir. Bu ağırlıkların aldığı değerlere göre gizli katmandaki nöronların sonuca olan etkisi gözlenebilir. Şekil 2.2 'de gösterilen yapıda 2 gizli katman bulunmaktadır ve her iki gizli katman da 4 adet nöron içermektedir.

Çıktı Katmanı (Output Layer): Verilen girdi değerine ait çıktı değeri veya değerlerinin tutulduğu katmandır. Bu katmanda, seçilen hata hesaplama fonksiyonu kullanılarak, sinir ağı tarafından üretilen çıktı değeri ile beklenen çıktı değeri arasındaki fark hesaplanır. Hata hesaplama fonksiyonu sonucunda elde edilen hata değeri ağın performansının değerlendirilmesi, istenen sonuçlara ne kadar uzak olduğumuzun belirlenmesi için kullanılır. Elde edilen hata değerine göre ağırlıkların güncellenmesi gerekmektedir. Ağırlıkların güncellenmesi, yani yapay sinir ağının öğrenme işlemi, seçilecek optimizasyon fonksiyonu ile gerçekleştirilir. Ağırlıkların güncellenmesi için farklı yöntemler kullanılabilir. Bazı yöntemler şunlardır;

- **İleri Beslemeli Ağlar:** Akışın girdi katmanından çıktı katmanına doğru gerçekleştiği ağ yapısıdır. Verilen x girdi vektöründen $\hat{y}$ çıktısının elde edildiği yapay sinir ağlarında, x girdi vektörü sinir ağı için ilk bilgiyi sağlar. Girdinin

sağladığı bu ilk bilgi gizli katmanlarda bulunan gizli düğümlere, birimlere yayılır ve $\hat{y}$ çıktısı üretilir [12]. Bu yönteme ileri yayılım, bu yöntemi kullanan ağlara ise ileri beslemeli ağlar denir. Bir katmandan gelen nöronların çıktıları, diğer katmandaki nöronlara ağırlıklar aracılığıyla girdi olarak verilir. Denklem 2.5'de, daha önce Şekil 1.1'de gösterilen *perceptron* ile n özelliğe sahip bir x girdisi için ($i \in 1,..n$) çıktı değerinin nasıl hesaplandığı gösterilmiştir; w ağırlıkları, b yanlılık (bias) değerini ve f seçilen aktivasyon fonksiyonunu temsil etmektedir. Yanlılık değeri aktivasyon fonksiyonunun sağa veya sola kaydırılmasını sağlar. Aynı zamanda yanlılık değeri yapay sinir ağlarında aşırı öğrenmeyi (*overfitting*) engellemek adına hesaplanan çıktıya eklenen sabit veya rastsal bir değerdir.

$$z = \sum_{i=1}^{n} x_i\, w_i + b,\ a = f(z) \qquad \textbf{(2.5)}$$

Yapay sinir ağlarında öğrenme adımlarının hızlandırılması için ağırlık başlatıcı (weight initiator) yönteminin doğru seçilmesi çok önemlidir. Sinir ağında bulunan ağırlıkların başlatılması için kullanılan birçok yöntem bulunmaktadır. En çok kullanılan yöntemlerden biri ağdaki ağırlıkların rastgele değerler ile başlatılmasıdır. Ağırlıklara atanan rastgele değerler belirli aralıklarla ([-1, 1]) sınırlandırılabilir. Bunun dışında Gauss dağılımı ve varyans değerlerini kullanarak ağırlıkların daha akıllıca bir şekilde başlatılmasını sağlayan Xavier (Glorot) [15] gibi ağırlık başlatıcılar yapılan çalışmalarda sıklıkla kullanılmaktadır.

- **Geri Beslemeli Ağlar:** Akışın yalnızca ileriye doğru değil aynı zamanda geriye doğru da olabildiği ağ yapısıdır. Ağırlıkların güncellenmesi geriye yada ileriye doğru gerçekleştirilebilir. Geri besleme yöntemiyle, hesaplanan hatanın ağın geriye kalan kısımlarına dağıtılması sağlanır. Kısmi türev ve zincir kuralı kullanılarak ağırlıkların güncellenmesi gerçekleştirilir. Kullanılan bu yöntem ile hesaplanan toplam hataya, her bir ağırlığın etkisi hesaplanır ve bu hesaplamalara göre ağırlıklar güncellenir. Bölüm 2.3'de daha detaylı anlatılmaktadır.

2.3 Hata Hesaplama

YSA'larda üretilen çıktı değerlerinin, beklenen çıktı değerlerine ne kadar uzak olduğunun belirlenebilmesi için hata hesaplama işlemi gerçekleştirilir. Gerçekleştirilen bu işlemlerde YSA'nın maliyetinin (cost) ölçülmesi için sınıf sayısı ve problemin tipine bağlı olarak (sınıflandırma, regresyon); *Çapraz entropi (Cross Entropy)* ve *ortalama kare hata (Mean Squared Error), kök ortalama kare hata (Root Mean Squared Error)* gibi yöntemler sıklıkla kullanılır.

Softmax yöntemi, olasılık temelli bir fonksiyondur. Problem tipine bağlı olarak Sınıflandırma işleminin gerçekleştirileceği problemin içerdiği sınıf sayısı kadar çıktı birim oluşturulur. Oluşturulan değerler için üretilen olasılıklar ile en çok benzerliğe sahip olan düğüm belirlenmeye çalışılır [17]. Örneğin 5 sınıftan oluşan bir problem için 5 düğüm oluşturulur ve çıktı katmanından önceki katmanda bulunan düğümler ile ağırlıklar denklem 2.6' da görüldüğü üzere çarpılır ve böylece her bir *i* sınıfı için a_i aktivasyon değerleri hesaplanır. Daha sonra her bir *i* sınıfı için 0 ile 1 arasındaki olasılık değeri, p_i, denklem 2.7' de gösterildiği hesaplanır ve en yüksek p_i değerine sahip sınıf tahmin edilen sınıf olarak belirlenir.

$$a_i = \sum_j z_j W_{ji} \quad \textbf{(2.6)}$$

$$p_i = \frac{exp(a_i)}{\sum_{k=1}^{5} exp(a_j)} \quad \textbf{(2.7)}$$

Olasılık değerlerinin hesaplanmasının ardından "Cross Entropy" yöntemi ile her bir girdi için hata hesaplanması denklem 2.8'e göre gerçekleştirilir ($\hat{y}$: hesaplanan değer, *y*: gerçek değer) [10]. Toplam hata (maliyet) denklem 2.9'a göre hesaplanır (*m*: örnek sayısı).

$$L(\hat{y}, y) = -y^{(i)} log\left(\hat{y}^{(i)}\right) - \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right) \quad \textbf{(2.8)}$$

$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) \quad \textbf{(2.9)}$$

Yapay sinir ağlarının sınıflandırma başarılarını arttırmak için bazen *düzenlileştirme* (*regularization*) yöntemleri kullanılabilir. Maliyet fonksiyonunun sonuna bir ceza eklenerek yapay sinir ağı modelleri genelleştirilmeye çalışılır, yani eğitilen yapay sinir ağının sadece eğitim örnekleri için değil daha önce hiç görmediği örnekler için de başarılı sonuçlar vermesi hedeflenir. Aşırı öğrenmeyi (*overfitting*) engellemek

amacıyla en sık kullanılan düzenlileştirme yöntemleri şunlardır: *L2 Düzenlileştirme (Ridge Regression), L1 Düzenlileştirme (Lasso Regression)* ve *Seyreltme (Dropout).*

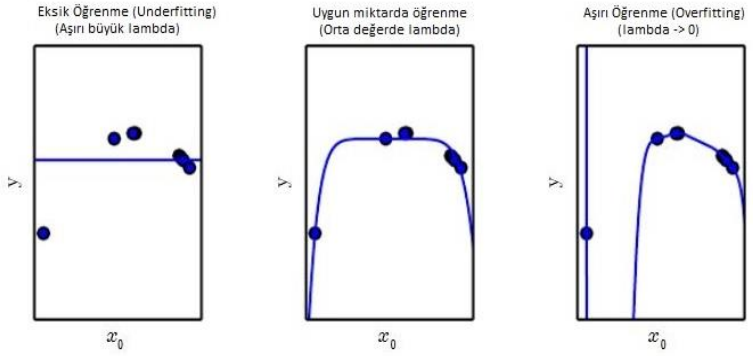
- **L2 ve L1 Düzenlileştirme:** L1 ve L2 düzenlileştirme fonksiyonlarında düzenlileştirme işlemi toplam maliyet fonksiyonuna eklenen ekstra hesaplamalar ile gerçekleştirilir. Denklem 2.10'da L2 düzenlileştirme, denklem 2.11'de ise L1 düzenlileştirme işlemlerinin nasıl gerçekleştirildiği gösterilmektedir. Bu denklemlerde W ağırlıkları, k ise toplam ağırlık sayısını temsil etmektedir. Bu iki yöntemde de lambda değerinin doğru seçilmesi çok önemlidir. Şekil 2.3'de gösterildiği gibi, lambda çok büyük seçilirse eksik öğrenme (underfitting) durumunun ortaya çıkmasına sebep olabilir. Çünkü büyük lambda değeri maliyet fonksiyonunun değerini yükseltir. Maliyet fonksiyonu sonucu minimize etmeye çalıştığından ağırlık değerlerini çok fazla küçültmeye, sıfıra doğru yaklaştırmaya başlayacaktır. Bu durumda verilen farklı girdiler için sınıflandırma işlemi sonucunda elde edilen sonuçlar birbirine çok benzer, grafiğe dökülmek istendiğinde ise neredeyse düz bir çizgi halinde gözükecektir. Lambda değerinin çok küçük seçilmesi de aşırı öğrenme durumunun ortaya çıkmasına sebep olabilir. Bu yüzden lambda değerleri eksik öğrenme ve aşırı öğrenmeyi engelleyecek şekilde seçilmelidir.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) + \lambda \sum_{j=1}^{k} W_j^2 \qquad \textbf{(2.10)}$$

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) + \lambda \sum_{j=1}^{k} \left|W_j\right| \qquad \textbf{(2.11)}$$

- **Seyreltme (Dropout) Yöntemi:** Seyreltme yöntemi [18] yapay sinir ağlarında aşırı öğrenmeyi azaltmak için sıklıkla kullanılan bir düzenlileştirme yöntemidir. Seyreltme yöntemi sahip olduğu seyreltme oranı (dropout rate) hiper-parametresine göre gerçekleştirilir. Amaç, 0 ve 1'den oluşan maskeler oluşturarak sinir ağını oluşturan bazı düğümlerin öğrenme, optimizasyon işlemine dahil edilmesini engellemek, bu düğümlere gelen ve giden bağlantıları, ağırlıkları sinir ağından çıkarmaktır [12]. Örneğin: 3 gizli katmandan oluşan yapay sinir ağının 2. gizli katmanında 100 düğüm bulunsun ve bu katmanda seyreltme işlemi yapılmak istensin. Seyreltme oranının 0.4 seçildiğini varsayarsak bu yapay sinir ağının 2. gizli katmanından rastgele seçilen 40 düğümün 1. ve 3. gizli katmanlar ile olan bağlantıları düşürülür. Bu

sayede, bu düğümler öğrenme, optimizasyon veya geri yayılma işlemlerine katılmaz.



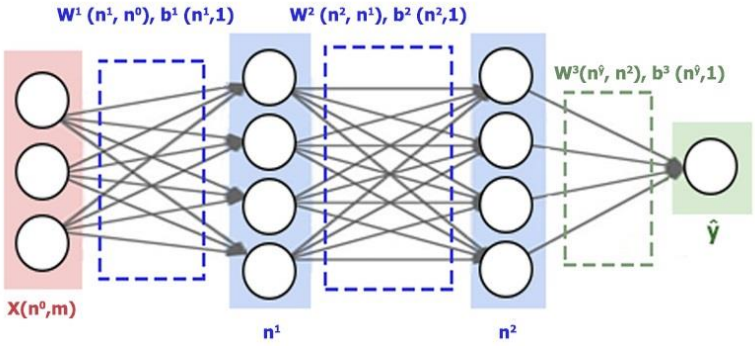
Şekil 2.3: Lambda değerlerinin öğrenmeye olan etkisi [12]

2.4 Optimizasyon Metodları

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 [2]. Geri Yayılım Algoritması ile sinir ağındaki her bir ağırlığın, hesaplanan hataya olan etkisini hesaplamak için gradyan tabanlı Stokastik Gradyan Azaltma [19], RMS-Prop, Adam [20] ve Adadelta [21] metodları sıklıkla kullanılmaktadır.

2.4.1 Gradyan Azaltma (Gradient Descent)

Yapay sinir ağlarında hata optimizasyonu için kullanılan yöntemlerin en temelinde Gradyan azaltma algoritması bulunmaktadır. Bu optimizasyon yöntemlerinin en temelindeki amaç, toplam hatayı geriye doğru yayarak, her bir ağırlığın toplam hataya olan etkisinin hesaplanmasıdır. Bu şekilde ağın optimizasyonu, öğrenme işlemi gerçekleştirilir. Seçilen optimizasyon yöntemine göre optimizasyon süreçleri değişse de kullanılan optimizasyon yöntemlerinin temelinde türev, zincir alma kuralı ve geri yayılım bulunmaktadır.



Şekil 2.4: İki gizli katmanlı bir YSA parametreleri

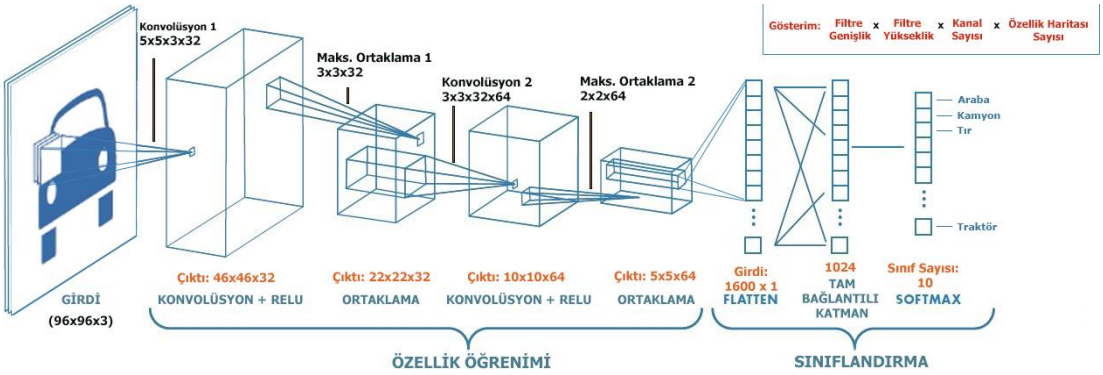
Sinir ağının toplam maliyeti/hatası *(J)* denklem 2.9'a göre hesaplanır. Gradyan azaltma metodunda hesaplanan toplam hata geriye doğru yayılarak, her bir ağırlığın toplam hataya olan etkisi kısmi türev ve zincir kuralı kullanılarak hesaplanır. Şekil 2.4 'de gösterilen iki katmanlı sinir ağında *X*, girdi matrisini, W^1, ilk gizli katman ağırlık matrisini, b^1, ilk gizli katman için yanlılık vektörünü, W^2, ikinci gizli katman ağırlık matrisini, b^2, bu katman için yanlılık vektörünü, $\hat{y}$ ise çıktı değerlerini göstermektedir. Şekil 2.4'deki n^0 özellik sayısını, n^1 ilk gizli katman nöron sayısını, n^2 ikinci gizli katman nöron sayısını, *m* ise örnek sayısını ifade etmektedir. Mevcut parametreler için maliyetin, $J(W^1, b^1, W^2, b^2)$, sinir ağında geriye doğru yayılması, ağırlık ve yanlılık değerlerinin hata üzerindeki etkilerinin hesaplanması gerekmektedir. Bunun için, toplam hatanın ağdaki parametrelerin her birine göre kısmi değişim oranı ($\partial J/\partial W^2$, $\partial J/\partial b2$, $\partial J/\partial W^1$, $\partial J/\partial b^1$) zincir türev kuralı ile bulunur [12]. Her hangi bir *k* katmanında *g*radyanı hesaplanan ağırlıklar belirlenen bir öğrenme oranı, α, ile güncellenir (denklem 2.12).

$$\boldsymbol{W^{(k)} = W^{(k)} - \alpha \frac{\partial J}{\partial W^{(k)}}} \qquad \textbf{(2.12)}$$

Tüm girdilerin kullanılmasının ardından toplam hata belirlenen bir değerin altına ulaşana kadar toplam hatanın hesaplanması, mevcut parametrelerin hata üzerindeki etkisinin hesaplanması ve parametrelerin güncellenmesi işlemi tekrar ettirilir. Bazen belirlenen sayıda iterasyonun tamamlanması veya CPU zamanı da durdurma kriteri olarak verilebilir.

3. KONVOLÜSYONEL SİNİR AĞLARI

Konvolüsyonel sinir ağları (KSA) [22, 23], görüntü sınıflandırma, nesne konumlandırma (localization), nesne algılama gibi bilgisayarlı görü çalışmalarında çokça kullanılmakta olan bir derin öğrenme mimarisidir. KSA, bilgisayarlı görü çalışmaları için hazırlanan birçok hazır veri setinde başarılı ve rekabetçi sonuçlar vermektedir [12, 23]. KSA, kendisini oluşturan katmanların en az birinde, filtrelerin girdiler üzerinde dolaşması (konvolüsyon) işleminin uygulandığı ve sonucunda özellik çıkarımının gerçekleştirildiği derin sinir ağlarıdır. KSA, anlatılan bu özelliği (konvolüsyon) ile diğer YSA'lardan ayrılır. KSA'lar bir tane girdi katmanı, çok sayıda konvolüsyon ve ortaklama katmanları ile tam bağlantılı katmandan oluşur. KSA yapısı şekil 3.1'de gösterilmiştir.



Şekil 3.1: Konvolüsyonel sinir ağlarının yapısı

3.1 Girdi Katmanı

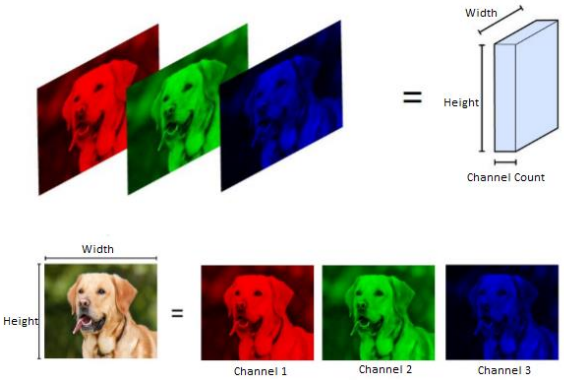
Konvolüsyonel sinir ağının ilk katmanını oluşturur. Sinir ağının bir çıktı elde edebilmesi için gerekli olan görüntü girdi olarak verilir. Verilen girdinin bilgisayar tarafından anlaşılabilmesi için resmin pikselleri sayısal ifadelere dönüştürülmelidir. Verilen girdinin renkli bir resim olduğunu kabul edersek her bir piksel 0-255 arasında sayısal değerler ile ifade edilir.

Girdinin her bir pikselini temsil eden sayısal değerler matrislerde saklanır. Görüntünün genişliği, yüksekliği, renkli olup olmadığı gibi özellikler, görüntülerin saklandığı matrisin boyutlarını değiştirmektedir. Örneğin: 28x28 piksel boyutunda gri tonlamalı bir resim, girdi için 28x28x1 boyutunda bir matris yeterlidir. Vektör haline getirirsek 28x28x1 = 784 elemanlı bir vektör ile bu girdi temsil edilebilir. Fakat aynı boyutlardaki resmin renkli olduğunu düşünürsek 28x28x3 boyutunda bir matris kullanmamız gerekir. Yine bu matrisi vektör haline getirmek istersek 28x28x3 = 2352 elemanlı bir vektör elde etmiş oluruz. Renkli olan girdiler direkt olarak 3 boyutta temsil edilmektedir. Her bir boyut, kanal (channel) olarak isimlendirilir. Bunun nedeni

her bir girdinin RGB (Red, Green, Blue) kanallarının olmasındandır. Yani renkli bir girdinin her bir pikseli "Red" kanalında ayrı, "Blue" kanalında ayrı, "Green" kanalında ayrı bir sayısal değer ile ifade edilmektedir (Şekil 3.2). Bu yüzden seçilen girdi boyutları sinir ağının hızı ve çıktıların doğruluğu için çok önemlidir. Büyük girdi boyutları, çok fazla bellek kullanımına neden olup, ağın eğitilmesi sırasında yapılacak hesaplamaların yavaşlamasına neden olur. Fakat çıktıların doğru bir şekilde tahmin edilebilmesi için daha fazla özellik çıkarılabileceğinden etkili sonuçlar verebilir. Aynı şekilde küçük girdi boyutları, daha az bellek kullanımı sağlayıp, ağın eğitilmesi için yapılan hesaplamaları hızlandırabilir [2]. Fakat tahmin edilen çıktıların doğruluk oranı istenilen kadar başarılı olmayabilir. Bu yüzden sinir ağının başarısı ve hesaplama maliyeti gibi kriterler göz önünde bulundurularak doğru bir girdi boyutu seçilmelidir.

3.2 Konvolüsyon Katmanı

Bu katman KSA'ların en önemli katmanlarından birini oluşturur. Bir önceki katmandan gelen girdiler üzerinde seçilen filtreler (kernel, receptive field size) uygulanarak yeni özellik kümeleri oluşturulur. Filtre boyutları değişkenlik gösterebilir ve matris yapısında tutulur (2x2, 3x3, 5x5 vb). Seçilen filtreler (ağırlık matrisleri) ile verilen girdi üzerinde dolaşılarak farklı özellikler elde edilmeye çalışılır. Girdiye uygulanan konvolüsyon filtreleri sonucunda her biri farklı özelliği temsil etmeye çalışan özellik haritaları elde edilir.



Şekil 3.2: Renkli girdiler için kanal, genişlik ve yükseklik kavramlarının gösterimi

Girdiler üzerine uygulanacak filtrelerin nasıl seçileceği ve boyutlarının ne olacağı çok önemli bir noktadır. Filtrelerin genişlik ve yükseklik değerlerinin çok yüksek seçilmesi, girdi boyutunun çok hızlı bir şekilde küçülmesine neden olabilir. Düşük girdi boyutları sinir ağındaki hesaplama hızını arttırırken, girdi üzerinden elde edilebilecek ayırt edici özelliklerin kaybedilmesine sebep olabilir. Konvolüsyonel sinir ağlarındaki filtreler, sinir ağlarındaki ağırlık matrislerine denk olarak düşünülebilir.

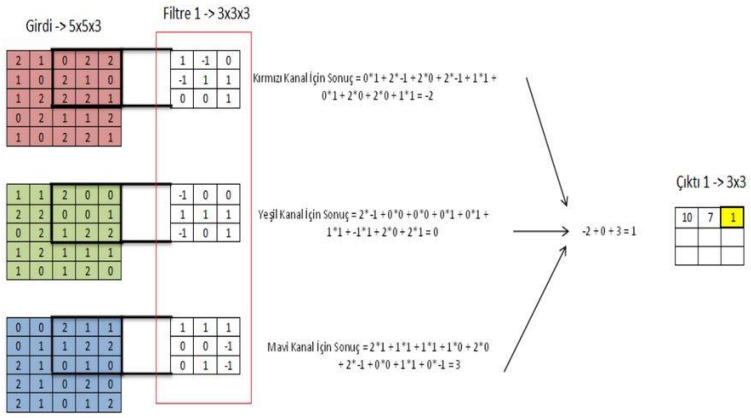
Yapay sinir ağlarında olduğu gibi konvolüsyonel sinir ağlarında da ağırlıklar yani filtreler hesaplanan hataya göre belirlenen çevrimlerde güncellenir. Güncellenen filtre değerleri ile öğrenme işlemi devam ettirilmiş olur. Güncellenen filtre ağırlıkları, girdi üzerinde hangi özelliklerin daha önemli olduğunu belirtir. Öğrenilen bu bilgiler ilerleyen katmanlarda ağın geriye kalan kısımları ile paylaşılır (weight sharing). Güncellenen ağırlıklar sonucunda oluşan filtreler girdi üzerinde yeni özelliklerin keşfedilmesini sağlar.

Konvolüsyon katmanında önemli olan bazı kavramlar vardır ve bu kavramlar şunlardır:

- **Filtre boyutu:** Girdi üzerine uygulanacak filtrenin yükseklik ve genişlik değerlerini temsil eder. Genelde kare matris olarak seçilir.
- **Aralık Değeri (Stride):** Filtre matrisinin, girdi üzerinde dolaşırken kaç adım kaydırılacağını belirler.
- **Dış Boşluk Sayısı (Padding):** Bu değer girdi matrisinin dışına eklenecek olan 0'lardan oluşan değer sayısını belirtir. Örneğin: 3x3 boyutunda bir girdiye padding = 1 uygulanırsa filtrenin yeni boyutu 5x5 olur. Bu özellik girdi boyutu ile çıktı boyutunun aynı kalmasının istendiği durumlarda kullanılır.
- **Özellik Haritaları (Feature Maps):** Girdi üzerine filtreler uygulandıktan sonra elde edilen özellik haritalarının sayısını belirtir.
- **Kanal Sayısı:** Girdi ve filtrelerin kaç kanaldan oluştuğunu temsil eder. Girdinin kanal sayısı ile girdiye uygulanacak olan filtrelerin kanal sayısı birbirine eşit olmalıdır. Örneğin: Girdi, 28x28x3 boyutunda ise yani 3 kanala sahip ise filtre ?x?x3 boyutunda olmalıdır (?: opsiyonel değerleri temsil eder, 3x3, 5x5, 7x7 vb).

Konvolüsyon işleminde yeni çıktılar, noktasal çarpım ve toplama işlemleri ile elde edilir. Konvolüsyon işleminde, belirlenen aralık hiper-parametresine (stride) göre konvolüsyon filtresi girdi üzerinde gezinir. Bu işlem sırasında, konvolüsyon filtresinin bulunduğu yere karşılık gelen alandaki girdi ağırlıkları ile konvolüsyon filtresinin ağırlıkları çarpılıp, toplanır. Bu işlem girdi boyunca her bir kanal için devam eder. Her bir kanal için elde edilen çarpımların toplamı alınarak, çıktı verisinde karşılık gelen noktaya bu toplam değer yazılır. Girdi üzerindeki her bir katman için farklı filtre

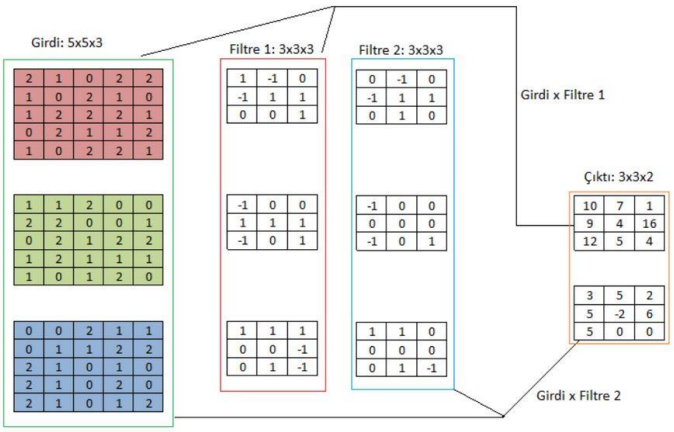
katsayıları seçilebilir. Örneğin: Konvolüsyon işlemi için kullanılacak olan girdinin genişliği 640 piksel, çıktının genişliği ise 638 piksel olsun. Her iki görüntü yüksekliği de 480 piksel olsun. Girdi görüntüsünden çıktıya dönüşüm gerçekleştirilmesi için konvolüsyon işlemi uygulanırsa; 638 x 480 x 3 = 918.720 kayan noktalı işlem gerekir. Aynı dönüşüm ağırlıkların tamamının birbirine bağlı olduğu bir sistemde matris çarpımı ile 94 milyardan (640 x 480 x 638 x 480) fazla hesaplama gerektirir. Bu durum konvolüsyon işlemini çok daha verimli hale getirir.



Şekil 3.3: Girdi için 3 kanaldan oluşan filtrenin uygulanması (aralık = 1, dış boşluk = 0)

Denklem 3.1'e bakıldığında girdi matrisi ve ağırlık matrislerinin çarpılmasıyla yeni oluşan özellik haritalarının nasıl hesaplandığı görülmektedir. Bu notasyonda, $W^{[i-1]}$, i-1. Katmanda bulunan her bir filtre için oluşturulan ağırlıkların saklandığı çok boyutlu bir matristir $W^{[i-1]} = [W_1^{[i-1]}, W_2^{[i-1]}, \dots, W_k^{[i-1]}]$. $Z^{[i-1]}$ matrisinde ise filtre ağırlık matrisi ile girdi ağırlık matrislerinin çarpılıp, önyargı vektörü ile toplanması sonucunda elde edilen yeni özellik haritaları saklanır ($Z^{[i-1]} = [Z_1^{[i-1]}, Z_2^{[i-1]}, \dots, Z_k^{[i-1]}]$). Bu notasyonda k değeri kernel, filtre sayılarını belirtmektedir. Verilen girdiler için konvolüsyon işleminin nasıl uygulandığı Şekil 3.3 ve Şekil 3.4 'de gösterilmiştir.

$$\boldsymbol{Z^{[i]} = X^{[i]} W^{[i-1]} + b^{[i-1]}} \tag{3.1}$$



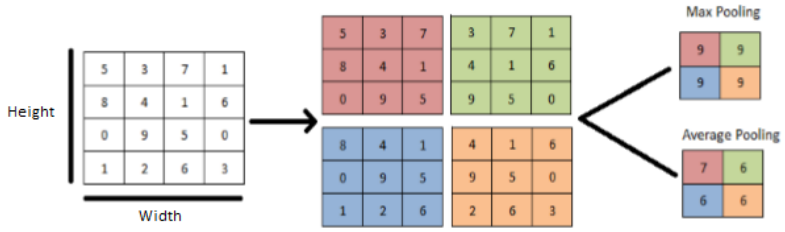
Şekil 3.4: Birden fazla filtre için konvolüsyon işlemin uygulanması ve çıktılar (aralık = 1, dış boşluk = 0)

3.3 Ortaklama Katmanı

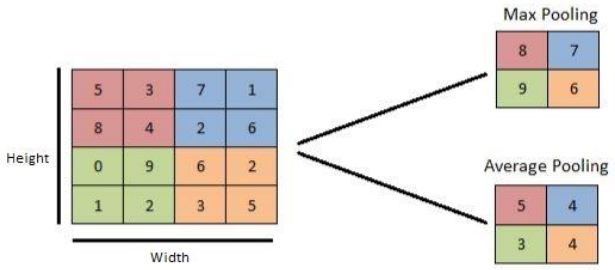
Ortaklama (pooling) işlemi ya da bazı örneklerde alt örnekleme (subsampling) olarak geçen yöntem konvolüsyonel sinir ağlarında girdi ve özelliklerin boyutunun düşürülmesi için kullanılır. Girdi, ortaklama işlemine tabi tutulduğunda belirli bir matematiksel formüle göre yükseklik ve genişlik değerleri azalır. Özellik düşürme işlemi, özellik çıkarımı (öğrenimi) işlemleri sonucunda elde edilen özelliklerin kaybına neden olsa da KSA'nın eğitim süresini azaltır. Ortaklama işleminin en ilginç yanı öğrenilecek, elde edilen hataya göre güncellenecek hiper-parametrelerin olmayışıdır. Ortaklama katmanı, filtre boyutu, ortaklama yöntemi ve aralık değeri hiper-parametrelerine sahiptir. Fakat bu hiper-parametreler ağın optimizasyonuna dahil edilmez.

Ortaklama işleminin uygulanması için farklı yöntemler seçilebilir. Bu yöntemler ortaklama tipi, ortaklama yöntemi olarak adlandırılabilir ve KSA optimizasyonunda ortaklama katmanında seçilmesi gereken bir hiper-parametredir. Maksimum ortaklama (max pooling) ve ortalama ortaklama (average pooling) yöntemleri KSA'da sıklıkla kullanılmaktadır. Konvolüsyon katmanında olduğu gibi seçilen ortaklama filtreleri, girdiler üzerinde dolaşarak çıktıları elde eder. Ortalama ortaklama işleminde, seçilen aralık hiper-parametresine (stride) göre filtre girdi üzerinde gezinir. Bu işlem sırasında, girdi üzerinde ortaklama filtresinin bulunduğu yere karşılık gelen alandaki değerlerin ortalaması hesaplanıp, çıktı üzerinde karşılık gelen bölgeye yerleştirilir. Bu işlem girdi boyunca devam eder. Aynı işlem maksimum ortaklama işlemi için de geçerlidir. Fakat bu sefer elemanların ortalaması yerine girdide filtrenin bulunduğu

alana karşılık gelen kısımdaki maksimum değer çıktı olarak verilir. Farklı aralık (stride) değerleri için girdilere "Max ve Average Pooling" işlemlerinin uygulanması Şekil 3.5 ve 3.6 'da gösterilmiştir.



Şekil 3.5: 3x3 Filtre boyutu ve 1 aralık (stride) değeri için maksimum ve ortalama ortaklama çıktıları



Şekil 3.6: 2x2 Filtre boyutu ve 2 adım (stride) değeri için maksimum ve ortalama ortaklama çıktıları

3.4 Tam Bağlantılı Katman

Konvolüsyonel sinir ağlarında özellik çıkarımı işleminden sonra, elde edilen özelliklerin sınıflandırılması için kullanılan son katmanları içerir. Tam bağlantılı katmandan önce, verilen girdi üzerinden özelliklerin çıkarılması, öğrenilmesi için konvolüsyon, ortaklama, normalizasyon ve aktivasyon işlemleri ağın yapısına bağlı olarak birkaç defa tekrarlanır. Elde edilen özellikler sonucunda artık bir tahmin işleminin gerçekleştirilmesi gerekmektedir. Tam bağlantılı katman kendinden önce gelen diğer katmanlarla tamamen bağlıdır. Bundan sonra kullanılacak olan yapı çıkartılan özelliklerin ağırlıklandırılması ve bir çıktının tahmin edilmesi işlemini gerçekleştirecektir. İstenilen doğruluk değeri elde edilene veya sonlandırma kriteri (iterasyon sayısı) sağlanana kadar ağırlıklar hesaplanan hataya göre güncellenir. Tam bağlantılı katmanın yapısı, yapay sinir ağlarının yapısı ile benzerlik göstermektedir. Bir giriş katmanı, isteğe bağlı olarak bir ya da birden fazla gizli katman ve son olarak çıktı katmanından oluşur.

Tam bağlantılı katmana gelene kadar elde edilen özelliklerin boyutunun 16x16x128 olduğunu varsayalım. Yani 16x16 boyunda 128 tane özellik elde edilmiş olsun. Bu matrisi bir vektör haline çevirmemiz gerekir. Bu yüzden elde edilen matrisi 16x16x128=32.768x1'lik bir vektöre çevirerek giriş katmanına veririz. Bu işlemin adına "Flattening" denir. Aynı zamanda giriş katmanı ile çıktı katmanı arasına bir tane gizli katman eklemek istersek ve bu katmanın 512x1 boyutunda bir vektör olduğunu varsayarsak toplamda 32.768x512 boyutunda bir ağırlık matrisi elde edilmiş olur [2]. Son olarak ta tahmin edilecek sınıf boyutunda bir çıktı katmanı eklenir.

Görüldüğü gibi elde edilen ağın toplam parametre sayısı yükseldiğinden, eğitilmesi için yüksek hesaplama gücü gerekmektedir. Bu yüzden tamamen bağlı katmana gelene kadar olan bölümde, yani özellik çıkarımının gerçekleştiği katmanlarda giriş katmanından ileriye doğru giderken kanal sayısı arttırılmalı, yükseklik x genişlik sayısı azaltılmalıdır. Yani girdinin yükseklik ve genişlik değerleri küçültülmeli, fakat özellik haritası sayısı arttırılmalıdır.

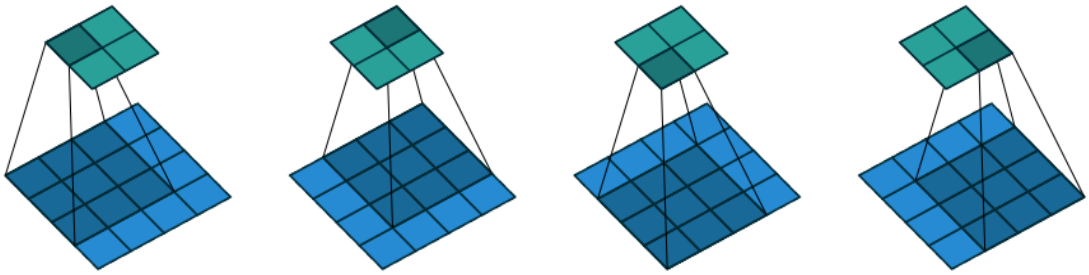
3.5 Konvolüsyon (Convolution) Aritmetiği

Bölüm 3.2 'de konvolüsyon işleminin ne olduğundan ve nasıl çalıştığından bahsedildi. Konvolüsyon işlemi yapıldıktan sonra elde edilen çıktıların boyutu neye göre belirlenir, konvolüsyon katmanında aralık (stride) ve dış boşluk (padding) değerleri ne için kullanılır ve konvolüsyon işleminden sonra elde edilen çıktının boyutunun değişmesini istemediğimiz durumlarda neler yapılması gerektiği bu kısımda anlatılacaktır.

3.5.1 Sıfır dış boşluk (padding) ve bir adım değeri

Şekil 3.7'de dış boşluk (padding) değerinin sıfır, adım (stride) değerinin 1 olduğu durum gösterilmiştir. Bu durumda, girdiye konvolüsyon işlemi uygulandıktan sonra elde edilecek çıktının boyutu denklem 3.2'de [24] gösterilen formüle göre hesaplanır (ç: çıktı boyutu, g: girdi boyutu, f: filtre boyutu).

Ç = (G – F) + 1 **(3.2)**



Şekil 3.7: 4x4 boyutunda bir girdi 3x3'lük bir filtre ile konvolüsyon işlemine sokuluyor (dış boşluk = 0, stride = 1, çıktı boyutu = 2x2) [24]

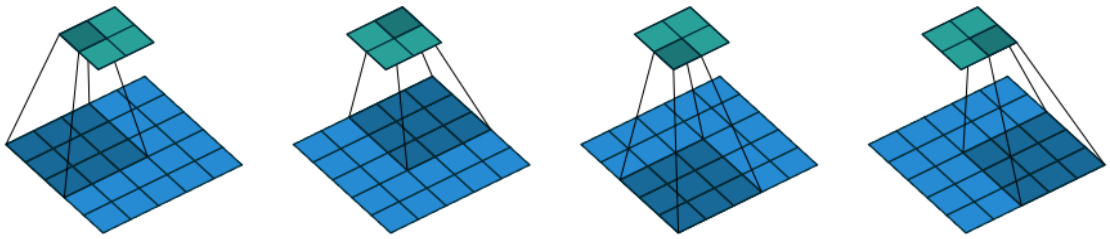
Şekil 3.7'de gösterilen örnek için elde edilecek çıktı Denklem 3.2'ye göre hesaplanır:

Çıktı Boyutu = (4 – 3) + 1 = 2

3.5.2 Sıfır dış boşluk (padding) ve birden farklı adım değeri

Dış boşluk (padding) değerini sıfır kabul edip, aralık sayısının değişken olduğunu kabul edersek verilen girdi için elde edilecek çıktı değerini Denklem 3.3'te [24] gösterilen formüle göre hesaplayabiliriz (ç: çıktı boyutu, g: girdi boyutu, f: filtre boyutu, a: aralık sayısı).

$$ç = \left\lfloor \frac{[g-f]}{a} \right\rfloor + 1 \quad \textbf{(3.3)}$$



Şekil 3.8: 5x5 boyutunda bir girdi 3x3'lük bir filtre ile konvolüsyon işlemine sokuluyor (dış boşluk = 0, stride = 2, çıktı boyutu = 2x2) [24]

Şekil 3.8'de gösterilen örnek için elde edilecek çıktı Denklem 3.3'e göre hesaplanır:

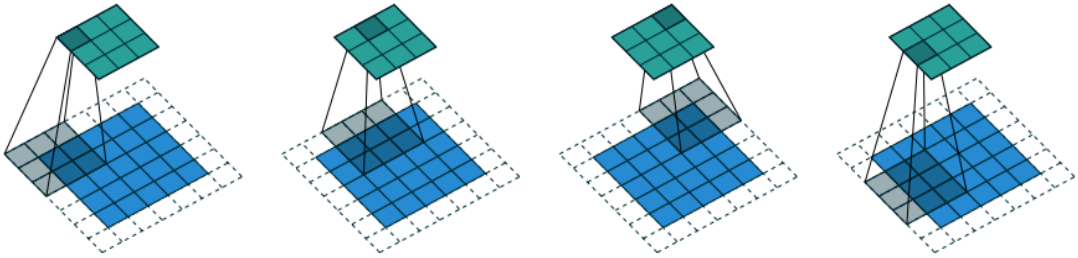
$$\text{Çıktı Boyutu} = \frac{[5-3]}{2} + 1 = 2$$

3.5.3 Sıfırdan farklı dış boşluk ve aralık sayısı

Girdiye uygulanan konvolüsyon işleminden sonra elde edilecek olan çıktı boyutunun hesaplanabileceği en genel formüldür. Konvolüsyon işleminin ardından elde edilen

çıktı boyutunun hesaplandığı genel formül denklem 3.4'te [24] verilmiştir (ç: çıktı boyutu, g:girdi boyutu, p: dış boşluk sayısı, f: filtre boyutu, a: aralık sayısı).

$$ç = \left\lfloor \frac{[g+2*p-f]}{a} \right\rfloor + 1 \quad \textbf{(3.4)}$$



Şekil 3.9: 5x5 boyutunda bir girdiye 1x1'lik kenar ekleniyor ve 3x3'lük bir filtre ile konvolüsyon işlemine sokuluyor. (dış boşluk = 1, stride = 2, çıktı boyutu = 3x3) [24]

Şekil 3.9'da gösterilen örnek için elde edilecek çıktı Denklem 3.4'e göre hesaplanır:

$$\text{Çıktı Boyutu} = \frac{[5+(2*1)-3]}{2} + 1 = 3$$

Görüldüğü gibi konvolüsyon işleminde farklı parametre değerleri için farklı çıktı boyutları elde edilmektedir. Elde edilecek olan çıktının mimari kurulurken hesaplanması, bilinmesi büyük bir önem arz etmektedir. Verilen girdi için çıktı katmanında elde edilecek boyutun hesaplanması için denklem 3.4 kullanılabilir. Bu denklem elde edilecek çıktının hesaplanması için kullanılan formüllerin en genel halidir. Denklem 3.4, çıktı boyutunu doğrudan etkileyen, girdi boyutu, filtre boyutu, aralık sayısı ve iç boşluk sayısı gibi bütün parametreleri içermektedir.

3.5.4 Girdi boyutunun korunması

Bazı durumlarda konvolüsyon işleminden elde edilen çıktının girdi ile aynı boyutta olmasını isteriz. Böyle durumlarda dış boşluk (padding) parametresini kullanırız. Bu duruma "same padding" denir. Girdi ile çıktının aynı boyutta olması için kullanılacak olan iç boşluk parametresi belirli bir denkleme göre hesaplanır. Bu denklem aralık sayısının bir olduğu durumlar için doğru sonucu vermektedir. Çünkü filtre boyutu ve aralık sayısı gibi değişkenlerin farklı olması sonucu değiştirebilir (g: girdi boyutu, p: dış boşluk sayısı, f: filtre boyutu).

$$g + 2 * p - f + 1 = g$$

$$2 * p = f - 1$$

$$p = \frac{f-1}{2} \quad \textbf{(3.5)}$$

Aralık sayısının 1 olduğu durumlarda dış boşluk (padding) sayısı denklem 3.5'e göre hesaplanır. Örneğin: 5x5 boyutundaki bir girdiye, aralık sayısı (stride) 1 iken, 3x3 boyutunda bir filtre uygulandığında tekrar 5x5 boyutunda bir çıktı elde etmek istersek kullanmamız gereken dış boşluk sayısı Denklem 3.5'e göre hesaplanabilir:

$$p = \frac{3-1}{2} = 1$$

Bulunan "p" değerini Denklem 3.4'te yerine koyarsak:

$$\text{Çıktı Boyutu} = \frac{[5+(2*1)-3]}{1} + 1 = 5$$

sonucu elde edilir. Görüldüğü gibi girdi boyutu ile çıktı boyutu aynı değerdedir.

3.6 Ortaklama (Pooling) Aritmetiği

Havuzlama işleminde iç boşluk(padding) değeri kullanılmadığından sadece filtre boyutu, girdi boyutu ve adım sayısı değerleri kullanılır. Konvolüsyon işlemindeki genel formülden (Denklem 3.4) tek farkı dış boşluk (padding) değerinin olmayışıdır. Bu nedenle ortaklama katmanında kullanılan girdi boyutu, filtre boyutu ve aralık sayısı değerleri için elde edilecek çıktının boyutu denklem 3.6'ya göre hesaplanır. Bu denklem 3.3 ile aynıdır (g: girdi boyutu, f: filtre boyutu, a: adım sayısı).

$$\text{ç} = \left\lfloor \frac{[g-f]}{a} \right\rfloor + 1 \qquad \textbf{(3.6)}$$

Örneğin: 32x32 boyutundaki bir girdi için, aralık değeri (stride) 2 olan 2x2 ortaklama boyutunda (pooling size) bir filtre uygulanırsa, elde edilecek çıktının boyutu denklem 3.6'ya göre hesaplanır:

$$\text{Çıktı Boyutu} = \frac{[32-2]}{2} + 1 = 16$$

3.7. Konvolüsyonel Sinir Ağı Modelleri

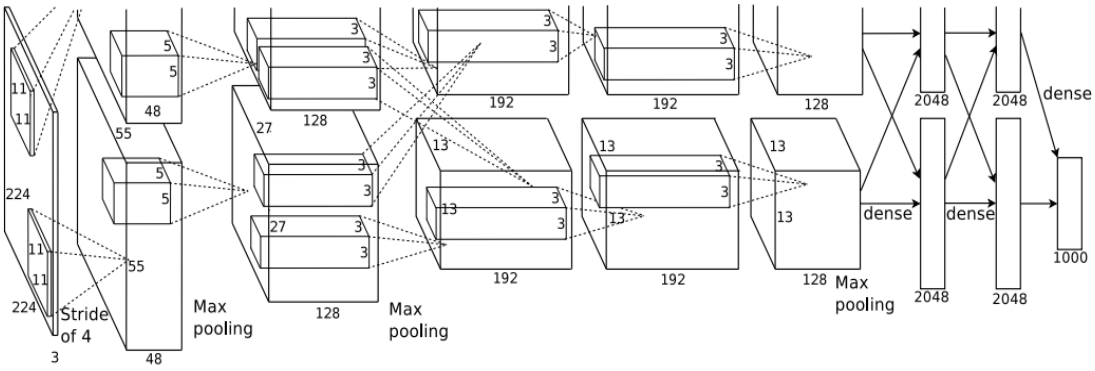
Bu bölümde konvolüsyonel sinir ağlarının eğitilmesi ve daha başarılı sonuçlar elde edebilmesi için geliştirilen modeller anlatılacaktır. Bilinen ilk konvolüsyonel sinir ağı modelinden başlanarak, büyük ölçekli resimlerin daha iyi bir şekilde tanınması için geliştirilen ve LSVRC yarışmasında en az hata oranlarıyla birincilik elde eden modellerin yapısından bahsedilecektir.

3.7.1 LeNet

1990'lı yıllarda Yann LeCun tarafından konvolüsyonel sinir ağı uygulamalarında kullanmak için geliştirilen başarılı bir modeldir. LeNet modeli posta kodları, el yazısı vb. gibi uygulamalarda başarılı sonuçlar elde edebilmiş bir konvolüsyonel sinir ağı modelidir. 1995 yılında Yann LeCun ve Yoshua Bengio tarafından yayınlanan "Convolutional networks for images, speech, and time series" isimli çalışmada LeNet-5 modeli resmedilmiştir. Bu model 2 konvolüsyon(convolution), 2 havuzlama(pooling, subsampling) ve 1 tam bağlantılı katman(fully connected layer) olmak üzere toplam 5 katmandan oluşmaktadır [8]. Verilen el yazısı girdilerinin doğru sınıflandırılması için başarılı sonuçlar elde etmiştir.

3.7.2 AlexNet

AlexNet 2012 yılında konvolüsyonel sinir ağlarını popülerleştiren bir çalışma olmuştur. ILSVRC-2012 yarışmasında sunulan AlexNet modeli, Alex Krizhevsky, Ilya Sutskever ve Geoff Hinton tarafından geliştirilmiştir. Geliştirilen bu model, en iyi 5 test hata oranında (top-5 error) 15.3%'lük bir değer elde etmiştir. Aynı yarışmada en iyi ikinci yarışmacı ise 26.2%'lik bir değer elde etmiştir. 60 milyon parametre ve 650.000 nöron içeren sinir ağı; 5 konvolüsyon katmanı (convolution layer), bunların birçoğunu takip eden maksimum havuzlama katmanı (max pooling layer) ve 3 tam bağlantılı (fully connected layer) katmandan oluşmaktadır [9]. ImageNet veri seti 1000 farklı örnek resim sınıfı içerdiğinden, çıktı katmanı 1000 birimden oluşmaktadır. 1000 farklı nesne sınıflandırılmaktadır. Elde edilen model, LeNet'e çok benzer bir mimariye sahip gibi görünmesine rağmen daha derin ve daha büyük bir modeldir. Diğer modellerden farklı olarak bu modelde iki farklı GPU (GTX 580) arasında bir işleyiş görülmektedir. Modelin üst kısmını farklı bir GPU, alt kısmını farklı bir GPU eğitmektedir. Bu iki GPU sadece belirli katmanlarda iletişim kurmaktadır [9]. Bu durum modelin daha hızlı eğitilmesi için avantaj sağlamaktadır. Aktivasyon fonksiyonu olarak ReLU, aşırı öğrenmeyi (overfitting) engellemek için seyreltme (dropout) yöntemi uygulanmıştır.

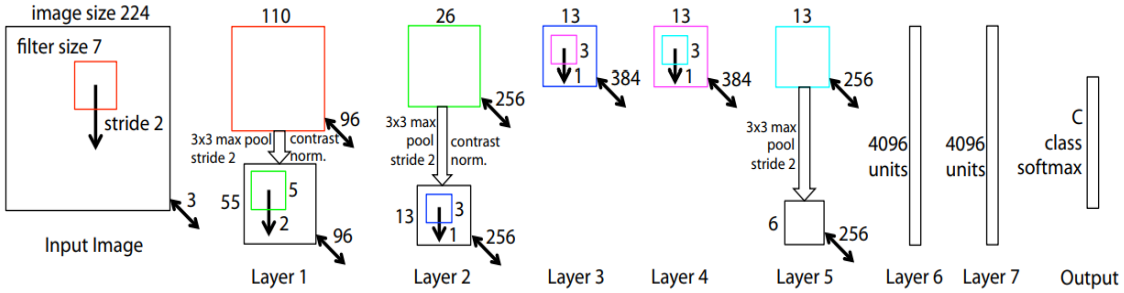


Şekil 3.10: AlexNet modeli [9]

3.7.3 ZFNet

ILSVRC 2013 yılının kazananı olan bu konvolüsyonel sinir ağı modeli Matthew Zeiler ve Rob Fergus tarafından geliştirilmiştir. Özellikle orta katmanların kanal sayısı, derinliğini arttırarak ve giriş katmanındaki aralık sayısı (stride) ve filtre boyutlarını küçülterek AlexNet modelinin geliştirilmesini sağlamıştır. AlexNet modelinde giriş katmanında filtre boyutu 11x11, aralık sayısı (stride) ise 4 olarak seçilmiştir. Fakat bu modelde giriş katmanında filtre boyutu 7x7 ve aralık sayısı (stride) 2 olarak seçilmiştir [25]. Daha küçük filtre boyutları ve adım sayısı giriş katmanından daha fazla bilginin korunmasına katkı sağlamıştır. Büyük filtre boyutları ve adım sayıları, girdi boyutlarını küçültüp hesaplama yükünü azaltsa da özellik kaybına neden olabilir. ZFNet 11.2%'lik bir hata oranı elde etmiştir. Aktivasyon fonksiyonu olarak ReLu, hata (kayıp) fonksiyonu olarak çapraz entropi kaybı (Cross-Entropy Loss) ve toplu stokastik gradyan (Batch Stochastic Gradient Descent) alçalması kullanılarak eğitilmiştir. Daha önceki yöntemlerden farklı olarak "Deconvolution Network" olarak isimlendirilen farklı bir yaklaşım ortaya koymuştur. Bu yöntem kabaca özellikleri pikseller ile eşleştirir. Konvolüsyon katmanının tersidir (Deconv). ZFNet, giriş ve sınıflandırma katmanları dahil edilmediğinde toplamda 7 katmandan oluşur. 1, 2 ve 5. Katmanlarda konvolüsyon (convolution) işleminden sonra maksimum havuzlama(max pooling) işlemi uygulanmıştır. 3. ve 4. konvolüsyon katmanlarından sonra havuzlama işlemi uygulanmamıştır. Ekran kartı olarak yine GTX 580 kullanılmış ve modelin eğitimi 12 gün boyunca sürmüş, geliştiren ekip tarafından 70.

eğitim dönemi (epoch) değerinde durdurulmuştur. Şekil 3.11'de ZFNet modeli görülmektedir.

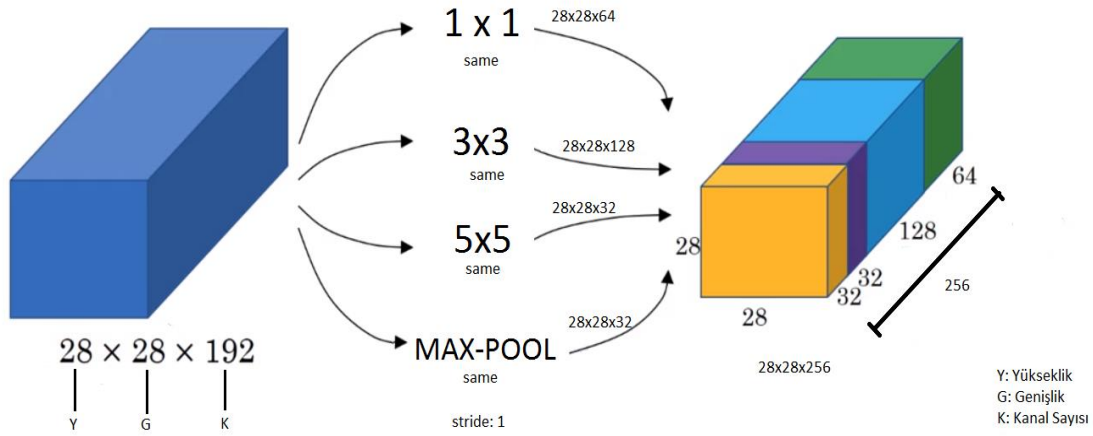


3.11: ZFNet modeli [25]

3.7.4 GoogleNet

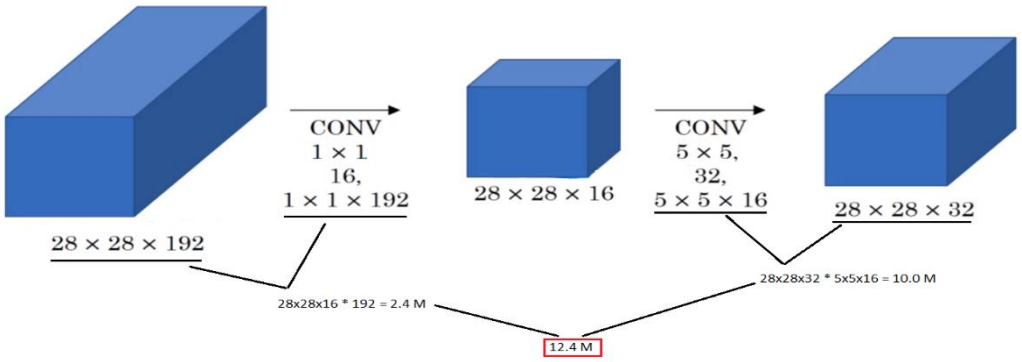
ILSVRC 2014 yılının kazananı olan bu konvolüsyonel sinir ağı modeli Christian Szegedy ve arkadaşları tarafından geliştirilmiştir. Inception kod adı verilen konvolüsyonel sinir ağı modeli önerilmiştir. GoogleNet, 22 katmanlı bir konvolüsyonel sinir ağı modelidir. ILSVRC 2014'de ilk 5 test hata oranında (top-5 error) 6.7%'lik bir değer elde ederek birinci olmuştur. Daha önceki klasikleşmiş konvolüsyon ve havuzlama katmanlarının ard arda dizilmesiyle oluşturulan modellerden farklı bir yaklaşım getirmiştir. Önceki modellerde görüldüğü gibi, her şeyin sıralı olarak gerçekleşmediği görülmektedir. Bu modelde paralel olan ağ parçaları mevcut. Bu parçalar, modüller "inception" olarak adlandırılır [26].

Geleneksel bir konvolüsyon ağında her katmanda konvolüsyon işlemi ya da havuzlama işlemi için seçim yapmanız istenir. Ayrıca bir filtre boyutu seçimi de istenir. Bir "inception" modülü tüm bu işlemleri paralel olarak gerçekleştirmenize izin verir. Aynı zamanda AlexNet'e göre 12 kat daha az parametre kullanılır. Bu da işlem yükünün azalmasını sağlar ve bellek kullanımını azaltır.



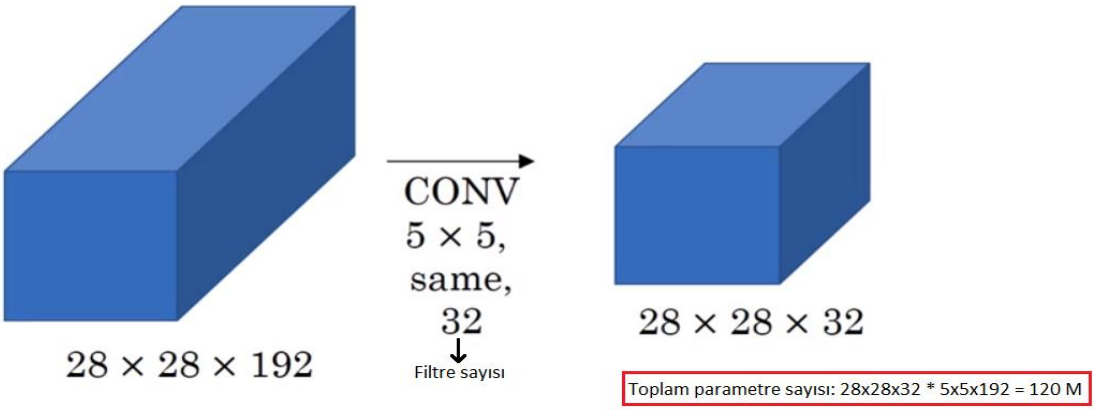
Şekil 3.12: Inception modül çalışma yapısı [27]

Şekil 3.12'ye bakıldığında verilen girdi için 3 farklı filtre boyutu ile konvolüsyon işlemi ve 1 adet maksimum havuzlama işlemi uygulanıyor. Elde edilen çıktıların genişlik ve yükseklik değerleri girdi ile aynı. Fakat her bir işlem kendi belirlediği derinlikte çıktı oluşturuyor (32,64,128 vb.). Oluşturulan çıktıların hepsi birleştirilip tek bir modül haline getiriliyor ve bir sonraki adıma girdi olarak verilmek üzere hazır tutuluyor [26, 27]. Fakat bu işlemler her ne kadar paralel olarak gerçekleştirilebilse de matris boyutlarından dolayı yüksek hesaplama gücü gerektirir. Bu zorluk Şekil 3.13'te gösterilmektedir.



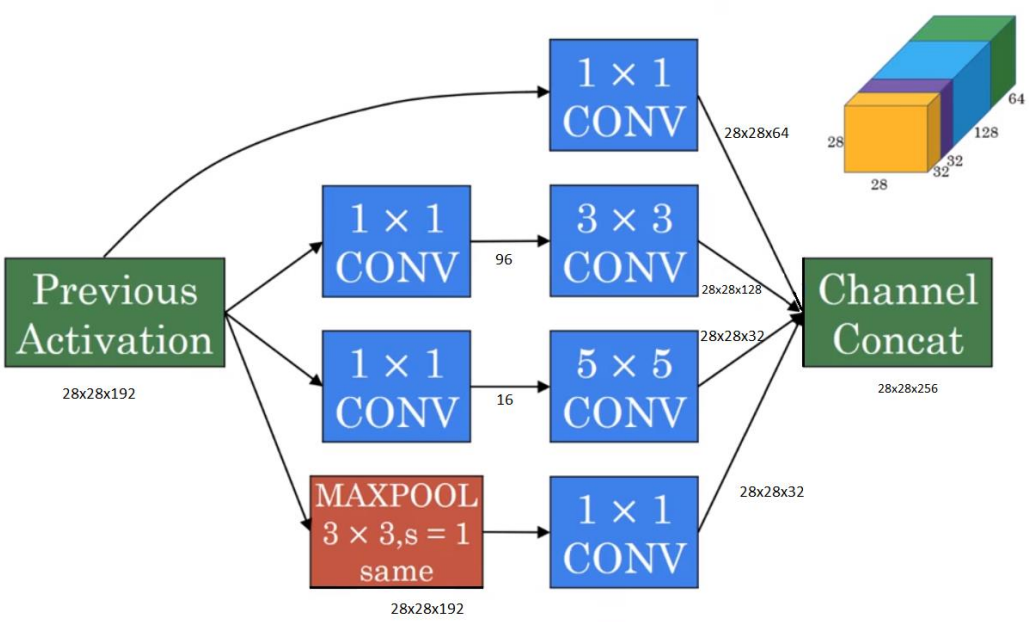
Şekil 3.13: Yüksek matris boyutları için parametre sayısı [27]

Şekil 3.13'te görüldüğü gibi kanal sayısı ve filtre boyutu arttıkça parametre sayısı artmakta ve bu durumda yüksek hesaplama gücü gerektirmektedir. Bu duruma çözüm olarak GoogleNet "darboğaz" olarak isimlendirilen bir çözüm geliştirmiştir. Inception modül içerisinde girdiye direkt olarak yüksek filtreli konvolüsyon işlemi uygulanmaz. İlk olarak 1x1 boyutunda konvolüsyon katmanı oluşturularak, konvolüsyon işlemi gerçekleştirilir. Daha sonra elde edilen çıktılar için istenilen filtre boyutları uygulanır [26, 27]. İlk başta uygulanan 1x1 boyutunda filtre katmanı, parametre sayısının 12 kata kadar azaltılmasına yardımcı olur. Bu durum Şekil 3.14'te görülmektedir.

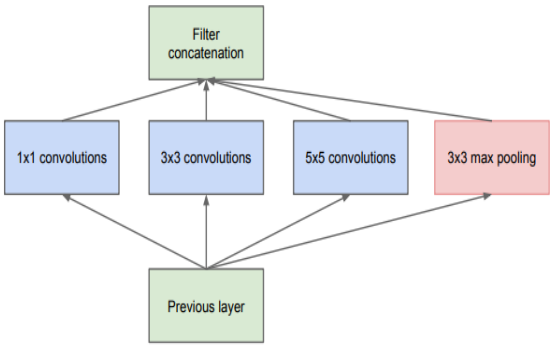


Şekil 3.14: Parametre sayısının azaltılması [27]

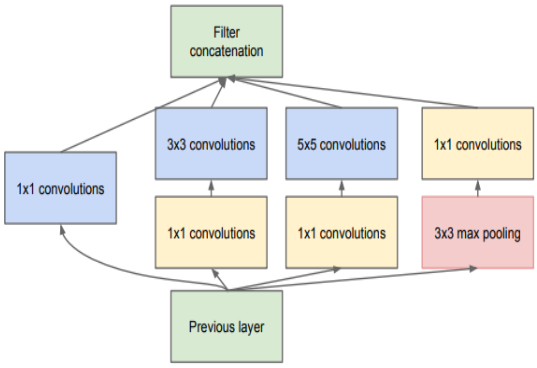
Şekil 3.13 ve Şekil 3.14 sonucunda elde edilen parametre sayılarına bakıldığında neredeyse 10 katlık bir fark bulunmaktadır. Inception modül içerisinde kullanılan 1x1 boyutundaki filtreler boyut indirgemeyi sağlamıştır. Aynı zamanda farklı filtre boyutları aynı örnek üzerinde kullanılarak daha fazla özellik çıkarımı gerçekleştirilmiştir. Yapılan bu işlemler, inception modüller sayesinde çok daha az hesaplama gücü ve performanslı bir şekilde gerçekleştirilmiştir. Şekil 3.15'te Inception modülün yapısı ve şekil 3.16'da GoogleNet mimarisi gösterilmektedir.



a) Boyut indirgemeli inception modül gösterimi [27]

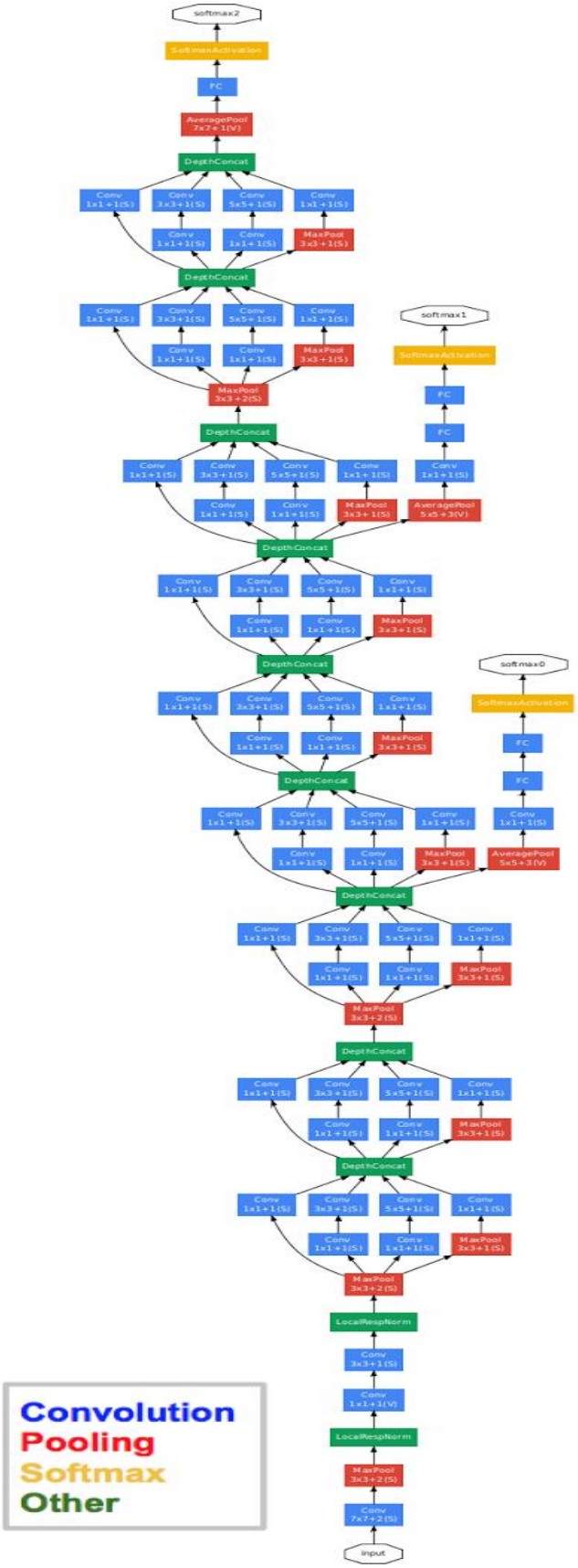


b) Inception Modül, Naive versiyon [26]



c) Boyut azaltma özelliğine sahip Inception Modülü [26]

Şekil 3.15: Inception modül

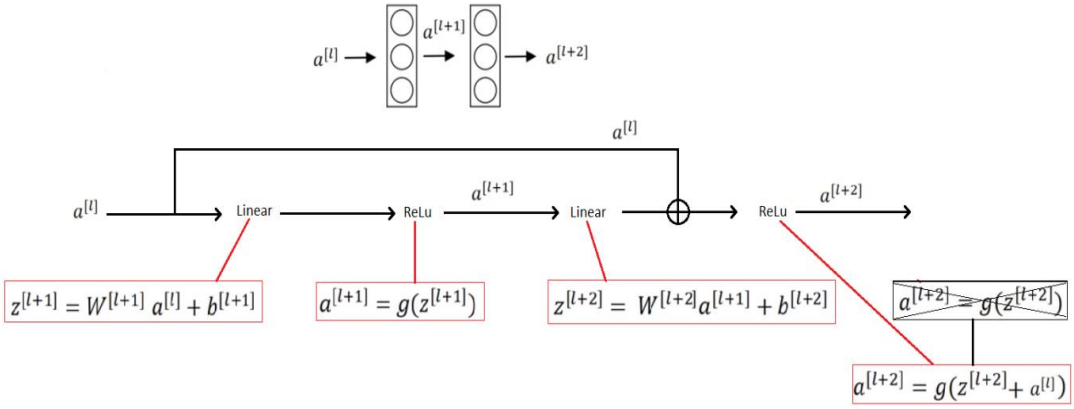


Şekil 3.16: GoogleNet mimarisi [26]

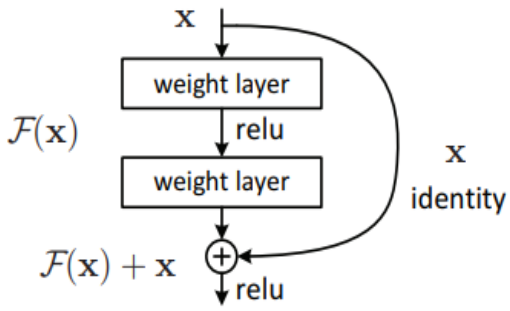
3.7.5 ResNet

ILSVRC 2015 yılının kazananı olan bu konvolüsyonel sinir ağı modeli Microsoft Asya Araştırma ekibi tarafından geliştirilmiştir. ResNet yeni bir yaklaşım getiren, 152 katmandan oluşan bir konvolüsyonel sinir ağı modelidir. Oluşturulan bu model, ImageNet veri seti için 2015 yılında 3.57%'lik inanılmaz bir hata oranı elde etti [28]. ImageNet veri seti için ise insanların hata oranı 5%-10% arasında değişmektedir (Andrej Karpathy). Oluşturulan bu yeni model ImageNet veri seti için insanlardan daha iyi sonuçlar elde etmiştir.

Klasik konvolüsyonel sinir ağlarında derinlik arttıkça hatanın belirli bir adım sayısından sonra aşırı öğrenmeden (overfitting) dolayı artmaya başladığı görülmüştür. Bu modelde bunun önüne geçebilmek için "Residual Block" adında yeni bir yaklaşım getirilmiştir. Bu yaklaşımda x girdisi konvolüsyon (weight layer) -> ReLu -> konvolüsyon (weight layer) işlemlerinden sonra F(x) olarak isimlendirilen bir çıktı elde eder. Bu çıktı direkt olarak diğer bloğa verilmez. Diğer blok için x + F(x) sonucunun ReLu işlemine sokulması ile elde edilen çıktı verilir. Bu çıktı H(x) olarak ifade edilir ve şu şekilde gösterilir; H(x) = F(x) + x [28]. Yani F(x) çıktısı diğer bloğa verilmeden önce x girdisi ile birleştirilip, ReLu fonksiyonuna sokulur. Artık (Residual) blokların içyapısı Şekil 3.17 ve 3.18 'da gösterilmektedir.



Şekil 3.17: Residual Block yapısı [27]



Şekil 3.18: Artık (Residual) Öğrenme: Blok yapısı [28]

3.8. Kullanılan Veri Setleri

Bu bölümde konvolüsyonel sinir ağlarının eğitilmesi için sıklıkla çalışılan ve bu çalışmada kullanılan veri setleri anlatılacaktır. Kullanılan veri setlerinin nasıl oluşturulduğu, veri setinde bulunan örnek sayısı, veri setinin hangi tür verileri ve sınıfları içerdiği gibi bilgiler verilecektir. Artan işlem gücü ve veri sayısı konvolüsyonel sinir ağlarının başarısını giderek arttırmaktadır. Bu gün görsel sınıflandırma konusunda eğitilen derin öğrenme modelleri, insanlardan daha iyi sonuçlar elde edebilmektedir (ILSVRC 2015 - ResNet). Bu durum günlük yaşamda karşılaşılan birçok problemin çözümü için yeni arayışlara yönelmemize neden olmuş. Bu sorunlara çözüm üretebilmek adına geliştirilen modelleri eğitebilmek için veri setleri oluşturulması önem kazanmıştır.

3.8.1 MNIST el yazısı rakamlar (mnist handwritten digits)

MNIST (Modified National Institute of Standards and Technology) veri seti, görüntü işleme uygulamaları için sıklıkla kullanılan büyük bir veri setidir. Konvolüsyonel sinir ağlarına yeni başlayanlar tarafından sıklıkla kullanılmaktadır. Veri seti, el yazısı ile yazılmış olan 0-9 arasındaki rakamları içerir (şekil 3.19). Veri seti içerisindeki görüntüler gri tonlamalı, 28x28 boyutunda, 60.000 eğitim ve 10.000 test verisinden oluşmaktadır [29]. Veri setine ait özellikler Tablo 3.1'de ayrıntılı olarak görülmektedir.

Tablo 3.1: MNIST veri seti özellikleri

MNIST El Yazısı Rakamlar Veri Seti Özellikleri	
Tanım:	Çeşitli görüntü işleme sistemlerini eğitmek için yaygın olarak kullanılan el yazısı rakamlardan oluşan büyük bir veri setidir.
Boyut:	28x28
Eğitim Veri Sayısı:	60.000
Test Veri Sayısı:	10.000
Tip:	Gri Tonlamalı (Tek kanal)
Sınıf Sayısı:	10

Şekil 3.19: MNIST handwritten veri seti [23]

3.8.2 EMNIST veri seti

MNIST veri seti, bilgisayarlı görü, sınıflandırma gibi çalışmalarda standart haline gelen bir karşılaştırma veri seti haline gelmiştir. MNIST veri setinin düşük saklama alanı, düşük hesaplama gücü gerektirmesi gibi nedenler bu veri setinin standart haline gelmesindeki en önemli etkenlerdendir. Yıllar içerisinde gelişen donanım teknolojisi ve gelişen makine öğrenmesi teknikleri bu veri seti için yüksek doğruluk oranları elde edilmesini kolaylaştırmıştır. Bu nedenle 2017 yılında Cohen ve arkadaşları tarafından EMNIST (Extended MNIST) veri seti oluşturulmuştur. MNIST veri setinin genel karakteristik özelliklerini taşıyan bu veri seti orijinal MNIST veri setinden daha fazla örneğe ve daha fazla sınıf sayısına sahiptir. Mevcut MNIST veri seti sadece 0-9 arasında el yazısı ile yazılmış rakamlardan oluşan bir veri seti özelliğini taşımaktaydı. Oluşturulan yeni veri setine el yazısı rakamlar yanında el yazı harflerde eklendi. Veri setinin içerdiği örnek sayısı ve örneklerin zorluğu arttırıldı. Yaptığımız bu çalışmada da kullandığımız EMNIST veri seti için oluşturulmuş bazı veri kümeleri ve bunlara ait bilgiler Tablo 3.2'de verilmiştir [30].

Tablo 3.2: EMNIST veri seti için oluşturulan bazı veri kümeleri ve özellikleri

İsim	Sınıf	Eğitim Örnek Sayısı	Test Örnek Sayısı
Balanced	47	112.800	18.800
Digits	10	240.000	40.000
Letters	37	88.800	14.800

3.8.3 Fashion-MNIST veri seti

Fashion-MNIST veri seti 28x28 boyutunda, gri tonlamalı 70.000 adet moda ürününden oluşan 10 sınıflık bir veri setidir. Her kategori için 7.000 resim vardır. Eğitim için 60.000, test için 10.000 resim bulunmaktadır. MNIST el yazısı veri setinin günümüzde kullanılan konvolüsyonel sinir ağları için kolay tahmin edilebilir olması ve yüksek doğruluk oranları vermesi yeni veri seti ihtiyacı doğurmuştur. Bu ihtiyacı karşılamak için Zalando giyim markasının web sitesi kullanılarak Fashion-MNIST veri seti oluşturulmuştur [31].

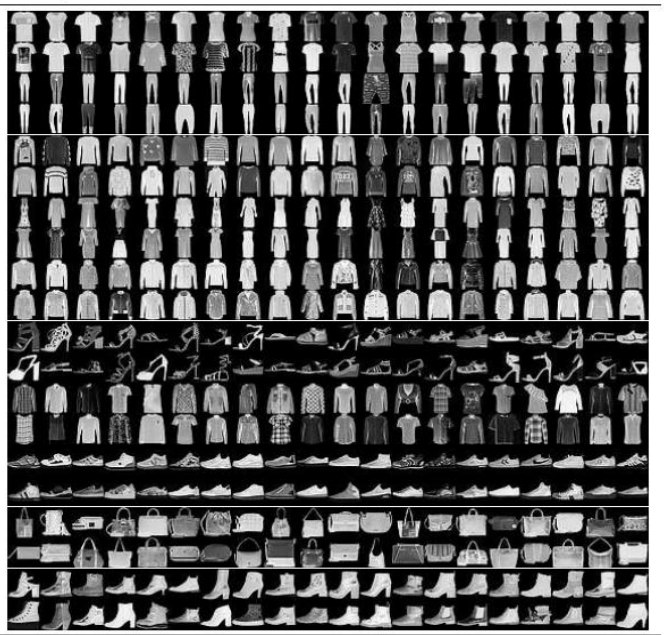
Diğer veri setlerine göre çok daha yakın bir tarihte oluşturulan ve 2017 yılında herkese açık bir şekilde yayınlanan Fashion-MNIST [31] veri seti konvolüsyonel sinir ağı çalışmalarında yeni yeni kullanılmaya başlanmıştır. Veri setine ait özellikler Tablo 3.3'de, örnek görüntüler ise şekil 3.20'de gösterilmiştir.

Tablo 3.3: Fashion-MNIST veri seti özellikleri

Fashion MNIST Veri Seti Özellikleri	
Tanım:	Zalando giyim markasının web sitesi kullanılarak oluşturulan veri setidir. 10 farklı sınıf için moda ürünlerine ait görüntüleri içermektedir.
Boyut:	28x28
Eğitim Veri Sayısı:	60.000
Test Veri Sayısı:	10.000
Tip:	Gri Tonlamalı (Tek kanal)
Sınıf Sayısı:	10

Fashion-MNIST veri seti 10 sınıf içermektedir. Bu sınıflar şunlardır:

- T-Shirt
- Pantolon
- Süveter
- Elbise
- Ceket
- Sandalet
- Gömlek
- Spor ayakkabı
- Çanta
- Ayak bileği çizmeler

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Şekil 3.20: Fashion-MNIST veri seti sınıflar ve örnek görüntüler [31]

3.8.4 ImageNet veri seti

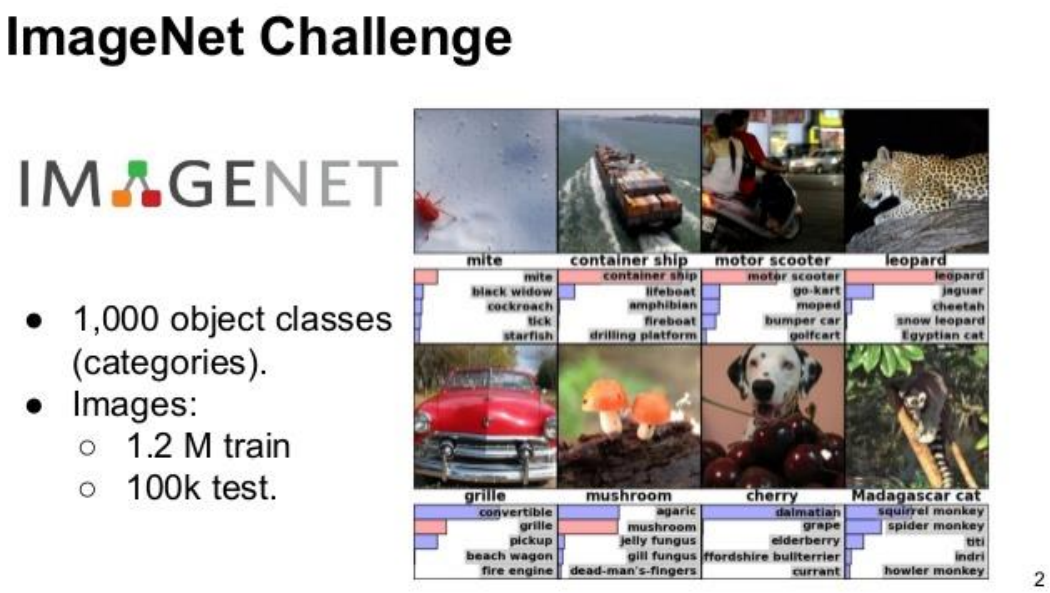
ImageNet veri seti, renkli, 15 milyondan fazla yüksek çözünürlüklü resimden oluşan 22.000 kategorilik bir veri setidir. Görüntüler internet üzerinden toplanan resimlerin kullanıcılar tarafından etiketlenmesiyle oluşturulmuştur. Bu projenin adı Amazon Mechanical Turk'tür [9]. ImageNet veri setinin belirli bir kısmı 2010 yılından beri ILSVRC yarışmasında modellerin eğitilmesi için kullanılmaktadır.

2010 yılında Fei-Fei Lee ve Stanford üniversitesi çalışanları tarafından geliştirilen veri seti ILSVRC yarışmaları ile popülerlik kazanmıştır. Her sene yapılan yarışmalarda birincilik elde eden çalışmalar yayınlanmaktadır[9, 25, 26, 28]. 2012 yılında yayınlanan çalışma [9] ile daha önceki senelerde elde edilen hata oranlarına göre daha iyi bir hata oranı elde edilmiş ve aşırı öğrenmeyi azaltmak için farklı teknikler kullanılmıştır. Kullanılan bu teknikler bir sonraki çalışmalara öncülük etmiştir. Özellikle 2014 ve 2015 yıllarında ortaya çıkan modeller klasik konvolüsyonel sinir ağı modellerini farklı bir noktaya getirmiştir. 2012 [9] yılında 15.3%, 2013 [25] yılında 11.2%, 2014 [26] yılında 6.4% ve 2015 [28] yılında 3.57%'lik hata oranları elde edilmiştir. 2015 yılında "Deep Residual Learning for Image Recognition" ismi ile yayınlanan bu çalışma, 2015 ILSVRC yarışmasında birinci olan konvolüsyonel sinir ağı modelini açıklamaktadır. Yayınlanan çalışmada bu veri seti için 3.57%'lik bir hata oranı elde edilmiştir

[28]. Bu değer insanın tahmin değerinin de altındadır. Bu yarışmalarda kullanılan veri seti özellikleri Tablo 3.4'te, veri setine ait örnek görüntüler şekil 3.21'de gösterilmiştir.

Tablo 3.4: ILSVRC ImageNet veri seti özellikleri [9]

ILSVRC ImageNet Veri Seti Özellikleri	
Tanım:	ILSVRC yarışmasında kullanılan ImageNet veri seti, 1000 farklı kategoriye ait renkli görüntüleri içermektedir.
Boyut:	256x256
Eğitim Veri Sayısı:	1.2 milyon
Test Veri Sayısı:	100.000
Tip:	Renkli (Çok kanal)
Sınıf Sayısı:	1000



Şekil 3.21: ILSVRC yarışmasında (ImageNet Challenge) kullanılan veri seti özellikleri ve örnek birkaç görüntü

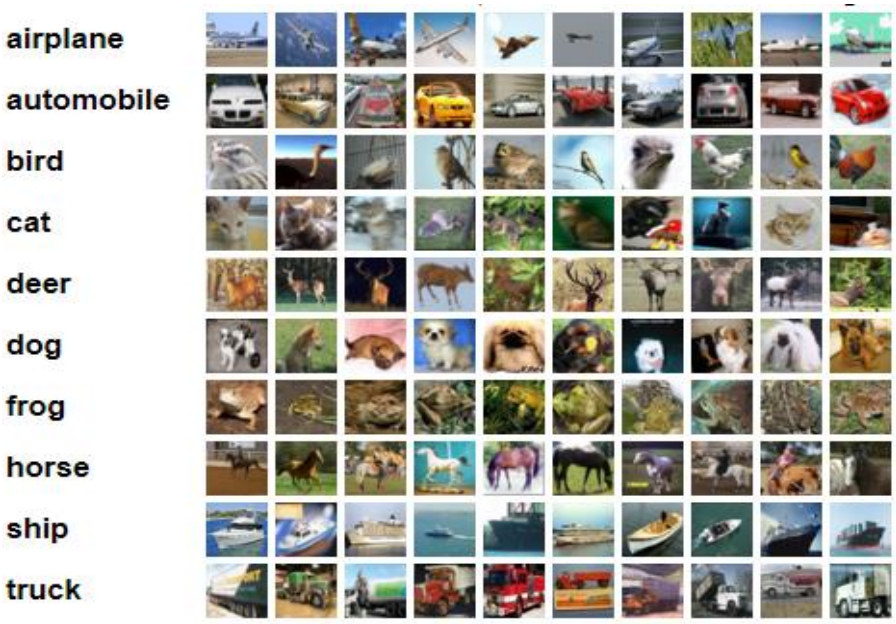
3.8.5 CIFAR veri seti

CIFAR (Canadian Institute For Advanced Research) veri seti, bilgisayarlı görü çalışmaları için geliştirilen bir veri setidir. Veri seti Alex Krizhevsky, Vinod Nair ve Geoffrey Hinton tarafından toplandı. CIFAR-10 ve CIFAR-100 olarak ikiye ayrılmaktadır. 10 ve 100 sayıları veri setinde bulunan sınıf, kategori sayılarını temsil etmektedir. CIFAR-10 ve CIFAR-100, 80 milyon küçük görüntüden oluşan veri setinin alt kümeleridir [32].

3.8.5.1 CIFAR-10 veri seti

CIFAR-10 veri seti 32x32 boyutunda renkli küçük görüntülerden oluşmaktadır. Veri seti 10 sınıf içermektedir ve sınıf başına 6.000 olmak üzere toplamda 60.000 görüntüden oluşmaktadır. Bunların 50.000'i eğitim, 10.000'i test için kullanılmaktadır. CIFAR-10 veri seti şu sınıfları içermektedir [32]:

- Uçak
- Otomobil
- Kuş
- Kedi
- Geyik
- Köpek
- Kurbağa
- At
- Gemi
- Tır



Şekil 3.22: CIFAR-10 veri seti özellikleri ve örnek birkaç görüntü [32]

Şekil 3.22'de CIFAR-10 veri setine ait özellikler ve örnek birkaç görüntü gösterilmiştir. CIFAR10 veri setine ait özellikler Tablo 3.5'te gösterilmiştir.

Tablo 3.5. CIFAR-10 veri seti özellikleri

CIFAR-10 Veri Seti Özellikleri	
Tanım:	CIFAR-10 veri seti, 10 farklı kategoriye ait renkli görüntüleri içermektedir.
Boyut:	32x32
Eğitim Veri Sayısı:	50.000
Test Veri Sayısı:	10.000
Tip:	Renkli (Çok kanal)
Sınıf Sayısı:	10

3.8.5.2 CIFAR-100 veri seti

CIFAR-100 veri seti 32x32 boyutunda renkli küçük görüntülerden oluşmaktadır. Veri seti 100 sınıf içermektedir ve sınıf başına 600 görüntüden oluşmaktadır. Bunların 500'ü eğitim, 100'ü test için kullanılmaktadır. CIFAR-100 veri setinin içerdiği süper sınıflar ve sınıflar şekil 3.23'de görülmektedir. Aynı şekilde CIFAR-100 veri setine ait özellikler tablo 3.6'da gösterilmiştir.

Tablo 3.6: CIFAR-100 veri seti özellikleri [32]

CIFAR-100 Veri Seti Özellikleri	
Tanım:	CIFAR-100 veri seti, 100 farklı kategoriye ait renkli görüntüleri içermektedir.
Boyut:	32x32
Eğitim Veri Sayısı:	50.000
Test Veri Sayısı:	10.000
Tip:	Renkli (Çok kanal)
Sınıf Sayısı:	100

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Şekil 3.23: CIFAR-100 süper sınıfları ve süper sınıfların içerdiği sınıflar [32]

4. LİTERATÜR ARAŞTIRMASI

KSA çok sayıda hiper-parametre içermektedir ve bu hiper-parametrelerin optimizasyonu başarı oranlarını doğrudan etkilemektedir. Her bir hiper-parametre için seçilebilecek değerlerin oluşturduğu arama uzayı çok geniş olduğundan bütün seçenekleri denemek çok fazla hesaplama zamanı gerektirmektedir. Bu nedenle hiper-parametrelerin optimizasyonu çalışmalarında genellikle üst-sezgisel yöntemler kullanılsa da istatistiktik tabanlı yaklaşımlar ve pekiştirmeli öğrenme gibi yöntemler de kullanılmaktadır. Bu bölümde, literatürde yapılmış olan çalışmalar incelenip, özetlenmiştir.

4.1 Hiper-parametre Optimizasyonu için Kullanılan Üst-sezgiseller

KSA hiper-parametre optimizasyonu çalışmalarında üst-sezgisel yöntemler sıklıkla kullanılmaktadır. Bu bölümde, KSA hiper-parametre optimizasyonu çalışmalarında sıklıkla kullanılan üst-sezgiseller yöntemler ve literatürde bu üst-sezgisel yöntemler kullanılarak yapılan çalışmalar anlatılmıştır.

4.1.1 Genetik algoritmalar

John Holland tarafından yazılan Adaption in Natural and Artifical Systems [33] kitabında Genetik Algoritmaların (GA) yapısı anlatılmış ve bu yöntemin günümüzde bilinen temelleri atılmıştır. Anlatılan bu yöntem, konvolüsyonel sinir ağlarında hiper-parametre optimizasyonunun gerçekleştirilmesi için yapılan çalışmalarda yoğun bir şekilde kullanılmış ve başarılı sonuçlar elde edilmiştir. GA popülasyon tabanlı bir optimizasyon yöntemidir. Temel olarak, en iyi çözümlerin yaşamasına izin verip, jenerasyonlar boyunca çözüm uzayında, daha iyi sonuçlar verebilecek çözümleri aramaya çalışan bir optimizasyon yöntemidir [34, 35]. Algoritmanın çalışması boyunca her jenerasyonda sabit sayıdan oluşacak şekilde popülasyonlar üretilir. Bu problem için popülasyondaki bireylerin her biri en iyi çözüm olmaya aday, KSA topolojileridir. Popülasyondaki çözümler birbirleri ile çaprazlama ve mutasyon işlemlerine uğratılarak, yeni çözümler, KSA topolojileri üretilir. Üretilen bu KSA topolojileri arasında hayatta kalanlar (daha iyi amaç fonksiyonu değerine sahip olanlar) gelecek jenerasyonları oluşturur ve bu sayede gelecek jenerasyonlarda daha iyi sonuçlar üreten çözümlerin çoğunlukta olması ümit edilir. GA, karar verilmesi gereken birçok parametre (popülasyon büyüklüğü, çaprazlama olasılığı, mutasyon olasılığı vb.) ve yöntem (çaprazlama yöntemi, seçim algoritması, mutasyon yöntemi

vb.) dolayısıyla kendi içerisinde de bir parametre optimizasyonu gerektirebilir. Bu parametre ve yöntemlerin doğru seçilmesi ekstra bir uğraş gerektirmektedir fakat seçilen parametre ve yöntemler sonucu doğrudan etkileyebilmektedir. Bu yüzden optimizasyon çalışmaları için GA seçilirken bu parametre ve yöntemlerin doğru bir şekilde belirlenmesine dikkat edilmelidir.

2016 yılında Ijjina ve arkadaşları [36] sabit boyutta bir KSA topolojisi seçerek bu mimaride konvolüsyon katmanı için filtre boyutu ve tam bağlantılı katmanda başlangıç ağırlıklarının oluşturulacağı "seed" gibi hiper-parametreler GA ile optimize edilmeye çalışılmıştır. Önerilen yöntem UCF50 veri seti için test edilmiştir. Literatürde GA ile KSA optimizasyonu çalışmalarında populasyon büyüklüğü, kromozom yapısı, çaprazlama ve mutasyon oranları gibi parametreler genellikle sabit seçilmiştir.

2017 yılında Dufourq ve arkadaşları [37] dinamik olarak genişleyebilen konvolüsyonel sinir ağlarının oluşturulması ve oluşturulan ağlarda GA ile hiper-parametrelerin otomatik seçilmesine yönelik bir çalışma önermişlerdir. Gerçekleştirilen çalışmada, kromozomlar değişken boyutlarda oluşabilmektedir. Bu yüzden sabit bir kromozom boyutu yoktur. Genlerin 1 tanesi öğrenme katsayısını, 1 tanesi ise KSA mimarisini temsil etmek için kullanılmıştır. Optimizasyon için, konvolüsyon katmanında; filtre boyutu, aktivasyon fonksiyonu, özellik haritası sayısı hiper-parametreleri, ortaklama katmanında; filtre boyutu hiper-parametresi, tam bağlantılı katmanda; düğüm sayısı ve aktivasyon fonksiyonu hiper-parametreleri seçilmiştir. Yapılan çalışmada, MNIST veri seti için en düşük hata oranını veren ağ topolojisi ve hiper-parametreler şu şekilde belirtilmiştir: KSA 3 konvolüsyon katmanı, 1 maksimum ortaklama katmanı ve 1 tam bağlantılı katmandan oluşmaktadır. Konvolüsyon katmanları için sırasıyla şu hiper-parametreler seçilmiştir (*özellik haritası sayısı, filter boyutu, aktivasyon fonksiyonu*): {(49, 5x5, Lineer), (43, 1x1, ReLU), (93, 4x4, ReLU)}. Maksimum ortaklama katmanı için 2x2 kernel boyutu, tam bağlantılı katman için nöron sayısı 10 ve aktivasyon fonksiyonu softmax olarak seçilmiştir.

2017 yılında Sun ve arkadaşları [38] KSA topolojilerinin otomatik oluşturulması ve oluşturulan topolojilerdeki hiper-parametrelerin otomatik seçilmesini sağlayan GA tabanlı bir çalışma önermişlerdir. Önerilen bu yöntem çok daha az parametre ile güçlü mimarilere karşı rekabetçi sonuçlar elde etmiştir. Değişken uzunlukta tanımlanan kromozom ile KSA blokları olarak temsil edilmiştir. Konvolüsyon katmanı için seçilen

hiper-parametreler bir blok, ortaklama katmanı için seçilen hiper-parametreler bir blok ve tam bağlantılı katman için seçilen hiper-parametreler ayrı bir blok olarak temsil edilmiştir. Konvolüsyon katmanında; filtre boyutu, özellik haritası sayısı ve aralık değeri, ortaklama katmanında; filtre boyutu ve aralık değeri, tam bağlantılı katmanda da düğüm sayısı hiper-parametreleri optimizasyon için seçilmiştir.

2017 yılında Silva ve arkadaşlarının [39] önerdiği çalışmada GA yardımıyla akciğer nodüllerini başarılı şekilde tespit edebilen bir KSA oluşturulmaya çalışılmıştır. LIDC-IDRI veri seti kullanılarak elde edilen sonuçlar farklı performans metrikleri ile değerlendirilmiştir. Önerilen çalışmada sabit bir konvolüsyonel sinir ağı mimarisi seçilmiştir. Optimize edilmek için konvolüsyon katmanında; özellik haritası sayısı, tam bağlantılı katmanda ise düğüm sayısı hiper-parametreleri seçilmiştir.

2017 yılında Bochinski ve arkadaşları [40] değişken boyutlarda kromozom yapısına sahip bir GA ile KSA topolojilerinin otomatik olarak oluşturulduğu ve hiper-parametrelerin otomatik olarak seçildiği bir çalışma önermişlerdir. Genlerin 1 tanesi konvolüsyon katmanı hiper-parametrelerini, 1 tanesi ise tam bağlantılı katman hiper-parametrelerini temsil etmek için kullanılmıştır. Temsil edilen genlerin sayısı katman sayısına göre artabilmektedir. Yapılan çalışmada, son popülasyondaki bireyler 4-5 konvolüsyon katmanı ve 1-3 (sıklıkla 2) tam bağlantılı katmandan oluşmuştur. Son popülasyonda üretilen topolojiler şu şekilde belirtilmiştir: Konvolüsyon katmanları sırasıyla (*özellik haritası sayısı, filtre boyutu*) = {(107, 7), (88, 7), (123, 6), (102, 3)} ve tam bağlantılı katman (düğüm sayısı) = {104}.

2017 yılında Fujino ve arkadaşları [41], AlexNet KSA mimarisi üzerinde GA ile gerçekleştirdikleri çalışmalarında en iyi sonucu veren KSA topolojisinin 5 konvolüsyon bloğu ve 2 tam bağlantılı katmandan oluştuğunu ortaya koymuşlardır. Konvolüsyon ve ortaklama katmanları için sırasıyla şu hiper-parametreler kullanılmıştır (*özellik haritası sayısı, filtre boyutu*, ortaklama kernel boyutu): {(64, 3, 7), (64, 7, 7), (16, 7, -), (32, 11, -), (32, 3, -)}. Tam bağlantılı katmanlar için sırasıyla şu hiper-parametreler kullanılmıştır (düğüm sayısı): {1024, 512}. İkinci konvolüsyon katmanı ve ortaklama katmanları hariç diğer bütün katmanlarda ReLU aktivasyon fonksiyonu kullanılmıştır. Yapılan bu çalışmada konvolüsyon katmanında; özellik haritası sayısı ve filtre boyutu hiper-parametreleri, ortaklama katmanında; filtre boyutu hiper-parametresi, tam bağlantılı katmanda; düğüm sayısı hiper-parametresi, bunların

dışında yığın boyutu ve aktivasyon fonksiyonu hiper-parametreleri optimize edilmek için seçilmiştir.

2018 yılında Rincon ve arkadaşları [42] tarafından KSA mimarisi sabit seçilerek, oluşturulan KSA topolojileri üzerinde hiper-parametre optimizasyonu çalışması gerçekleştirilmiştir. Önerilen yöntem tıp alanındaki çalışmalarda kullanılan bir veri seti üzerinde kullanılmış; farklı makine öğrenmesi algoritmaları ile karşılaştırıldığında mikro ortalamada çok daha iyi sonuçlar üretirken makro ortalamada aynı başarıya ulaşamamıştır. Bu çalışmada konvolüsyon katmanında; filtre boyutu ve özellik haritası sayısı hiper-parametreleri ve ortaklama katmanında ise filtre boyutu hiper-parametresi optimize edilmek için seçilmiştir.

2018 yılında Ma ve arkadaşları [43] tarafından, sabit bir KSA mimarisi yerine belirlenen sınırlar içerisinde genişleyebilen KSA mimarisi önerilmiştir. Önerilen yöntem "state-of-the-art" olarak kabul edilen algoritmalara karşı rekabetçi sonuçlar vermiştir. Değişken uzunlukta kromozomlar ile KSA blokları özellikleri temsil edilmiştir. Her bir konvolüsyon bloğu liste tipindedir. Kromozomların her biri en az 3 genden oluşur. Genlerin 1 tanesi konvolüsyon bloğu hiper-parametrelerini, 1 tanesi tam bağlantılı katman bloğu hiper-parametrelerini ve 1 tanesi ise optimizasyon yöntemini temsil etmek için kullanılmıştır. Yapılan çalışmada, MNIST veri seti için en düşük hata oranını veren konvolüsyonel sinir ağı topolojisi 3 konvolüsyon bloğu ve 3 tam bağlantılı bloktan oluşmaktadır. Konvolüsyon blokları için belirlenen hiper-parametreler sırasıyla şu şekildedir (*özellik haritası sayısı, filtre boyutu, (ortaklama tipi, aralık), aktivasyon, seyreltme*): {(419, 5, -, ELU, 0.2), (403, 5, -, ELU, 0), (288, 7, (Avg, 2), PreLU, 0)}. Tam bağlantılı bloklar için belirlenen hiper-parametreler ise sırasıyla şu şekildedir (*düğüm sayısı, aktivasyon, seyreltme*): {(194, ReLU, 0.3), (414, ELU, 0.45), (356, TreLU, 0.05)}. Tüm konvolüsyon ve tam bağlantılı bloklarda yığın normalizasyonu uygulanmıştır. Bu çalışmaya benzer bir çalışma 2018 yılında Assunçao ve arkadaşları [44] tarafından önerilmiştir.

2018 yılında Baldominos ve arkadaşlarının [45] önerdiği dinamik ağ topolojisinde ise oluşturulan topolojilerin değerlendirilmesi için farklı bir yöntem önerilmiştir. Üretilen bireye daha önceki bireylere benzeme derecesine göre ceza puanı verilerek sürekli aynı bireylerin elde edilmesi engellenerek genetik çeşitlilik oluşturulmaya çalışılmıştır. Değişken uzunluklu ikili (binary) yapıda kromozomlarda KSA blokları temsil edilmiştir. En düşük hata oranını veren KSA topolojisi 3 konvolüsyon katmanı, 1

ortaklama katmanı ve 1 tam bağlantılı katmandan oluşmaktadır. Konvolüsyon katmanları için belirlenen hiper-parametreler sırasıyla şu şekildedir (*özellik haritası sayısı, filtre boyutu, aktivasyon*): {(64, 4, lineer), (256, 2, ReLU), (256, 7, lineer)}. Sadece son konvolüsyon katmanına ortaklama işlemi uygulanmış ve kernel boyutu 6 olarak seçilmiştir. Tam bağlantılı katman için hiper-parametreler şu şekildedir (*düğüm sayısı, aktivasyon, seyreltme*): {(1024, ReLU, 0.5)}. Yığın sayısı 100, öğrenme oranı 0.001 ve en iyileme yöntemi *adamax* seçilmiştir.

4.1.2 Parçacık sürü optimizasyonu

Dr. Eberhart ve Dr. Kennedy tarafından, 1995 yılında önerilen bu optimizasyon yöntemi sürülerin davranışları gözlemlenerek geliştirilmiştir. Önerilen bu yöntem GA'da olduğu gibi popülasyon tabanlı bir yöntemdir. GA'dan farklı olarak burada popülasyon yerine sürü, üretilen çözüm veya bireyler yerine parçacık kavramı kullanılmaktadır. Sürü halinde yaşayan balık ve kuş türleri, tehlikeler karşısında birlikte hareket ederek hayatta kalmaya çalışırlar. Önerilen optimizasyon yöntemindeki her bir parçacık, sürüdeki tek bir canlıyı temsil etmektedir. Her bir parçacık daha uzun süre hayatta kalabilmek için sürü içerisindeki en iyi parçacığın hareketlerini taklit eder. Bunun dışında her bir parçacığın daha önceki adımlarda almış olduğu kararlar, yapmış olduğu hareketler de saklanır. Bu bilgiler ışığında her bir parçacık, atacağı sonraki adımlarda, sürünün kararını veya kendi kararını değerlendirerek hareket eder. Her adım sonunda, her bir parçacık vermiş olduğu kararların faydasını hesaplar (amaç fonksiyonu). Daha önce atılan adım ile yeni atılan bu adımı kıyaslayarak, parçacıklar daha uzun süre hayatta kalmak, daha iyi sonuçlar elde edebilmek için atılması gereken en iyi adımı belirlemeye çalışır [46, 47]. PSO, GA'ya göre daha az optimize edilecek parametreye sahiptir ve GA ile rekabetçi sonuçlar elde etmektedir. Bu nedenle optimizasyon çalışmalarında sıklıkla kullanılmaktadır.

2017 yılında Lorenzo ve arkadaşları [48] tarafından PSO ile KSA hiper-parametre optimizasyonu çalışması gerçekleştirilmiştir. Bu çalışmada farklı hiper-parametre seçim algoritmalarının doğruluk değerleri ve hesaplama zamanları karşılaştırılmıştır.PSO'nun diğer hiper-parametre optimizasyon algoritmalarından, doğruluk oranı bakımından 0.0026 daha kötü olmasına ragmen, hesaplama zamanı bakımından 94 kat iyi olduğu öne sürülmüştür. Yapılan çalışmada, MNIST veri seti için en düşük hata oranını veren konvolüsyonel sinir ağı hiper-parametreleri

belirtilmiştir. Konvolüsyon katmanı için seçilen hiper-parametreler (*özellik haritası sayısı, filtre boyutu*) sırasıyla {(38, 6), (96, 9)}, ortaklama katmanları için hiper-parametreler (*filtre boyutu, aralık)* sırasıyla {(4, 2), (2, 4)} ve tam bağlantılı katman nöron sayısı 23 olarak seçilmiştir. Yapılan bu çalışmada konvolüsyon katmanında; filtre boyutu ve özellik haritası sayısı hiper-parametreleri, ortaklama katmanında ise filtre boyutu ve aralık hiper-parametreleri optimize edilmek için seçilmiştir.

2017 yılında Yamasaki ve arkadaşları [49] AlexNet mimarisi üzerinde çalışmışlardır. Bu çalışmada, CIFAR10, CIFAR100, Subset10, Subset30, Subset50 veri setleri kullanılmıştır. Optimizasyon sonucunda, 2012 yılında yayınlanan AlexNet mimarisinden daha iyi sonuçlar elde edilmiştir. Bu çalışmada, daha iyi doğruluk oranları elde edilmesine rağmen AlexNet mimarisine göre daha fazla hesaplama zamanı harcanmıştır. Önerilen çalışmada, konvolüsyon katmanında; filtre boyutu, özellik haritası sayısı ve dış dolgu hiper-parametreleri, ortaklama katmanında ise filtre boyutu ve ortaklama tipi hiper-parametreleri optimizasyon için seçilmiştir.

2017 yılında Sun ve arkadaşları [50], KSA mimarisinin dinamik olarak oluşturulduğu bir çalışma önermişlerdir. CIFAR10, MNIST, STL-10 ve Caltech-101 veri setleri için sırasıyla; 81.5, 118, 230 ve 22.5 saat hesaplama süreleri elde edilmiştir. Yapılan çalışmada, MNIST veri seti için en düşük hata oranını veren konvolüsyonel sinir ağı parametreleri belirtilmiştir. Konvolüsyon katmanı için hiper-parametreler (*özellik haritası sayısı, L2 katsayısı)* sırasıyla şu şekilde: {(100, 0.001), (82, 0.0018), (100, 0.001), (100, 0.001)} ve ortaklama katmanı için hiper-parametreler (*filtre boyutu, aralık*): {(2, 2)} şeklinde seçilmiştir. Bütün konvolüsyon katmanları için filtre boyutu 2 olarak seçilmiştir. Önerilen bu çalışmada, state-of-the-art algoritmalarla rekabetçi sonuçlar elde edilmiştir.

2018 yılında Silva ve arkadaşları [51] önerdikleri PSO tabanlı yöntemi tıbbi bir veri seti (LIDC-IDRI) üzerinde test etmişlerdir. Önerilen çalışmada her parçacık 8 koordinat ile temsil edilmiştir. Yapılan bu çalışmada, konvolüsyon katmanında; filtre boyutu ve özellik haritası sayısı hiper-parametreleri, ortaklama katmanında; ortaklama tipi hiper-parametresi ve bunlar dışında yığın boyutu ve seyreltme oranı hiper-parametreleri optimizasyon için seçilmiştir.

2018 yılında Lorenzo ve arkadaşları [52] tarafından KSA hiper-parametre optimizasyonu için PSO' nun kullanıldığı bir çalışma önerilmiştir. MNIST ve CIFAR-

10 veri setleri kullanarak eğittikleri topolojileri Xeon E5-2698v3 işlemci, 128 GB bellek ve 24 GB Nvidia Tesla K80 GPU ve i7-6850K işlemci, 32 GB bellek ve 12 GB Nvidia Titan X ultimate Pascal GPU'dan oluşan iki farklı sistem üzerinde çalıştırmışlardır. Konvolüsyon katmanında; filtre boyutu ve özellik haritası sayısı hiper-parametreleri, ortaklama katmanında ise filtre boyutu ve aralık hiper-parametreleri optimizasyon için seçilmiştir.

2018 yılında Wang ve arkadaşları [53] dinamik olarak genişleyebilen bir KSA önermişlerdir. Seçilen veri setlerinin eğitilmesi için 2 adet GTX1080 GPU'dan oluşan bir sistem kullanılmıştır. Kullanılan MNIST veri seti için en düşük hata oranını veren hiper-parametreler, konvolüsyon katmanı için (*özellik haritası sayısı, filtre boyutu, aralık)* sırasıyla {(26, 2, 1), (82, 6, 3), (114, 8, 4), (107, 7, 4)} ve tam bağlantılı katman için nöron sayısı 1686 olarak seçilmiştir. Yapılan çalışmada, konvolüsyon katmanında; filtre boyutu, özellik haritası sayısı ve aralık hiper-parametreleri, ortaklama katmanında; filtre boyutu, aralık ve ortaklama tipi hiper-parametreleri ve tam bağlantılı katmanda ise düğüm sayısı hiper-parametresi optimizasyon için seçilmiştir.

4.1.3 Diferansiyel evrim algoritması

Price ve Storn tarafından gerçekleştirilen bir çalışma ile ortaya atılan bu optimizasyon yöntemi Diferansiyel Evrim (DE) algoritması olarak adlandırılır. GA'ya çok benzeyen bu yöntem, GA ve PSO'da olduğu gibi popülasyon tabanlı bir optimizasyon yöntemidir. DE'de popülasyonları oluşturan yeni bireylerin üretilmesi sırasında rastgele seçilen 4 kromozom kullanılır ve bu kromozomların 3 tanesi yeni bireyin oluşturulması için çaprazlama ve mutasyon işlemlerine dahil edilir. Dördüncü kromozom ise hedef kromozom olarak adlandırılır ve çaprazlama ve mutasyon işlemlerinden sonra elde edilen yeni bireyle karşılaştırma işleminin gerçekleştirilmesi için kullanılır. Hedef kromozom ile yeni bireyin amaç fonksiyonları, KSA hiper-parametre optimizasyonu için doğruluk oranı veya hata değeri, karşılaştırılarak daha başarılı olan gelecek jenerasyona aktarılmak üzere seçilir [54, 55].

Literatürde, DE algoritması kullanılarak yapılan çalışmaları araştırdığımda kısıtlı sonuçlara ulaşabildim. Bu üst-sezgisel algoritmayı kullanarak KSA hiper-parametre optimizasyonu gerçekleştiren sadece bir çalışma bulabildim. 2018 yılında Wang ve arkadaşları [56] tarafından gerçekleştirilen KSA hiper-parametre optimizasyonu çalışmasında iki farklı üst-sezgisel yöntem karşılaştırılmıştır. Çalışma boyunca

üretilen ağlar dinamik bir şekilde oluşturulmuştur. Yani üretilen topolojiler başlangıçta belirlenen sabit bir uzunluğa sahip değildir. Yapılan çalışmada, konvolüsyon katmanında; filtre boyutu, özellik haritası sayısı ve aralık hiper-parametreleri, ortaklama katmanında; filtre boyutu, aralık ve ortaklama tipi hiper-parametreleri ve tam bağlantılı katmanda ise düğüm sayısı hiper-parametresi optimizasyon için seçilmiştir.

4.1.4 Harmonik arama

Geem ve arkadaşları tarafından 2001 yılında önerilen bu optimizasyon yöntemi, caz orkestrasının serbest performansı göz önüne alınarak geliştirilmiştir. Bu yeni optimizasyon yönteminde adım sayısı azaltılarak, iyi sonuçlara daha hızlı yakınsama hedeflenmiştir. HA optimizasyon yönteminde, her bir çözüm caz orkestrasındaki müzisyenlere karşılık gelmektedir. Müzisyenlerin kullandığı çalgı aletlerinin çıkardığı seslerin kalınlık ve incelik derecesi, seçilebilecek üst ve alt sınırları temsil etmektedir. Bir grup caz müzisyeninin ortaya koyduğu performansların her biri çözümleri temsil eder ve bu çözümlerin yani caz orkestrasındaki müzisyenlerin sergiledikleri uyum seyirciler tarafından değerlendirilir. Değerlendirme sonucunda seyirciler tarafından verilen puan, oluşturulan çözümlerin elde ettiği amaç fonksiyonuna karşılık gelir. Her bir çözümü oluşturan müzisyenler elde ettikleri amaç fonksiyonu değerine göre daha iyi sonuçlar elde etmeye çalışırlar [57].

2018 yılında Lee ve arkadaşları [58], farklı veri setleri için dinamik bir şekilde genişleyebilen (sabit topoloji uzunluklarına sahip olmayan) KSA topolojilerinin optimize edilmesine yönelik bir çalışma önermiştir. Önerilen çalışmada topolojilerin eğitilmesi için Intel Core i5-7500 işlemci, 8 GB bellek ve Nvidia GeForce GTX 970 GPU'dan oluşan bir sistem kullanılmıştır. Önerilen bu çalışmada konvolüsyon katmanında; filtre boyutu, filtre sayısı ve aralık değeri hiper-parametreleri, ortaklama katmanında; filtre boyutu ve aralık değeri hiper-parametreleri seçilmiştir. Seçilen bu hiper-parametrelere ait aralıklar net olarak belirtilmemiştir.

4.2 Hiper-parametre Optimizasyonu için Kullanılan Diğer Yaklaşımlar

Daha önceki bölümlerde anlatılan yöntemler dışında KSA hiper-parametre optimizasyonu çalışmalarında farklı optimizasyon yöntemleri de kullanılmaktadır.

2012 yılında Bengio ve arkadaşları [59], Sıralı Manuel Optimizasyon (Sequential Manual Optimization), Izgara Arama ve Rastgele Arama yöntemlerinin

karşılaştırıldığı bir çalışma önermişlerdir. Oluşturulan KSA modelleri, 8 farklı veri seti ile eğitilip, test edilmiştir Yapılan çalışmanın sonuçlarına bakıldığında; Rastgele Arama, optimize edilmek için 7 hiper-parametrenin seçildiği bir çalışmada Izgara Arama yöntemine göre daha hızlı yakınsayarak, bu yöntemden daha iyi doğruluk değerleri elde etmiştir. Farklı bir durumda, rastgele arama, optimize edilmek için 32 hiper-parametrenin seçildiği bir çalışmada Derin İnanç Ağı (Deep Belief Network) yöntemine göre rekabetçi sonuçlar elde etmiştir.

2015 yılında Domhan ve arkadaşları [60], *"Sequential Model-based Algorithm Configuration (SMAC)", "Tree Parzen Estimator (TPE)",* ve Rastgele Arama yöntemlerinin karşılaştırıldığı bir çalışma önermiştir. Yapılan çalışmada karşılaştırma işlemi, kullanılan hiper-parametre sayısı ve mimari büyüklüğü bakımından daha geniş olan KSA modeli ile diğerine göre daha küçük olan bir KSA modeli üzerinden gerçekleştirilmiştir. Oluşturulan topolojiler 4 adet Nvidia Titan GPU ile paralel olarak eğitilmiştir.

2018 yılında Saranyaraj ve arkadaşları [61] ise üst-sezgisel yöntemler dışında yeni bir yöntem önermişlerdir. Başlangıçta sabit olarak belirlenen bir konvolüsyonel sinir ağı mimarisi kullanılmıştır. Optimizasyon algoritmasının çalışması boyunca, elde edilen bütün KSA modelleri göğüs kanserine ait görüntüleri içeren *"Digital Database for Screening Mammography"* veri seti ile test edilmiştir. Yapılan çalışmada en iyi sonuçları veren hiper-parametreler, konvolüsyon katmanı kernel sayısı=5, maksimum ortaklama kernel boyutu=2, seyreltme oranı=0.5, tam bağlantılı katman aktivasyon fonksiyonu=ReLU ve nöron sayısı=256 olarak belirlenmiştir.

2018 yılında Neary ve arkadaşları [62], KSA hiper-parametre optimizasyonu için Pekiştirmeli Öğrenme (Reinforcement Learning – RL) yönteminin kullanıldığı bir çalışma önermişlerdir. Oluşturulan topolojiler Nvidia GPU ile eğitilmiştir. Üretilen topolojilere bakıldığında 2 konvolüsyon katmanı ve 2 tam bağlantılı katmandan oluşan topolojiler en uygun yapılandırma ayarı olarak seçilmiştir. İterasyon sayısı arttıkça konvolüsyon katmanı kernel sayısı sıklıkla 2 ve 3 olarak seçilmiştir. Aynı zamanda, seçilen kernel sayısı değerlerinin 32'den 72'ye ve seçilen nöron sayılarının 512'den 1280'e doğru yükseldiği görülmüştür. Yapılan bu çalışmada, konvolüsyon katmanında; katman sayısı, filtre boyutu ve özellik haritaları sayısı hiper-parametreleri, tam bağlantılı katmanda ise düğüm sayısı ve katman sayısı hiper-parametreleri optimizasyona dahil edilmiştir.

2018 yılında van Stein ve arkadaşları [63], KSA hiper-parametre optimizasyonu için Verimli Global Optimizasyon (Efficient Global Optimization - EGO) yönteminin kullanıldığı bir çalışma önermişlerdir. EGO yönteminde her bir iterasyonda yalnızca üretilen bir topoloji değerlendirilirken, yapılan çalışmada ise oluşturulan topolojilerin paralel olarak değerlendirilebildiği, yani her iterasyonda tek bir aday çözüm yerine birkaç aday topolojiyi elde edebilmek için EGO algoritmasının uygulanması önerilmiştir. Daha kısa bir şekilde, bu optimizasyon yöntemi öngörü ve modelin belirsizliğini hesaba katarak yeni aday topolojileri önermektedir. Oluşturulan topolojiler Nvidia K80 GPU ile eğitilmiştir. Yapılan çalışmada, konvolüsyon katmanında; filtre boyutu, özellik haritası sayısı ve aralık hiper-parametreleri optimizasyon için seçilirken bunun dışında öğrenme süreçleri için; seyreltme oranı, öğrenme katsayısı, L2 düzenlileştirme katsayısı ve hata optimizasyonu yöntemi hiper-parametreleri seçilmiştir.

2018 yılında Hinz ve arkadaşları [64] *"Sequential Model-based Algorithm Configuration (SMAC)", "Tree Parzen Estimator (TPE)",* Rastgele Arama ve GA yöntemlerinin karşılaştırıldığı bir çalışma önermişlerdir. Yapılan bu çalışmada, elde edilen KSA modelleri 3 farklı veri seti kullanılarak eğitilip, test edilmiştir. Çalışma sonuçlarına bakıldığında, KSA hiper-parametre optimizasyonu için gerçekleştirilen çalışmalarda TPE ve SMAC metotlarının hesaplama zamanını azaltmak için seçilebileceği öne sürülmüştür. Hesaplama zamanının çok kısıtlı olmadığı, yüksek doğruluk oranlarının asıl amaç, gereklilik olduğu durumlarda ise GA ve Rastgele Arama metotlarının seçilebileceği öne sürülmüştür.

5. DENEYSEL SONUÇLAR

Bu bölümde tez çalışmasında KSA hiper-parametre optimizasyonu için kullanılan optimizasyon yöntemi anlatılmıştır. μO kullanılarak elde edilen sonuçlar gösterilmiş, elde edilen sonuçlar literatürdeki çalışmalar ve bu tez çalışmasında kullanılan diğer yöntemler ile karşılaştırılıp, yorumlanmıştır.

5.1 Kullanılan Yöntemler

Bu bölümde tez çalışmasında KSA hiper-parametre optimizasyonu için kullanılan farklı optimizasyon yöntemleri anlatılmıştır. Yapılan tez çalışmasında, ana yöntem olarak bölüm 5.1.1'de anlatılan μO-Kısıtlı yöntemi seçilmiştir. Sadece bu yöntem için hassasiyet analizi gerçekleştirilmiştir. Çalışmada kullanılan diğer optimizasyon yöntemleri hassasiyet analizi gerçekleştirilmeden elde edilen sonuçlar üzerinden karşılaştırılmıştır.

5.1.1 Mikrokanonikal tavlama

Benzetimli tavlama (Simulated Annealing) [65], üst-sezgiseli bir olasılığa dayalı olarak iyileştirici olmayan hareketleri kabul etmek için Metropolis algoritmasının bir uyarlamasını kullanır. Eğer rastgele bir hareket mevcut amaç fonksiyonunu iyileştirirse, kabul edilir. Ancak kötüleşen hareketler, amaç fonksiyonundaki kötüleşme miktarı ve sıcaklık olarak adlandırılan *T* parametresi ile önyargılı bir üstel olasılık dağılımına göre kabul edilir. Başlangıçta, sistem yüksek bir sıcaklıkla başlatılır ve öngörülen tavlama programına göre sıcaklık kademeli olarak düşürülür. Sıcaklık, minimum sıcaklık değerine (T_{min}) ulaştığında arama işlemi durdurulur ve bu durum kötüleşen çözümlerin kabul edilmesini önler. BT çalışması sırasında, her sıcaklıkta, sistemin dengelenmesini sağlamak için yeterli sayıda hareket (move) yapılmalıdır. Optimizasyon başarısı için başlangıç sıcaklığı, soğutma hızı ve son sıcaklık gibi parametrelerin seçimi çok önemlidir.

Öte yandan, BT'nin bir varyantı olan Mikrokanonikal Tavlama (MT) algoritması sistemi, sıcaklığı ile değil iç enerjisiyle kontrol etmek için Creutz tekniğini [66, 67] kullanır. Creutz'un algoritmasında, sistemin toplam enerjisi korunur. Bunu sağlamak için mevcut sistemin toplam enerjisi, mevcut çözüm *S*'nin gerçek amaç fonksiyon değeri olan potansiyel enerji (E_P (S)) ve değişken enerji miktarı olan kinetik enerjinin (E_D) toplamı ile elde edilir (Denklem 5.1).

$$\mathbf{E}\,(\mathbf{S}) = \mathbf{E_P}(\mathbf{S}) + \mathbf{E_D} \qquad \textbf{(5.1)}$$

Mikrokanonik Tavlama yönteminde, S' olarak gösterilen yeni bir durum rastgele seçilir ve $\boldsymbol{\Delta E = E(S') - E(S)}$ olarak hesaplanan enerjideki değişime bağlı olarak kabul edilir veya reddedilir. Eğer $\boldsymbol{\Delta E \leq 0}$ ise, S' kabul edilir ve toplam enerjiyi korumak için sistemin kinetik enerjisi, E_D arttırılır. Eğer $\boldsymbol{\Delta E > 0}$ ise S' için kabul durumu E_D'ye bağlıdır. Eğer $\boldsymbol{\Delta E \leq E_D}$ ise değişiklik kabul edilir ve kinetik enerji (deamon energy) azalır, aksi takdirde S' reddedilir. MT periyodik olarak sistemden enerji çekilerek kontrol edilir. Belirli bir enerji seviyesinde bir dizi yinelemeden sonra, sistemdeki enerji, kinetik enerjiden sabit bir miktar çıkarılarak azaltılır. Algoritma aşağıdaki koşullar altında sonlandırılır: maksimum yineleme sayısına ulaşılırsa veya en iyi çözüm belirtilen adım boyunca sabit kalırsa. MT yönteminin sözde kodu şekil 5.1'de gösterilmiştir.

Algoritma 1: Mikrokanonikal Tavlama

```
1  E_D için başlangıç değerini ata;
2  while Sonlandırma koşulu gerçekleşene kadar do
3      repeat
4          S' çözümünü üret;
5          △E ← E(S') - E(S);
6          if △E ≤ 0 then
7              S ← S';
8              E_D ← E_D - △E;
9          else
10             if E_D - △E ≥ 0 then
11                 S ← S';
12                 E_D ← E_D - △E;
13     until Sistem dengeye ulaşana kadar;
14     E_D değerini azalt;
```

Şekil 5.1: Mikrokanonikal tavlama sözde kodu

Algoritma, muhtemelen yüksek enerjili rastgele bir durumla başlar, bu nedenle ilk E_D genellikle sıfıra ayarlanır. Şekil 5.1'de gösterildiği gibi, sistem belirli bir enerji seviyesinde dengeye ulaştıktan sonra, sistemin enerjisi belirli bir miktarda azalır. Barnard [67], denge (equilibrium) için yüksek enerji durumlarında kabul edilen hamle oranının dikkate alınmasının test için makul olacağını önerir. Ayrıca, başlangıçta belirlenen kinetik enerjiyle dengeye ulaştıktan sonra sistemden çıkarılacak enerjiyi de $\boldsymbol{E_{D0} / 300}$ olarak belirler. MT algoritması, SA algoritmasına göre birçok avantaja sahiptir. İlk olarak, MT, yüksek kalitede rasgele sayılar veya ağır hesaplamalar gerektirmeyen Creutz algoritmasına dayanır. İkincisi, creutz yöntemi üzerindeki deneyler, ayrık sistemler için geleneksel Metropol yönteminden daha hızlı bir büyüklük sırasını çalıştırmak için programlanabileceğini göstermektedir [68]. Bu avantajlarına rağmen, hesaplama zorlukları nedeniyle, MT algoritması, problemimiz

için umut verici sonuçlar sağlamadı. Bu nedenle, bu algoritmanın biraz değiştirilmiş bir versiyonunu bu çalışmada kullanıldı [69]. Algoritma dönüşümlü olarak uygulanan iki prosedürden oluşur: Başlatma ve örnekleme (Şekil 5.2). Başlatma aşamasında, açgözlü (greedy) bir yerel arama kullanılır. Algoritma yerel bir minimumda sıkışıp kaldığında, örnekleme aşaması başlar. Bu aşama, Creutz'un algoritmasına başvurarak çözümü yerel minimumdan kurtarmaya çalışır. Yazarlar, Torreao ve Roe [69], yöntemlerinin tavlamaya başvurmadığını belirtirler, bu yüzden bu yöntemi *Mikrokanonikal Optimizasyon Algoritması (**μO**)* olarak adlandırırlar. İki faz olan başlatma ve örnekleme adımları aşağıdaki gibi açıklanmıştır:

Algoritma 2: Mikrokanonikal Optimizasyon

```
1 while sonlandırma koşulu sağlanmadığı sürece do
2 |   Başlatma();
3 |   Örnekleme();
4 return S_Best
```

Şekil 5.2: Mikrokanonikal optimizasyon sözde kodu

Başlatma Fazı: Bu aşamada MT, yalnızca düşük maliyetli bir duruma neden olan hareketlerin kabul edildiği yerel bir arama yapar (YA). Sonlandırma kriteri olarak Linhares ve Torreao [70], bu fazı kesmek için üst üste reddedilen hareketlerin kullanılmasını önerir. Bu aşama için bir maksimum yineleme limiti de ayarlanabilir. Başlatma adımları şekil 5.3'te gösterilmiştir. Sözde kodda ***L***,daha sonra örnekleme aşamasındaki parametreleri tanımlamak için kullanılan reddedilen hareketlerin listesini belirtmek için kullanılır.

Algoritma 3: Başlatma Prosedürü

```
   Girdi: S, max_nbr_rejected, max_init_iter
1  ℒ ← []
2  nbr_rej ← 0;
3  nbr_iter ← 0;
4  while (nbr_rej < max_nbr_rejected and nbr_iter< max_init_iter) do
5  |   nbr_iter ← nbr_iter + 1;
6  |   Generate S';
7  |   △E ← E(S') - E(S);
8  |   if △E ≥ 0 then
9  |   |   Add △E to ℒ ;
10 |   |   nbr_rej ← n + 1;
11 |   else
12 |   |   nbr_rej ← 0;
13 |   |   S ← S';
```

Şekil 5.3: Başlatma prosedürü sözde kodu

Örnekleme Fazı: Bu aşama, Creutz'un algoritmasına başvurarak çözümü yerel minimumlardan kurtarmaya çalışır. Verilen sabit enerji seviyesi için (bkz. Denklem 5.1), bir çözüm örneği üretilir, ancak bu çözümün kabul edilmesi E_D değerine bağlıdır. İyileşen tüm hamleler kabul edilir ve kötüye giden hamleler, eğer kinetik enerji (deamon) oluşan maliyet farkını absorbe edebiliyorsa kabul edilir. Linhares ve Torreao [70], başlangıç aşamasında reddedilen hareketleri gözlemleyerek, kinetik enerjiyi (E_D) tanımlamak için uyarlanabilir bir strateji kullanmayı önermiştir. Reddedilen hareketler ***L*** ile gösterilen bir listeye yerleştirilir, daha sonra bu hareketlerle ilişkilendirilen maliyet artışları, örnekleme aşamasında kullanılacak ilk kinetik enerji değerini (D_I) tanımlamak için kullanılır. Başlangıçta, E_D, D_I değerine atanır. Örnekleme aşamasında, E_D'nin aynı listeden seçilen maksimum kinetik enerji değeri (D_{max}) ile sınırlanan değerleri almasına izin verilir. Örnekleme fazının adımları şekil 5.4'te gösterilen sözde koda göre gerçekleştirilmektedir.

Algoritma 4: Örnekleme Prosedürü

```
1  D_MAX ve D_I değerlerini list-of-rejected-moves listesinden seç;
2  max_samp_iter örnekleme adımlarının maksimum sayısını saklar;
3  S örnekleme fazında kullanılan başlangıç çözümünü temsil eder;
4  num_iter ← 0;
5  E_D ← D_I;
6  while (num_iter < max_samp_iter) do
7      Generate S';
8      △E ← E(S') - E(S);
9      if △E ≤ 0 then
10         S ← S';
11         E_D ← E_D - △E;
12
13     else
14         if E_D - △E ≥ 0 then
15             S ← S';
16             E_D ← E_D - △E;
17     num_iter ← num_iter + 1;
```

Şekil 5.4: Örnekleme prosedürü sözde kodu

Bölüm 5.2'de anlatıldığı gibi konvolüsyonel sinir ağı topolojilerinin oluşturulması sırasında belirli hiper-parametre ve topoloji kısıtları eklenmiştir. Bu sınırlamalar kullanılarak yapılan optimizasyon µO-Kısıtlı, hiper-parametre sınırlamaları olmadan yapılan optimizasyon ise µO-Kısıtsız olarak adlandırılmıştır.

5.1.2 Tree parzen estimator

Yapılan çalışmada µO'ya ek olarak bir yöntem daha denenmiştir. Bu yöntem Tree Parzen Estimator (TPE) olarak adlandırılır. TPE, "Bayesçi (Bayesian) Model Tabanlı Optimizasyon" algoritmalarının bir alt kümesinde yer alan optimizasyon algoritmasıdır. Bayesçi Hiper-parametre optimizasyonu şu şekilde tanımlanabilir: Amaç fonksiyonunun olasılıksal bir modeli oluşturulur. Oluşturulan bu model asıl

amaç fonksiyonunda değerlendirilmek üzere, en umut verici hiper-parametreleri seçmek için kullanılır. Hiper-parametre optimizasyonu için kullanılan genel notasyon denklem 5.2'de gösterilmiştir [71, 72].

$$x^* = argmin_{x \epsilon X} f(x) \quad (5.2)$$

- **f(x):** Amaç fonksiyonu
- **X:** Alan (domain - hiper-parametreler kümesi)
- **x:** X domaini içerisindeki herhangi bir alt küme
- **Amaç:** Validasyon seti için en iyi sonucu verecek hiper-parametre kümesini bulmak

Hiper-parametre optimizasyonu için Izgara Arama (Grid Search) ve Rastgele Arama (Random Search) yöntemleri sıklıkla kullanılmaktadır. Fakat bu yöntemler bazen verimsiz olabiliyor. Çünkü bu algoritmalar, önceki sonuçlara dayanarak bir sonraki hiper-parametreleri seçmezler, değerlendirmezler. Izgara ve Rastgele arama geçmişteki değerlendirmelerden, seçimlerden (evaluations) habersizdir. Bu nedenle harcanan zamanın önemli bir kısmı kötü hiper-parametreleri değerlendirmeyle geçer. Bayesçi optimizasyonda ise durum tam tersidir. Geçmişteki hareketler kaydedilir ve sonraki hareketler, geçmişteki hareketler göz önüne alınarak belirlenir. Aynı şekilde Izgara Arama ve Rastgele Arama gibi yöntemlerde çok sık bir şekilde "amaç fonksiyonunu (objective function)" çağırma ihtiyacı duyulur. Büyük hiper-parametre kümeleri için ve derin öğrenme ağları gibi yapıların optimizasyonu için her adımda "amaç fonksiyonunu" çağırmak çok maliyetlidir. Bunun yerine "Model Tabanlı Algoritmalar" ile bu çağırmalara bir limit koymak istenir [71, 72].

Bayes konseptine dayanan "Model Tabanlı Optimizasyon Algoritmalarında", amaç fonksiyonuna yapılan önceki çağrılara dayanarak, değerlendirmek (evaluate) için yalnızca en umut verici hiper-parametre setini seçerek amaç fonksiyonuna yapılan çağrı sayısı azaltılmaya çalışılmaktadır. Bir sonraki hiper-parametre seti, vekil (surrogate - bu çalışma için TPE) olarak adlandırılan amaç fonksiyonu istatistiksel

modeline dayanarak seçilir. Şekil 5.5'de "Sıralı Model Tabanlı Optimizasyon" algoritmasının adımları gösterilmiştir.

SMBO(f, M_0, T, S)

1 $\mathcal{H} \leftarrow \emptyset$, // Boş bir H listesi oluştur.
2 For $t \leftarrow 1$ **to** T, // Belirlenen sabit T sayısı kadar döngüyü çalıştır.
3 $x^* \leftarrow \text{argmin}_x\ S(x, M_{t-1})$, // S(x, M)'yi, S'yi en aza indiren x*'ı bulmak için birden fazla x örneği için çalıştırılır.
4 Evaluate $f(x^*)$, // x* için gerçek değerini hesapla
5 $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$, // x*, f(x*) değerleri için H listesini güncelle
6 Fit a new model M_t to $\mathcal{H}$. // Her yinelemeden sonra M, f'nin daha iyi bir yaklaşımı olacak şekilde güncellenir.
7 **return** $\mathcal{H}$

H: Gözlem geçmisi
T: Deneme sayısı
f: Objektif fonksiyon
M: Yaklaşık f değerini hesaplayan vekil fonksiyon (Tree Parzen Estimator)
S: Bir sonraki hiper-parametre seçimini hesaplar
x*: M'nin en aza indirildiği örnek.

Şekil 5.5: Sıralı Model Tabanlı Optimizasyon sözde kodu [71]

Şekil 5.5'deki adımlar kısaca özetlenmek istenirse:

1. Gerçek amaç fonksiyonunun, olasılıksal bir vekil modeli (M - bu çalışma için TPE) oluşturulur.
2. Vekil model üzerinde en iyi sonuçları veren hiper-parametreler bulunur (Adım 3).
3. Bu hiper-parametrelerin gerçek amaç fonksiyonu değeri hesaplanır (Adım 4).
4. Vekil modelin seçtiği hiper-parametre seti ve bu set için elde edilen gerçek amaç fonksiyonu değeri gözlem listesine eklenir (Adım 5).
5. Elde edilen yeni sonuçlara göre daha iyi bir yaklaşım üretmek için vekil model güncellenir (Adım 6).
6. 3-6 adımları maksimum iterasyon veya zaman limitine ulaşıncaya kadar tekrarlanır.

Algoritmanın sözde kodunda 3. adımda, bu çalışma için TPE yöntemi çalıştırılır ve sözde koda göre bu yöntem vekil fonksiyon olarak adlandırılır. TPE'nin başlatılabilmesi için en başlarda hiper-parametre seçimi için algoritmaya bir önceki dağılım (prior distribution) tanımlanılması gerekir. Yani TPE uygulamadan önce başlangıçtaki iterasyonlarda birkaç veri toplanması, TPE algoritmasının uyarılması, başlatılması gerekir. Bunu yapmanın en iyi ve en basit yolu, yalnızca Rasgele Arama'nın birkaç yinelemesini gerçekleştirmektir. Başlangıçta rastgele arama yönteminin ne kadar iterasyon çalıştırılacağı, TPE algoritması için kullanıcı tarafından tanımlanan bir parametredir. Bu çalışmada bu parametre değeri toplam

üretilecek çözüm sayısının %10'u olacak şekilde seçildi. Yani ilk 20 çözüm rastgele arama ile oluşturulacak. Rastgele Arama sırasında ne kadar fazla tekrar kullanılırsa, başlangıçta o kadar iyi bir dağılım elde edilir [71, 72, 73].

TPE'de direkt olarak en iyi gözlemleri kullanmak yerine en iyi gözlemlerin bir dağılımı kullanılır. TPE'yi başlatmak için yeterli veri toplandıktan sonra bir sonraki adımda gözlemler iki gruba ayrılır. İki grubun her biri için olabilirlik (likelihood) olasılığı modellenir. İlk grup, *l(x)*, değerlendirmeden sonra en iyi skorları veren gözlemleri içermektedir. Burada, *l(x),* $\{x^{(i)}\}$ gözlemleri kullanılarak oluşturulan yoğunluktur. Öyle ki $f\left(x^{(i)}\right) < y^*$ (eşik değeri) olmalı. İkinci grup, *g(x),* diğer bütün gözlemlerin yoğunluğudur. Her yinelemede, her hiper-parametre için TPE, en küçük (en iyi), kayıp (loss) değerleri ile ilişkilendirilmiş hiper-parametre değerleri kümesine bir *l(x)* ve diğer hiper-parametre değerlerine bağlı başka bir *g(x)* dağılımı uydurur. Ve amaç, *l(x) / g(x)* oranını maksimize eden "x" hiper-parametre değerinin seçilmesi yani birinci grupta olması muhtemel olan ve ikinci grupta bulunma olasılığı daha düşük olan bir dizi hiper-parametre bulmaktır. Denklem 5.3'de "x" hiper-parametreleri için oluşturulan dağılımların nasıl belirlendiği gösterilmiştir. En iyi gözlemlerin oranı kullanıcı tarafından TPE algoritması için bir parametre olarak tanımlanır. Bu çalışmada en iyi gözlemlerin oranı parametresi %20 olarak seçilmiştir [71, 72, 73].

$$p\,(x \mid y) = \begin{cases} l\,(x)\ eğer\ y < y^* \\ g(x)\ eğer\ y \geq y^* \end{cases} \tag{5.3}$$

- $\boldsymbol{y < y^*}$**:** Amaç fonksiyon değerinin, eşik değerinden daha küçük olduğunu temsil eder.

y^* genel olarak gözlemlenen en iyi kayıp (loss) değerinden daha az seçilir. Eğer amaç fonksiyonu değeri eşik değerinden küçük ise *l(x),* amaç fonksiyonu değeri eşik değerinden büyük ise *g(x)* dağılımı kullanılır.

Bu adayların hangi dağılım alanı içerisinde, yakınında olduğunun hesaplanabilmesi için seçim fonksiyonu kullanılır. Bu seçim fonksiyonuna göre bir sonraki hiper-parametre kümesinin seçilmesi sağlanır. Bu çalışmada daha önce yapılan çalışmalarda sıklıkla kullanılan "Beklenen İyileştirme (Expected Improvement)" seçim fonksiyonu kullanılmıştır (Şekil 5.6). Eldeki gözlemlere dayanarak bir sonraki adımda seçilecek hiper-parametre kümesinin belirlenmesi için her yinelemede belirlenen sayıda aday

çözümler üretilir (bu çalışma için 1000). Bu aday çözümler için vekil olasılık modeli ile yaklaşık amaç fonksiyonu değerleri hesaplanır. Aşağıdaki formül ile her aday için Beklenen İyileştirme (Expected Improvement) tanımlanır [71, 72, 73]:

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy = \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} dy$$

Şekil 5.6. Beklenen iyileştirme [71]

- **y*:** Amaç fonksiyonu için eşik (threshold) değeri
- **x:** Önerilen hiper-parametreler kümesi
- **y:** "x" hiper-parametreleri kullanıldığında amaç fonksiyonu için elde edilen gerçek (actual) değer
- **p (y | x):** Vekil olasılık modeli. "x" verildiğinde "y" olasılığını ifade eder
- **Amaç:** x'e göre Beklenen İyileştirmeyi en üst seviyeye çıkarmaktır. Yani; vekil fonksiyon (p (y | x)) altındaki en iyi hiper-parametreleri bulmaktır. Her bir aday konfigürasyon (hiper-parametre kümesi) için Beklenen İyileştirme hesaplanır. Her yinelemede TPE, en yüksek iyileştirmeye sahip olan konfigürasyonu, bir sonraki yineleme için kullanılacak konfigürasyon olarak seçer.

5.2 Çözümlerin Gösterimi

Konvolüsyonel sinir ağlarının mimarileri, sabit boyutta olmayabilir. Farklı çalışmalarda farklı yapılarda konvolüsyonel sinir ağları oluşturulabilir. Literatürdeki çalışmalar incelendiğinde, konvolüsyonel sinir ağlarının optimizasyonu için sabit ve değişken boyutta konvolüsyonel sinir ağları oluşturulup, oluşturulan bu ağların optimize edildiği görülmüştür. Yapılan bu tez çalışmasında oluşturulan topolojiler dinamik olarak değiştirilebilir ve değişken boyutlu bir yapıya sahiptir. Oluşturulan topolojilerde, konvolüsyon bloklarının sayısı minimum 2, maksimum 4 ve tam bağlantılı blokların sayısı minimum 0 ve maksimum 2 olacak şekilde tasarlanmıştır. Bu değerleri belirlerken, 2014 yılında ImageNet Büyük Ölçekli Görsel Tanıma Yarışması (ILSVRC), görüntü sınıflandırma kategorisinde ikinci en düşük hata değerini veren VGGNet [74] mimarisi örnek alınmıştır. Yapılan çalışmada konvolüsyon katmanı, aktivasyon fonksiyonu, yığın (batch) normalizasyonu, ortaklama katmanı ve seyreltme işlemi tek bir blok altında toplanmıştır ve bu blokların her biri konvolüsyon blok olarak adlandırılmıştır (Şekil 5.6). Aynı şekilde tam

bağlantılı katman, aktivasyon fonksiyonu ve seyreltme yöntemi de tam bağlantılı blok olarak adlandırılan tek bir blokta toplanmıştır (Şekil 5.7). Bu bloklar kullanılarak oluşturulan her bir çözüm şu notasyona göre oluşturulacaktır: [(CONV - BN - ACT) * N - (SUBS - DROP)] * M - [(FC - BN - ACT - DROP)] * K. Bu notasyonda kullanılan kısaltmalar tablo 5.1'de açıklanmıştır.

Tablo 5.1: Çözümler için kullanılan notasyon ve açıklamalar

Notasyon	Açıklama
CONV	Konvolüsyon Katmanı (Convolution Layer)
BN	Yığın (Batch) Normalizasyon
ACT	Aktivasyon Fonksiyonu
SUBS	Boyut Düşürme Katmanı (Pooling Layer, Strive Layer)
DROP	Seyreltme (Dropout) İşlemi
FC	Tam Bağlantılı Katman (Fully Connected Layer)

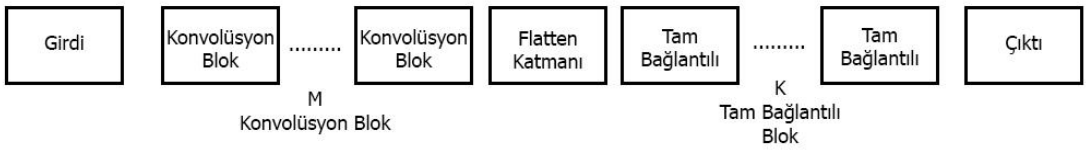
Aynı zamanda çözümlerin oluşturulması için kullanılan notasyondaki N, M ve K parametrelerinin aldığı değer aralıkları tablo 5.2'de gösterilmiştir. Bu parametrelerin aldığı değerlere göre oluşturulan konvolüsyonel sinir ağlarının mimarileri, boyutları değişkenlik göstermektedir.

Tablo 5.2: Çözümler için kullanılan notasyondaki parametreler ve değer aralıkları

Parametre	Değer Aralıkları
N	[2, 3]
M	[2, 4]
K	[0, 2]

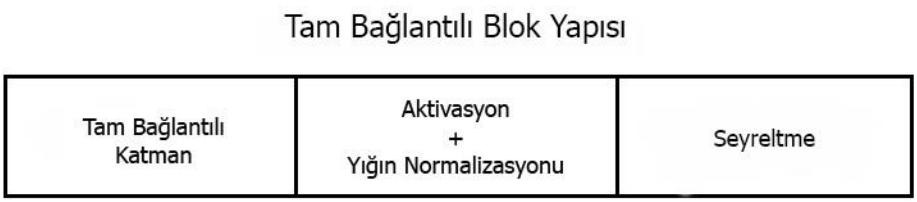
Yeni bir çözüm üretilirken, ilk olarak, kullanılacak olan aktivasyon fonksiyonu belirlenir. Seçilen aktivasyon fonksiyonu, tüm konvolüsyon ve tam bağlantılı katmanlarda ortak olarak kullanılır. Benzer şekilde, yeni çözümde, ortaklama ya da strive yöntemlerinden hangisinin seçileceğine en başta karar verilir. Daha sonra, seçilen yöntem tüm konvolüsyon bloklarında ortak olarak kullanılır. Bu çalışmada boyut düşürme işlemi için sadece ortaklama işlemi kullanılmamıştır. Ortaklama işlemine alternatif olarak *strive* [75] yapısı kullanılmıştır. Springenberg ve arkadaşları [75] tarafından yayınlanan çalışmada anlatılan bu yöntemde boyut düşürme fonksiyonu olarak kullanılabilecek bir alternatif önerilmiştir. Klasik konvolüsyonel sinir ağlarında boyut düşürme yöntemi olarak maksimum veya ortalama ortaklama katmanı kullanılır. Yapılan bu çalışmada ise boyut düşürme için ortaklama katmanı yerine, aralık değeri 2 olan, 3x3 veya 2x2 filtre boyutlarından oluşan konvolüsyon

katmanı kullanımı önerilmiştir. Bir bakıma konvolüsyon katmanı kullanılarak özellik öğrenimi işlemi devam ettirilmeye çalışılmıştır. Aynı zamanda konvolüsyon katmanının aralık değeri 2 olarak belirlendiğinden boyut düşürme işlemi de gerçekleştirilmiştir. Boyut düşürme işlemi için iki farklı alternatif sunularak farklı boyut düşürme yöntemlerinin sonuçları nasıl etkilediği gözlemlenmeye çalışılmıştır. Üretilen rastgele çözümlerin yapısı şekil 5.7'te gösterilmiştir. Aynı şekilde konvolüsyon ve tam bağlantılı blokların yapısı şekil 5.8 ve 5.9'da gösterilmiştir.



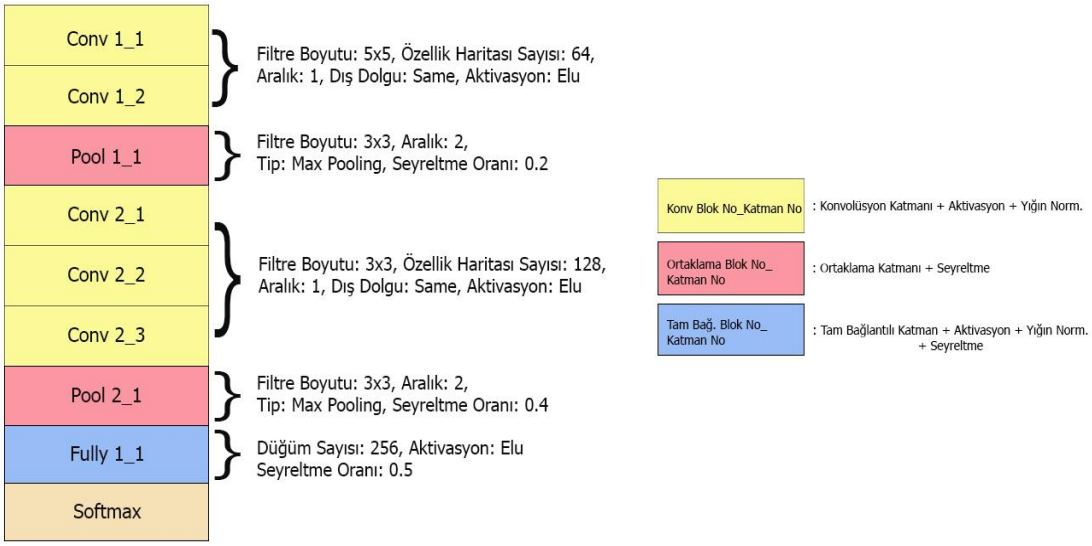
Şekil 5.7: Üretilen rastgele çözümlerin yapısı

Şekil 5.8: Konvolüsyon blok yapısı



Şekil 5.9: Tam bağlantılı blok yapısı

Konvolüsyonel sinir ağlarında hiper-parametrelerin optimizasyonu için kullanılan üst-sezgiseller optimizasyon işlemine belirli veya her çalıştırmada rastgele oluşturulan bir başlangıç çözümü ile başlatılır. Bu çalışmada kullanılan üst-sezgisel yöntem için sabit bir başlangıç çözümü seçilmiştir. Her bağımsız çalıştırma için seçilen bu başlangıç çözümü kullanılmıştır. Optimizasyon yönteminin başlangıç çözümünü ne kadar iyiye götürdüğü gözlemlenmeye çalışılmıştır. Optimizasyon boyunca başlangıç çözümünün değişmesine imkan tanındığından çalışma boyunca üretilen çözümler başlangıç çözümü ve birbirinden farklı olmaktadır. Şekil 5.10 'da başlangıç çözümü gösterilmiştir. Bu başlangıç çözümü 2 konvolüsyon bloğu ve 1 tam bağlantılı bloktan oluşur.



Şekil 5.10: Başlangıç çözümünün yapısı

Mevcut çözümden üst-sezgisel algoritmalarla yeni çözümler oluştururken, ağ topolojisi ve hiper-parametrelerin rastgele değişimini sınırlamak için bazı kısıtlamalar getirildi. Bu kısıtlamalar belirlenirken daha önce bu alanda yapılan çalışmalar göz önünde bulunduruldu ve ağın hiper parametrelerinin belirli kurallar çerçevesinde seçilmesinin tamamen rastgele seçim yerine daha başarılı sonuçlar verdiği gözlemlendi. Bu çalışmalar göz önüne alınarak belirlenen sınırlamalar şu şekildedir: Hiper-parametre kısıtlamaları ve topoloji kısıtlamaları.

Hiper-Parametre Kısıtlamaları:

1. Aralık (Stride) hiper-parametresi, ortaklama katmanı ve ortaklama katmanı yerine kullanılan konvolüsyon katmanları (strive layer) hariç 1 olarak sabitlenir ve aralık hiper-parametresi, ortaklama katmanı ve ortaklama katmanı yerine kullanılan konvolüsyon katmanları için 2 olarak sabit seçilir.
2. Aynı konvolüsyon bloğu içindeki konvolüsyon katmanları aynı hiper-parametre değerlerini kullanır [74, 75].
3. Konvolüsyon katmanlarının filtre boyutu belirlenirken, önceki konvolüsyon bloğundaki konvolüsyon katmanlarının filtre boyutundan daha küçük veya eşit bir filtre boyutu seçilir. Önceki konvolüsyon katmanının, filtre boyutundan daha küçük bir hiper-parametre değeri yoksa en küçük veya aynı hiper-parametre değeri seçilir.
4. Seyreltme (Dropout) işlemi, yalnızca her bir ortaklama katmanından sonra uygulanır (veya ortaklama yerine kullanılan *strive* katmanı) [75].

5. Ortaklama katmanı için seyreltme değeri belirlenirken, önceki konvolüsyon bloğundaki ortaklama katmanının seyreltme değeri veya bu değerin üstündeki hiper-parametre değeri seçilecektir. Maksimum seyreltme değeri 0.5 olarak seçilebilir [18, 76, 77, 78].
6. Kısıt 5, yalnızca ilk ortaklama (veya ortaklama yerine kullanılan strive) katmanına uygulanmaz. İlk katmanda kullanılan seyreltme değeri 0.2 olarak seçilecektir [75, 76, 79].
7. Her bir ortaklama katmanından (veya ortaklama yerine kullanılan strive) sonra, özellik haritası sayısı (kernel count) hiper-parametre değeri, önceki konvolüsyon bloğundaki özellik haritası sayısı değerinden en az otuz iki daha fazla olacaktır. Maksimum 256 değeri seçilebilir [74, 75].
8. Tam bağlantılı katman düğüm sayısı (unit count) değeri, önceki tam bağlantılı katman değerinin 2 katına eşit veya daha büyük olabilir. Düğüm sayısı değeri en az 128 ve en fazla 512 olabilir.
9. Tam bağlantılı katman, seyreltme (dropout) değeri önceki tam bağlantılı katman değerine göre belirlenir. Seçilen katman, varsa kendinden önceki tam bağlantılı katmandaki düğüm sayısı değeri ile aynı düğüm sayısı değerine sahipse, önceki tam bağlantılı katmanla aynı veya üzerindeki seyreltme değeri seçilir. Seyreltme değeri maksimum 0.5 olabilir [18, 74, 76]. Mevcut tam bağlantılı katman kendinden önce herhangi bir tam bağlantılı katmana sahip değil ise varsayılan olarak 0.3 seyreltme oranı seçilir.

Topoloji Kısıtları:

1. Her bir çözüm en az 2, en fazla 4 konvolüsyon bloğundan oluşabilir. Aynı zamanda her bir çözümde hiç tam bağlantılı blok olmayabilir veya en fazla 2 tam bağlantılı blok olabilir. Tam bağlantılı blok içermeyen çözümlerde, en son konvolüsyon katmanından sonra çözümün genelleştirme performansını iyileştirmek adına 0.5 seyreltme oranına sahip seyreltme işlemi eklenir.
2. Konvolüsyon bloklarında ortaklama katmanlarının silinmesine izin verilmez. Ortaklama katmanları, yalnızca aralık (stride) değeri 2 olan konvolüsyon katmanları (strive) ile değiştirilebilir.

3. Konvolüsyon bloğunda 3'ten fazla konvolüsyon katmanı olamaz. Mevcut konvolüsyon bloğunda 2 konvolüsyon katmanı varsa 0.5 olasılıkla yeni konvolüsyon bloğu eklenmesine izin verilebilir. Aynı şekilde mevcut konvolüsyon bloğunda 3 konvolüsyon katmanı varsa 0.5 olasılıkla bir konvolüsyon katmanının silinmesine izin verilebilir.
4. Yeni bir konvolüsyon bloğu eklemek çok düşük olasılıkta (0.0625 olasılık) yapılabilir.
5. Her konvolüsyon bloğu, en az 2, en fazla 3 konvolüsyon katmanı ve en fazla 1 ortaklama katmanı ya da strive katmanı içerebilir. Ortaklama katmanı veya strive katmanı olmayan konvolüsyon bloğu oluşturulamaz.

Konvolüsyonel sinir ağları, optimize edilecek birçok hiper-parametreye sahiptir. Bu çalışmada seçilen hiper-parametreler ve değer aralıkları sözlük (dictionary) yapısında gösterilmiştir. Şekil 5.5'te gösterilen yapıda rastgele çözümler üretmek için, sözlük yapısında tutulan hiper-parametre değerleri seçilir. Bu hiper-parametreler ve değer aralıkları tablo 5.3'de gösterilmiştir.

Tablo 5.3: Optimizasyon için seçilen hiper-parametreler ve değer aralıkları

Katman	Hiper-Parametre	Değer Aralığı
Konvolüsyon Katmanı	Aralık	1, 2
	Dış Dolgu	SAME
	Filtre Boyutu	3, 5, 7
	Özellik Haritası Sayısı	32, 64, 128, 160, 192, 224, 256
Ortaklama Katmanı	Aralık	2
	Filtre Boyutu	2, 3
	Ortaklama Tipi	MAX, AVG
Tam Bağlantılı Katman	Düğüm Sayısı	128, 256, 512
	Seyreltme Oranı	0.3, 0.4, 0.5
Öğrenme Süreci	Yığın Boyutu	32, 64, 128
	Öğrenme Oranı	1e-2, 1e-3, 1e-4
	Aktivasyon Fonksiyonu	ReLU, Leaky-ReLU, Elu

Konvolüsyonel sinir ağlarında hiper-parametre optimizasyonu gerçekleştirilen çalışmalarda hiper-parametrelerin ve değer aralıklarının seçimi hesaplama gücüne ve kullanılan veri setlerine de bağlı olabilmektedir. Örneğin: Özellik haritası sayısı hiper-parametresinde yüksek değerlerin seçilmesi hesaplama yükünü ve modelin toplam parametre sayısını arttıracaktır. Aynı şekilde tam bağlantılı katmanda düğüm sayısı

hiper-parametresinin seçimi de hesaplama yükünü ve toplam parametre sayısını doğrudan etkilemektedir. Bu nedenle bu hiper-parametreler için değer aralıkları belirlenirken kullanılan veri setinin zorluğu ve sahip olunan hesaplama gücü de dikkate alınmalıdır. MNIST gibi yüksek doğruluk oranlarının kolay bir şekilde elde edildiği veri setleri için 1024, 2048 gibi yüksek hiper-parametre değerlerinin seçilmesi gereksiz olabilir. Aynı şekilde konvolüsyon katmanında kullanılan filtre boyutu hiper-parametresi kullanılan veri setindeki örneklerin yükseklik ve genişlik değerleri göz önüne alınarak seçilmelidir. CIFAR10, FashionMNIST ve MNIST gibi nispeten küçük görüntü boyutlarından oluşan veri setleri için 11 veya üstü filtre boyutlarının seçilmesi görüntülerin aşırı küçülmesine ve böylece özellik kaybına neden olabilir. Aynı zamanda yüksek değerlerde seçilen filtre boyutları, yüksek özellik harita sayıları ile beraber kullanılırsa hem hesaplama yükünü hem de modelin toplam parametre sayısını arttırır. Yapılan bu çalışmada, kullanılan veri setleri ve sahip olduğumuz hesaplama gücü dikkate alınarak tablo 5.3'te ki hiper-parametreler ve değer aralıkları belirlenmiştir.

5.3 Mikrokanonikal Optimizasyon Parametre Seçimi

Yapılan çalışma boyunca üretilen topolojilerin eğitilip test edilmesi için Google tarafından araştırmacılara ücretsiz olarak sunulan Google Colab servisi kullanılmıştır. Aynı sanal makine üzerinde günlük 12 saatlik bir kullanım hizmeti sağlayan servisin donanım özellikleri şu şekildedir:

- **İşlemci:** Intel(R) Xeon(R) CPU @ 2.20GHz
- **Bellek:** 13 GB
- **Ekran Kartı:** Tesla T4 GPU
- **Depolama:** 358 GB

Yapılan çalışmada, iki farklı deneysel ortam hazırlandı. Seçilen optimizasyon yöntemi ile KSA hiper-parametre optimizasyonunun gerçekleştirilmesi sırasında birçok KSA topolojisi, çözüm eğitilip test edileceğinden ve bu durum yüksek hesaplama gücü, zamanı gerektirdiğinden optimizasyon algoritmasının çalışması boyunca üretilen KSA topolojileri için eğitim dönemi (epoch) 5 olarak belirlendi. Aynı zamanda, KSA hiper-parametre optimizasyonu adımlarında hız kazanmak için, çalışma boyunca kullanılan her bir veri seti için rastgele seçilmiş örneklerle veri setlerinin yarısı büyüklüğünde bir veri kümesi oluşturuldu. Seçilen eğitim dönemi ve veri setleri ile eğitilen topolojiler,

eğitim (train) veri setindeki toplam örnek sayısının % 10'u kullanılarak oluşturulan rastgele bir örnek kümesi ile test edildi. Eğitim veri setinden ayrı olan bu küme, ağırlıkların öğrenilmesine dahil edilmedi. Bu küme doğrulama seti (validation set) olarak adlandırılır. Her bağımsız çalıştırmada, eğitilen modelleri aynı örnekler ile test edebilmek için sabit bir "random seed" değeri belirlendi. Bu sayede optimizasyon çalışması boyunca, aynı veri seti için her çalıştırmada aynı eğitim, test ve doğrulama kümeleri elde edilmiş oldu. Eğitim dönemleri boyunca her model için eğitim ve validasyon hatası değerleri hesaplandı. Son eğitim dönemindeki (beşinci dönem) validasyon hatası, eğitilen modelin amaç fonksiyonu değeri olarak belirlendi. Optimizasyon algoritmasının çalışması boyunca kullanılan bu ilk deneysel ortamda oluşturulan topolojilerin test edilmesinde, veri setleri için sunulan test kümesi değil, eğitim kümesinden ayrılan validasyon kümesinin kullanıldığı dikkate alınmalıdır.

Optimizasyon algoritmasının çalışma hızının arttırılması için kurulan bu ilk deneysel ortamın yanında, optimizasyon algoritması sonucunda her bir veri seti için elde edilen en iyi KSA topolojilerinin eğitilmesi için ise farklı bir ortam hazırlanmıştır. Üretilen en iyi KSA topolojilerinin eğitilmesi için eğitim dönemi sayısı 200 olarak belirlenmiştir. Aynı zamanda ilk deneysel ortamdan farklı olarak bu ortamda veri setlerindeki bütün örnekler eğitim ve test için kullanılmıştır. Eğitilen modellerin, aşırı öğrenmeye uğrayıp uğramadığını kontrol etmek için eğitim veri kümesinden ayırarak oluşturduğumuz validasyon kümesini kullandık. Validasyon kümesi, ağırlıkların öğrenilmesine dahil edilmedi. 200 epoch sonunda eğitilen KSA topolojisi, veri setini oluşturan kişiler tarafından belirlenen orijinal test kümesi ile test edildi. Elde edilen değerler performans karşılaştırılması yapılmak üzere, KSA topolojisinin doğruluk oranı ve hata değeri olarak kaydedildi.

KSA topolojilerinin optimizasyonu dışında optimizasyon için kullanılan sezgisel yöntemler kendi içerisinde de belirlenmesi gereken parametrelere sahip olduğundan dolayı bu parametrelerin de doğru seçilmesi gerekmektedir. μO'da kullanılan bazı kritik parametreler şunlardır: *max_rejected_moves*; başlatma aşamasını sonlandırmak için başlatma aşamasında kaç ardışık çözümün reddedildiğini belirler, *max_init_iteration*; başlatma aşamasında maksimum yineleme sayısını belirler, max_{samp}; örnekleme aşamasındaki maksimum yineleme sayısını belirler. Bu parametrelerin doğru seçilmesinin sonuç üzerinde olumlu bir etkisi olacağı düşünüldü. Bu nedenle, bu parametrelerin değerleri değiştirilip, sonuçlar gözlemlendi.

Yapılan bu çalışmada, μO için seçilecek parametreleri belirlemek için bazı konfigürasyonlar oluşturuldu. Konfigürasyonların test edilmesi sırasında, her konfigürasyon için aynı sayıda minimum çözüm üretilmesini sağlamak için bütçe (her çalıştırmada üretilecek minimum çözüm sayısı), 200 olarak belirlendi. Oluşturulan yapılandırmalar ve seçilen parametre değerleri, Tablo 5.4'te gösterilmiştir. Bu parametrelerin ne anlama geldiğini açıklamak gerekirse: *"min_nbr_cycle"* parametresi, çalışma sırasında gerçekleşecek minimum döngü sayısını gösterir. Ardışık olarak *başlatma* ve *örnekleme* prosedürlerinin çalıştırılması 1 döngü olarak adlandırılır. Bu nedenle, 10 döngü, *başlatma* ve *örnekleme* adımlarının art arda 10 kez çalıştırıldığı anlamına gelir. Seçilen bütçe (önceden tanımlanmış 200 çözüm bütçesi) ve "*min_nbr_cycle"* parametreleri, döngü başına oluşturulacak maksimum çözüm sayısını belirtir. Örneğin, *bütçe* parametresi 200 olarak ayarlanmışsa ve *"min_nbr_cycle"* parametresi 20 ise, döngü başına maksimum 10 çözüm üretilebilir (*bütçe / min_nbr_cycle*). Aynı zamanda, "*init_iteration_ratio"* ve "*sampling_iter_ratio*" parametreleri, döngü başına üretilecek maksimum çözüm sayısının başlangıç ve örnekleme adımlarına dağılımını belirler. Örneğin, her bir döngü için maksimum 20 çözüm üretiliyorsa ve "*init_iteration_ratio"* değeri 0.7 ve *"sampling_iter_ratio"* değeri 0.3 ise *başlatma* aşamasında en fazla 14, *örnekleme* aşamasında en fazla 6 çözüm üretilir. Yapılan çalışmada, *"max_nbr_rejected"* parametresi, *başlatma* aşamasında oluşturulacak maksimum çözüm sayısının yarısı olacak şekilde ayarlanmıştır.

Tablo 5.4: μO için test konfigürasyonları

Konfigürasyon	Min_nbr_cycle	İnit_iteration_ratio	Sampling_iter_ratio
Konfig. 1		0.7	0.3
Konfig. 2	20	0.8	0.2
Konfig. 3		0.9	0.1
Konfig. 4		0.7	0.3
Konfig. 5	10	0.8	0.2
Konfig. 6		0.9	0.1

Yapılan çalışmada, bağımsız çalıştırmalar sonucunda elde edilen çözümlerin en iyisini depolamak yerine, kabul edilen çözümler için bir arşiv tutmaya karar verdik. Bağımsız çalıştırmalar bittiğinde, arşivdeki çözümler artan hata değeri ile sıralanır ve bu sıralamadaki ilk 3 çözümün topoloji bilgileri bir sonraki adımda daha uzun eğitim dönemlerinde eğitilmek için bir metin dosyasına kaydedilir. Bu strateji, μO için en iyi konfigürasyonun belirlenmesinde kullanıldı. μO için uygun konfigürasyonların

belirlenmesi dışında yapılan optimizasyon çalışmasında arşiv çözümlerinin sayısı 5'e çıkarıldı. µO'da parametrelerin belirlenmesi için oluşturulan konfigürasyonlar (Tablo 5.4), CIFAR10 ve FashionMNIST veri setleri ile test edilmiştir. Her bir konfigürasyon seçilen farklı veri setleri üzerinde çalıştırılmıştır. Farklı konfigürasyonlar için çalıştırılan µO'nun, çalışması boyunca üretilen bütün çözümler içerisinden kabul edilen çözümlerin ve elde edilen en iyi çözümlerin istatistikleri Tablo 5.5, 5.6, 5.7 ve 5.8'de gösterilmiştir (Daha düşük değerler daha iyi). Bu tablolarda elde edilen hata değerleri gösterilmiştir.

Tablo 5.5: CIFAR10 veri seti için kabul edilen çözümlerin istatistikleri

Kabul Edilen Çözümler					
Konfig.	**Min.**	**Maks.**	**Ortalama**	**Medyan**	**Std. Sapma**
Konfig 1	0.2778	0.4610	0.3592	0.3556	0.0413
Konfig 2	0.2966	0.4428	0.3765	0.3766	0.0343
Konfig 3	0.2824	0.4012	0.3466	0.3450	**0.0310**
Konfig 4	**0.2652**	**0.3928**	**0.3350**	**0.3379**	0.0322
Konfig 5	0.2848	0.4354	0.3609	0.3576	0.0387
Konfig 6	0.3102	0.5004	0.3683	0.3562	0.0478

Tablo 5.6: FashionMNIST veri seti için kabul edilen çözümlerin istatistikleri

Kabul Edilen Çözümler					
Konfig.	**Min.**	**Maks.**	**Ortalama**	**Medyan**	**Std. Sapma**
Konfig 1	0.0810	0.1156	0.0978	0.0966	0.0076
Konfig 2	0.0804	0.1088	0.0957	0.0955	0.0064
Konfig 3	0.0806	0.1062	0.0935	0.0935	0.0062
Konfig 4	**0.0750**	0.1028	**0.0928**	**0.0925**	0.0063
Konfig 5	0.0820	0.1058	0.0948	0.0948	0.0060
Konfig 6	0.0806	**0.0998**	0.0934	0.0934	**0.0050**

Tablo 5.7: CIFAR10 veri seti için elde edilen arşiv çözümlerinin istatistikleri

Konfig.	Minimum	Ortalama	Standart Sapma
Konfig. 1	0.2778	0.2820	0.0036
Konfig. 2	0.2966	0.3104	0.0125
Konfig. 3	0.2824	0.3004	0.0157
Konfig. 4	**0.2652**	**0.2729**	0.0094
Konfig. 5	0.2848	0.2927	0.0069
Konfig. 6	0.3102	0.3119	**0.0023**

Tablo 5.8: FashionMNIST veri seti için elde edilen arşiv çözümlerinin istatistikleri

Konfig.	Minimum	Ortalama	Standart Sapma
Konfig. 1	0.0810	0.0830	0.0017
Konfig. 2	0.0804	0.0816	**0.0012**
Konfig. 3	0.0806	0.0822	0.0022
Konfig. 4	**0.0750**	**0.0806**	0.0049

Konfig. 5	0.0820	0.0828	0.0013
Konfig. 6	0.0806	0.0842	0.0031

Her konfigürasyon için μO'yu çalıştırdıktan sonra, hangi konfigürasyonun daha başarılı olduğunu belirlemek için her konfigürasyon için elde edilen en iyi (en düşük hataya sahip) 3 modeli, 200 eğitim dönemi boyunca eğittik. Bu uzun eğitim dönemi için yine CIFAR10 ve FashionMNIST veri setlerini kullandık ve bu veri setlerini 3 bölüme (eğitim, doğrulama ve test) ayırdık. Eğitilen modellerin (200 eğitim dönemi) başarısını ölçmek için test setini kullandık. Doğrulama setini, modelin fazla mı yoksa tam mı uygun olduğunu belirlemek için kullandık. Tüm eğitim dönemlerinde optimizasyon yöntemi olarak *Adam* optimizer'ı (öğrenme oranı, 1e-4) kullandık. Ayrıca tüm eğitim dönemleri için *Batch Size* hiper-parametresi 32 olarak seçildi. Uzun eğitim dönemleri (200 epoch) için elde edilen konfigürasyon istatistikleri tablo 5.9 ve 5.10'da gösterilmiştir (Model 1, 2 ve 3 çalışma boyunca ilgili veri seti için elde edilen en iyi modelleri temsil etmektedir).

Tablo 5.9: CIFAR10 veri seti için uzun eğitim dönemleri sonucunda elde edilen konfigürasyon istatistikleri

	Konfig. 1	Konfig. 2	Konfig. 3	Konfig. 4	Konfig. 5	Konfig. 6
Model 1	0.1706	0.1416	0.1360	0.1484	0.1701	0.1760
Model 2	0.1356	0.1488	0.1549	0.1471	0.1939	0.1713
Model 3	0.1685	0.1792	0.1622	0.1677	0.1573	0.1298
Ortalama	0.1583	0.1566	**0.1510**	0.1544	0.1738	0.1591

Tablo 5.10: FashionMNIST veri seti için uzun eğitim dönemleri sonucunda elde edilen konfigürasyon istatistikleri

	Konfig. 1	Konfig. 2	Konfig. 3	Konfig. 4	Konfig. 5	Konfig. 6
Model 1	0.0684	0.0713	0.0743	0.0703	0.0715	0.0745
Model 2	0.0799	0.0726	0.0784	0.0723	0.0778	0.0656
Model 3	0.0696	0.0682	0.0697	0.0885	0.0726	0.0807
Ortalama	0.0726	**0.0706**	0.0741	0.0770	0.0739	0.0736

Elde edilen bilgilerden yola çıkarak hangi konfigürasyonun daha başarılı olduğunu belirlemek için Kruskal-Wallis H-test istatistik testini kullandık. Kruskal-Wallis testi, tek yönlü varyans analizi veya ANOVA analizinin parametrik olmayan bir versiyonudur [80, 81]. Bu istatistik yönteminde H'nin *Chi Square* dağılımına sahip olduğu varsayımı nedeniyle, her gruptaki örnek sayısı çok küçük olmamalıdır. Her grup için en az 5 örnek olmalıdır. Yapılan Kruskal testleri sonucunda (CIFAR10 ve FashionMNIST veri setlerinin her ikisi için de ayrı olarak gerçekleştirildi), konfigürasyonlar arasında hiçbir fark olmadığı hipotezi reddedildi (null hipotez, H_0)

ve 6 farklı konfigürasyon arasında anlamlı bir fark olduğu ortaya çıktı. Bu konfigürasyonların hangilerinin birbirinden farklı olduğunun anlaşılması için Dunn post-test [82] uygulandı**.** Bu testler için $\boldsymbol{\alpha = 0.05}$ ve 0.01 seçilerek elde edilen sonuçlar gözlemlendi. Gözlemlenen sonuçlara göre konfigürasyon 4'ün 1, 5 ve 6'dan anlamlı bir şekilde daha iyi olduğu, konfigürasyon 2'nin 3 ve 4'ten anlamlı derecede daha kötü olduğu görüldü. Konfigürasyon 3 ve 4 arasında anlamlı bir fark bulunamadı. Elde edilen sonuçlara göre konfigürasyon 3 veya 4'ten hangisinin seçileceğinin belirlenmesi için tablo 5.9 ve 5.10'daki sonuçlara ek olarak sadece konfigürasyon 3 ve 4 için ikişer sonuç daha eklendi. Daha önce µO çalışması boyunca elde edilen en iyi 3 model uzun çalıştırılıp sonuçlar bu tabloda gösterilmişti. Şimdi bu iki konfigürasyon arasından hangisinin daha iyi olduğunun anlaşılması için sadece bu konfigürasyonlar için µO çalışması boyunca elde edilen en iyi 5 model ele alındı (Tablo 5.11). Tablodaki veriler kullanılarak t-test uygulandı ve sonuçlara bakıldığında yine konfigürasyon 3 ve 4 arasında anlamlı bir fark olmadığı gözlemlendi. Sonuç olarak bu iki konfigürasyon arasından birini seçmek için tablo 5.11'deki sonuçlara bakılarak en iyi sonucu elde eden konfigürasyon 3, µO'nun çalışması boyunca kullanılacak konfigürasyon olarak seçildi.

Tablo 5.11: CIFAR10 ve FashionMNIST veri setleri için konfigürasyon 3 ve 4'ün karşılaştırılması

	CIFAR10		FashionMNIST	
	Konfig. 3	Konfig. 4	Konfig. 3	Konfig. 4
Model 1	0.1360	0.1484	0.0743	0.0703
Model 2	0.1549	0.1471	0.0784	0.0723
Model 3	0.1622	0.1677	0.0697	0.0885
Model 4	0.1634	0.1821	0.0775	0.0756
Model 5	0.1726	0.1802	0.0761	0.0844
Ortalama	**0.1578**	0.1651	**0.0752**	0.0782

5.4 Doğruluk Oranı Ve Hesaplama Zamanı Bakımından Performans Değerlendirmesi

Bu bölümde, tez çalışmasında kullanılan optimizasyon yöntemi ile diğer KSA hiper-parametre optimizasyonu gerçekleştirilen çalışmalar karşılaştırıldı. Karşılaştırma işlemi, yapılan çalışmaların hepsinde parametre sayısı ve hesaplama süreleri belirtilmediğinden yalnızca doğruluk oranı üzerinden gerçekleştirildi. Aynı zamanda KSA hiper-parametre optimizasyonu gerçekleştirilmeyen state-of-the-art mimariler ile hiper-parametre optimizasyonu işlemi gerçekleştirilen çalışmalar parametre sayısı ve doğruluk oranı bakımından kıyaslandı. Doğruluk oranı veya hata değeri, makine

öğrenmesi yöntemlerinde performans karşılaştırması için sıklıkla kullanılmaktadır. Fakat bu çalışma derin öğrenme ağlarının özelleşmiş bir mimarisi olan KSA'ları hedef aldığından hesaplama zamanı karşılaştırması da büyük önem taşımaktadır.

KSA hiper-parametre optimizasyonu çalışmalarında üretilen topolojiler için yüksek doğruluk oranları elde edilmesinin yanında, düşük parametre sayılarına sahip topolojilerin üretilmesi de önem arz etmektedir. Düşük parametre sayısına sahip topolojiler daha hızlı eğitim ve test süreleri sunar. Bu nedenle yapılan çalışmalarda en az parametre sayısına sahip olmasının yanında en yüksek doğruluk oranlarını veren topolojilerin üretilmesi hedeflenmelidir. Tablo 5.12'de gerçekleştirilen KSA hiper-parametre optimizasyonu çalışmaları için kullanılan yöntemler ve elde edilen doğruluk değerleri gösterilmiştir.

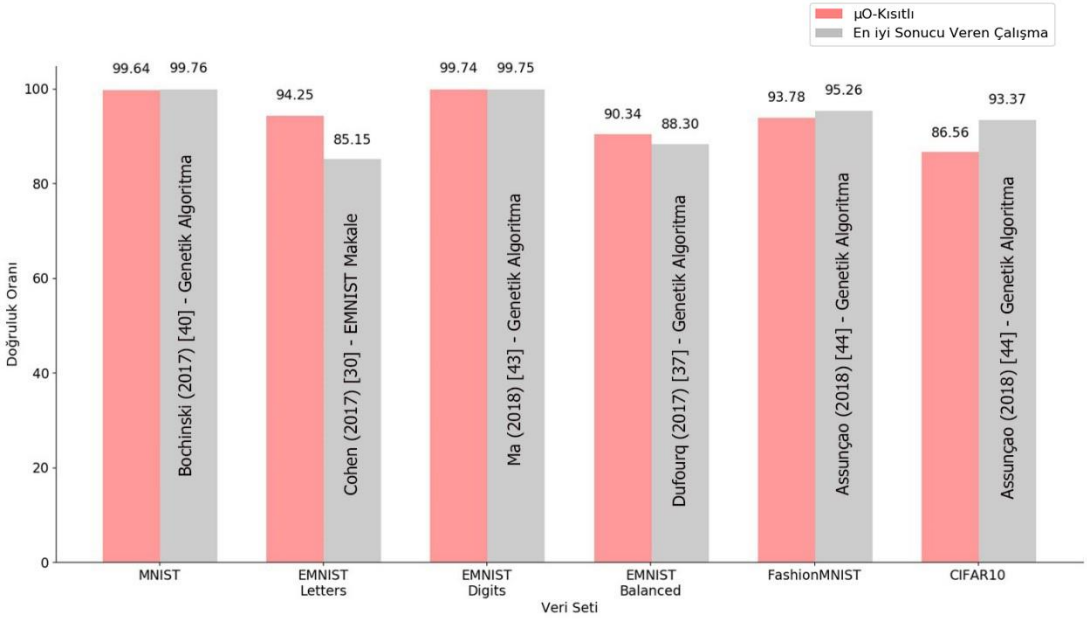
Tablo 5.12: KSA Hiper-Parametre optimizasyonu için gerçekleştirilen çalışmalar ve elde edilen doğruluk oranları

Veri Seti	Referans	Kullanılan Metot	Doğruluk Oranı (%)
CIFAR-10	Domhan (2015) [60]	Küçük KSA + TPE	81.88
		Küçük KSA + SMAC	82.53
		Büyük KSA+ SMAC	91.19
	Sun (2017) [50]	PSO	83.5
	Dufourq (2017) [37]	GA	75.5
	Lorenzo (2017) [48]	PSO	59.59
	Yamasaki (2017) [49]	PSO	80.15
	Assunçao (2018) [44]	GA	**93.37***
	Lorenzo (2018) [52]	PSO	58.47
	Lee (2018) [58]	HA	74.76
	Ma (2018) [43]	GA	89.23
	Van Stein (2018) [63]	EGO	86.46
	Yamasaki (2017) [49]	AlexNet	77.75**
	Lee (2018) [58]	CifarNet	73.32**
	Ma (2018) [43]	AlexNet	82.53**
	Ma (2018) [43]	VGGNet	84.62**
	Ma (2018) [43]	ResNet	90.61**
	Önerilen Yöntem	µO-Kısıtlı	86.56
Fashion-MNIST	Dufourq (2017) [37]	GA	90.6
	Sun (2017) [47]	GA	94.53
	Ma (2018) [43]	GA	94.59
	Assunçao (2018) [44]	GA	**95.26***
	Ma (2018) [43]	AlexNet	86.43**
	Ma (2018) [43]	VGGNet	90.45**
	Ma (2018) [43]	ResNet	94.39**
	Assunçao (2018) [44]	AlexNet	89.90**
	Assunçao (2018) [44]	VGGNet	93.50**
	Assunçao (2018) [44]	ResNet	94.90**
	Önerilen Yöntem	µO-Kısıtlı	93.78
MNIST	Sun (2017) [47]	GA	98.72
	Bochinski (2017) [40]	GA	**99.76***
	Lorenzo (2017) [48]	PSO	99.45
	Sun (2017) [50]	PSO	99.34
	Assunçao (2018) [44]	GA	99.70

	Baldominos (2018) [45]	GA	99.63
	Lorenzo (2018) [52]	PSO	98.82
	Wang (2018) [53]	PSO	98.79
	Wang (2018) [56]	DG	98.24
	Lee (2018) [58]	HA	99.25
	Ma (2018) [43]	GA	99.64
	Neary (2018) [62]	RL	95.8
	Van Stein (2018) [63]	EGO	99.39
	Lorenzo (2017) [48]	LeNet-4	98.9**
	Sun (2017) [50]	LeNet-4	98.9**
	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**
	Önerilen Yöntem	µO-Kısıtlı	99.64
EMNIST-Balanced	Cohen (2017) [30]	OPIUM Sınıflandırıcı	78.02
	Dufourq (2017) [37]	GA	88.3
	Önerilen Yöntem	µO-Kısıtlı	**90.34***
EMNIST-Digits	Cohen (2017) [30]	OPIUM Sınıflandırıcı	95.90
	Dufourq (2017) [37]	GA	99.3
	Ma (2018) [43]	GA	**99.75***
	Önerilen Yöntem	µO-Kısıtlı	99.74
EMNIST-Letter	Cohen (2017) [30]	OPIUM Sınıflandırıcı	85.15
	Önerilen Yöntem	µO-Kısıtlı	**94.25***

***** : *İlgili veri seti için en iyi doğruluk oranını elde eden çalışma*
****** : *Herhangi bir hiper-parametre optimizasyonu yapılmadan elde edilen doğruluk oranları*

Tablo 5.12 ve şekil 5.11'e bakıldığında, bu çalışmada kullanılan veri setleri için GA üst-sezgiselinin açık bir şekilde en iyi sonuçları verdiği gözlemlenmiştir. GA üst-sezgiseli kullanılarak CIFAR-10 veri seti için 93.37%, MNIST için 99.76%, FashionMNIST için 95.26%, EMNIST-Digits için 99.75% doğruluk oranları ile en iyi sonuçlar elde edilmiştir. Bu çalışmada önerilen yöntem 6 veri seti için de rekabetçi sonuçlar elde etmiştir.



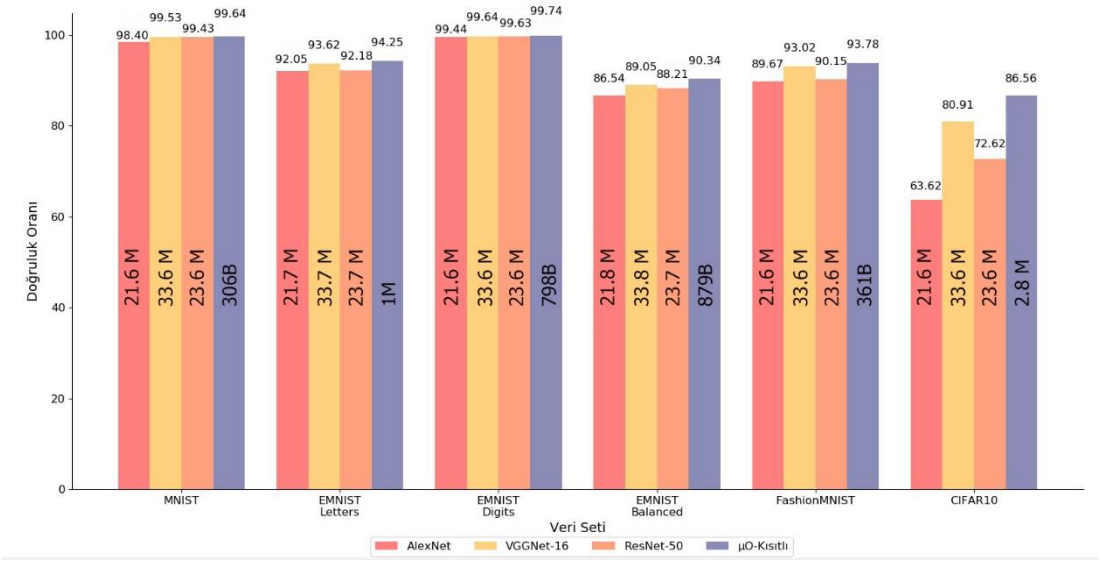
Şekil 5.11: KSA hiper-parametre optimizasyonu için literatürdeki çalışmalar ile µO-Kısıtlı yönteminin performans karşılaştırması

Tablo 5.13: Farklı veri setleri için eğitilip, test edilen topolojilerin parametre sayısı ve doğruluk oranı

Veri Seti	Yöntem	Parametre Sayısı	Doğruluk Oranı
MNIST	**µO-Kısıtlı**	**306,730**	**99.64**
	Dufourq (2017) [37]	1,857,601	98.4
	AlexNet	21,600,330	98.94
	VGGNet-16	33,637,066	99.53
	ResNet-50	23,601,930	99.43
EMNIST-Letters	**µO-Kısıtlı**	**1,005,381**	**94.25**
	AlexNet	21,710,949	92.05
	VGGNet-16	33,747,685	93.62
	ResNet-50	23,657,253	92.18
EMNIST-Digits	**µO-Kısıtlı**	**798,026**	**99.74**
	Dufourq (2017) [37]	3,001,576	99.3
	AlexNet	21,600,330	99.44
	VGGNet-16	33,637,066	99.64
	ResNet-50	23,601,930	99.63
EMNIST-Balanced	**µO-Kısıtlı**	**879,055**	**90.34**
	Dufourq (2017) [37]	1,688,43	88.3
	AlexNet	21,751,919	86.54
	VGGNet-16	33,788,655	89.05
	ResNet-50	23,677,743	88.21
FashionMNIST	µO-Kısıtlı	361,834	93.78
	Dufourq (2017) [37]	4,624,447	90.06
	Sun (2017) [47]	**6,680,000**	**94.53**
	AlexNet	21,600,330	89.67
	VGGNet-16	33,637,066	93.02
	ResNet-50	23,601,930	90.15
CIFAR10	**µO-Kısıtlı**	**2,845,962**	**86.56**
	Dufourq (2017) [37]	172,767	74.5
	AlexNet	21,623,562	63.62
	VGGNet-16	33,638,218	80.91
	ResNet-50	23,608,202	72.62

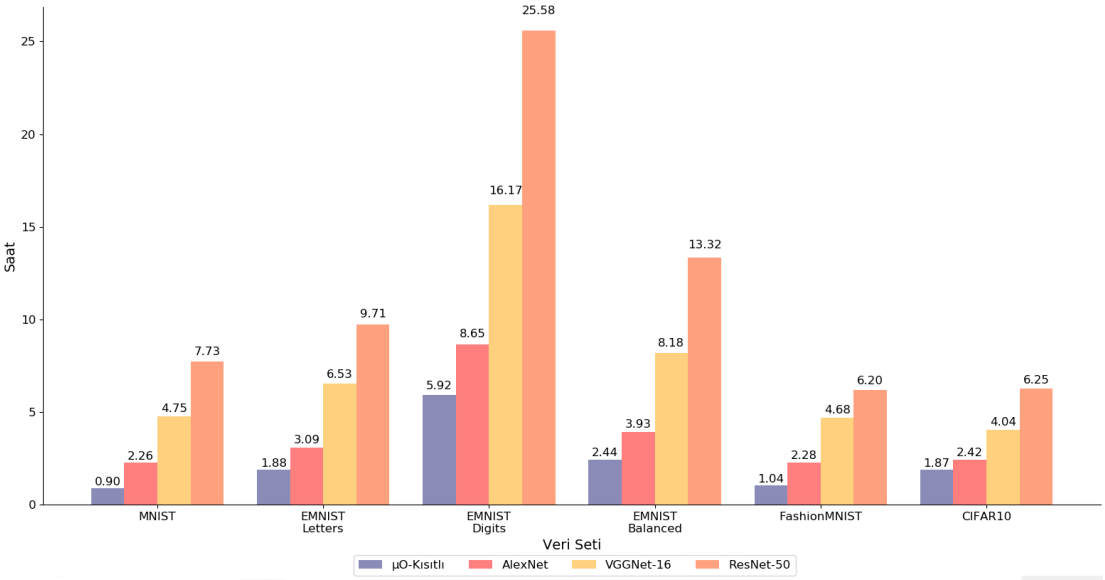
Tablo 5.13'e bakıldığında, bu çalışmada önerilen yöntemin KSA hiper-parametre optimizasyonu gerçekleştirmeyen state-of-the-art algoritmaların birçoğundan hem doğruluk oranı olarak, hem de parametre sayısı bakımından daha iyi olduğu gözlemlenmiştir. Tablo 5.13'deki veriler Şekil 5.12'de görselleştirilmiştir. Tablo 5.13'te daha önce KSA hiper-parametre optimizasyonu yapılan çalışmalardan, oluşan en iyi topoloji için parametre sayısını belirten çalışmalarda gösterilmiştir. Bu çalışmaların sonuçları ile tez çalışmasında önerilen yöntem kıyaslandığında, FashionMNIST veri seti hariç diğer bütün veri setlerinde doğruluk oranı bakımından daha iyi sonuçlar elde edilmiştir. FashionMNIST veri seti için tez çalışmasında önerilen yöntem, en iyi sonuçtan 0.75 daha kötü doğruluk oranı elde etmesine rağmen, parametre sayısı bakımından en iyi sonuçtan 18.5 kat daha iyidir. Doğruluk oranlarının yanında, tez çalışmasında önerilen yöntem CIFAR10 veri seti hariç diğer tüm veri setlerinde en az parametre sayısına sahip topolojileri üretmiştir. Tez çalışmasında önerilen yöntem, parametre sayısı bakımından diğer çalışmalardan en az 1.92 kat, en

fazla 109.6 kat daha iyi sonuçlar elde etmiştir. Önerilen bu tez çalışmasında rekabetçi doğruluk oranları yanında, düşük parametre sayılarına sahip topolojilerin üretildiği görülmektedir.



Şekil 5.12: State-of-the-art mimariler ile µO'nun doğruluk oranı açısından karşılaştırılması

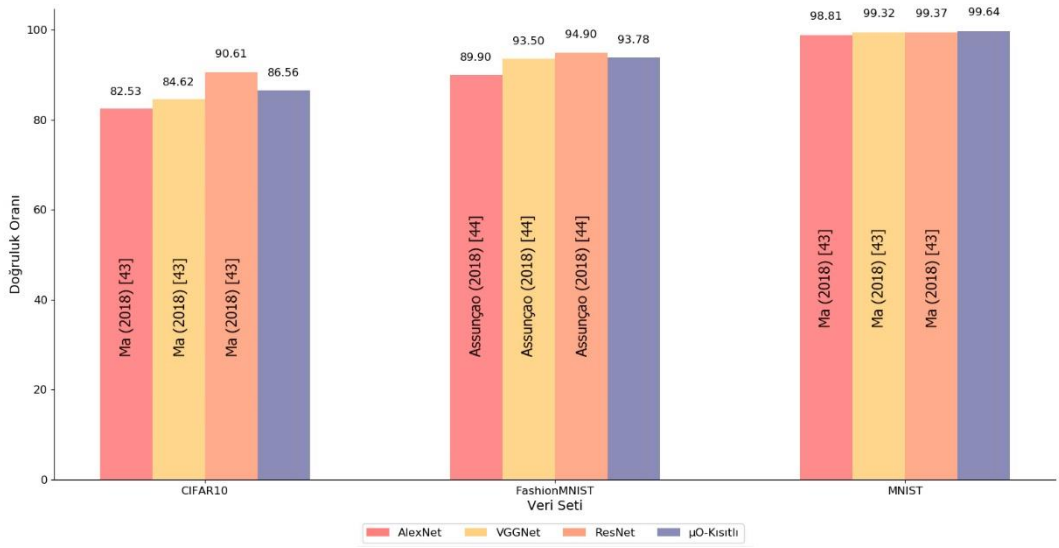
Tablo 5.12'de verilen çalışmalardaki bilgilerle kapsamlı bir şekilde karşılaştırma yapabilmek ne yazık ki mümkün değildir. KSA hiper-parametre optimizasyonu çalışmalarında sadece yüksek doğruluk oranları değil, aynı zamanda düşük toplam parametre sayısına sahip topolojilerin de üretilmesi hedeflenmelidir. Bu sayede hesaplama maliyeti azaltılarak, mobil cihazlarda daha düşük enerji tüketimi gerçekleştirilebilir. Bu durum, anlık, hızlı bir şekilde cevap üretilmesi, tepki verilmesi gereken nesne tespiti, nesne konumlandırma, işaretleme gibi bilgisayarlı görü uygulamaları için büyük önem arz etmektedir. Rekabetçi doğruluk oranları ile düşük parametre sayısına sahip, daha verimli KSA topolojilerinin üretilmesi için gerçekleştirilen çalışmalar bulunmaktadır [83].



Şekil 5.13: Farklı topolojilerin eğitilmesi için geçen toplam sürelerin karşılaştırılması

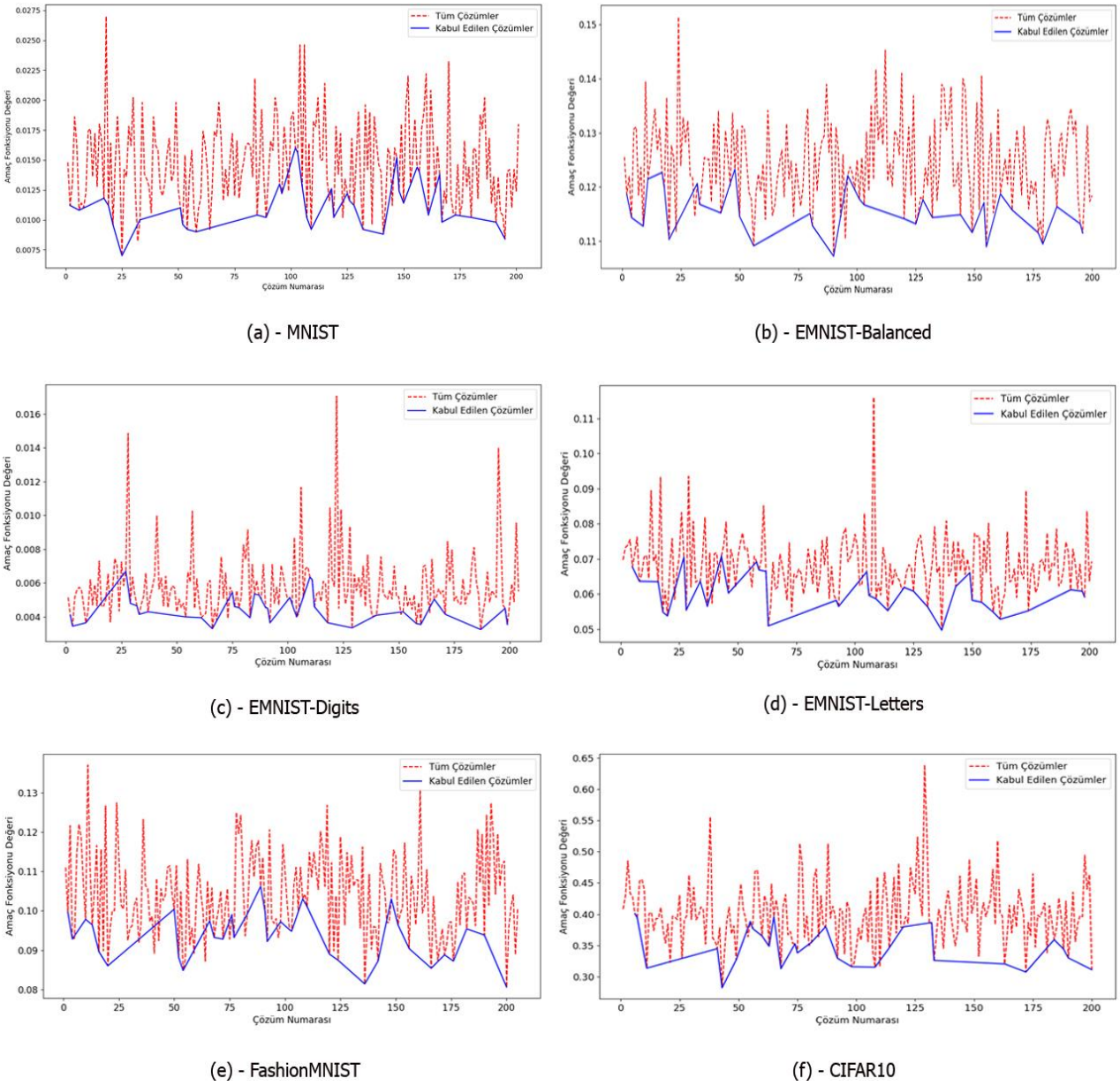
Şekil 5.13 ve tablo 5.13'e bakılarak, parametre sayısındaki artışın toplam eğitim sürelerini de etkilediği açık bir şekilde görülmektedir. Şekil 5.13'de görüldüğü üzere bu tez çalışmasında önerilen yöntem eğitim için geçen toplam sürede state-of-the-art algoritmalardan daha iyi sonuçlar elde etmiştir. Tablo 5.13 ve şekil 5.13'da gösterilen state-of-the-art (AlexNet, VGGNet-16, ResNet-50) mimariler, bu mimarilerin anlatıldığı makaleler [9, 28, 74] referans alınarak oluşturulmuştur. Bunun dışında Tablo 5.12'de gösterildiği gibi farklı çalışmalarda aynı veri seti için aynı state-of-the-art mimarisiyle farklı doğruluk oranları elde edilmiştir. Elde edilen bu farklılıkların, kullanılan ağırlık başlatıcı, eğitim dönemi ve rastsallıktan (seyreltme işleminde seçilecek düğümler gibi) kaynaklandığı düşünülmektedir. Oluşan bu farklılıklardan dolayı, yeniden oluştuduğum state-of-the-art mimariler dışında Tablo 5.12'de gösterilip, en iyi doğruluk oranlarını elde eden state-of-the-art mimariler µO-Kısıtlı yöntemi ile karşılaştırmalı olarak Şekil 5.14'de verilmiştir. Şekil 5.14'de CIFAR10 ve MNIST veri setleri için elde edilen, AlexNet, VGGNet ve ResNet doğruluk oranları *Ma (2018) [43]* referanslı çalışmadan, FashionMNIST için elde edilen AlexNet, VGGNet ve ResNet doğruluk oranları ise *Assunçao (2018)[44]* referanslı çalışmadan alınmıştır.

Bu çalışmada, state-of-the-art mimariler için direkt olarak Tablo 5.12'deki değerlerin kullanılmamasının ilk nedeni tablo 5.12'de yapılan çalışmalarda, bu state-of-the-art mimariler için oluşan toplam parametre sayısının belirtilmemesidir. İkinci olarak yine bu state-of-the-art mimariler için kullanılan yapının nasıl seçildiğinin net olarak belirtilmemesidir. Örneğin: VGGNet mimarisi için VGGNet-16 ve VGGNet-19 gibi farklı mimariler bulunmaktadır. Buradaki 16 ve 19 rakamları kullanılan mimarideki toplam katman sayısını belirtmektedir. Fakat bu çalışmalarda bu bilgi net olarak verilmemiş, sadece ilgili çalışmaya ait makalenin referansı verilmiştir. Aynı şekilde, ResNet mimarisi için de bu durum geçerlidir. ResNet mimarisinin 18, 34, 50, 101 ve 152 katmandan oluşan farklı yapıları bulunmaktadır. Yapılan bu çalışmalarda mimariye ait bu bilgi yine belirtilmemiştir. Bu nedenlerden dolayı yapılan bu tez çalışmasında ilgili state-of-the-art mimarilerin makaleleri referans alınarak oluşturulup, eğitilmesi ihtiyacı doğmuştur.



Şekil 5.14: µO'nun Tablo 5.12'de en iyi değeri elde eden state-of-the-art mimariler ile karşılaştırılması

µO algoritmasının çalışması boyunca oluşan adımlar, üretilen çözümler ve bu çözümlere ait amaç fonksiyonu değerleri şekil 5.15'te gösterilmiştir. Algoritmanın çalışması boyunca üretilen bütün çözümler ve bunlar arasından sadece kabul edilen çözümler aynı grafik üzerinde farklı veri setleri için gösterilmiştir.



Şekil 5.15: Farklı veri setleri için µO-Kısıtlı ile KSA hiper-parametre optimizasyonu adımları

5.5 Mikrokanonikal Optimizasyon için Hassasiyet Analizi

Bu tez çalışmasında, KSA hiper-parametre optimizasyonu sırasında üretilen bütün topolojilerin eğitilmesi için yığın boyutu 32, öğrenme oranı 1e-4 ve optimizasyon metodu Adam olacak şekilde seçilmiştir. Bu hiper-parametreler, yapılan KSA hiper-parametre optimizasyonu çalışmalarının birçoğunda sabit seçilmek yerine optimizasyon işlemine dahil edilir. Yapılan bu çalışmalar göz önüne alınarak, bu tez çalışmasında da yığın boyutu, öğrenme oranı ve optimizasyon metodu hiper-parametreleri optimizasyon işlemine dahil edildi (Tablo 5.3). Fakat µO-Kısıtlı'nın çalışması sırasında optimizasyon işlemine bu hiper-parametreleri dahil ederek, modellerin adil bir şekilde değerlendirilmesinin zorlaşacağını düşündüm. Çünkü aynı

hiper-parametre değerlerine sahip, birbirinin aynısı olan iki KSA topolojisi, öğrenme oranı veya yığın boyutu gibi öğrenme sürecini etkileyen hiper-parametrelerin farklı seçilmesiyle tamamen birbirinden farklı hata değerleri üretebiliyordu. Bu durum topoloji olarak daha kötü ama öğrenme oranı daha iyi seçilmiş bir modelin, topoloji olarak daha iyi olmasına rağmen öğrenme oranı daha kötü bir değer seçilmiş olan bir modele göre daha iyi sonuçlar vermesine neden olabiliyordu. Halbuki daha iyi topolojiye sahip olan bir model aynı öğrenme oranı ile daha iyi bir sonuç verebilirdi. Aynı durum yığın boyutu ve optimizasyon metodu hiper-parametreleri için de geçerli. Zaten bu hiper-parametreler birçok çalışmada öğrenme süreci başlığı altında gruplanıyor. Bu sebeplerden dolayı öğrenme oranı, yığın boyutu ve optimizasyon metodu hiper-parametreleri ayrı olarak ele alındı. Bu hiper-parametrelerin, elde edilen en iyi KSA topolojilerinin doğruluk değerlerini nasıl etkilediği ayrı olarak gözlemlendi. Hassasiyet analizi sadece μO-Kısıtlı yöntemi için gerçekleştirildi.

Bu hiper-parametrelerin doğruluk oranlarını nasıl etkilediğini gözlemlemeden önce bu çalışmada kullanılan veri setleri için optimizasyon gerçekleştirilmeden, bu hiper-parametreler için seçilen sabit değerlerle elde edilen doğruluk değerleri bilinmelidir. MNIST, EMNIST-Letters, EMNIST-Digits, EMNIST-Balanced, FashionMNIST ve CIFAR10 veri setleri için tablo 5.14'deki sabit değerler ile elde edilen doğruluk oranları tablo 5.15'te gösterilmiştir. Tablo 5.15'teki sonuçlar incelendiğinde, iki optimizasyon metodu için de hiper-parametre optimizasyonu gerçekleştirilmediğinde (Tablo 5.15'teki değerler kullanıldığında), Adam optimizasyon metodunun MNIST, MNIST-Letters, EMNIST-Digits ve CIFAR10 veri setleri için SGD optimizasyon metodundan daha iyi sonuçlar verdiği görülmektedir.

Tablo 5.14: Öğrenme sürecinde kullanılan ve optimize edilmek üzere sabit seçilen hiper-parametreler

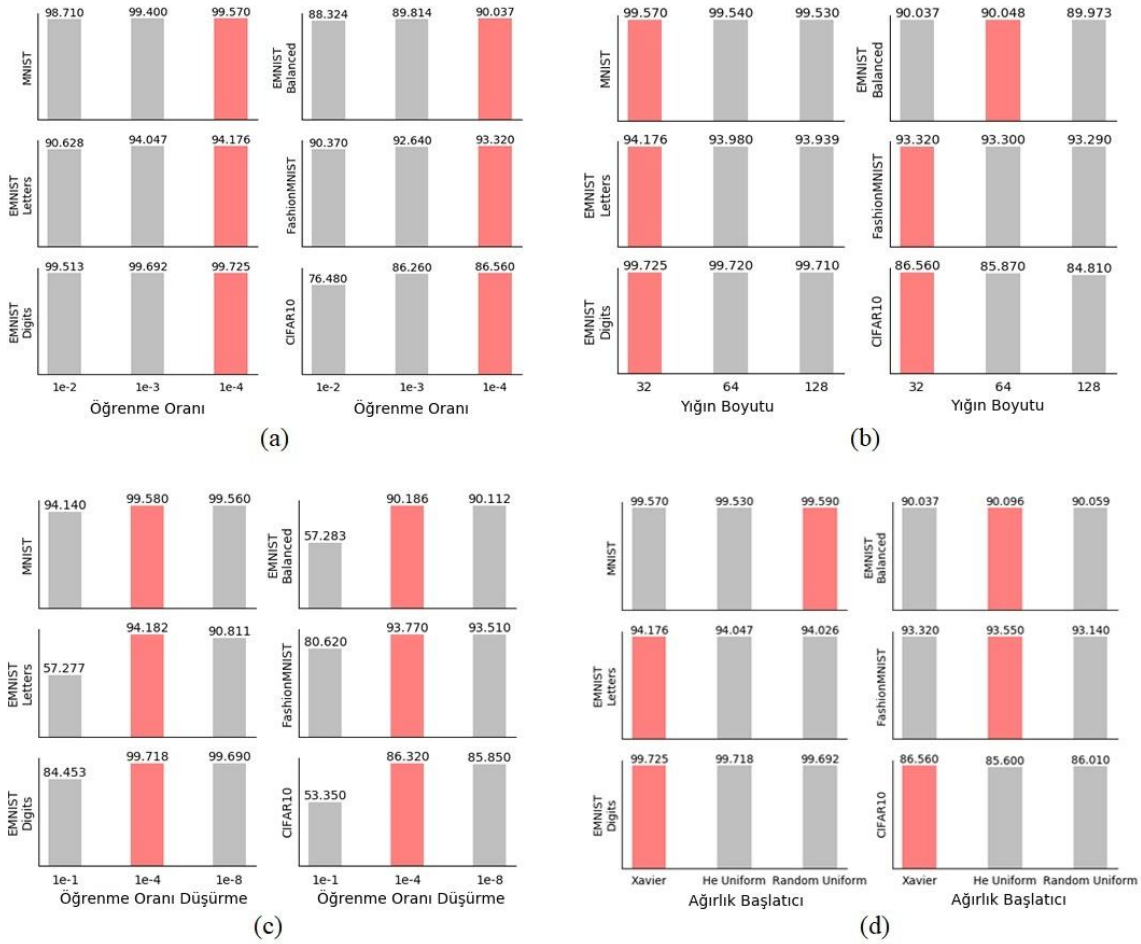
Optimizasyon Metodu	Öğrenme Oranı	Öğrenme Oranı Düşürme	Yığın Boyutu	Ağırlık Başlatıcı
Adam	1e-4	Yok	32	Xavier
SGD	1e-3	1e-6	32	Xavier

Tablo 5.15: Tablo 5.14'teki sabit değerler seçilerek elde edilen doğruluk oranları

Veri Seti	Doğruluk Oranı (Adam)	Doğruluk Oranı (SGD)
MNIST	99.57	99.57
EMNIST-Letters	94.176	94.142
EMNIST-Digits	99.725	99.720
EMNIST-Balanced	90.037	90.064
FashionMNIST	93.32	93.61
CIFAR10	86.56	85.81

Şekil 5.16'da yığın boyutu, öğrenme oranı, öğrenme oranı düşürme ve ağırlık başlatıcı hiper-parametrelerinin doğruluk oranına nasıl etki ettiği farklı veri setleri üzerinde gösterilmiştir. Şekil 5.16'da yapılan testlerde optimizasyon metodu olarak Adam kullanılmıştır. Şekil 5.16 (a)'da öğrenme oranının doğruluk oranına olan etkisi gösterilmiştir. Farklı öğrenme oranları arasında çok büyük farklar olmadığı görülmüştür. Adam optimizasyon metodu için öğrenme oranının 1e-4 seçilmesinin en iyi sonuçları verdiği, 1e-2 ve 1e-3 değerlerinin mevcut en iyi doğruluk değerini veren öğrenme oranının, 1e-4, (Tablo 5.15 - Adam) elde ettiği sonucu iyileştiremediği görülmüştür. Tablo 5.16 incelendiğinde hesaplama zamanı olarak çok büyük farklar olmasada 1e-2 öğrenme oranı değerinin daha az hesaplama zamanı elde ettiği görülmüştür.

Şekil 5.16 (b)'de yığın boyutu hiper-parametresinin doğruluk oranına olan etkisi gösterilmiştir. EMNIST-Balanced veri seti hariç diğer bütün veri setlerinde yığın boyutunun 32 olarak seçilmesinin en iyi doğruluk oranını verdiği, fakat hesaplama zamanı olarak diğerlerinin gerisinde kaldığı gözlemlenmiştir. Hesaplama zamanı olarak en iyi değeri 128 yığın boyutu elde etmiştir (Tablo 5.16). Elde edilen doğruluk değerleri birbirine çok yakın olduğundan hızlı hesaplama zamanları elde edebilmek için 32 değerine alternatif olarak yığın boyutu 128 seçilip, elde edilen kazanç ve kayıplar değerlendirilebilir.



Şekil 5.16: Farklı hiper-parametrelerin doğruluk oranına olan etkisi

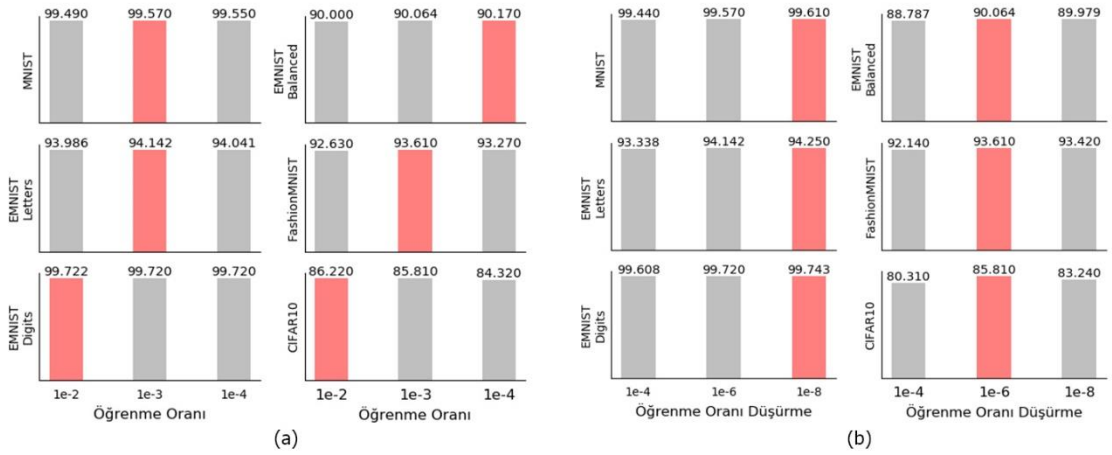
Tablo 5.16: Adam optimizasyon metodu için öğrenme sürecini etkileyen parametrelerin hesaplama zamanına olan etkisi

	MNIST	E-Letters	E-Digits	E-Balanced	FashionMNIST	CIFAR10
Değer	Yığın Boyutu Hiper-Parametresinin Hesaplama Zamanına (Saat) Etkisi					
32	1.02	1.89	6.93	2.68	**1.15**	1.87
64	0.65	1.31	9.04	4.11	1.79	3.66
128	**0.50**	**1.13**	**3.83**	**1.64**	1.39	**1.35**
Değer	Öğrenme Oranı Hiper-Parametresinin Hesaplama Zamanına (Saat) Etkisi					
1e-2	**0.95**	1.93	**6.36**	**2.66**	**1.14**	1.85
1e-3	0.98	**1.85**	6.48	2.71	1.16	**1.84**
1e-4	1.02	1.89	6.93	2.68	1.15	1.87
Değer	Ağırlık Başlatma Hiper-Parametresinin Hesaplama Zamanına (Saat) Etkisi					
Xavier	1.02	**1.89**	6.93	**2.68**	**1.15**	1.87
Random	0.96	1.97	6.91	**2.68**	**1.15**	**1.83**
He	**0.95**	1.97	**6.43**	2.73	2.82	1.84

Şekil 5.16 (c)'de öğrenme oranı düşürme hiper-parametresinin doğruluk oranına olan etkisi gösterilmektedir. Öğrenme oranı düşürme parametresinin 1e-1 gibi büyük değerler seçilmesi, yani öğrenme oranının hızlı bir şekilde düşürülmesi şekil 5.15 (c)'de görüldüğü gibi düşük doğruluk değerlerinin elde edilmesine neden olmuştur.

Aynı zamanda, bu hiper-parametre için 1e-4 değeri seçilerek, EMNIST-Digits ve CIFAR10 veri setleri hariç diğer veri setlerinde, tablo 5.15 (*Adam kolonu*)'deki doğruluk oranlarından daha iyi değerler elde edilmiştir.

Şekil 5.16 (d)'de ağırlık başlatma yönteminin doğruluk oranına olan etkisi gösterilmiştir. KSA'larda ağırlıklar rastsal bir şekilde başlatılır. Bu nedenle ağırlıkların doğru bir şekilde başlatılması doğruluk oranını ve ağın yakınsama hızını olumlu anlamda etkilemektedir. Elde edilen sonuçlara bakıldığında; Xavier ağırlık başlatma yönteminin 6 farklı veri setinin 3'ünde en iyi doğruluk oranlarını verdiği gözlemlenmiştir. Random uniform ağırlık başlatma yönteminin EMNIST-Balanced ve FashionMNIST veri setleri için Tablo 5.15 (*Adam kolonu*)'deki değerleri ileriye götürdüğü gözlemlenmiştir. Bu anlamda yapılan çalışmalar için ağırlık başlatıcı olarak Xavier ve Random uniform yöntemleri test edilip, sonuçlar değerlendirilebilir.



Şekil 5.17: SGD için farklı hiper-parametrelerin doğruluk oranına olan etkisi

Yapılan çalışmada, Adam optimizasyon metodu için öğrenme oranı, öğrenme oranı düşürme parametrelerinin doğruluk oranına olan etkisi gözlemlendi. Bunun dışında farklı optimizasyon metodunun doğruluk oranına etkisi var mı gözlemleyebilmek adına, yapılan diğer çalışmalarda sıklıkla kullanılan SGD optimizasyon metodu ayrı olarak test edildi. Şekil 5.17 (a)'da SGD optimizasyon metodu için farklı öğrenme oranı ve öğrenme oranı hiper-parametrelerinin sonuca olan etkisi gösterilmiştir. Gösterilen sonuçlar arasında anlamlı, çok büyük bir fark görülmemektedir. Şekil 5.17 incelenirse MNIST, EMNIST-Letters ve FashionMNIST veri setleri için 1e-3 (SGD için önerilen öğrenme oranı değeri), EMNIST-Digits ve CIFAR10 veri setleri içinse 1e-2 değerinin en iyi doğruluk oranı değerlerini verdiği gözlemlenmiştir. Hesaplama zamanı bakımından ise 1e-2 öğrenme oranı 3 farklı veri setinde en iyi hesaplama

zamanı değerlerini elde ederken, diğer 3 veri setinde 1e-3 doğruluk oranı değeri en iyi hesaplama zamanı değerlerini elde etmiştir (Tablo 5.17). Şekil 5.17 (a)'da öğrenme oranı için çoğunluk olarak 1e-3 değeri en iyi doğruluk oranlarını verse de, yapılan çalışmalarda, 1e-2 değeri de test edilip, sonuçlar gözlemlenebilir.

Tablo 5.17: SGD optimizasyon metodu için öğrenme oranı hiper-parametresinin hesaplama zamanına olan etkisi

	Öğrenme Oranı Hiper-Parametresinin Hesaplama Zamanına (Saat) Etkisi					
Değer	**MNIST**	**E-Letters**	**E-Digits**	**E-Balanced**	**FashionMNIST**	**CIFAR10**
1e-2	**0.90**	1.81	**5.85**	2.55	1.08	**1.71**
1e-3	0.91	**1.75**	5.86	**2.47**	**1.07**	1.76
1e-4	0.97	1.82	5.93	2.56	1.09	1.82

Şekil 5.17 (b)'de ise öğrenme oranı düşürme hiper-parametresinin doğruluk oranına olan etkisi gösterilmektedir. Farklı öğrenme oranı düşürme hiper-parametrelerinin elde ettiği doğruluk değerleri arasında anlamlı derecede bir fark görülmemektedir. Öğrenme oranı düşürme parametresinin 1e-4 gibi diğerlerine (1e-6, 1e-8) göre nispeten daha büyük değerler seçilmesi, yani öğrenme oranının hızlı bir şekilde düşürülmesi, şekil 5.17 (b)'de görüldüğü gibi diğerlerine göre daha düşük doğruluk değerlerinin elde edilmesine neden olmuştur. Aynı zamanda, bu hiper-parametre için 1e-8 değeri seçilerek, MNIST, EMNIST-Letters ve EMNIST-Digits veri setlerinde tablo 5.15 (*SGD kolonu*)'deki doğruluk oranlarından daha iyi değerler elde edilmiştir. Bu hiper-parametre değeri için en iyi doğruluk oranını veren 1e-8 değerine alternatif olarak 1e-6 değeri de test edilip, sonuçlar gözlemlenebilir.

Şekil 5.17'de elde edilen sonuçlar ve state-of-the-art [9, 26, 28, 74] mimarilerde kullanılan yaklaşımdan yola çıkarak son olarak farklı bir yapılandırma oluşturuldu. Oluşturulan bu yapılandırma ile 4 farklı veri seti için, bu çalışmada Tablo 5.12'de belirtilen en iyi doğruluk oranları elde edilmiştir. AlexNet, VGGNet ve ResNet gibi çok fazla sayıda atıf alan ve ImageNet yarışmasında yüksek doğruluk oranları elde eden mimarilerin yayınlanan makaleleri okunduğunda, öğrenme oranının iterasyon sayısı arttıkça, belirli bir oranda düşürüldüğü görülmektedir. Bu yaklaşım, KSA çalışmalarında hiper-parametre seçiminde sıklıkla belirtilmektedir. KSA'lar eğitilirken ilk olarak yüksek öğrenme oranları (0.1, 0.01 gibi) seçilip eğitim adımlarının hızlandırılması, daha sonra sonlara yaklaştıkça öğrenme oranının belirli adımlarda, seçilen sabit bir oranda düşürülmesi önerilmektedir. Bu sayede ilk adımlarda yüksek öğrenme oranları seçilerek hesaplama zamanından kazanç sağlanır.

Sonlara doğru yaklaştıkça daha küçük öğrenme oranları seçilerek daha küçük adımlar atılır. Böylece hedefe (yüksek doğruluk oranı) yaklaştıkça oluşacak sapmalar azaltılmış olur. Bu bilgiler göz önüne alınarak, SGD optimizasyon yönteminde öğrenme oranı hiper-parametresi 1e-2 ve öğrenme oranı düşürme hiper-parametresi 1e-4 olacak şekilde seçilip, 4 farklı veri seti için şu ana kadar elde edilen ve bu çalışmada en iyi değerler olarak bildirilen doğruluk oranları elde edildi. Elde edilen bu değerler öğrenme sürecindeki hiper-parametrelerin (yığın boyutu, öğrenme oranı, öğrenme oranı düşürücü, ağırlık başlatıcı), doğru bir şekilde seçilmesinin doğruluk oranına olan etkisini göstermek amacıyla tablo 5.18'de gösterilmiştir. Elde edilen en iyi değerler için seçilen hiper-parametreler bir sonraki bölümde anlatılmıştır.

Tablo 5.18: Öğrenme süreci hiper-parametreleri için optimizasyon öncesi ve sonrasında elde edilen en iyi doğruluk oranları

Veri Seti	Optimizasyon Öncesi Doğruluk Oranı	Optimizasyon Sonrası Doğruluk Oranı	Fark
MNIST	99.57	99.64	+ 0.07
EMNIST-Letters	94.176	94.25	+ 0.074
EMNIST-Digits	99.725	99.743	+ 0.018
EMNIST-Balanced	90.037	90.34	+ 0.303
FashionMNIST	93.32	93.78	+ 0.46
CIFAR10	86.56	86.56	0

5.6 Mikrokanonikal Optimizasyon ve TPE Yöntemlerinin Karşılaştırılması

Gerçekleştirilen tez çalışmasında, performans karşılaştırması için μO'nun farklı versiyonları dışında TPE yöntemi de denenmiştir. Elde edilen performans karşılaştırmaları tablo 5.19'da gösterilmiştir. Fark sütununda gösterilen + ve – işaretleri μO-Kısıtlı yönteminin karşılaştırılan yöntemden ne kadar daha az ya da daha fazla iyi değer verdiğini göstermektedir. Örneğin ilk satırda μO-Kısıtlı, TPE yönteminden 0.05 daha az doğruluk oranı değeri vermiştir. Aynı zamanda μO-Kısıtlı yöntemi taraıfından üretilen model, TPE'den 16.99 kat daha az parametre sayısına sahiptir. Burada doğruluk oranları için + değeri (yani diğer yöntemlere göre daha iyi doğruluk oranına sahip olması), parametre sayısı için – değeri (yani diğer yöntemlere göre daha az parametre sayısına sahip olması) istenen durumlardır. Tablo 5.19'a bakıldığında μO yöntemine başlangıçta uygulanan hiper-parametre kısıtları kaldırıldığında doğruluk oranı bakımından daha başarılı sonuçların elde edildiği görülmüştür. μO-Kısıtsız ile doğruluk oranı bakımından daha başarılı sonuçlar elde edilmesine rağmen parametre sayısı bakımından en az 1.07, en çok ise 22.38 kat daha fazla parametre sayısına sahip modeller üretilmiştir. Bu sonuca bakılarak şu çıkarımda

bulunulabilir; üretilen topolojilerde parametre sayısının belirli bir noktaya kadar arttırılması, elde edilen sonuçları doğruluk oranı bakımından daha iyileştirebilir.

Fakat her zaman daha fazla parametre sayısına sahip modellerin, daha iyi doğruluk oranları vereceği söylenemez. Tablo 5.19'a bakıldığında TPE, EMNIST-Balanced veri seti için en yüksek parametre sayısına sahip modeli üretmesine rağmen doğruluk oranı bakımından en iyi sonuçları üretememiştir. Buna rağmen TPE yöntemi 6 farklı veri setinin 2'sinde en iyi doğruluk oranlarını elde etmiştir. Aynı zamanda geriye kalan 4 farklı veri setinin 3'ünde en iyi yöntemin ardından en iyi doğruluk oranlarını veren ikinci yöntem olmuştur.

Yapılan çalışmada, referans yöntem olarak μO-Kısıtlı yöntemi seçilmiştir. Yapılan karşılaştırmalar ve hassasiyet analizi için seçilen bu yöntem kullanılmıştır. μO-Kısıtlı yöntemi kullanılarak daha az parametre ile diğer yöntemler ile rekabetçi sonuçlar elde edildiği için bu yöntem referans olarak seçilmiştir. Daha iyi doğruluk oranları için tablo 5.19'da görüldüğü gibi μO-Kısıtlı ve TPE yöntemleri kullanılabilir. Bu yöntemler kullanılırken parametre sayısı ve doğruluk oranı gibi kriterler göz önüne alınmalıdır. Örneğin: Daha az hesaplama zamanı harcanarak en iyi modeller ile rekabetçi sonuçların elde edilebildiği bir durum kabul ediliyorsa, μO-Kısıtlı yöntemi kullanılabilir. Fakat hesaplama zamanı bakımından herhangi bir kısıtlama yoksa ama doğruluk oranı en önemli kriter ise μO-Kısıtsız ve TPE yöntemleri kullanılabilir. Bunu belirlerken ise parametre sayısı ve doğruluk oranı arasında iyi bir denge kurulmalıdır. Örneğin: Tablo 5.19'a bakıldığında TPE yöntemi ile MNIST veri seti için referans yöntemden (μO-Kısıtlı) 0.05 daha iyi doğruluk oranı elde edilmesine rağmen 16.99 kat daha fazla parametre sayısı elde edilmiştir. Bu durumda elde edilen küçük bir doğruluk oranı artışı, hesaplama zamanından büyük kayıp gerektirir. Bu durum nispeten daha az tercih edilmelidir. Fakat farklı bir veri setinde, TPE yöntemi ile CIFAR10 veri seti için referans yöntemden 2.03 daha iyi doğruluk oranı elde edilmesine rağmen 1.89 kat daha fazla parametre sayısı elde edilmiştir. Bu durumda elde edilen doğruluk oranı artışı kabul edilebilir derecede daha iyi iken parametre sayısı artışı diğer veri setine göre çok daha azdır. Bu durumda TPE yöntemi seçilerek, doğruluk oranı ve parametre sayısı arasındaki denge daha iyi kurulabilir.

Tablo 5.19: µO ve TPE yöntemlerinin parametre sayısı ve doğruluk oranı bakımından karşılaştırılması

Veri Seti	Yöntem	Parametre Sayısı	Doğruluk Oranı	Fark (µO Kısıtsız, TPE)
MNIST	µO Kısıtlı	306,730	99.57	**Doğruluk Oranı:** + 0.02, - 0.05 **Parametre Sayısı:** -2.9x, -16.99x
	µO Kısıtsız	895,402	99.55	
	TPE	**5,210,090**	**99.62**	
EMNIST-Letters	µO Kısıtlı	1,005,381	94.176	**Doğruluk Oranı:** - 0.142, - 0.006 **Parametre Sayısı:** -4.23x, -1.90x
	µO Kısıtsız	**4,260,123**	**94.318**	
	TPE	1,915,227	94.182	
EMNIST-Digits	µO Kısıtlı	798,026	99.725	**Doğruluk Oranı:** - 0.032, - 0.06 **Parametre Sayısı:** -1.07x, -2.79x
	µO Kısıtsız	859,338	99.757	
	TPE	**2,228,906**	**99.785**	
EMNIST-Balanced	µO Kısıtlı	879,055	90.037	**Doğruluk Oranı:** - 0.107, +0.026 **Parametre Sayısı:** +1.06x, -4.22x
	µO Kısıtsız	**828,687**	**90.144**	
	TPE	3,714,095	90.011	
FashionMNIST	µO Kısıtlı	361,834	93,32	**Doğruluk Oranı:** -0.72, -0.48 **Parametre Sayısı:** -22.38x, -14.06x
	µO Kısıtsız	**8,096,714**	**94.04**	
	TPE	5,089,610	93.80	
CIFAR10	µO Kısıtlı	2,845,962	86.56	**Doğruluk Oranı:** -2.03, - 1.39 **Parametre Sayısı:** -1.89x, -1.45x
	µO Kısıtsız	**5,390,506**	**88.59**	
	TPE	4,127,754	87.95	

5.7 Elde Edilen En İyi Topolojilerin İncelenmesi ve Yorumlanması

Bu bölümde, µO-Kısıtlı optimizasyon yöntemi ile KSA hiper-parametre optimizasyonu sonucunda farklı veri setleri için en iyi doğruluk oranlarını veren KSA topolojileri incelenmiştir. Farklı veri setleri için en iyi doğruluk oranlarını veren topolojilerin yapısı ve aldıkları hiper-parametre değerleri, elde ettikleri doğruluk oranları ve topolojinin toplam parametre sayısı şekil 5.17'de gösterilmiştir. Aynı

zamanda şekil 5.18'de gösterilen topolojilerin daha rahat anlaşılması için kullanılan notasyon, şekil 5.19'da gösterilmiştir.

Şekil 5.18'de gösterilen bütün KSA topolojileri için konvolüsyon işlemi aralık (stride) değeri 1 ve dış boşluk (padding) yöntemi "SAME" olarak seçilmiştir. Aynı şekilde, bütün topolojilerde boyut düşürme işlemleri (ortaklama ve strive) için aralık değeri 2 olarak seçilmiştir.

Şekil 5.18'de gösterilen bütün topolojilerin eğitilmesi sırasında yığın boyutu hiper-parametresi 32 ve ağırlık başlatıcı olarak Xavier metodu seçilmiştir. MNIST, EMNIST-Digits, EMNIST-Balanced ve FashionMNIST veri setleri için en iyi sonucu veren topolojilerin eğitilmesinde; optimizasyon metodu SGD, öğrenme oranı 1e-2, öğrenme oranını düşürme hiper-parametresi 1e-4 olacak şekilde seçilmiştir. EMNIST-Letters veri seti için en iyi sonucu veren topolojinin eğitilmesinde; optimizasyon metodu SGD, öğrenme oranı 1e-3 ve öğrenme oranını düşürme hiper-parametresi 1e-8 olarak seçilmiştir. Son olarak CIFAR10 veri seti için optimizasyon metodu olarak Adam, öğrenme oranı olarak 1e-4 değeri seçilmiştir (öğrenme oranı düşürme hiper-parametresi kullanılmamıştır).

Şekil 5.18: Farklı veri setleri için çalışma boyunca elde edilen en iyi doğruluk oranlarına sahip KSA topolojilerinin yapısı

Şekil 5.19: En iyi KSA topolojilerinin anlatılması için kullanılan notasyon

Elde edilen topolojilere bakıldığında oluşturulan bütün topolojilerin (EMNIST-Letters hariç) iki konvolüsyon katmanından oluştuğu görülmektedir. Şekil 5.17'de MNIST veri seti için elde edilen topolojiye bakıldığında *Conv 1_1, Conv 1_2 ve Pool 1_1* katmanlarının bir konvolüsyon bloğunu oluşturduğu unutulmamalıdır. Elde edilen topolojilerin hepsinde boyut düşürme işlemi olarak strive yerine ortaklama işlemi kullanılmıştır. Boyut düşürme işlemi olarak strive işlemi, oluşan topolojilerin hiçbirinde tercih edilmemiştir. Strive işlemini kullanan modeller, diğer modellerden parametre sayısı bakımından daha fazla parametreye sahip olmasına rağmen doğruluk oranı bakımından bu topolojiler ile rekabet edememiştir. Bu çalışmanın ilginç noktalarından biri ise ortaklama katmanlarında kullanılan ortaklama tipi (maksimum ortaklama ve ortalama ortaklama) hiper-parametresinin nasıl seçildiğidir. Çalışma boyunca bu hiper-parametrenin seçimi için herhangi bir zorlama, kısıt olmamasına rağmen, EMNIST-Digits için elde edilen topoloji hariç diğer bütün topolojilerde tam bağlantılı bloklar ya da çıktı katmanına bağlanana kadar ortaklama tipi olarak maksimum ortaklama hiper-parametresi seçilirken, son bloğa bağlanırken ortalama ortaklama hiper-parametresi seçilmiştir. Bu durum state-of-the-art mimarilerde sıklıkla kullanılan ve istenilen bir durumdur. Çünkü tam bağlantılı katmana bağlanana kadar yapılan işlem, verilen girdi, görüntü üzerinde özellik çıkarımı işlemidir. Bundan sonra elde edilen özelliklere göre sınıflandırma işlemi gerçekleştirilir. Bu yüzden tam bağlantılı katmana gelene kadar maksimum ortaklama işlemi seçilerek görüntülerdeki kenar ve köşelerin elde edilmesi, bu özelliklerin çıkarılması amaçlanır. Tam bağlantılı katmana bağlanırken ise elde edilen bu özelliklerin özet bir bilgisinin verilmesi istenir. Ortalama ortaklama işlemi görüntü üzerindeki bilgilerin ortalamasını verdiğinden

görüntüdeki her bir piksel sonucu etkilemiş olacaktır. Ama maksimum ortaklama işleminde durum bu şekilde değildir. Aynı zamanda, elde edilen topolojilere bakıldığında KSA topolojilerinde tam bağlantılı katmanlar olmadan da rekabetçi doğruluk oranlarının elde edilebileceğini göstermektedir. KSA topolojilerinde tam bağlantılı katmanların kullanılmaması aynı zamanda oluşan topolojideki toplam parametre sayısını önemli derecede azaltacak ve bu durum da hesaplama zamanında pozitif anlamda (hesaplama zamanını düşürücü) bir etki elde edilmesine neden olacaktır.

Yapılan tez çalışması sonucunda elde edilen sonuçlara bakıldığında her problem için en iyi doğruluk oranlarını verecek hiper-parametreler ve topolojilerden bahsetmenin mümkün olmadığı görülmüştür. Bunun yanında çalışma boyunca elde edilen sonuçlar, literatürde KSA hiper-parametre optimizasyonu için yapılan çalışmalar ve state-of-the-art mimariler incelendiğinde farklı hiper-parametrelerin nasıl seçileceğinin belirlenmesi için şu çıkarımlara ulaşılabilir:

- **Özellik Haritası Sayısı (Feature Map Count):** Bu hiper-parametre, KSA topolojilerinin elde ettiği doğruluk oranı değerlerini doğrudan etkileyen bir hiper-parametredir. Özellik haritaları görüntüden, girdiden elde edilen, öğrenilen özelliklerin sayısını temsil etmektedir. Her bir özellik haritası ile görüntünün farklı bir özelliğinin temsil edilmesi amaçlanır. Bu hiper-parametre değerinin yüksek değerler olarak seçilmesi doğruluk oranında genel olarak pozitif bir artış sağlayacaktır. Fakat özellik haritası sayısı doğruluk oranı ile ilişkili olduğu kadar hesaplama zamanı, maliyeti ile de doğrudan ilişkilidir. Bu hiper-parametrenin aldığı yüksek değerler girdi boyutlarına bağlı olarak yüksek hesaplama zamanlarına neden olabilir. Bu nedenle girdi boyutları (genişlik, yükseklik) yüksek iken daha az sayıda özellik haritaları seçilmelidir. Girdi boyutu azalmaya başladıkça, girdi boyutunun azalması ile ters orantılı bir şekilde özellik haritası sayıları arttırılabilir. Aynı zamanda bu hiper-parametrenin seçimi kullanılan veri setlerine de bağlı olarak gerçekleştirilebilir. Kolay öğrenilebilen veri setleri için 512, 1024, 2048 gibi yüksek özellik haritası değerlerinin seçilmesi çok mantıklı olmayacaktır. Özellik haritası hiper-parametresi seçilirken doğruluk oranı ve hesaplama maliyeti dengesine dikkat edilmelidir.

- **Filtre Boyutu (Filter Size):** KSA'ların genel yapısına bakıldığında, ağın derinliği artmaya başladıkça girdi boyutunun azaldığı ve özellik haritası sayılarının arttığı gözlemlenmektedir. Bu durum, başlangıçta yüksek boyutta filtreler seçilerek gerçekleştirilebilir. Başlangıçta girdi boyutları yüksek olduğundan hesaplama maliyetini düşürebilmek için yüksek filtre boyutları seçilebilir. Hesaplama zamanından kazanmak için filtre değerlerinin sürekli çok yüksek değerler seçilmesi bilgi kaybına neden olabilir. Yüksek filtre boyutları ile görüntüden elde edilecek bilgilerin kaçırılmasına neden olunabilir. Bu yüzden KSA'larının başlangıcında yüksek filtre boyutları seçilirken, ağ derinleşmeye başladıkça filtre boyutları küçültülebilir. Bu sayede hem hesaplama zamanından hem de görüntüden elde edilen bilgilerde kazanç sağlanabilir.
- **Öğrenme Oranı (Learning Rate):** Bu hiper-parametreyi, kabaca KSA'nın öğrenme süreci boyunca attığı adımların büyüklüğünü belirleyen bir hiper-parametre olarak düşünebiliriz. İstenen hedefe yaklaşmak için başlangıçta büyük adımlar atılırken sonlara doğru atılan öğrenme adımlarının boyutu küçültülmelidir. Bu durum öğrenme süreci boyunca eğitilen KSA topolojisi için, eğitim süresini azaltırken, sınıflandırma performansının arttırılmasını sağlayacaktır. Anlatılan bu strateji günümüzde adaptif (uyarlamalı) optimizasyon yöntemleri (Adam, Adamax vb.) ile öğrenme süreçlerine uygulanmaktadır.
- **Seyreltme Oranı (Dropout Rate):** Seyreltme oranı, katmanlarda kullanılan düğüm sayısı hiper-parametresi ile doğru bir orantı kuracak şekilde seçilmelidir. Tam bağlantılı katmanlarda, düğüm sayısının daha az olduğu durumlarda daha düşük seyreltme oranı seçilirken, düğüm sayısının daha fazla olduğu durumlarda daha yüksek seyreltme oranlarının seçilmesi önerilmektedir. Seçilen bu değer KSA'nın yetersiz öğrenmesine (underfitting) veya aşırı öğrenmesine (overfitting) neden olabilir. Örneğin: düğüm sayısı 32 olan bir katman için 0.8, 0.9 seyreltme oranı değerlerinin seçilmesi yetersiz öğrenmeye, düğüm sayısı 1024 olan bir katman için de 0.2, 0.3 gibi seyreltme oranlarının seçilmesi aşırı öğrenmeye neden olabilir.
- **Yığın Boyutu:** Bu hiper-parametre KSA topolojilerinin elde ettiği doğruluk oranı ve eğitim için harcanan hesaplama sürelerini doğrudan etkilemektedir.

Çok yüksek değerlerin seçilmesi elde edilen doğruluk oranının düşmesine neden olurken, çok düşük değerilerin seçilmesi eğitim sürelerini uzatır. Bu yüzden veri setinin büyüklüğünü ve elde edilecek doğruluk oranları göz önüne alınarak bir seçim yapılmalıdır. Yapılan çalışmalarda bu hiper-parametre için sıklıkla; 32, 64 ve 128 değerleri seçilmektedir.

6. DEĞERLENDİRME VE ÖNERİLER

Günümüzde *konvolüsyonel sinir ağları* kullanılarak yapılan bilgisayarlı görü çalışmalarında çok başarılı sonuçlar elde edilse de kullanılan yöntem hala birçok zorluk içermektedir. Bu zorluklardan biri gittikçe derinleşen sinir ağlarının gerektirdiği yüksek hesaplama maliyetidir. Aynı zamanda konvolüsyonel sinir ağları kurulurken ağ modelindeki katman sayısı, filtre boyutu, özellik haritası sayısı, öğrenme oranı ve daha birçok hiper-parametrenin en başarılı sonucu verecek şekilde seçilmesi gerekmektedir. Bu seçim işlemi, yani optimizasyon işlemi hesaplama maliyeti yüksek bir işlem olduğundan ızgara arama yöntemi gibi oluşabilecek bütün hiper-parametre kombinasyolarının eğitilip, test edildiği yöntemler uygulanabilir değildir. Böyle yöntemler yerine, bu optimizasyon problemi için, kabul edilebilir doğruluk oranlarının makul sürelerde elde edildiği üst-sezgisel yöntemler gereklidir ve yapılan hiper-parametre optimizasyonu çalışmalarında bu yöntemler sıklıkla kullanılmıştır.

KSA hiper-parametre optimizasyonu çalışmaları için GA ve PSO üst-sezgisel yöntemleri sıklıkla kullanılmış ve özellikle GA ile hiper-parametre optimizasyonu gerçekleştirilen çalışmalarda başarılı sonuçlar elde edilmiştir (Tablo A). Üst-sezgisel yöntemler kullanılarak yapılan çalışmalarda göze çarpan en büyük eksiklik, bu algoritmaların parametre seçimi için sistematik bir optimizasyon çalışması gerçekleştirilmemiş olmasıdır. Örneğin: *GA* için başlangıçta belirlenen popülasyon büyüklüğü, çaprazlama ve mutasyon olasılığı gibi parametreler kullanılmış, farklı parametrelerle algoritmanın nasıl sonuçlar verdiği test edilmemiştir. Aynı zamanda, KSA hiper-parametre optimizasyonu çalışmalarında optimize edilmeye çalışılan parametrelerde bir bütünlük görülmemektedir. Bu çalışmalarda genel olarak parametreler kısmi olarak ele alınmış; bir çalışmadan diğerine geçildiğinde çok farklı parametre alt kümeleri optimize edilmeye çalışılmıştır. Bazı çalışmalarda sabit mimari kullanılırken bazı çalışmalarda dinamik olarak genişleyen bir mimari kullanılmıştır. Hatta bazı çalışmalarda optimize edildiği söylenen parametreler için nispeten dar değer aralıkları seçilmiştir. Birçok çalışmada öğrenme oranı, yığın büyüklüğü ve optimizasyon metodu gibi önemli hiper-parametreler optimizasyon işlemine dahil edilmemiştir. Aynı zamanda elde edilen en iyi topolojiler için toplam parametre sayısı ve bu topolojiyi oluşturmak için gereken tüm hiper-parametre bilgileri eksiksiz olarak verilmemiştir. Çoğu çalışmada yapılan kabuller ile ilgili doyurucu sebepler

belirtilmemiştir. Bu sebeplerden dolayı literatürde gerçekleştirilen çalışmalar ile bu tez çalışmasında önerilen yöntemlerin kapsamlı bir şekilde karşılaştırılması zorlaşmaktadır.

KSA hiper-parametre optimizasyonu için optimizasyon yöntemi seçilirken dikkat edilmesi gereken önemli noktalardan biri optimize edilecek hiper-parametre kümesinin büyüklüğüdür. Daha dar, küçük hiper-parametre kümeleri için ızgara arama yöntemi seçilerek etkili sonuçlar elde edilebilirken, daha geniş, büyük hiper-parametre kümelerinde daha başarılı sonuçların elde edilebilmesi için *rastgele arama, üst-sezgisel algoritmalar* gibi yöntemler seçilebilir. GA, PSO, DE gibi üst-sezgisel yöntemler ile daha iyi doğruluk oranları elde edilirken, bu yöntemler kullanıldığında daha fazla hesaplama süresi ihtiyacı oluşmaktadır. Literatürdeki çalışmalar incelendiğinde daha az eğitim, hesaplama süreleri için TPE, SMAC gibi yöntemlerin önerildiği görülmüştür. Üst-sezgisel algoritmaların seçiminde, parametre ayarlaması zorluğundan dolayı daha az parametreye sahip üst-sezgisel yöntemler, daha çok parametreye sahip olan üst-sezgisel yöntemlere tercih edilebilir. Bunun yanında, implementasyon kolaylığı ve daha hızlı yakınsama süreleri, yani daha az sayıda iterasyonda iyi sonuçlar üretebilmesi de üst-sezgisel seçiminde önemli olabilmektedir. Örneğin: *PSO, GA'* ya kıyasla daha az sayıda parametreye sahiptir, hızlı çalışmakta ve *GA'* ya kıyasla rekabetçi sonuçlar vermektedir. Yüksek doğruluk oranı/ düşük hata oranı veren yöntemler arasından ise en hızlı çalışan yöntem çoğu zaman tercih edilen yöntem olabilir.

Literatürde sıklıkla kullanılan üst-sezgisel yöntemlerin aksine bu çalışmada KSA hiper-parametre optimizasyonu için Mikrokanonikal Optimizasyon yöntemi kullanıldı. Benzetimli tavlama yönteminin farklı bir varyasyonu olan bu yöntemin, benzetimli tavlama yöntemine göre kendi içerisinde daha az parametreye sahip olmasının daha avantajlı olabileceğini düşündüm. Aynı zamanda sahip olduğum hesaplama gücü, donanım desteği sınırlı olduğundan popülasyon tabanlı yöntemleri tercih etmedim. Yapılan bu tez çalışması sonucunda elde edilen sonuçlara bakıldığında doğruluk oranı bakımından literatürdeki çalışmalarla (CIFAR10 veri seti hariç) rekabetçi sonuçlar elde edildi. Aynı zamanda düşük parametre sayılarından oluşan topolojiler ile hesaplama süresi bakımından başarılı sonuçlar elde edildi. Literatürde state-of-the art olarak kabul edilen mimarilerin birçoğundan hem doğruluk oranı hem de hesaplama süresi bakımından daha iyi sonuçlar elde edildi. Donanım desteği ve

hesaplama gücünün yetersiz olması nedeniyle kısıtlanmak zorunda kalan topolojiler ve μO-Kısıtlı yönteminin, daha yüksek hesaplama gücü elde edildiğinde, daha fazla ve daha geniş topolojiye sahip çözümler üretileceğinden daha da iyi sonuçlar vereceğini düşünmekteyim.

Hiper-parametre optimizasyonu çalışmalarında üretilen topolojiler çok iyi doğruluk değerlerine sahip olsalarda çok uzun eğitim sürelerine ihtiyaç duyabilmektedirler. Bu durum yüksek hesaplama zamanı yanında mobil cihazlar için test sırasında yüksek enerji tüketimine de neden olmaktadır. Bu nedenle, yapılan KSA hiper-parametre optimizasyonu çalışmalarının aksine, daha az parametre sayısına sahip yani çok daha az eğitim süresine, hesaplama zamanına ihtiyaç duyan, eğitim adımlarını hızlandıran ve kabul edilebilir seviyede doğruluk oranlarının elde edilmeye çalışıldığı çok amaçlı KSA hiper-parametre optimizasyonu çalışmalarının ileride daha çok çalışılacağını düşünmekteyim.

KAYNAKLAR

1. McCulloch, W. S., Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.
2. ÜLKER, E. Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, *6*(3), 85-104.
3. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.
4. Minsky M., Papert S. (1969), Perceptrons. MIT Press, Cambridge, MA
5. Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533.
6. Cortes, C., V. Vapnik. Support-vector networks. *Machine learning*, *20*(3), (1995) 273-297.
7. DNI Instute. [Çevrimiçi] [Alıntı Tarihi: 28 05 2019] http://dni-institute.in/blogs/building-predictive-model-using-svm-and-r/
8. LeCun, Y., Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.
9. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
10. E. Alpaydin (2009). Introduction to machine learning. MIT press.
11. E. Öztemel (2003), Yapay Sinir Ağları. Istanbul: Papatya Yayincilik.
12. I. Goodfellow, Y. Bengio, A. Courville and Y. Bengio (2016), Deep learning (Vol. 1). Cambridge: MIT press
13. 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).
14. 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.
15. 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).

16. 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. DOI: 10.29109/http-gujsc-gazi-edu-tr.304938.
17. T. Yichuan, Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, (2013).
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), (2014) 1929-1958.
19. L. Bottou, Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010, Physica-Verlag HD, (2010) 177-186.
20. D. P. Kingma, J. L. Ba, Adam: Amethod for stochastic optimization. Proc. 3rd Int. Conf. Learn. Representations, (2014).
21. M. D. Zeiler, ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, (2012).
22. 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).
23. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86:11 (1998) 2278-2324
24. Dumoulin, V., Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.
25. Zeiler, M. D., Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.
26. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A. (2015, June). Going deeper with convolutions. Cvpr.
27. Andrew Ng. [Çevrimiçi] [Alıntı Tarihi: 13 04 2019] https://www.coursera.org/learn/convolutional-neural-networks/
28. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

29. LeCun, Y. [Çevrimiçi] [Alıntı Tarihi: 30 04 2019] http://yann.lecun.com/exdb/mnist/
30. Cohen, G., Afshar, S., Tapson, J., van Schaik, A. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373* (2017).
31. Xiao, H., Rasul, K., Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017)
32. Krizhevsky, A. [Çevrimiçi] [Alıntı Tarihi: 05 05 2019] https://www.cs.toronto.edu/~kriz/cifar.html
33. J. H. Holland (1992), Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence. MIT press.
34. J. H. Holland, Genetic algorithms. Scientific american, 267 (1992) 66-73.
35. D. E. Goldberg, J. H. Holland, Genetic algorithms and machine learning. Machine learning, 3:2 (1988) 95-99.
36. E. P. Ijjina, M. C. Krishna, Human action recognition using genetic algorithms and convolutional neural networks. Pattern recognition, 59 (2016) 199-212.
37. 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).
38. Y. Sun, X. Bing, M. Zhang, Evolving deep convolutional neural networks for image classification. arXiv preprint arXiv: 1710.10741, (2017).
39. 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.
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. 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.
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. arXiv preprint arXiv:1801.01563, (2018).
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. 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).
49. 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).
50. Y. Sun, B. Xue, M. Zhang, A Particle Swarm Optimization-based Flexible Convolutional Auto-Encoder for Image Classification. arXiv preprint arXiv:1712.05042, (2017).
51. 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.
52. 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).

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. J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13.Feb (2012), (281-305).
60. T. Domhan, J. T. Springenberg, F. Hutter, Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. IJCAI, 15 (2015).
61. 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.
62. P. Neary, Automatic Hyperparameter Tuning in Deep Convolutional Neural Networks Using Asynchronous Reinforcement Learning. 2018 IEEE International Conference on Cognitive Computing (ICCC), IEEE, (2018).
63. B. van Stein, H. Wang, T. Bäck, Automatic Configuration of Deep Neural Networks with EGO. arXiv preprint arXiv:1810.05526, (2018).
64. 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)

65. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing. *science*, *220*(4598), (1983) 671-680.
66. M. Creutz, Microcanonical monte carlo simulation. *Physical Review Letters*, *50*(19), (1983) 1411.
67. S. T. Barnard, *Stereo matching by hierarchical, microcanonical annealing* (No. TN-414). SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER. (1987).
68. G. Bhanot, M. Creutz, H. Neuberger, Microcanonical simulation of Ising systems. *Nuclear Physics B*, *235*(3), (1984) 417-434.
69. J. A. Torreão, E. Roe, Microcanonical optimization applied to visual processing. *Physics Letters A*, *205*(5-6), (1995) 377-382.
70. A. Linhares, J. R. Torreão, Microcanonical optimization applied to the traveling salesman problem. *International Journal of Modern Physics C*, *9*(01), (1998) 133-146.
71. J. S. Bergstra, , R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, (2011) 2546-2554.
72. J. Bergstra, D. Yamins, D. D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. (2013)
73. Shevchuk, Y. [Çevrimiçi] [Alıntı Tarihi: 25 06 2019] http://neupy.com/2016/12/17/hyperparameter_optimization_for_neural_networks.html
74. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
75. J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).
76. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).
77. H. Wu, X. Gu, Max-pooling dropout for regularization of convolutional neural networks. In *International Conference on Neural Information Processing*, Springer, Cham, (2015, November) 46-54.

78. H. Wu, X. Gu, Towards dropout training for convolutional neural networks. *Neural Networks*, *71*, (2015) 1-10.
79. S. Park, N. Kwak, Analysis on the dropout effect in convolutional neural networks. In *Asian Conference on Computer Vision*, Springer, Cham. (2016, November) 189-204.
80. N. Breslow. A generalized Kruskal-Wallis test for comparing K samples subject to unequal patterns of censorship. *Biometrika*, *57*(3), (1970) 579-594.
81. P. E. McKight, J. Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, (2010) 1-1.
82. A. C. Elliott, L. S. Hynan, A SAS® macro implementation of a multiple comparison post hoc test for a Kruskal–Wallis analysis. *Computer methods and programs in biomedicine*, *102*(1), (2011) 75-80.
83. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).