T.C.

EGE ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

GENETİK ALGORİTMALAR VE MAKİNE ÖĞRENMESİ YÖNTEMLERİYLE GÖRÜNTÜ SINIFLANDIRMA

Emre ALTAN

Danışman : Prof. Dr. M. Serdar KORUKOĞLU

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

İzmir
2019

Emre Altan tarafından Yüksek Lisans tezi olarak sunulan "**Genetik Algoritmalar ve Makine Öğrenmesi Yöntemleriyle Görüntü Sınıflandırma**" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **20.08.2019** tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Prof.Dr. M. Serdar KORUKOĞLU**	.....
Raportör Üye	**: Doç.Dr. Hasan BULUT**	.....
Üye	**: Dr. Öğr. Üyesi Korhan KARABULUT**	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "Genetik Algoritmalar ve Makine Öğrenmesi Yöntemleriyle Görüntü Sınıflandırma" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

20 / 08 / 2019

Emre Altan

ÖZET

GENETİK ALGORİTMALAR VE MAKİNE ÖĞRENMESİ YÖNTEMLERİYLE GÖRÜNTÜ SINIFLANDIRMA

ALTAN, Emre

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. M. Serdar KORUKOĞLU

Ağustos 2019, 51 sayfa

Yeni araç ve teknolojilerin gelişmesiyle insanlar olağanüstü görsel veri artışına tanık olmaktadır. Bu görsel verilerin bilgisayarlar tarafından anlamlandırılabilmesi için, görüntü sınıflandırma yöntemlerine ihtiyaç duyulmaktadır. Görüntü sınıflandırma; verilen görüntülerden öznitelik çıkarımı yapılarak önceden belirlenmiş sınıflara görüntülerin atanması işlemidir.

Bu tez kapsamında görüntü sınınflandırma problemine çözüm üretebilmek amacıyla, derin öğrenmenin bir alt dalı olan evrişimli yapay sinir ağları kullanılmıştır. Sinir ağlarının modellerinde kullanılacak hiperparametrelerin belirlenmesi işleminin optimizasyonu için ise genetik algoritmalardan faydalanılmıştır.

Çalışmada en uygun evrişimli sinir ağı modelinin bulunması için, katman sayısı, katmanlardaki nöron sayıları, aktivasyon fonksiyonları ve iyileştirici fonksiyonlar ayrı ayrı parametre setleri olarak sinir ağı oluşturan birime verilir. İlk olarak sinir ağı popülasyonları rastgele üretilir. Daha sonra bu popülasyonlara genetik algoritma ile seçilim uygulanır ve en iyi sonucu veren parametreler belirlenir. Deneysel çalışmada, geliştirilen sistemin farklı verisetleri üzerindeki başarıları ve çalışma süreleri ölçülmüştür.

Anahtar sözcükler: Görüntü Sınıflandırma, Genetik Algoritmalar, Hiperparametre Optimizasyonu, Evrişimli Yapay Sinir Ağları.

ABSTRACT

IMAGE CLASSIFICATION WITH GENETIC ALGORITHMS AND MACHINE LEARNING METHODS

ALTAN, Emre

MSc, Computer Engineering

Supervisor: Prof. M. Serdar KORUKOĞLU

August 2019, 51 pages

With the development of new tools and technologies, people are witnessing an extraordinary increase in visual data. In order for these visual data to be interpreted by computers, image classification methods are needed. Image classification; is the process of assigning images to predetermined classes by extracting attribute from given images.

In this thesis, convolutional neural networks, which is a sub-branch of deep learning, has been used in order to produce a solution to the image classification problem. Genetic algorithms were used for optimization of hyperparameters to be used in neural network models.

In order to find the most suitable convolutional neural network model in the study, the number of layers, the number of neurons in the layers, activation functions and optimization functions were given to the neural network modelling units as separate sets of parameters. Firstly, neural network populations has been generated randomly. Then the selection has been applied to these populations with genetic algorithm and the parameters that yield the best results are determined. In the experimental study, the successes and working times of the developed system on different datasets were measured.

Keywords: Image Classification, Genetic Algorithms, Hyperparameter Optimization, Convolutional Neural Networks.

ÖNSÖZ

Son yıllarda dijital ortamdaki veriler hızla artış göstermektedir. Bu verilerin otomatik olarak anlamlandırılma çalışmaları da paralel olarak artmıştır. Yüksek lisans ders alma sürecinde aldığım sayısal görüntü işleme dersinde görüntü işleme çalışmalarının, tıp alanında, endüstriyel alanlarda ve günlük hayatta bir çok faydalı uygulaması olabileceğini gördüm. Değerli danışmanım Prof. Dr. M. Serdar Korukoğlu ile görüşerek bu alanda çalışma hususunda karar aldık.

Teknolojinin gelişmesiyle birlikte donanım maliyetleri düşmüştür. Bunun sonucunda yüksek donanım özelliklerine gereksinim duyan derin öğrenme algoritmaları kabul edilebilir zaman dilimlerinde oldukça başarılı sonuçlar vererek görüntü tanıma çalışmalarında popüler hale gelmiştir. Bu sebeple görüntü tanıma problemini derin öğrenme yaklaşımıyla çözmeye karar verdik. Görüntü işleme çalışmalarına derin öğrenme ile çözüm getirilmeye çalışılırken en önemli husus derin öğrenme parametrelerinin uygun seçilmesidir. Bu parametrelerin seçiminde birden fazla yaklaşım mevcuttur. Araştırmalarım sonucu parametre seçimi işleminde göreceli olarak daha hızlı ve uygun değerlerin bulunmasında daha güçlü bir teknik olduğunu gördüğüm genetik algoritmaları kullanmaya karar verdim.

Çalışma sürecim boyunca daha önce yapılmış bir çok çalışmayı araştırdım ve bu bilgi mirasından oldukça faydalandım. Ben de yapmış olduğum çalışmamın bu bilim mirasına katkı sağlamasını umuyor, gelecek nesiller için faydalı bir çalışma olmasını diliyorum.

İZMİR

20/08/2019

Emre Altan

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

Tablo Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar

CNN	Evrişimli sinir ağları
ELU	Üstel doğrusal birimler
MLP	Çok katmanlı algılayıcı
RAM	Rastgele erişimli bellek
ReLU	Doğrultulmuş lineer birimler
SVC	Destek vektör sınıflandırıcı
SVM	Destek vektör makineleri
YSA	Yapay sinir ağları

1. GİRİŞ

Günümüzde, fotoğraf albümlerinden web içeriklerine birçok kaynakta görüntüler insanların kullanımına sunulmaktadır. Bu resimlerin makineler tarafından anlamlandırılması, robotikten tıbba kadar birçok farklı alanda fayda sağlamaktadır. Resimlerin anlamlandırılması ise görüntü sınıflandırma çalışmaları ile mümkün olabilmektedir. Görüntü sınıflandırma; görüntülerin, önceden belirlenmiş etiket veya sınıflara göre kategorizasyon işlemidir.

Görüntü sınıflandırma işleminin, endüstride bozuk hammaddelerin tespiti, otonom araçlarda yolda görülen engellerin ve trafik uyarıları ile işaretlerinin tanınması, insansız hava araçları ve keşif araçları gibi robotlarda nesne tanıma, tıpta hastalıklı hücrelerin belirlenmesi gibi birçok uygulaması günümüzde mevcuttur. Bu uygulamaların tümü, kendi alanlarında hem iş gücü hem de zaman tasarrufu sağlamaktadır.

Görüntü sınıflandırma işlemi pek çok makine öğrenmesi yöntemi ile gerçekleştirilebilmektedir. Son zamanlarda bu yöntemlerden derin öğrenme (deep learning) ve bunun alt dalı olan evrişimli sinir ağları (Convolutional Neural Networks - CNN), özellikle görüntü işleme çalışmalarında başarılı sonuçlar vermesinden ötürü oldukça popüler bir hale gelmiştir.

Evrişimli sinir ağları, yapısı gereği modeli oluştururken kullanılan parametrelere ve modelin yapısına göre farklı verisetlerinde farklı sonuçlar vermektedir. Dolayısıyla model yapısı ve aktivasyon ile iyileştirici fonksiyonlar gibi ağda kullanılan parametreler, çalışmaların başarılarını doğrudan etkileyen faktörlerdir. Çalışmada, başarılı sonuç elde edilen parametrelerin belirlenmesi için genetik algoritmalar ile hiperparametre optimizasyonu yaklaşımı önerilmiştir.

Çalışma kapsamında, Python programlama dili ve Keras kütüphanesi ile bir sistem geliştirilmiştir. Bu sistemde girdi parametresi olarak verilen popülasyon nüfusu (population) adedince sinir ağı üretilmektedir. Bu ağlar üretilirken, her bir ağın modelindeki gizli katmanların oluşturulması için parametre kümelerinden faydalanılmaktadır. Bu parametre kümeleri; katman sayısı, her bir katmandaki nöron sayısı, her bir katmanda kullanılan aktivasyon fonksiyonu ve iyileştirici (optimization) fonksiyonu setleridir. İlk nesildeki (generation) tüm ağlar rastgele

parametrelerle üretilirken, bu ağlar arasından en başarılıların bir kısmı bir sonraki nesle aktarılır. Geriye kalan ağlar ise çarprazlama (crossover) ve mutasyon (mutation) ile üretilir ve yeni nesil oluşturulmuş olur. Toplam nesil sayısı, eğitim ve testte kullanılacak veri seti, mutasyon oranı ve en başarılı ağların aktarım oranı da sisteme girdi olarak verilmektedir.

Ağlarda uygunluk fonksiyonu (fitness function) olarak ağların test veri setlerindeki doğruluk oranları (accuracy ratio) kullanılmıştır. Her bir nesil üretildikten sonra nesilde bulunan tüm ağlar, girdi olarak verilen veri setindeki eğitim verisiyle eğitime, ardından da test verileriyle teste tabi tutulur. Tüm nesiller üretilip eğitildikten ve teste tabi tutularak başarıları ölçüldükten sonra, sistem en başarılı 5 ağ modelini, doğruluk oranları ve kullanılan parametrelerle birlikte çıktı olarak sunar. Bu sayede hangi veri setinde hangi parametrenin daha iyi sonuç verdiği tespit edilebilmektedir.

Tez çalışmasının ikinci bölümünde, bu konuda daha önceki dönemlerde yapılmış yol gösterici çalışmalardan bahsedilmiştir. Üçüncu bölümde; makine öğrenmesi, yapay sinir ağları ve evrişimli sinir ağları anlatılmıştır. Dördüncü bölümde; genetik algoritmalar ve hiperparametre optimizasyonu yöntemlerine değinilmiştir. Beşinci bölümde; tez kapsamında geliştirilen, evrişimsel sinir ağları ve genetik algoritmalar ile görüntü sınıflandırma yaklaşımı açıklanmıştır. Altıncı bölümde ise; kullanılan veri setleri ile bu veri setleri üzerinde elde edilen deneysel sonuçlar belirtilerek sonuçların analizi ve yorumlanmasına yer verilmiştir. Ayrıca ileride bu alanda yapılacak olan çalışmalara yol gösterici olması amacıyla, karşılaşılan zorluklardan ve geliştirilebilecek bölümlerden bahsedilmiştir.

2. ÖNCEKİ ÇALIŞMALAR

Makine öğrenmesi yöntemleri; doğal dil işleme, veri madenciliği, görüntü tanıma, zeki sistemler gibi kritik alanlarda yapılan çalışmalarda uzun bir süredir etkin olarak kullanılmaktadır. Makine öğrenmesi, bilgisayarların veriler aracılığıyla öğrenmesini, veri odaklı kararlar verebilmesini ve öngörülerde bulunabilmesini sağlayan birçok farklı yapıda algoritmaları içinde barındırır. Birçok alanda başarılı sonuçlar veren makine öğrenmesi algoritmaları, konuşma tanıma (speech recognition) veya bilgisayarlı görü (computer vision) gibi insani becerilere daha yakın olan alanlarda ise pek verimli değildir. Bu gibi problemlerin üstesinden gelebilmek için makine öğrenmesinin en yeni alt dallarından birisi olan derin öğrenme algoritmaları ortaya atılmıştır.

Derin öğrenme, çoklu soyutlama seviyelerine sahip verilerin gösterimini öğrenmek için çoklu işleme katmanlarından oluşan hesaplama modellerine izin verir. Bu yöntemler konuşma tanıma, görsel nesne tanıma, nesne algılama ile ilaç keşfi ve genomik gibi diğer birçok alanda son teknolojiyi önemli ölçüde geliştirmiştir. Derin öğrenme; her katmandaki parametreleri, geri yayılım (backpropogation) yöntemiyle bir önceki katmandan gelen çıktılara göre düzenleyerek büyük verisetlerinde bulunan karmaşık yapıları ve desenleri keşfeder. Derin evrişimli ağlar, görüntü, video, konuşma ve ses işlemede çığır açarken, tekrarlayan ağlar metin ve konuşma gibi ardışık veriler üzerinde ışık tutmaktadır. (LeCun et al., 2015)

Görüntü sınıflandırma üzerinde daha önce yapılan çalışmalarda, destek vektör makineleri (support vector machines, SVM), rastgele ormanlar (random forests) ve karar ağaçları (decision trees) gibi, farklı alanlarda genelde başarılı sonuçlar veren makine öğrenmesi algoritmaları kullanılırken, bu algoritmaların yanında farklı öznitelik çıkarımı metodları deneyerek, geleneksel yöntemlerin görüntü sınıflandırma işlemi üzerindeki başarılarını arttırmaya çalışmışlardır.

Chapelle et al. (1999), Corel veri setleri üzerinde farklı çekirdek (kernel) değerlerine sahip destek vektör makineleri ile histogram bazlı görüntü sınıflandırma işlemini test etmişlerdir. Corel7 veri seti üzerinde elde ettikleri en iyi başarı %16,3 hata oranıyken (error rate), Corel14 veri seti ile elde ettikleri hata oranı %11'dir.

Marée et al. (2004), görüntü sınıflandırmada en çok kullanılan veri setlerinden birisi olan MNIST elyazısı rakam veri setinin yanında, ORL (insan yüzü veriseti), COIL-100 (3 boyutlu nesne veri seti) ve OUTEX (doku -texture- veri seti) veri setleri ile yaptıkları çalışmada karar ağacı topluluğu (decision tree ensembles) ve yerel alt-pencereler (local sub-windows) kullanarak görüntü sınıflandırma üzerinde çalışmışlardır. Karar ağacı topluluğuyla MNIST veri seti için elde ettikleri hata oranı %3,26 iken, yerel alt-pencerelerle elde ettikleri hata oranı %2,63'tür.

Yine MNIST veri seti üzerinde çalışan Milgram et al. (2005), hiperdüzlemler (hyperplanes) ve destek vektör makineleri ile iki aşamalı bir sınıflandırma yöntemi önermişlerdir. Elde ettikleri hata oranı %1,50'dir.

Bosch et al. (2007) ise rastgele ormanlar ve rastgele eğreltiotları (random ferns) algoritmaları ile görüntü sınıflandırma işlemini gerçekleştirmeye çalışmışlardır. Caltech-101 ve Caltech-256 veri setleri üzerinde çalışmalarını test eden araştırmacılar, rastgele orman algoritması ile Caltech-101 üzerinde %80, rastgele eğreltiotu algoritması ile %79 doğruluk oranı (accuracy) elde ederken, Caltech-256 veri setinde bu oranlar sırasıyla %45.3 ve %44'tür.

Geleneksel makine öğrenmesi yöntemlerinin, diğer problemlerin aksine görüntü sınıflandırma problemlerinde görece daha az başarılı olduğu görülürken, bu alandaki çalışmalar yapay sinir ağları ve özellikle derin öğrenme yöntemlerine kaymıştır. Aslında derin öğrenmenin temelleri ilk olarak Rumelhart et al.'in 1986 yılında yaptığı çalışmada atılmışsa da, bu yöntemin makine öğrenmesi ve görüntü işleme alanlarındaki yüksek başarısı, daha sonraki yıllarda bilgisayar donanımlarının ucuzlamasıyla ve buna bağlı olarak, büyük ve gelişmiş yapay sinir ağı modellerinin çalışmalarda kullanılabilmesiyle anlaşılmış ve bu alanlarda büyük bir etki yaratmıştır.

Krizhevsky ve Hinton (2009), çalışmalarında CIFAR-10 ve CIFAR-100 adlarında iki yeni görüntü veri seti yaratmışlardır. Krizhevsky (2010), daha sonra yaptığı çalışmada, modellediği 2 katmanlı evrişimli derin inanç ağını (Deep Belief Network) CIFAR-10 veri seti ile eğitmiştir ve %78,90 oranında doğruluk elde etmiştir.

Cireşan et al (2012), çok sütunlu derin yapay sinir ağları ile MNIST ve CIFAR-10 veri setleri üzerinde görüntü sınıflandırma yapmışlardır. Birden fazla evrişimli sinir ağının eğitilmesi ve bu ağların çıktılarının ortalamasının alınmasıyla sonuca ulaşılması prensibini ortaya atan bu çalışma, MNIST veri setinde insan beynine yakın sonuçlar veren ilk çalışma olmuş, aynı zamanda CIFAR-10 veri setinde de o zamana kadar literatürde yer alan tüm çalışmalardan daha iyi bir doğruluk oranı vermiştir. Deneysel çalışmada önerilen sistemin, 35 farklı sinir ağı ile çalıştığında MNIST veri setinde %0,23'lük bir hata oranıyla çalıştığı, CIFAR-10 veri setinde ise 8 farklı sinir ağı ile %11,21'lik hata oranı verdiği görülmüştür. Ancak elbette bu kadar büyük bir sinir ağları sisteminin çalışma zamanı da büyük olmuştur. Çalışmada sunulan bilgiye göre, MNIST veri seti için eğitilen 35 ağdan her birini eğitmek 14 saat sürmektedir.

El Kessab et al. (2013) ise çok katmanlı algılayıcılara (multilayer perceptron) sahip yapay sinir ağları kullanarak MNIST veri seti ile yaptıkları çalışmada ön işleme (preprocessing) aşamasına ağırlık vermişlerdir. Öncelikle medyan filtreleme yöntemiyle görüntüler üzerindeki gürültüyü azaltıp, daha sonra görüntülere uyguladıkları genişleme (dilation) yöntemi ile öznitelik çıkarımı yapmışlardır. Deneysel çalışmada önerdikleri modelin MNIST veri seti üzerinde %80 doğruluk oranı ile çalıştığını gözlemlemişlerdir.

McDonnell and Vladusich (2015), sığ evrişimli sinir ağları (shallow convolutional neural networks) ile hızlı öğrenebilen bir sistem önermiş ve bu sistemi, MNIST, CIFAR-10, NORB ve SVHN (Google Street View House Number) veri setleri üzerinde denemişlerdir. Önerilen sistem, MNIST veri setinde %0,37 hata oranı verirken, CIFAR-10 veri setinde ise %75,68 doğruluk oranıyla çalışmıştır.

Springenberg et al. (2015); önerdikleri evrişimli sinir ağlarında, havuz katmanları (pooling layers) ve tam bağlı katmanlar (fully connected layers) kullanmak yerine, ağın tamamını evrişim katmanlarından (convolutional layers) oluşturmuşlardır. Önerdikleri sistemi, CIFAR-10 ve CIFAR-100 veri setleri üzerinde test etmiş ve sırasıyla %4,41 ve %33,71 hata oranı tespit etmişlerdir.

Ronao and Cho (2016), akıllı telefon sensörleri yardımıyla elde edilen verileri kullanarak, insan aktivitesi tanıma (human activity recognition) problemi

üzerine eğilmişlerdir. Çalışmada, evrişimsel sinir ağları kullanılmış olup, verilerin ön işleme aşamasında ise geçici hızlı Fourier dönüşümü (temporal fast Fourier transform) kullanılmış ve %95,75 başarı elde edilmiştir.

Xiao et al. (2017), derin öğrenmede en sık kullanılan veri seti olan MNIST veri seti üzerinde mevcut çalışmaların çok yüksek başarılar elde edebilmesinden dolayı, aynı yapıya sahip, fakat daha zorlayıcı bir veri setini literatüre kazandırmışlardır. 28x28 boyutlarında moda ürünleri resimlerini içeren bu verisetinin adı Fashion-MNIST'tir. Çalışmada aynı zamanda, bu yeni veri seti üzerinde Logistic Regression, MLP Classifier, KneighborsClassifier ve SVC gibi 13 farklı makine öğrenmesi algoritması test edilmiştir. Çalışma kapsamında destek vektör sınıflandırıcısı (support vector classifier - SVC) en yüksek başarıları vermiştir. Bu sınıflandırıcının MNIST üzerindeki doğruluk oranı %97,8 iken, Fashion-MNIST verisetinde elde ettiği başarı %89,7'dir.

Fashion-MNIST, literatüre sunulmasından itibaren, derin öğrenme ve görüntü sınıflandırma çalışmalarında MNIST'in yerine kullanılmaya başlanmıştır. Bu yeni veri seti üzerinde elde edilen en kayda değer başarılardan biri, Zhong et al. (2017)'in evrişimli sinir ağları ve rastgele silme (random erasing) yöntemleriyle yapmış olduğu çalışmadır. Çalışmada, tipik bir evrişimli sinir ağına sahip sistem ile resmin üzerinde rastgele olarak üretilen bir dikdörtgen yardımıyla silme işlemi yapan ve rastgele silme olarak adlandırılan ikinci bir yöntem ayrı ayrı test edilmiştir. Rastgele silme yönteminin evrişimli sinir ağları üzerinde uygulanmasıyla başarının arttığı gözlemlenmiştir. Çalışmada, Fashion-MNIST, CIFAR-10 ve CIFAR-100 üzerinde elde edilen hata oranları sırasıyla; %4,05, %3,08 ve %17,73'tür.

Görüntü sınıflandırma, tıp alanında birtakım hastalıkların tespiti için de kullanılmaktadır. Sun et al. (2017), evrişimli sinir ağlarını 100'ü etiketli olmak üzere 1874 çift göğüs resmi ile eğitmişler ve %82,43 doğruluk oranı elde etmişlerdir. Evrişimli sinir ağını modellerken, çizge tabanlı yarı-öğreticili (graph-based semi-supervised) bir öğrenme yaklaşımı benimsemişlerdir.

Agarap (2019), derin öğrenmede genellikle aktivasyon fonksiyonu olarak kullanılan düzeltilmiş doğrusal birimleri (rectified linear units - ReLU) sınıflandırıcı fonksiyonu olarak kullanmışlardır. Çalışmada ileri beslemeli (feed-

forward) sinir ağları ile evrişimli sinir ağları birlikte kullanılmıştır. Önerilen sistem; MNIST verisetinde %97,98, Fashion-MNIST veri setinde ise %89,35 başarı elde etmiştir.

Genetik algoritmalar, optimizasyon problemlerinin çözümü için bilinen ve sık kullanılan bir yöntemdir. Derin öğrenme çalışmalarında genetik algoritmalar, genellikle katmanların ağırlıklarını ayarlama amaçlı kullanılmıştır. David and Greental (2014), katman ağırlık setlerinin (weight sets) her birini bir birey olarak tanımlayıp, genetik algoritmayla bu ağırlık setlerinin optimizasyonunu sağlamaya çalışmışlardır. Öğreticisiz yapay sinir ağlarının bir çeşidi olan otokodlayıcılar (autoencoders) üzerinde çalışan sistem, genetik algoritma ile ağırlık optimizasyonu olmadan çalıştırıldığında MNIST veri seti üzerinde %1,85 hata oranı verirken, genetik algoritma yardımıyla bu oran %1,44'e inmiş tir.

Such et al. (2017) ise destekleyicili öğrenme (reinforcement learning) uygulanan derin yapay sinir ağlarında, ağırlıkların optimizasyonu için Q-Learning gibi geleneksel destekleyicili öğrenme algoritmaları yerine, genetik algoritmalardan faydalanmışlardır. Daha sonra bu sistemi, 13 farklı Atari oyununda denemişler ve bu oyunların 4 tanesinde en iyi skorları kendi önerdikleri sistemin aldığını belirtmişlerdir.

Genetik algoritmaların, evrişimli yapay sinir ağlarının optimizasyonunda kullanılması, çoğunlukla katman ağırlıklarının optimizasyonu bazında olmuştur. Bunun yerine, tüm ağın evrilme yoluyla oluşturularak, en iyi sonucu veren ağ modelini bulan çalışma sayısı azdır. Young et al. (2015)'in çalışması, ağın yapısını oluşturan hiperparametrelerin optimizasyonu için genetik algoritma yöntemini kullanan ilk çalışmalardandır. Ancak önerilen yöntemde, genetik algoritma katmanların sayısı veya nöron sayısı, iyileştirici fonksiyonu gibi yapısını belirleyen parametreler yerine, evrişimli katmanların çıktı boyutunu ve çekirdek (kernel) boyutunu optimize etmektedir. Çalışmada üretilen tüm ağlarda varsayılan model olarak, Jia et al. (2014)'in çalışmalarında sundukları Caffe kütüphanesi ve CIFAR-10 veri seti için önermiş olduğu evrişimli sinir ağı modeli kullanılmıştır. Deneysel sonuçlarda, genetik algoritma tarafından optimize edilen ve en iyi sonucu veren ağların CIFAR-10 veri seti üzerindeki başarı oranları ve çalışma süreleri verilmezken, yalnızca genetik algoritma optimizasyonunun

başarıyı arttırdığı belirtilmiştir. Çalışmada, üretilen nesil (generation) sayısı 35 iken, her bir nesilde üretilen birey sayısı 500'dür.

Genetik algoritmaları, görüntü sınıflandırma alanında, hiperparametreleri optimize etmek ve yeni ağlar üretmek yoluyla en başarılı ağı tespit etme işlemi için kullanan en kapsamlı çalışma, Suganuma et al. (2017)'in hazırladığı çalışmadır. Çalışmada evrişimli yapay sinir ağları ile birlikte, Miller ve Harding'in (2000) önerdiği kartezyen genetik programlama (cartesian genetic programming) kullanılmıştır. Önerilen yöntemde, her bir nesilde yalnızca bir ebeveyn (parent) birey ve o bireyin mutasyonu ile oluşan iki adet çocuk birey bulunur. İlk nesilde ebeveyn birey rastgele olarak üretilir. Bu bireyler eğitime tabi tutulur ve doğruluk oranları uygunluk fonksiyonu (fitness function) olarak belirlenir. Bu üç adet bireyden en başarılı olan birey bir sonraki neslin ebeveyn bireyi olarak seçilir ve yeni neslin diğer iki üyesi yine bu ebeveynin mutasyonu ile üretilir. Çalışmada derin sinir ağı olarak, evrişimli yapay sinir ağı ve artık ağlar (residual networks) kullanılmış, bu iki ağı baz alan ve genetik algoritmanın ürettiği sistemler CIFAR-10 veri seti üzerinde test edilmiştir. Evrişimli sinir ağı ile üretilen sistem, veriseti üzerinde %6,75 hata oranıyla çalışırken, artık ağ ile üretilen sistem ise %5,98 hata oranı vermiştir. Evrişimli sinir ağının optimizasyonu için 500 nesil üretilirken, atık ağların optimizasyonu için ise 300 nesil üretilmiştir. Çalışmanın dezavantajı ise çalışma süreleridir. En iyi sonuçları veren evrişimli sinir ağının optimizasyonu 15,2 gün, atık ağların optimizasyonu ise 13,7 gün sürmüştür.

3. MAKİNE ÖĞRENMESİ YÖNTEMLERİ

Makine öğrenimi, sistemlere açıkça programlanmadan, otomatik olarak ve deneyimler yoluyla öğrenme yeteneği sağlayan yapay zekanın bir uygulamasıdır. Makine öğrenmesi, verilere erişebilen ve kendileri için öğrendiklerini kullanabilen bilgisayar programlarının geliştirilmesine odaklanır. Birincil amaç, bilgisayarların insan müdahalesi veya yardımı olmadan otomatik olarak öğrenmesini sağlamak ve eylemleri buna göre ayarlamaktır.

Çalışma kapsamında makine öğrenmesi metotları geleneksel yöntemler ve derin öğrenme yöntemleri olarak ele alınmış, deneysel çalışmalar derin öğrenme mimarisi kullanılarak yürütülmüştür. Derin öğrenme yöntemleri de makine öğrenmesi yöntemlerinin bir alt dalı olarak ele alınsa da, geleneksel yöntemlerden ayrılan yönleri mevcuttur. Geleneksel makine öğrenmesi, verileri ayrıştırmak, bu verilerden öğrenmek ve öğrendiklerine dayanarak bilinçli kararlar vermek için algoritmalar kullanır. Derin öğrenme yapıları, kendi başına akıllı kararlar alabilen ve öğrenebilen bir "yapay sinir ağı" oluşturmak için katmanlar halinde algoritmalar oluşturur. Bu özelliğiyle derin öğrenme insan benzeri yapay zekaya en çok benzeyen makine öğrenmesi yöntemlerindendir.

3.1. Geleneksel Yöntemler

Geleneksel makine öğrenmesi yöntemleri kabaca, Öğreticili öğrenme, öğreticisiz öğrenme, yarı öğreticili öğrenme ve destekleyicili öğrenme olmak üzere dört ana başlık altında toplanabilir.

Öğreticili öğrenme önceden etiketlenmiş sınıf verilerinden öğrenme işlemidir. Öğreticili öğrenme algoritmaları, geçmişte öğrenilenleri gelecekteki olayları tahmin etmek için etiketli örnekler kullanarak yeni verilere uygulayabilir. Bilinen bir eğitim veri setinin analizinden başlayarak, öğrenme algoritması çıktı değerleri hakkında tahminler yapmak için çıkarımlı bir fonksiyon üretir. Sistem yeterli eğitimden sonra herhangi bir yeni girdi için etiketleme işlemini gerçekleştirir. Öğreticili öğrenme algoritmaları temel olarak sınıflandırma ve eğri uydurma olarak ayrılır. Sınıflandırma algoritmaları, çıkışlar sınırlı bir değer setiyle sınırlandırıldığında kullanılır. Eğri uydurma algoritmaları ise, çıkışlar bir aralık içinde herhangi bir sayısal değere sahip olduğunda kullanılır.

Öğreticisiz öğrenme öğrenme algoritmaları, sınıf bilgisi verilmeyen girdiler içeren bir veri kümesi alır ve veri setindeki tanımlanabilir elementleri gruplayarak veya kümeleyerek verilerdeki genel yapıyı tahmin etmeye çalışır. Bu nedenle algoritmalar etiketlenmemiş, kategorize edilmemiş veya sınıflandırılmamış test verilerinden öğrenirler. Öğreticisiz öğrenme algoritmaları geri bildirime cevap vermek yerine, verilerdeki ortak noktaları tanımlar ve her yeni veri parçasındaki bu ortak noktaların varlığına veya yokluğuna dayanarak tepki verir.

Yarı öğreticili öğrenme, öğreticili ve öğreticisiz makine öğrenmesi yöntemlerinin bir birleşimidir. Yarı öğreticili öğrenmede, bir algoritma hem etiketli hem de etiketlenmemiş verileri içeren bir veri kümesinden öğrenir. Doğru bir model üretmek için yeterli etiketli veri olmadığında, eğitim verilerinin boyutunu artırmak için yarı öğreticili öğrenme teknikleri kullanılır.

Destekleyicili öğrenme, yazılım ajanlarının bir ortamda kümülatif ödül kavramını en üst düzeye çıkarmak için nasıl bir eylemde bulunmaları gerektiği ile ilgili bir makine öğrenmesi alanıdır. Ajanın öğrenebilmesi için ödül geri bildirimi verilir ve bu bildirim takviye sinyali olarak adlandırılır. Birçok destekleyicili öğrenme algoritması dinamik programlama tekniklerini kullanır. Destekleyicili öğrenme algoritmalarının, genellikle otonom araç tasarımı ve makinenin insan rakiplerine karşı oyun oynamayı öğrenmesi gibi uygulama alanları vardır.

3.2. Derin Öğrenme Yöntemleri

Derin öğrenme yaklaşımı, çoklu soyutlama seviyelerine sahip verilerin gösterimini öğrenmek için çoklu işleme katmanlarından oluşan hesaplama modellerinden meydana gelir. Derin öğrenme yöntemleri, konuşma tanıma, görsel nesne tanıma, nesne algılama, ilaç keşfi ve gen bilimi gibi birçok alanda son teknolojiyi önemli ölçüde geliştirmiştir. Derin öğrenme, makinenin her katmandaki gösterimi önceki katmandaki gösterimden hesaplamak için kullanılan dahili parametrelerini nasıl değiştirmesi gerektiğini göstermek için geri yayılma algoritmasını kullanarak karmaşık veri kümelerindeki karmaşık yapıyı keşfeder. Derin evrimsel ağlar, görüntü, video, konuşma ve ses işlemede çığır açarken, tekrarlayan ağlar metin ve konuşma gibi ardışık veriler üzerinde oldukça başarılı sonuçlar vermektedir. (LeCun et al, 2015)

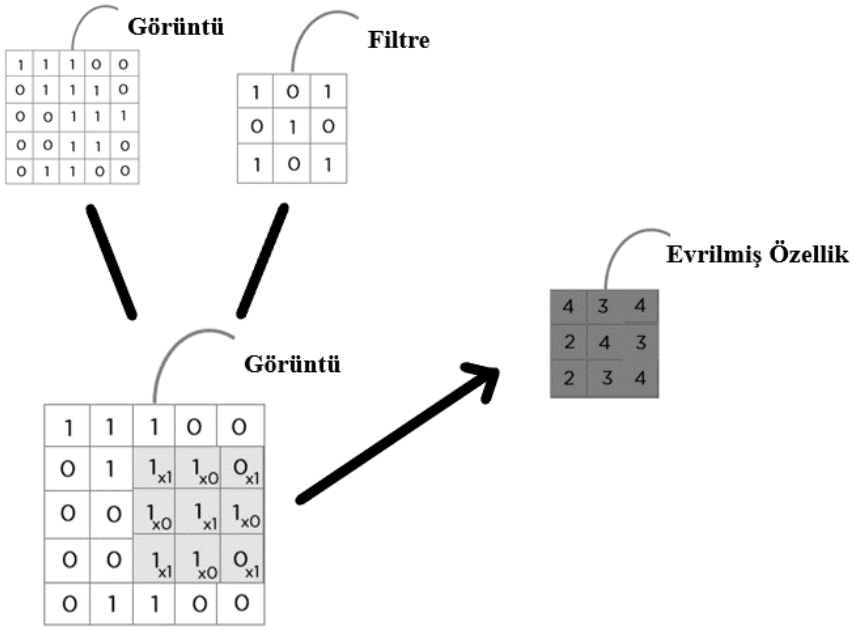
Derin öğrenme ile öznitelik çıkarım işlemi derin öğrenme mimarisi içerisinde otomatik olarak yapılmaktadır. Son zamanlarda derin öğrenme çalışmaları geleneksel yöntemlerle kıyaslandığında, derin öğrenme yöntemlerinin daha başarılı sonuçlar verdiği görülmüştür. Fakat eğitim için çok miktarda eğitim verisi gerektirmesi ve uzun eğitim süreleri derin öğrenme yöntemlerinin dezavantajı olarak görülmektedir.

3.2.1. Evrişimli yapay sinir ağları

Evrişimli yapay sinir ağları, üç renk kanalında piksel yoğunluğunu içeren üç adet 2 boyutlu diziden meydana gelen renkli bir görüntü gibi, çok sayıda dizi biçimindeki verileri işlemek için tasarlanmıştır. Birçok veri çeşidi farklı boyutlarda çoklu diziler biçimindedir: sinyaller ve diziler 1 boyutlu, görüntüler ve ses spektrogramları 2 boyutlu, videolar ve volumetrik görüntüler ise 3 boyutlu dizilerle temsil edilebilir.

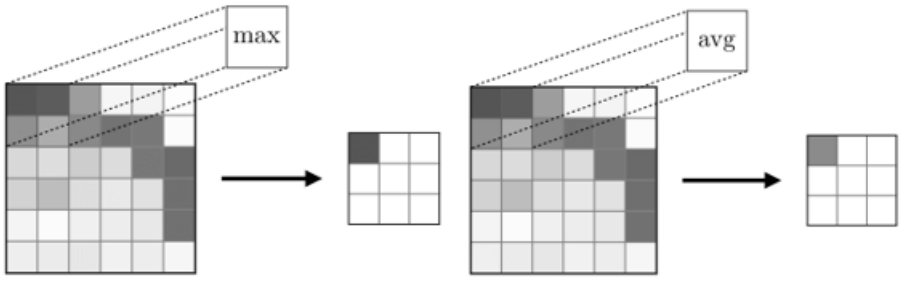
Evrişimli yapay sinir ağlarında gizli katman, özellik çıkarımı yapan birkaç farklı katmandan oluşur. Evrişim operasyonu her evrişim sinir ağının çekirdeğini oluşturur. Bir CNN'de farklı türlerde katmanlar bulunur. Bunlar arasında en sık kullanılan türler; evrişim katmanı (convolution layer), havuzlama katmanı (pooling layer) ve tam bağlı (fully connected) katmanlardır. Bu katmanların haricinde, normalizasyon katmanı, düzleştirme (flattening) katmanı gibi katmanlar da yerine göre kullanılabilmektedir.

Evrişim katmanı, CNN'in çekirdeğini oluşturur. Bu katman, görüntü pikselleri dizisi üzerinde bir filtre matrisi kullanır ve bir evrilmiş özellik haritası elde etmek için evrişim işlemini gerçekleştirir. Yani görüntü üzerinde bir filtreyi kaydırarak uygular ve bu sayede görüntüye filtre uygulanmış olur. Şekil 3.1'de, evrişim katmanında 5x5'lik bir görüntü üzerinde 3x3'lük bir filtre uygulanması tasvir edilmiştir.



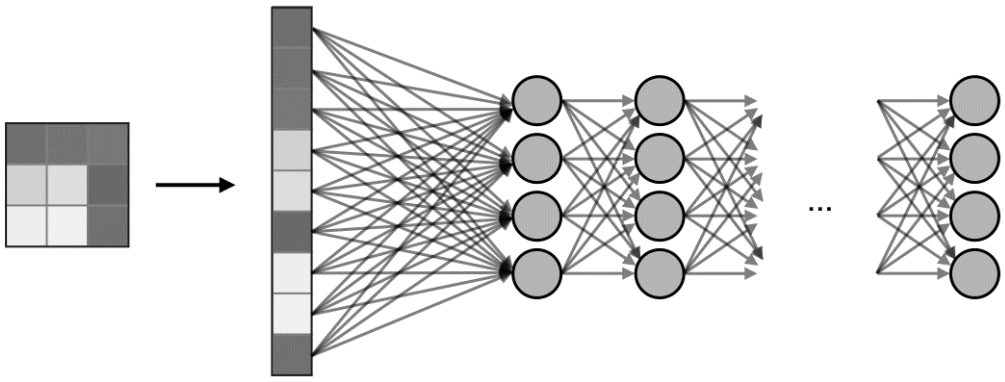
Şekil 3.1 5x5'lik bir görüntü üzerinde 3x3'lük bir filtre uygulanması.

Havuzlama katmanı, kısaca örnekleme işlemi için kullanılır. Bir görüntünün hangi sınıfa ait olduğu konusunda belirleyici özellik olabilecek bir bölümü, görüntü döndürüldüğünde ya da o kısmın pozisyonu değiştiğinde, evrişim katmanı farklı bir öznitelik haritası çıkarır. Örneğin bir kuşun gagası bir görüntüde soldayken bir görüntüde sağda ise, hatta aynı resim 45 derece döndürüldüğünde bile, bir evrişim katmanı bu iki görüntü için farklı öznitelik matrisi çıkarır. Bu şekilde birbirine benzer görüntülerin birbirine yakın öznitelik haritalarına sahip olabilmesi için, aşağı örnekleme (down sampling) gereklidir. Havuzlama katmanı bu aşağı örnekleme işlemini gerçekleştirir. Havuzlama katmanının, maksimum havuzlama (max pooling) ve ortalama havuzlama (average pooling) gibi türleri bulunur. Maksimum havuzlama, belirlenen alandaki piksel değerlerinden en büyük olanı baz alır. Ortalama havuzlama ise belirlenen alandaki değerlerin ortalamasını alır. Şekil 3.2'de maksimum ve ortalama havuzlamanın çalışma şekli gösterilmiştir.



Şekil 3.2 Maksimum ve ortalama havuzlama çalışma şekli. (Amidi, 2019)

Tam bağlı katmanlar (fully connected layers ya da dense layers) ise her bir girdinin tüm nöronlara bağlı olduğu bir katman yapısıdır. Bir CNN ağında tam bağlı katman varsa, bu katman genellikle ağın sonuna doğru bulunur ve sınıf skoru çıktılarını optimize etmeye yarar.

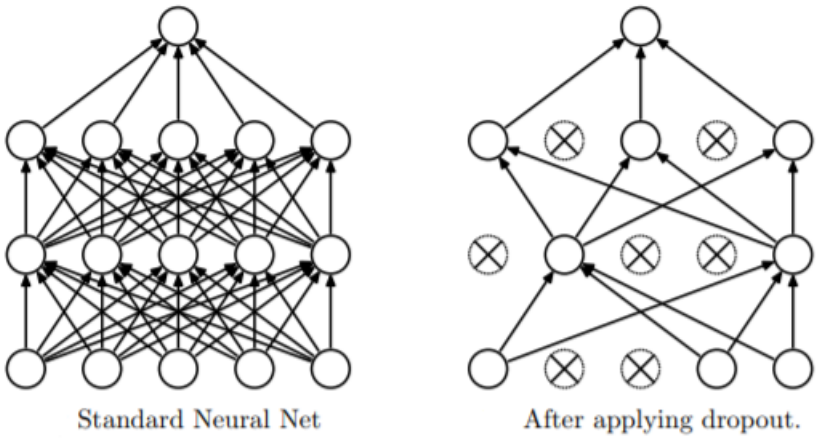


Şekil 3.3 Tam bağlı katmanlar.

Normalizasyon katmanı; daha hızlı ve daha esnek bir eğitime izin vermek için, çıkışın sıfır ortalamaya (zero mean) ve birim standart sapmaya (unit standard deviation) yakın olacak şekilde girişi ölçeklendirir.

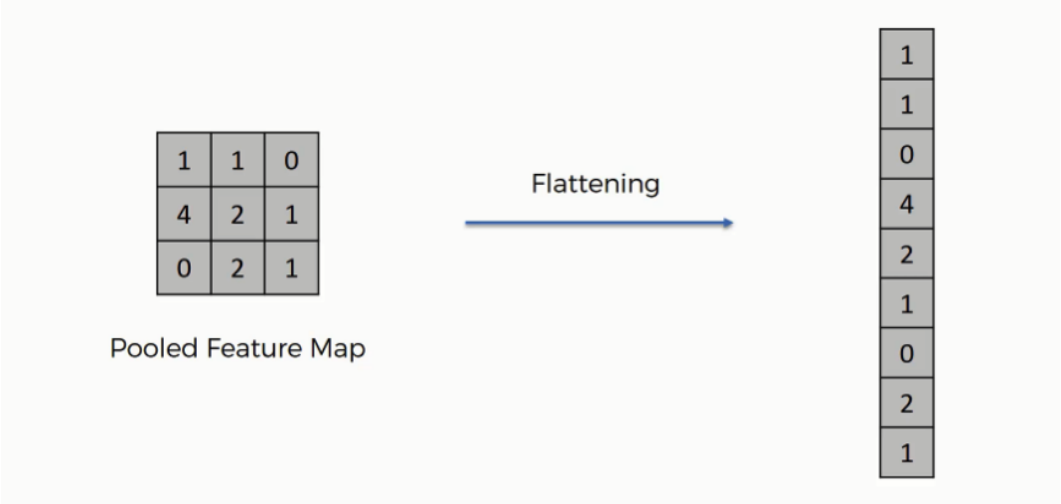
Seyreltme (dropout) katmanı, aşırı uyum (aşırı öğrenme, overfitting) problemini ortadan kaldırmaya yarayan katmandır. Aşırı uyum; ağın, eğitim verisi üzerinde çok iyi sonuçlar verirken, daha önce görmediği test verileri üzerinde düşük sonuçlar vermesi durumudur. Seyreltme katmanı, eğitim sırasında rastgele

seçtiği bazı nöronları pasif hale getirir. Kaç adet nöronun pasif hale getirileceği seyreltme oranına (dropout rate) göre belirlenir.



Şekil 3.4 Seyreltme çalışma mantığı. (Srivastava et al, 2014)

Düzleştirme (flattening) katmanı ise, bir önceki katmandan gelen n boyutlu girdiyi, tek bir kolon haline getirir.



Şekil 3.5 Düzleştirme işlemi. (Superdatascience Team, 2018)

4. MAKİNE ÖĞRENMESİ YÖNTEMLERİNDE HİPERPARAMETRE OPTİMİZASYONU

Hiperparametreler; öğrenme sırasında güncellenmeyen, öğrenme süreci başlamadan önce değerleri belirlenmiş değişkenlerdir. Hiperparametrelerin değeri, geliştirilen modelin problem çözmedeki becerisini belirler. Bu bakımdan hiperparametrelerin doğru ayarlanması modelin performansını doğrudan etkilemektedir.

Hiperparametreler kullanılan algoritmaya veya yönteme göre değişiklik gösterirler. Yapay sinir ağlarında kullanılan ağırlıklar veya öğrenme oranı, destek vektör makinelerinde kullanılan destek vektörleri veya C ve sigma parametreleri, doğrusal veya lojistik regresyonda kullanılan katsayılar, k-en yakın komşuluk algoritmasında kullanılan k değeri hiperparametrelere örnek olarak gösterilebilir. Bir yazılım modeli veya algoritma için en uygun hiperparametre değerlerinin seçilmesi işlemi hiperparametre optimizasyonu olarak adlandırılır.

Her makine öğrenim sistemi hiperparametrelere sahiptir ve otomatik makine öğreniminde temel görevlerden biri bu hiperparametreleri otomatik olarak ayarlamaktır. Özellikle son zamanlarda, çalışmalarda sıklıkla kullanılan derin sinir ağları için mimari geliştirme, düzenlenme ve optimizasyon işlemleri çok çeşitli hiperparametre seçeneklerine bağlıdır. Otomatik hiperparametre optimizasyonu makine öğrenmesi algoritmasını belirli bir probleme uygularken gereken insani çabayı azaltarak maliyetlerde düşüş sağlar. Bununla birlikte algoritmanın eldeki probleme göre ayarlanmasını sağlayarak makine öğrenmesi algoritmalarının performasını iyileştirir. (Feurer et al., 2019)

4.1. Derin Öğrenme Hiperparametreleri

Derin öğrenmede hiperparametreler, genellikle ağ modelinin yapısını belirler. Ağdaki gizli katman sayısı, kullanılan iyileştirici fonksiyonu, seyreltme değeri (dropout rate), gizli katmanlardaki birim sayıları, aktivasyon fonksiyonları ve çekirdek boyutları (kernel size) bir derin yapay sinir ağındaki hiperparametrelere örnek olarak verilebilir. Bu hiperparametrelerin her birinin yanlış ayarlanması performans düşüklüğüne sebebiyet verebilir.

Girdi ve çıktı katmanlarının arasındaki her bir katman gizli katmandır. Gizli katman sayısının azlığı yetersiz uyuma (underfitting – yetersiz öğrenme) sebep olur. Bu da sistemin istenen başarıya erişememesine neden olur. Gizli katman sayısı gerekenden fazla olursa da eğitim süreleri uzar. Katmanlardaki birim (nöron) sayısının düzgün ayarlanmaması da yine aşırı uyum ya da yetersiz uyuma neden olabilir.

Gizli katmanlarda kullanılan çekirdek boyutları, katmanların verdikleri çıktılara direkt olarak etki eder. Öznitelik çıkarımının daha iyi yapılabilmesi için gizli katmanların çekirdek boyutlarının doğru ayarlanmasına bağlıdır. Dolayısıyla başarı oranının artması için çekirdek boyutları iyi ayarlanmalıdır.

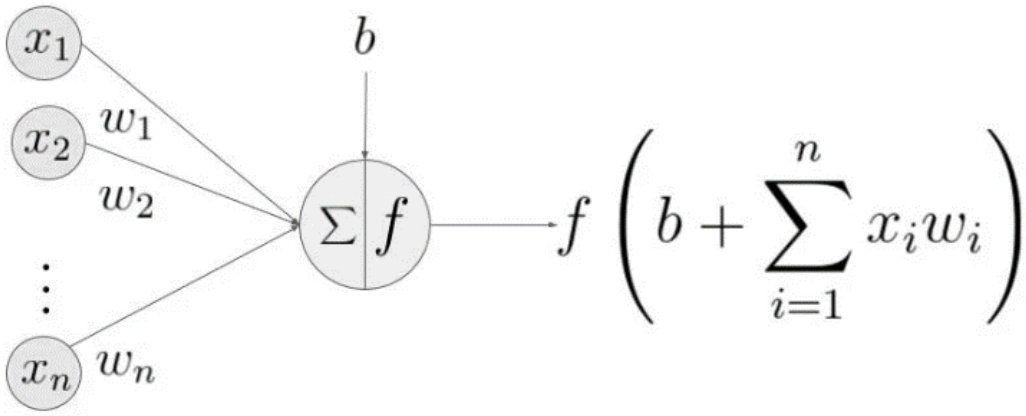
Hiperparametre olarak aktivasyon ve iyileştirici fonksiyonlar da hiperparametreye örnektir. Bu iki fonksiyon tipi ve bu fonksiyonların çeşitleri alt başlıklar altında incelenecektir.

4.1.1. Aktivasyon fonksiyonları

Aktivasyon fonksiyonları, yapay sinir ağlarının önemli özelliklerinden biridir. Aktivasyon fonksiyonlarının yapay sinir ağlarında kullanım nedenini anlamak için, öncelikle biyolojik sinir hücrelerindeki veri iletimini anlamak faydalı olabilir.

Sinir hücreleri, diğer hücrelerdeki veriyi ağaç dalı benzeri bir yapıda olan dendritler yoluyla alır. Yani veri iletimi kaynak sinir ucu ile hedef sinir dendriti arasında gerçekleşir. Dendritler bu veriyi kimyasal taşıyıcılar sayesinde (nörotransmitter) alırlar. Her bir dendritin kendi sinaptik ağırlığı vardır. Gelen sinyalin kuvvetiyle bu sinaptik kuvvet çarpılır ve hücre gövdesine iletilir. Hücre gövdesinde bu ağırlıklandırılmış kuvvetin, belirli bir eşik değerini geçip geçmediği kontrol edilir. Eğer sinyal, eşik değerinin üzerindeyse nöron mesajı aksona iletir. Aksi takdirde, sinyal nöron tarafından öldürülür ve daha fazla yayılmaz. Yapay sinir ağlarında aktivasyon fonksiyonları da tam olarak bu görevi yaparlar, sinyalin iletilip iletilmeyeceğine karar verirler. Aslında bu fonksiyonlar, tek bir parametreye (eşik değeri) sahip basit birer adım fonksiyonudur.

Sistem, yeni bir şey öğrendiğinde, eşik değerleri ve bazı nöronların sinaptik ağırlıkları değişir. Şekil 4.1'de, bir yapay sinir ağında aktivasyon fonksiyonunun çalışma şekli gösterilmektedir.



Şekil 4.1 Aktivasyon fonksiyonunun çalışma mekanizması.

Burada $(x_1, x_2, \ldots, x_n)$ girdi olarak verilen sinyalin vektör değerleri iken, $(w_1, w_2, \ldots, w_n)$ ise her bir nöronun ağırlığıdır. Her bir vektör değeri, ağırlık değeriyle çarpılır ve toplama birimine aktarılır. Son olarak, toplama işleminden çıkan sonuca aktivasyon fonksiyonu uygulanır.

Yapay sinir ağları çalışmalarında, farklı aktivasyon fonksiyonu tipleri kullanılır. Bu çalışmada hiperparametre kümesinde ReLu, Elu, Tanh ve Sigmoid olmak üzere dört farklı aktivasyon fonksiyonu kullanılmıştır.

4.1.1.1. Rectified Linear Units (RELU)

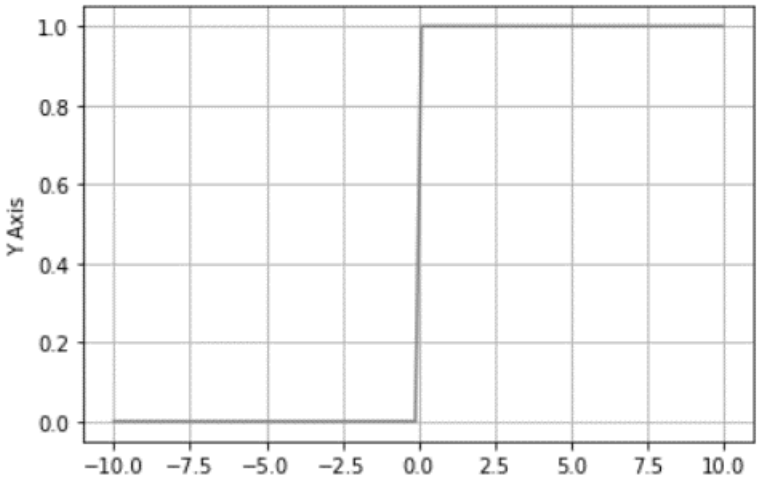
RELU fonksiyonunun formülü şu şekildedir:

$$f(x) = \max(0, x)$$

Yani x değeri 0'dan küçükse 0, 0'dan büyükse x değeri döner. Bu aktivasyon fonksiyonu, ağın çok hızlı yakınsamasını sağlar. Yalnızca basit bir eşik değeri kullandığı için perofrmanslı çalışan bir fonksiyondur, ancak bazı dezavantajları bulunur.

- Sıfır merkezli olmayan bir fonksiyondur.

- RELU ile ilgili diğer bir konu ise, ileri geçiş sırasında x <0 olması durumunda, nöronun aktif kalmaması ve geri geçiş sırasında degradeyi öldürmesidir. Böylece ağırlıklar güncellenmez ve ağ öğrenmez. X = 0 olduğunda eğim bu noktada tanımlanmaz, ancak bu problem uygulama sırasında sol veya sağ degrade seçilerek halledilir.



Şekil 4.2 ReLU fonksiyon çıkış grafiği. (Fortuner 2017)

4.1.1.2. Exponential Linear Units (ELU)

Üstel Doğrusal Birim veya yaygın olarak bilinen adı ELU, maliyeti sıfıra daha hızlı bir şekilde dönüştürme ve daha doğru sonuçlar üreten bir işlevdir. Diğer aktivasyon fonksiyonlarından farklı olarak, ELU pozitif sayılar olması gereken ekstra bir alfa sabitine sahiptir.

$$R'(\mathrm{z}) = \begin{cases} \mathrm{z}, & \mathrm{z} > 0 \\ \alpha.e^{z}, & \mathrm{z} < 0 \end{cases}$$

ELU, negatif girişler dışında RELU'ya çok benzer. Her ikisi de negatif olmayan girdiler için kimlik işlevi biçimindedir. Öte yandan, ELU çıkışı - α 'ya eşit olana kadar yavaş yavaş pürüzsüzleşirken RELU keskin bir şekilde pürüzsüzleşir. Ayrıca RELU'nun aksine ELU, negatif çıktı üretebilmektedir.

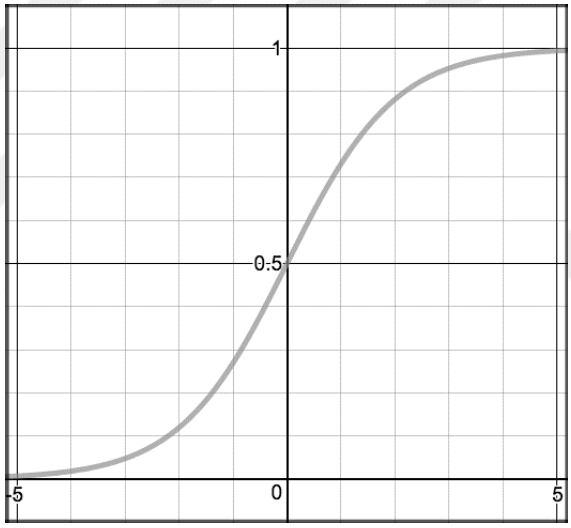
4.1.1.3. Sigmoid

Sigmoid, giriş olarak gerçek bir değer alır ve 0 ile 1 arasında başka bir değer verir. Çalışması kolaydır ve etkinleştirme işlevlerinin tüm güzel

özelliklerine sahiptir: doğrusal olmayan, sürekli farklılaşabilen, monoton ve sabit bir çıktı aralığına sahiptir.

$$S(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid, sınıflandırıcılar için genellikle iyi bir fonksiyondur. Doğrusal olmayan bir fonksiyondur. Ayrıca pürüzsüz bir gradyana sahiptir. Ancak bazı eksileri de vardır. Sigmoid fonksiyonunun herhangi bir ucuna doğru, Y değerleri X'deki değişikliklere çok daha az tepki verme eğilimindedir. Çıkışı sıfır merkezli değildir. Degrade güncellemelerinin farklı yönlerde çok ileri gitmesini sağlar. Kullanım yerine göre, ağ öğrenmeyi reddedebilir ya da çok yavaş öğrenebilir.



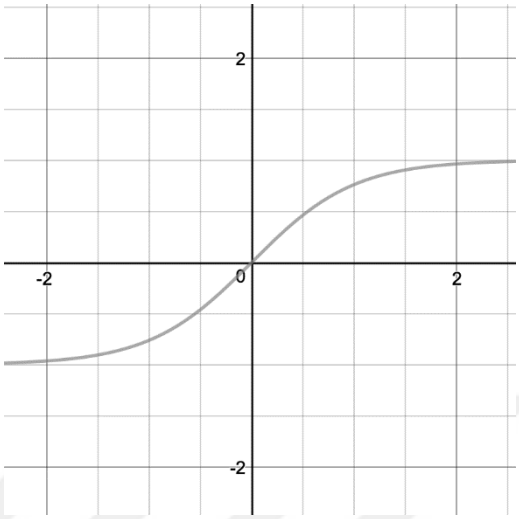
Şekil 4.3 Sigmoid fonksiyon çıkış grafiği. (Fortuner 2017)

4.1.1.4. Tanh

Tanh, [-1, 1] aralığına gerçek değeri olan bir sayıyı sıkıştırır. Doğrusal değildir. Ancak Sigmoid'in aksine çıkışı sıfır merkezlidir. Sigmoid fonksiyonunun biraz değiştirilmiş ve eksi değerler alabilen hali olarak düşünülebilir.

$$T(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Tanh fonksiyonunun gradyanı Sigmoid'den daha güçlüdür. Ancak kaybolan gradyan problemine sahiptir.



Şekil 4.4 Tanh fonksiyon çıkış grafiği. (Fortuner 2017)

4.1.2. İyileştirici (Optimizer) Fonksiyonlar

Bir makine öğrenmesi utgulamasının temel hedefi, tahmin edilen çıktı ile gerçek çıktı değeri arasındaki farkı azaltmaktır. Bu hedefi ne kadar iyi gerçekleştirdiğini, yani tahmin edilen değer ile beklenen değer arasındaki farkı veren fonksiyona kayıp fonksiyonu (loss function) ya da maliyet fonksiyonu (cost function) denir.

İyileştirici fonksiyonlar, eğitim süreci boyunca, parametreleri (ağırlıkları) değiştirerek kayıp fonksiyonunu minimize etmeye ve sistemin tahminlerinin doğruluğunu arttırmaya çalışır. Kayıp fonksiyonu, iyileştirici fonksiyonların kılavuzu hükmündedir, kayıp fonksiyonundan dönen değere göre iyileştirici fonksiyonlar doğru veya yanlış yönde gittiğini anlar.

Çalışmada; Rmsprop, Adam, Sgd, Adagrad, Adadelta, Adamax ve Nadam olmak üzere 7 farklı iyileştirici fonksiyon türü kullanılmıştır.

4.1.2.1. Adagrad (Adaptive Gradient Algorithm)

Adagrad, öğrenme oranını özellikle bireysel özelliklere uyarlar. Bu, veri kümenizdeki ağırlıkların bir kısmının diğerlerinden farklı öğrenme oranlarına

sahip olacağı anlamına gelir. Bu çalışma mantığı, birçok giriş örneğinin eksik olduğu seyrek veri kümeleri için gerçekten işe yarar. Bununla birlikte, Adagrad'ın büyük bir dezavantajı da vardır: Uyarlanabilir öğrenme oranı zaman içinde gerçekten çok küçük olma eğilimindedir.

4.1.2.2. RMSprop

RMSprop, Adagrad'ın özel bir versiyonudur ve Geoffrey Hinton tarafından geliştirilmiştir (Hinton et al, 2019). Tüm degradelerin momentum için birikmesine izin vermek yerine, yalnızca sabit bir pencerede degradeler biriktirir. RMSprop, Adagrad'ın açık bıraktığı bazı sorunları çözmeyi amaçlayan bir başka optimize edici olan Adaprop'a da benzer.

4.1.2.3. Adam (Adaptive Moment Estimation)

Adam, uyarlanabilir moment kestirimi anlamına gelir ve mevcut degradeleri hesaplamak için geçmiş gradyanları kullanmanın başka bir yoludur. Adam aynı zamanda mevcut gradyanlara önceki gradyanların kesirlerini ekleyerek momentum kavramını kullanır. Sinir ağlarının eğitimi esnasında yaygın olarak kullanılan bir iyileştiricidir.

4.1.2.4. SGD (Stochastic Gradient Descent)

Tüm eğitim örnekleri için degradeleri her degrade iniş geçişinde hesaplamak yerine, her seferinde yalnızca eğitim örneklerinin bir alt kümesini kullanmak daha verimli olur. Stokastik gradyan inişi, bir seferde örnek yığınlarını kullanan her uygulamada rastgele örnekler kullanan bir uygulamadır.

4.1.2.5. Adadelta

Adadelta, tüm geçmiş gradyanları biriktirmek yerine, biriken geçmiş gradyanların penceresini bir miktar sabit boyuta sınırlayan daha güçlü bir Adagrad uzantısıdır. Bu sayede Adadelta birçok güncelleme yapıldığında bile öğrenmeye devam eder (Decaying Learning Rate Problem). Adagrad'a kıyasla, Adadelta'da başlangıç öğrenme oranı ayarlamaya gerek yoktur.

4.1.2.6. Adamax

Adam fonksiyonu ile aynı çalışmada ortaya atılan Adamax, Adam iyileştiricisinin sonsuzluk normuna dayanan bir çeşididir.

4.1.2.7. Nadam

Dozat'ın (2016) çalışmasında önerilen Nadam iyileştiricisi, tıpkı Adam gibi aslında RMSprop'un momentumlu bir türüdür. Nadam için kısaca Nesterov momentumlu Adam RMSprop denilebilir.

4.2. Hiperparametre Optimizasyon Yöntemleri

Hiperparametre optimizasyon işlemi 1990'lı yıllara dayanan uzun bir geçmişe sahip olsada halen üzerinde çalışılan ve güncelliğini koruyan bir alandır. (Ripley, 1993) Zaman içinde bu problemi çözmek için bir çok yöntem geliştirilmiştir. Izgara arama, rastgele arama, bayes optimizasyon yöntemi, genetik algoritmalar ile optimizasyon, sezgisel parametre uydurma, sıralı model tabanlı optimizasyon bu yöntemlere örnek olarak verilebilir. Bunlardan ızgara arama, rastgele arama, bayes optimizasyon yöntemi ve genetik algoritmalar ile optimizasyon yöntemleri bu bölümde açıklanmıştır.

4.2.1. Izgara arama (Grid Search)

Izgara arama, hiperparametre optimizasyonunu gerçekleştirmenin geleneksel bir yoludur. Belirlenmiş bir hiperparametre alt kümesi üzerinden ayrıntılı olarak arayarak çalışır. Hiperparametrenin her değeri için modeli yeniden eğiterek birden fazla hiperparametre değerini test etmeyi amaçlar. Böylelikle sorunlu alanın haritasını çıkararak optimizasyon işlemini gerçekleştirir. Çok sayıda hiperparametre değeri olduğunda çalışması yavaş olabilir.

4.2.2. Rastgele arama (Random Search)

Rastgele arama, esas olarak, belirlenmiş bir hiperparametreler alt kümesini ayrıntılı olarak değil, rastgele olarak aramasından dolayı, ızgara aramadan farklıdır. İşlem süresi buna bağlı olarak kısalır fakat her zaman optimum sonucu vereceğini garantileyemez. Aramanın performansı rastgele seçilen parametrelerle yakından ilişkilidir.

4.2.3. Bayes optimizasyon yöntemi

Bayes optimizasyon yöntemi, rastgele veya ızgaralı aramaların aksine, olasılıklı bir model haritalama hiperparametreleri oluşturmak için kullandıkları

geçmiş değerlendirme sonuçlarını takip eder. Öncelikle bir model oluşturulur ve öncül model olarak adlandırılır sonrasında ise deneyimlenen sonuçlara dayanarak model güncellenir. Güncellenmiş yeni model ardıl model olarak isimlendirilir. Bayes yöntemi bu işlemleri olasılık hesaplarıyla gerçekleştirir. Yaygın olarak kullanılan bir modeldir ve diğer yöntemlere göre hızlı çalışır. (Snoek et al, 2012)

4.2.4. Genetik algoritmalar ile optimizasyon

Genetik algoritmalar ile optimizasyon işlemi rastgele parametrelerle oluşturulmuş modeller ile başlar. Sonrasında rasgele oluşturulan bu modellerden başarılı olanları çarprazlama (crossover), mutasyon ve gen seçilimi gibi genetik algoritma operatörleriyle işleme tabi tutularak bir sonraki nesle aktarılır. Böylelikle nesiller boyu en uygun parametrelerin yakalanarak modelin iyileştirilmesi amaçlanır. Nispeten hızlı bir tekniktir.

5. GENETİK ALGORİTMALAR

Genetik algoritmalar Charles Darwin'in evrim teorisinden ilham alan sezgisel bir arama yöntemidir. Bu algoritma gelecek neslin yavrularını üretmek için en uygun bireylerin üreme amacıyla seçildiği doğal seleksiyon sürecinden esinlenir. Genetik algoritmalar aşağıda belirtilen adımlardan oluşurlar.

Adım 1: Problem değişken alanı sabit uzunlukta bir kromozom olarak temsil edilir. Kromozom popülasyonun boyutu (N), çarprazlama olasılığı (Pc) ve mutasyon olasılığı (Pm) belirlenir.

Adım 2: Sorunlu alandaki tek bir kromozomun performansını veya uygunluğunu ölçmek için uygunluk fonksiyonu (fitness function) tanımlanır. Uygunluk fonksiyonu, üreme sırasında eşleştirilecek olan kromozomların seçilmesinde temel oluşturur.

Adım 3: Rastgele bir başlangıç popülasyonu oluşturulur. Adım 1'de belirlenen popülasyon boyutu kadar birey (kromozom) üretilir.

Adım 4: Uygunluk fonksiyonu kullanılarak her bir kromozomun uygunluk değeri hesaplanır.

Adım 5: Yeni bireyler üretmek için mevcut popülasyondan bir çift kromozom belirlenir. Kromozomlar, uygunlukları ile ilgili bir olasılık ile seçilirler.

Adım 6: Seçilen kromozomlara genetik operatörler (çaprazlama ve mutasyon) uygulanarak bir çift yavru kromozomu oluşturulur.

Adım 7: Oluşturulan yavru kromozomlar yeni popülasyona atanır.

Adım 8: Yeni kromozom popülasyonunun boyutu, ilk popülasyonun boyutuna eşit olana kadar Adım 5 tekrarlanır.

Adım 9: İlk (ebeveyn) kromozom popülasyonunu yeni (yavru) popülasyonla değiştirilir.

Adım 10: Adım 4'e gidilir ve sonlandırma kriteri yerine getirilinceye kadar kalan adımlar tekrarlanır.

5.1. Çarprazlama (Crossover)

Genetik algoritmalarda ve evrimsel hesaplamada, rekombinasyon olarak da adlandırılan çaprazlama (crossover), iki ebeveynin genetik bilgilerini yeni yavrular üretmek için kullanan genetik bir operatördür. Tek noktalı çarprazlama, k noktalı çarprazlama, tek tip çarprazlama gibi türleri vardır.

Tek noktalı çarprazlama işlemi için; her iki ebeveynin kromozomları üzerine bir nokta rastgele seçilir ve bu nokta 'çarprazlama noktası' olarak adlandırılır. Bu noktanın sağındaki bitler, iki ana kromozom arasında değiştirilir. Bu işlem, her iki ebeveynden de bazı genetik bilgiler taşıyan iki yavru ile sonuçlanır.

K noktalı çarprazlama işleminde, ana kromozomlardan k adet çarprazlama noktası rastgele seçilir. k nokta arasındaki bitler ana kromozomlar arasında değiştirilir.

Tek tip çarprazlama işleminde rekombinasyon için kromozomlar parçalanmaz. Yavrularda her bir gen, ana kromozomların uzunluğu ile aynı uzunlukta olan ikili çaprazlama maskesinde karşılık gelen bit'e göre seçilen ebeveynden kopyalanarak yaratılır. (Sivanandam et al., 2007) Çapraz maskedeki bit 1 ise, o zaman elde edilen gen birinci ebeveynden kopyalanır ve çapraz maskedeki bit 0 ise, o zaman elde edilen gen ikinci ebeveynden kopyalanır.

5.2. Mutasyon (Mutation)

Mutasyon operatörü, bir kromozomda rastgele seçilen bir geni değiştirir. Görevi, arama algoritmasının yerel bir optimumda takılmadığını garanti etmektir. Mutasyon olasılığı doğada oldukça küçüktür ve genetik algoritmalar için de düşük tutulur, tipik olarak 0,001 ila 0,01 arasındadır.

6. GELİŞTİRİLEN YÖNTEM

Bu çalışmada, görüntü sınıflandırma problemi için eğitilecek olan evrişimsel yapay sinir ağlarının hiperparametrelerinin genetik algoritmalar yoluyla optimizasyonuna dayalı bir yöntem önerilmiştir.

Görüntü sınıflandırma probleminde evrişimsel yapay sinir ağlarının başarılı sonuçlar verdiği daha önceki çalışmalarda görülmüştür. Ancak yapay sinir ağı modellenirken kullanılan hiperparametreler, sistem başarısına direkt olarak etki etmektedir. Hangi parametrelerin daha iyi sonuç vereceğinin belirlenmesi ise, ağı modellerken karşılaşılan en büyük zorluktur. Sezgisel bir arama yöntemi olan genetik algoritmalar, optimizasyon işlemi için farklı alanlarda sıkça kullanılmaktadır. Bu çalışmada da, ağın yapısını meydana getiren hiperparametrelerin optimizasyonu için genetik algoritmalardan faydalanılmıştır.

Çalışma kapsamında oluşturulan parametre kümeleri; ağdaki gizli katman sayısı, iyileştirici fonksiyon, gizli katmanlarda kullanılacak olan aktivasyon fonksiyonları, gizli katmanların nöron sayıları ve çekirdek (kernel) boyutlarıdır. Parametre kümelerinin içindeki değerler, Tablo 6.1'de gösterilmiştir.

Tablo 6.1 Kullanılan hiperparametre setleri.

Katman Sayısı	**İyileştirici Fonksiyon**	**Aktivasyon Fonksiyon**	**Nöron Sayısı**	**Çekirdek Boyutu**
1	Rmsprop	Relu	16	1
2	Adam	Elu	32	2
3	Sgd	Tanh	64	3
4	Adagrad	Sigmoid	128	4
	Adadelta			5
	Adamax			
	Nadam			

Sisteme girdi olarak verilen parametreler ise, kullanılacak veri seti, nesil (generation) sayısı ve her bir nesildeki birey sayısıdır (population). Sistem ilk

nesli parametre kümelerinden yararlanarak rastgele üretir. Rastgele üretilen her bir ağ, girdi olarak belirlenen veri setinin eğitim kümesi ile ile eğitime tabi tutulur.

Eğitim aşamasındaki adım (epoch) değeri Keras'ın "EarlyStopping" adlı sınıfı tarafından belirlenir. Bu sınıf, verilen "monitor", "min_delta" ve "patience" parametrelerine göre her adımda öğrenme başarısının artışını gözler. Eğer artış yeterli değilse eğitimi durdurur. Buradaki "monitor" değeri gözlenmesi gereken değeri (kayıp fonksiyonu veya doğruluk oranı), "min_delta" parametresi ise iki çağ arasındaki sağlanması gereken minimum başarı artış oranını temsil ederken, "patience" parametresi ise, "min_delta" oranı sağlanmadan geçilebilecek maksimum adım sayısını verir. Örneğin; "monitor" değeri "val_loss" (kayıp oranı), "min_delta" oranı 0.1 ve "patience" değeri 3 olsun. X adımında (epoch) kayıp oranı 0.21 ise, X+1, X+2 ve X+3'üncü adımlarda bu değer 0.20'ye inmezse eğitim "EarlyStopping" sınıfı tarafından sonlandırılır. Çalışmada, en iyi ağlar bulunana kadar tüm nesillerin eğitiminde "val_loss" değeri 0.1, "patience" değeri ise 10 olarak verilmiştir. Bu sayede her bir ağın eğitim süresi kısaltılmıştır.

Eğitim aşamasının ardından test kümesiyle doğruluk oranı (accuracy) belirlenir. Ağların doğruluk oranları uygunluk fonksiyonu (fitness function) olarak kullanılmıştır.

İlk nesildeki tüm bireylerin eğitim ve test süreçleri tamamlandıktan sonra, uygunluk fonksiyonuna göre ağlar sıralanır. Ağların en başarılı %40'ı (örneğin 10 ağdan en başarılı 4 tanesi) bir sonraki nesle aktarılır. Ancak başarılı ağ olarak seçilmenin kıstası olarak sisteme bir eşik değeri (threshold) verilmiştir. Doğruluk oranı bu değerin altında kalan ağlar, en başarılı ağların arasına girse bile bir sonraki nesle aktarılmaz. Aktarılmayan her bir en başarılı ağın yerine rastgele parametrelerle yeni bir birey oluşturulur ve yeni nesle aktarılır. Başarısız bireylerin de %20 olasılıkla bir sonraki nesle aktarılma şansları vardır. Yeni neslin geriye kalan bireyleri için ise, çarprazlama yoluyla yeni çocuk bireyler üretilir.

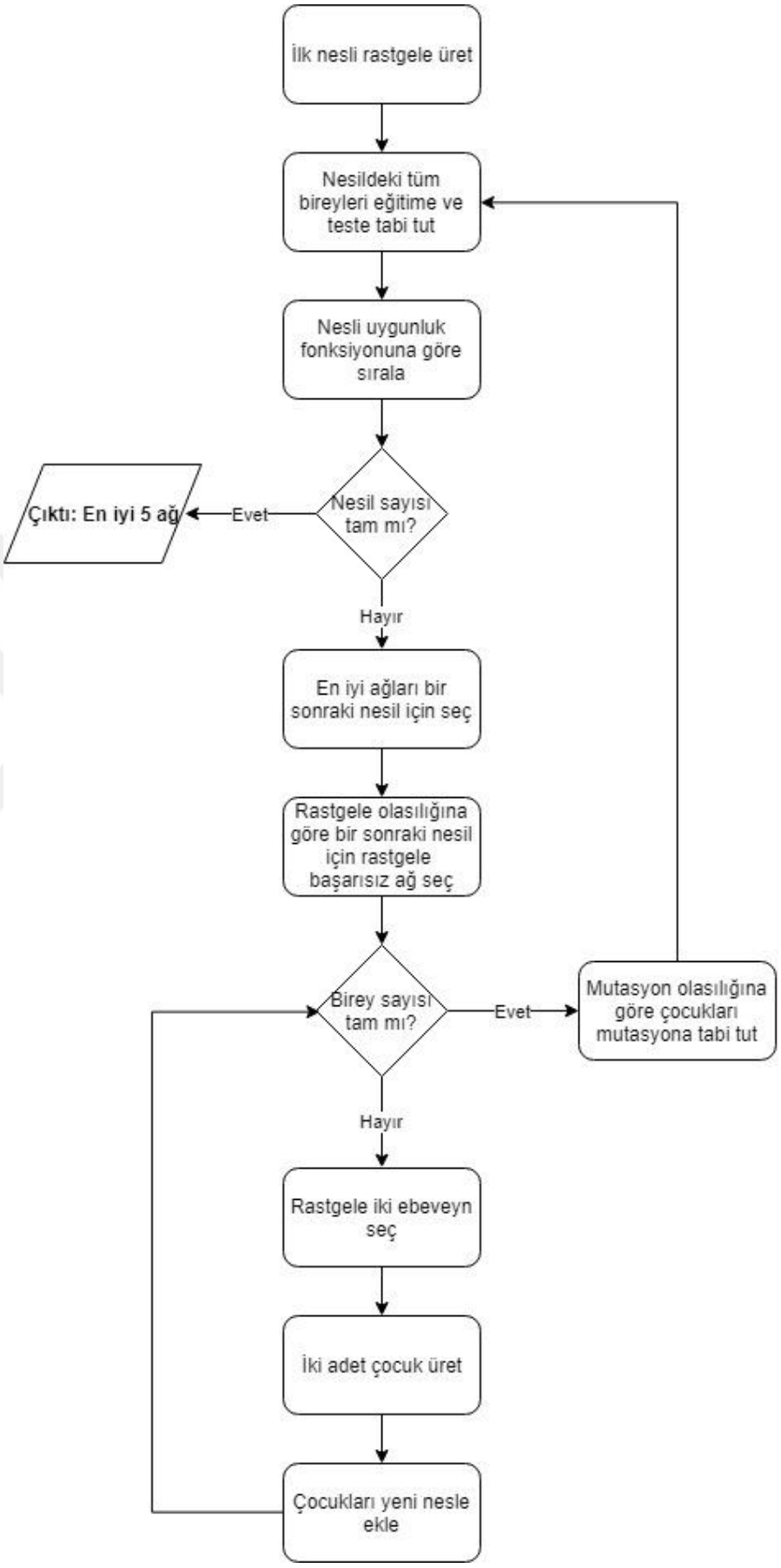
Çarprazlama işlemi için rastgele iki ebeveyn belirlenir. Bu ebeveynler, bir önceki nesilde başarılı olmuş ve bir sonraki nesle aktarılmak için seçilen ağlar arasından belirlenir. Her bir hiperparametre değeri, ebeveynlerden rastgele seçilerek belirlenir. Her bir ebeveyn çiftinden iki adet çocuk üretilir. Bir sonraki nesil sayısı tamamlanıncaya kadar çarprazlama yoluyla çocuk üretimi devam eder.

Genetik algoritmanın yerel minimuma takılmaması için mutasyon işlemi yapılmıştır. Çocuk bireylerin mutasyona uğrama olasılığı %10 olarak verilmiştir. Mutasyon iki şekilde yapılmaktadır. İlk mutasyon türü, tüm parametrelerin rastgele olarak yeniden seçildiği toplu mutasyondur. İkinci mutasyon ise, rastgele seçilen bir gizli katmanın; nöron sayısı, çekirdek boyutu veya aktivasyon fonksiyonu parametrelerinden birinin rastgele bir değerle değiştirildiği kısmi mutasyondur. Çocuk bireylerden biri mutasyon için seçildiğinde, bu iki mutasyon türünün de seçilme olasılığı %50 olarak belirlenmiştir.

Mutasyon işleminin tamamlanmasının ardından, yeni neslin üretimi de tamamlanmış olur. Sisteme girdi olarak verilen nesil sayısı kadar bu işlem tekrar edilir. Bir önceki nesilde doğruluk oranı elde edilen bireyler tekrar eğitime tabi tutulmaz. Bu sayede sistemin çalışma zamanı iyileştirilmiştir.

Tüm nesiller üretilip içindeki bireyler değerlendirildikten sonra, sistem en iyi beş ağın yapısını ve aldıları doğruluk oranını çıktı olarak verir. Tez kapsamında geliştirilen modelin mimarisi Şekil 6.1'deki gibidir.

En iyi beş ağ bulunduktan sonra bu ağlar "EarlyStopping" parametreleri değiştirilerek tekrar eğitime tabi tutulur. "val_loss" değeri 0.001, "patience" değeri ise 20 olarak verilir.



Şekil 6.1 Genetik algoritma çalışma mekanizması.

6.1. Ön İşleme Aşaması

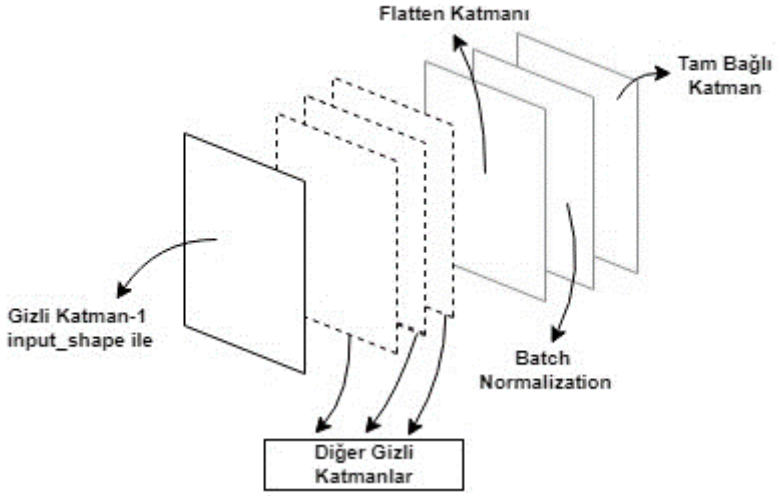
Veriler yapay sinir ağlarına verilmeden önce ön işlemeye tabi tutulmuştur. Ön işleme için Keras kütüphanesinin "ImageDataGenerator" isimli sınıfı kullanılmış ve bu sayede gerçek zamanlı veri ağırlıklandırılması yapılmıştır. Ön işleyici sınıfın parametreleri Chollet'in (2019) çalışmasında önerdiği şekilde kullanılmıştır.

6.2. Ağ Modeli

Geliştirilen sistemde ağlar sıralı (sequential) yapıda olup, her bir ağdaki tüm gizli katmanlar genetik algoritma ile optimize edilmiş hiperparametreler doğrultusunda üretilmektedir. Bu gizli katmanların tümü, iki boyutlu evrişimli katman tipindedir (Conv2D). Yapıda belirli bir giriş katmanı yoktur, gizli katman için verilmiş katman parametre kümelerinden ilki giriş katman olarak kullanılır. Giriş katmanlarının diğer gizli katmanlardan tek farkı belirli bir giriş formu (input_shape) olmasıdır. Bu giriş formu, kullanılan veri setinin yapısına göre belirlenmektedir.

Her bir gizli katmanın sonunda bir adet 2 boyutlu maksimum havuzlama (MaxPooling2D) katmanı ile 0,25 seyreltme oranına sahip seyreltme (dropout) katmanı yer alır.

Modelde çıktı katmanı olarak ise 3 adet katman bulunmaktadır. Bunlardan ilki, kendisine verilen girdiyi tek boyutlu hale getiren düzleştirici katmandır (Flatten). İkinci katman olarak; Ioffe and Szegedy (2014)'nin önerdiği, her partide (batch) bir önceki katmanın aktivasyonunu normalize eden Batch Normalization katmanı kullanılmıştır. Son katman ise Softmax aktivasyon fonksiyonu ile çalışan bir tam bağlı katmandır. Ağların temel yapısı Şekil 6.2'de gösterilmiştir.



Şekil 6.2 Oluşturulan ağların temel yapısı

6.3. Geliştirilen Uygulama

Deneysel çalışma için geliştirilen uygulama, Python programlama dili ile yazılmış olup, veri setlerinin elde edilmesi, yapay sinir ağının oluşturulması, eğitilmesi ve testi için Keras kütüphanesi kullanılmıştır. (Keras Team, 2015)

Program, "main.py" dosyasından başlatılmaktadır. Bu program "optimizer.py" dosyasını çalıştırıp evrimsel algoritma ile nesillerin üretiminin başlamasını sağlar. "optimizer" sınıfı "main" kodundan gelen parametre kümeleriyle, "network.py" dosyasındaki sınıfı kullanarak bireyler üretir. Veri setlerinin ilgili kaynaklardan indirilmesi, bireylerin parametre bilgileriyle yapay sinir ağlarının oluşturulması, ağların eğitim ve test işlemleri "train.py" dosyasındaki kodlar aracılığıyla gerçekleştirilir.

Uygulama, yüksek veri işleme gücü gerektirdiğinden, Google'ın Colaboratory adlı bulut platformu üzerinde çalıştırılmıştır. Buradaki sanal bilgisayarda; 12 GB RAM ve Tesla K80 grafik kartı bulunmakta olup, program bu grafik kartı üzerinde çalıştırılmaktadır.

7. DENEYSEL SONUÇLAR

Bu çalışmada, evrişimli yapay sinir ağlarıyla görüntü tanıma işlemlerinde genetik algoritmalar yardımıyla hiperparametre optimizasyonunun başarıya etkileri incelenmiştir. 6. bölümde anlatılan Python uygulaması geliştirildikten sonra deneyler, Google Colaboratory bulut ortamında gerçekleştirilmiştitr.

Bu bölümde ilk olarak çalışma kapsamında kullanılan veri setleri anlatılmakta, daha sonra ise geliştirilen uygulamanın bu veri setleri üzerindeki başarıları ve çalışma süreleri verilmektedir.

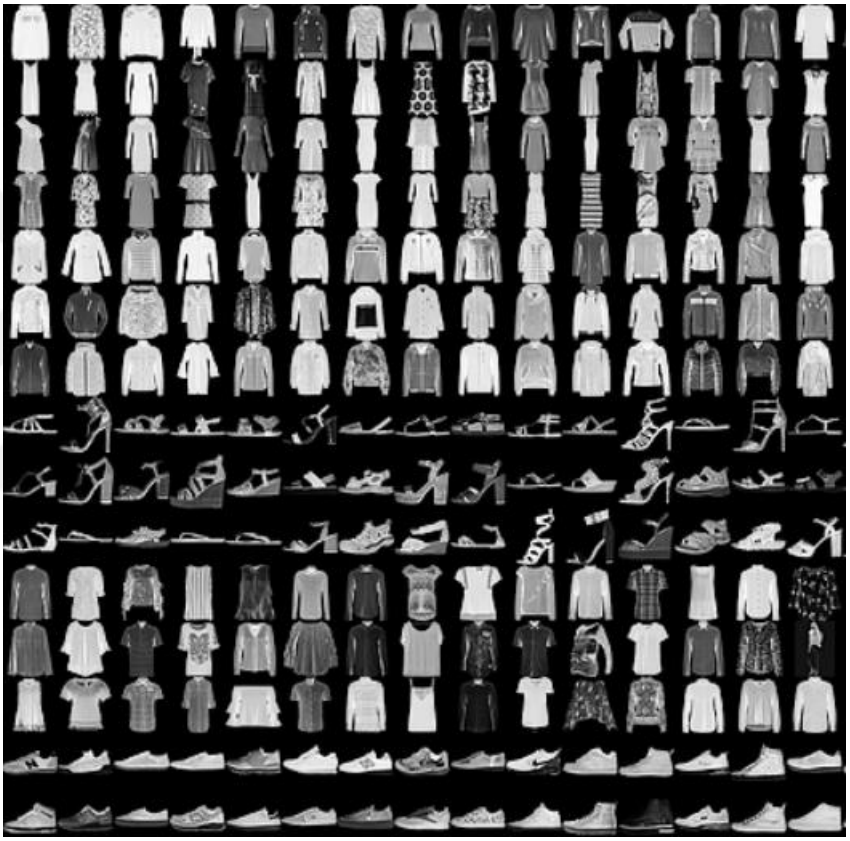
7.1. Veri Setleri

Deneysel çalışmalar kapsamında görüntü sınıflandırma çalışmaları; "MNIST", "Fashion-MNIST" ve "CIFAR-10" olmak üzere üç farklı veri seti üzerinde gerçekleştirilmiştir.

MNIST veri seti, LeCun et al (1998)'in yapmış oldukları çalışmada önerilmiş, el yazısı rakam görüntülerinden oluşan bir görüntü veri setidir. Veri setinde 28x28 boyutlarında 60.000 adet eğitim ve 10.000 adet test verisi olmak üzere toplamda 70.000 adet görüntü bulunmaktadır. Bu görüntüler, el yazısı ile yazılmış sıfırdan dokuza kadar rakamları içermektedir. Yani toplamda 10 adet sınıf bulunmaktadır.

Şekil 7.1 MNIST veri seti örnekleri.

Fashion-MNIST, Xiao et al. (2017)'in MNIST veri setinde yüksek başarılar elde edilmesinin ardından, aynı yapıya sahip daha zorlayıcı bir veri seti olması için oluşturulmuş bir veri setidir. Bu veri setinde MNIST'ten farklı olarak, moda ürünleri görüntülerini içermektedir. Yine MNIST'e benzer şekilde 60.000 adet eğitim ve 10.000 adet test verisi içeren Fashion-MNIST, literatüre sunulduğundan beri özellikle görüntü sınıflandırma çalışmalarında sıkça kullanılmaktadır.



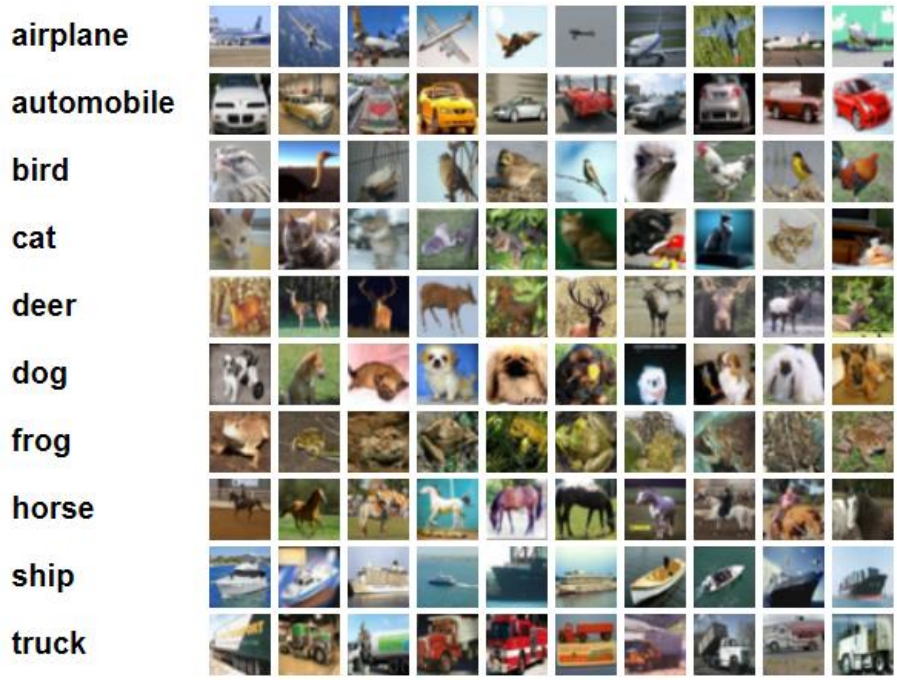
Şekil 7.2 Fashion-MNIST veri seti örnekleri.

Veri setinde toplam 10 adet etiket bulunmaktadır. Tablo 7.1'de bu etiket ve değerleri verilmiştir.

Tablo 7.1 Fashion-MNIST etiketleri.

Etiket	Değeri
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Krizhevsky ve Hinton (2009), görüntü sınıflandırma işlemi için bazı obje ve canlıların küçük görüntülerini içeren iki adet zorlayıcı veri seti meydana getirmişlerdir. Bunlardan CIFAR-10; on adet sınıf içerirken, CIFAR-100 ise 100 adet sınıftan oluşmaktadır. Her iki veri seti de 32x32 boyutlarında, 50.000 adet eğitim ve 10.000 adet test verisi olmak üzere toplamda 60.000 adet görüntüden oluşmaktadır. Çalışma kapsamında yalnızca CIFAR-10 veri seti kullanılmıştır. Aşağıda CIFAR-10 veri setinin etiket değerleri ve her bir etikete ait veri örnekleri bulunmaktadır.



Şekil 7.3 CIFAR-10 veri seti örnekleri.

7.2. Başarı Değerlendirme Ölçütü

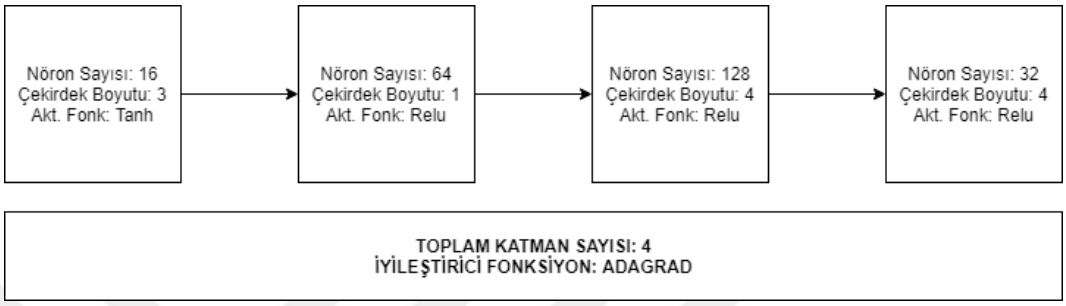
Deneysel çalışmalarda ağların görüntü sınıflandırma başarıları ana başarı ölçütü olarak sunulsa da, sistemin çalışma zamanı da başarı karşılaştırma ölçütü olabilmesi açısından deneysel sonuçlara eklenmiştir. Ağların sınıflandırma başarıları hesaplanırken, doğru sınıflanan veri örneklerinin yüzdesini veren doğruluk oranı (accuracy) kullanılmıştır. Bu oranlar aynı zamanda hiperparametre optimizasyonunu sağlayan genetik algoritmaya uygunluk fonksiyonu olarak verilmiştir.

Deneysel sonuçlar bölümlerinde, her veriseti için elde edilen en başarılı sonuçlar paylaşılacaktır. Ancak genetik algoritmalar metasezgisel (metaheuristic) olduğu için her zaman en iyi sonuçlar elde edilemeyebilir.

7.3. MNIST Veri Seti Deneysel Sonuçlar

MNIST veri seti, evrişimsel sinir ağlarının kolaylıkla öğrenebildiği ve başarılı sonuçlar elde edebildiği bir veri setidir. Sistemin başarılı sonuç elde edebilmesi için 5'er adet bireyden oluşan 5 farklı nesil üretmek yeterli olmuştur.

Sistemin elde ettiği en iyi doğruluk oranı %99,07'dir. Bu oran, görüntü sınıflandırma alanındaki en gelişmiş çalışmaların başarı oranına yakındır. Sistem 5 bireyden oluşan 5 neslin eğitim ve test işlemini 50 dakikada tamamlamıştır. Şekil 7.4'te sistemin MNIST veri setinin sınıflandırılması için ürettiği en iyi sonucu veren ağın yapısı gösterilmiştir.



Şekil 7.4 MNIST veri setinde en başarılı olan ağın modeli

Sistem, MNIST veri seti ile toplam 5 defa test edilmiştir. Sistemin her çalışmasında 5 bireyden oluşan 5'er adet nesil üretildiği için, toplamda 125 adet evrişimli yapay sinir ağı oluşturulmuş ve bu veri seti ile teste tabi tutulmuştur. Tablo 7.2'de MNIST veri seti için üretilen en başarılı 3 ağın hiperparametreleri ve doğruluk oranları verilmiştir.

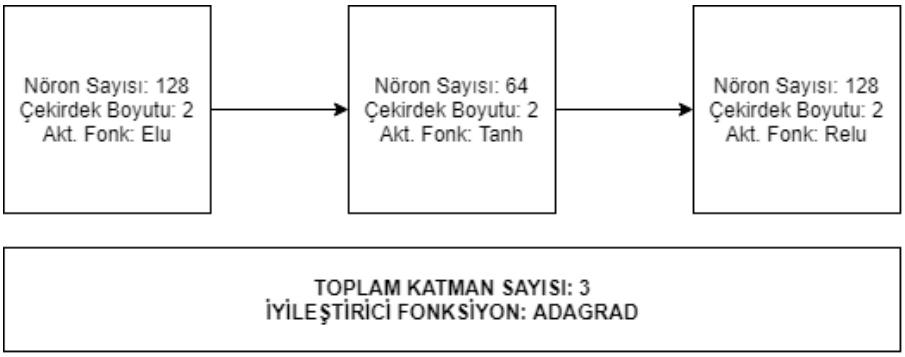
Tablo 7.2 MNIST veri setinde en başarılı üç ağ

1		**1. Katman**	**2. Katman**	**3. Katman**	**4. Katman**
	Nöron Sayısı	16	64	128	32
	Çekirdek Boyutu	3	1	4	4
	Aktivasyon Fonk.	Tanh	Relu	Relu	Relu
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%99,07			
2		**1. Katman**	**2. Katman**	**3. Katman**	**4. Katman**
	Nöron Sayısı	32	64	32	64
	Çekirdek Boyutu	2	2	2	2
	Aktivasyon Fonk.	Elu	Relu	Tanh	Elu
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%99,04			
3		**1. Katman**	**2. Katman**	**3. Katman**	**4. Katman**
	Nöron Sayısı	16	64	64	64
	Çekirdek Boyutu	3	1	5	3
	Aktivasyon Fonk.	Tanh	Elu	Sigmoid	Elu
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%98,54			

7.4. Fashion-MNIST Veri Seti Deneysel Sonuçlar

Fashion-MNIST veri seti, MNIST veri setine kıyasla yapay sinir ağlarını daha zorlayan bir veri setidir. Deneylerde daha başarılı sonuç elde edebilmek amacıyla 10 bireyden oluşan 5 nesil üretilmiştir.

Sistemin elde ettiği en yüksek doğruluk oranı %92.80'dir. Bu oran da yine bu alandaki önde gelen çalışmaların elde ettiği sonuçlara yakın bir sonuçtur. Sistem bu sonucu 1 saat 58 dakikalık bir sürede elde etmiştir. Şekil 7.5'te sistemin Fashion-MNIST veri setinin sınıflandırılması için ürettiği en iyi sonucu veren ağın yapısı gösterilmiştir.



Şekil 7.5 Fashion-MNIST veri setinde en başarılı olan ağın modeli

Sistem, Fashion-MNIST veri seti ile toplam 15 defa test edilmiştir. Yani toplamda 750 adet evrişimli yapay sinir ağı oluşturulmuş ve bu veri seti ile teste tabi tutulmuştur. Tablo 7.3'te Fashion-MNIST veri seti için üretilen en başarılı 3 ağın hiperparametreleri ve doğruluk oranları verilmiştir.

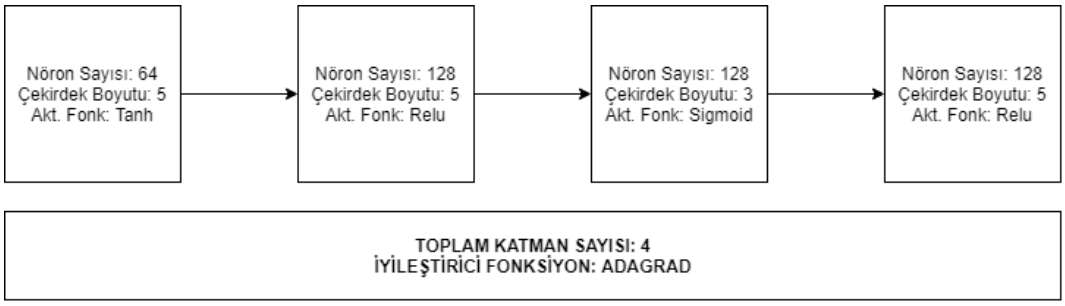
Tablo 7.3 Fashion-MNIST veri setinde en başarılı üç ağ

		1. Katman	**2. Katman**	**3. Katman**	**4. Katman**
1	**Nöron Sayısı**	128	64	128	-
	Çekirdek Boyutu	2	2	2	-
	Aktivasyon Fonksiyonu	Elu	Tanh	Relu	-
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%92,80			
		1. Katman	**2. Katman**	**3. Katman**	**4. Katman**
2	**Nöron Sayısı**	128	32	128	-
	Çekirdek Boyutu	2	2	2	-
	Aktivasyon Fonksiyonu	Elu	Tanh	Relu	-
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%92,35			
		1. Katman	**2. Katman**	**3. Katman**	**4. Katman**
3	**Nöron Sayısı**	128	64	-	-
	Çekirdek Boyutu	5	4	-	-
	Aktivasyon Fonksiyonu	Relu	Sigmoid	-	-
	İyileştirici	SGD			
	Doğruluk Oranı	%92,22			

7.5. CIFAR-10 Veri Seti Deneysel Sonuçlar

CIFAR-10 veri seti, kullanılan üç veri seti içerisinde eğitimi en zor olan veri setidir. Özellikle veri setindeki görüntülerin karmaşıklığı, görüntü sınıflandırma işlemini zorlaştırmaktadır. Ayrıca MNIST ve Fashion-MNIST'in aksine CIFAR-10'daki görüntüler 32x32'lik boyutta olduğu için, eğitimler daha uzun sürmektedir (Diğer iki veri setinde görüntü boyutları 28x28'dir). Sistem; kabul edilebilir sürelerde başarılı sonuçlar elde etmek amacıyla 10'ar bireyden oluşan 5 nesil ile çalıştırılmıştır.

Sistemin elde ettiği en yüksek doğruluk oranı %82.42'dir. Bu oran, önde gelen çalışmaların gerisinde kalsa da, özellikle çalışma süresi bakımından hızlı olarak kabul edilebilir. Örneğin Cireşan et al (2012)'in önerdiği sistem, %89,79 doğruluk oranıyla çalışırken, önerdikleri sistemin MNIST veri seti üzerinde çalıştırılması yaklaşık 20,41 gün sürmüştür. Suganuma et al (2017)'in önerdiği sistem ise, %94,02 gibi çok yüksek bir başarı oranı ile çalışmasına karşın, bu sistemin de çalışma süresi 13,7 gün olmuştur. Bu çalışmada önerilen sistem ise, en başarılı sonucu 4 saat 28 dakikada elde etmiştir. Şekil 7.6'da sistemin CIFAR-10 veri setinin sınıflandırılması için ürettiği en iyi sonucu veren ağın yapısı gösterilmiştir.



Şekil 7.6 CIFAR-10 veri setinde en başarılı olan ağın modeli

Sistem, CIFAR-10 veri seti ile toplam 10 defa test edilmiştir. Yani toplamda 500 adet evrişimli yapay sinir ağı oluşturulmuş ve bu veri seti ile teste tabi tutulmuştur. Tablo 7.4'te CIFAR-10 veri seti için üretilen en başarılı 3 ağın hiperparametreleri ve doğruluk oranları verilmiştir.

Tablo 7.4 CIFAR-10 veri setinde en başarılı üç ağ

1		**1. Katman**	**2. Katman**	**3. Katman**	**4. Katman**
	Nöron Sayısı	64	128	128	128
	Çekirdek Boyutu	5	5	3	5
	Aktivasyon Fonk.	Tanh	Relu	Sigmoid	Relu
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%82,42			
2		**1. Katman**	**2. Katman**	**3. Katman**	**4. Katman**
	Nöron Sayısı	128	64	128	128
	Çekirdek Boyutu	2	2	2	2
	Aktivasyon Fonk.	Sigmoid	Elu	Tanh	Sigmoid
	İyileştirici	ADAGRAD			
	Doğruluk Oranı	%81,83			
3		**1. Katman**	**2. Katman**	**3. Katman**	**4. Katman**
	Nöron Sayısı	16	64	128	64
	Çekirdek Boyutu	2	2	5	4
	Aktivasyon Fonk.	Elu	Relu	Relu	Tanh
	İyileştirici	ADAM			
	Doğruluk Oranı	%81,66			

Deneysel çalışmanın sonuçları incelendiğinde, rastgele parametrelerle başlayan genetik algoritma optimizasyonunun, nesiller ilerledikçe başarıyı arttırdığı görülmüştür. Tablo 7.5'te üç veri seti için de; en yüksek, en düşük ve ortalama başarı oranları ile, standart sapma oranları verilmiştir.

Tablo 7.5 Veri setlerine göre başarı oranı istatistikleri

	MNIST	**FASHION**	**CIFAR-10**
En Yüksek Başarı	0,9907	0,928	0,8242
En Düşük Başarı	0,3256	0,1003	0,1007
Ortalama Başarı	0,934	0,8591	0,64
Standart Sapma	0,0924	0,071	0,1194

Deneysel çalışmada elde edilen başarı oranları, literatürdeki çalışmalarda elde edilen sonuçlarla kıyaslanabilecek düzeyde iken, sistemin çalışma süreleri de literatürdeki başarılı sonuçlar veren çalışmalara nazaran daha kısadır. Tablo

7.6'da, veri setlerine göre en başarılı çalışmaların başarı oranları ve çalışma süreleri ile geliştirilen sistemin başarı oranı ve çalışma süreleri karşılaştırılmak üzere verilmiştir.

Tablo 7.6 Önerilen sistem ile literatür karşılaştırması

	MNIST	**FASHION**	**CIFAR-10**
En Başarılı Literatür Çalışması	Cireşan et al (2012)	Zhong et al. (2017)	Suganuma et al. (2017)
Literatür Başarı Oranı	0,9977	0,9595	0,9402
Literatür Çalışma Süresi	35 sa.	10 sa. 4 dk.	328 sa. 48 dk.
Önerilen Sistem Başarı Oranı	0,9907	0,928	0,8242
Önerilen Sistem Çalışma Süresi	50 dk.	1 sa. 58 dk.	4 sa. 28 dk.

8. TARTIŞMA VE SONUÇ

Bu tez çalışmasında, evrişimli yapay sinir ağları üzerinde genetik algoritmalar kullanılarak hiperparametre optimizasyonu yapılması ve bu sayede görüntü sınıflandırma işlemi için etkili bir yapay sinir ağı modeli üretilmesi amaçlanmıştır. Bu kapsamda, verilen popülasyon sayısınca rastgele yapay sinir ağları üreten ve en iyi sınıflandırma sonuçlarını veren ağların bir sonraki nesle aktarılması ile bu başarılı ağların çarprazlanarak yeni nesiller üretilmesi yoluyla her yeni nesle başarılı hiperparametrelerin aktarılması sağlanmıştır. Sonuç olarak nesiller ilerledikçe yeni üretilen ağların başarılarının arttığı görülmüş ve kullanılan üç veri setinde de literatürdeki sonuçlarla karşılaştırılabilecek nitelikte sonuçlar elde edilmiştir.

Genetik algoritmalar ile optimizasyonun sağlanmasının yanında, oluşan nesilde başarılı gözükmesine karşın aslında yeteri kadar başarılı olmayan hiperparametrelerin bir sonraki nesle aktarılmasını önlemek amacıyla bir eşik değeri kullanılmıştır. Bu eşik değerinin altında kalan ve bir sonraki nesle seçilmesi hedeflenen her bir ağ yerine, tamamıyla rastgele hiperparametrelerden oluşan yepyeni bir ağ üretilmiş ve bu yeni ağ yeni nesle aktarılmıştır.

Eğitim süresi ve başarı oranıyla doğru orantılı olan bir diğer parametre olan adım sayısı (epoch) için ise, Keras kütüphanesinin EarlyStopping adlı sınıfından faydalanılmıştır. Bu sınıf, iki farklı şekilde kullanılmıştır. İlk önce, genetik algoritmalar yoluyla üretilen ağların eğitim aşamasında bu sınıfın parametreleri daha hızlı sonuçlar elde edilebilecek şekilde ayarlanmıştır. Nesillerin tümünün eğitilmesinin ardından en başarılı beş ağ için bu parametreler, daha uzun eğitime tabi tutulacak şekilde yeniden ayarlanmıştır. Bu sayede Izgara Arama (Grid Search) yöntemine benzer bir yaklaşım benimsenmiştir.

Başarılı ağ kriteri olarak eşik değeri belirlenmesi ve adım sayısı için iki aşamalı bir yöntem kullanılmasıyla, sistemin hem genetik algoritmaların metasezgisel özelliğinden gelen dezavantajlardan kurtulması, hem de çalışma süresinin de optimize edilmesi hedeflenmiştir.

Tez kapsamında yapılan deneysel çalışmalar, Google Colaboratory ortamında gerçekleştirilmiştir. Google'ın ücretsiz olarak sunduğu bu bulut

ortamında maksimum oturum süresi 12 saattir. Dolayısıyla deneysel çalışmalarda verilen nesil ve nesillerdeki birey sayısı değerleri düşük tutulmuş, bu sayede sistemin yarım günden kısa sürede başarılı sonuçlar verebilmesi hedeflenmiştir. İleriki çalışmalarda böyle bir kısıtın olmadığı bir bulut ortamı veya grafik kartı güçlü bir bilgisayarda, daha çok bireye sahip çok nesil ile bu tezde önerilen yöntem tekrar test edilebilir.

Çalışmanın odak noktası genetik algoritmalar ile evrişimli sinir ağlarının hiperparametrelerinin optimizasyonu olup, bu bağlamda oluşturulan yapay sinir ağı modelleri mümkün olduğunca basit tutulmuştur. Bu aynı zamanda çalışma süresinin de kısalmasını sağlamıştır. Yine ileriki çalışmalarda, görüntü sınıflandırma problemi üzerinde daha yüksek sonuçlar elde edebilmek amacıyla oluşturulan ağların modelleri üzerinde geliştirmeler yapılabilir.

Çalışma, paralel programalama ile de geliştirilmeye müsait bir durumdadır. Yukarıda belirtilen nesillerin büyütülmesi ve ağ modellerinin detaylandırılması gibi geliştirmelerin yine bu çalışmadaki gibi kabul edilebilir düzeyde çalışma sürelerine sahip olabilmesi için, her bir bireyin farklı bir bilgisayarda eğitilmesi ve elde edilen sonuçların bir ana bilgisayara (master) gönderilmesi, ardından da yeni neslin üretilip bireylerinin tekrar paylaştırılması yoluyla hem daha başarılı sonuçlar veren, hem de literatürdeki çalışmalara göre çok daha hızlı çalışan sistemler tasarlanabilir.

KAYNAKLAR DİZİNİ

Amidi, A., "*Evrişimli Sinir Ağları El Kitabı*", https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks (Erişim Tarihi: 10 Temmuz 2019)

Bosch, A., Zisserman, A. and Munoz, X., 2007, Image Classification Using Random Forests And Ferns, IEEE 11th International Conference on Computer Vision, Rio De Janeiro, 1-8 p.

Chapelle, O., Haffner P., and Vapnik, V., 1999, Support Vector Machines for Histogram-Based Image Classification, IEEE Transactions on Neural Networks 10.5, 1055-1064 p.

Chollet, F., *"Train a simple CNN-Capsule Network on the CIFAR10 small images dataset"*, https://keras.io/examples/cifar10_cnn_capsule/ (Erişim Tarihi: 16 Temmuz 2019)

Cireşan, D., Meier, U. and Schmidhuber, J., 2012, Multi-Column Deep Neural Networks for Image Classification, ArXiv Preprint, 1202.2745, New York, 20 p.

David, O. E. and Greental, I., 2014, Genetic Algorithms for Evolving Deep Neural Networks, Annual Conference on Genetic and Evolutionary Computation, Vancouver, 1451-1452 p.

Dozat, T., 2016, Incorporating Nesterov Momentum Into Adam, International Conference on Learning Representations, San Juan, 4 p.

El Kessab, B., Daoui, C., Bouikhalene, B., Fakir, M. and Moro, K., 2013, Extraction Method of Handwritten Digit Recognition Tested on the Mnist Database, International Journal of Advanced Science & Technology, 50, Washiongton, 99-110 p.

Feurer, M. and Hutter, F., 2019, Hyperparameter Optimization, Automated Machine Learning, Long Beach, 3-33 p.

Fortuner, B., *"Activation Functions"*, https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html (Erişim Tarihi: 14 Temmuz 2019)

KAYNAKLAR DİZİNİ (devam)

Fred, A. and Agarap, M., 2019, Deep Learning using Rectified Linear Units (ReLU), ArXiv Preprint, 1803.08375, New York, 7p.

Hinton, B., Srivastava, N. and Swersky, K., *"Neural Networks for Machine Learning",* http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (Erişim Tarihi: 14 Temmuz 2019)

Ioffe, S. and Szegedy, C., 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Arxiv Preprint, 1502.03167, 11p.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. and Darrell, T., 2014, Caffe: Convolutional Architecture for Fast Feature Embedding, 22nd ACM international conference on Multimedia, Orlando, 675-678 p.

Keras Team, *"Keras: Deep Learning for humans",* https://github.com/keras-team/keras (Erişim Tarihi: 16 Temmuz 2019)

Krizhevsky, A. and Hinton, G., 2009, Learning Multiple Layers Of Features From Tiny Images, Technical Report University of Toronto, 1.4, Toronto, 7p.

Krizhevsky, A. and Hinton, G., 2010, Convolutional Deep Belief Networks On Cifar-10, University of Toronto, 9p (unpublished).

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998, Gradient-Based Learning Applied to Document Recognition, IEEE, 86(11), Trieste, 2278-2324 p.

LeCun, Y., Yoshua, B. and Hinton, G., 2015, Deep learning, Nature, 521,436-444 p.

Marée, R., Geurts, P., Piater, J., Wehenkel, L., Hong, K. S., and Zhang, Z., 2004, A Generic Approach for Image Classification Based on Decision Tree Ensembles and Local Sub-Windows, 6th Asian Conference on Computer Vision, 2, Jeju, 860-865 p.

KAYNAKLAR DİZİNİ (devam)

McDonnell, M. D. And Vladusich, T., 2015, Enhanced Image Classification with a Fast-Learning Shallow Convolutional Neural Network. International Joint Conference on Neural Networks (IJCNN), Killarney, 1-7 p.

Milgram, J., Sabourin, R. and Cheriet M., 2005, Combining Model-based and Discriminative Approaches in a Modular Two-stage Classification System: Application to Isolated Handwritten Digit Recognition., ELCVIA Electronic Letters on Computer Vision and Image Analysis, 5.2, Barcelona, 1-15.

Miller, J. F., & Harding, S. L., 2008, Cartesian Genetic Programming, 10th Annual Conference Companion on Genetic and Evolutionary Computation, Atlanta, 2701-2726 p.

Ripley, B. D., 1993, Statistical Aspects Of Neural Networks, Networks And Chaos Statistical And Probabilistic Aspects, 50, Boca Raton, 40-123 p.

Ronao, C. A. and Cho, S. B., 2016, Human Activity Recognition with Smartphone Sensors Using Deep Learning Neural Networks, Expert Systems with Applications, 59, 235-244 p.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1985, Learning İnternal Representations By Error Propagation, California Univ San Diego La Jolla Inst for Cognitive Science, 8506, San Diego, 49p.

S.N. Sivanandam and S. N. Deepa., 2007, Introduction to Genetic Algorithms, Springer-Verlag Berlin Heidelberg, Berlin, 442 p.

Snoek, J., Larochelle, H. and Adams, R. P., 2012, Practical Bayesian Optimization Of Machine Learning Algorithms. Neural Information Processing Systems, Stateline, 2951-2959 p.

Springenberg, J. T., Dosovitskiy, A., Brox, T. and Riedmiller, M., 2014, Striving for Simplicity: The All Convolutional Net, ArXiv Preprint, 1412.6806, New York, 14p.

KAYNAKLAR DİZİNİ (devam)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014, Dropout: A Simple Way To Prevent Neural Networks From Overfitting, The Journal Of Machine Learning Research, 15(1), 1929-1958 p.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O. and Clune, J., 2017, Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning, Arxiv Preprint, 1712.06567, New York, 16 p.

Suganuma, M., Shirakawa, S. and Nagao, T., 2017, A Genetic Programming Approach To Designing Convolutional Neural Network Architectures, Genetic and Evolutionary Computation Conference, Berlin, 497-504 p.

Sun, W., Tseng, T. L. B., Zhang, J. and Qian, W., 2017, Enhancing Deep Convolutional Neural Network Scheme for Breast Cancer Diagnosis with Unlabeled Data, Computerized Medical Imaging and Graphics, 57, 4-9 p.

Superdatascience Team, *"Convolutional Neural Networks (CNN): Step 3 - Flattening",* https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening (Erişim Tarihi: 12 Temmuz 2019)

Xiao, H., Rasul, K. and Vollgraf, R., 2017, Fashion-MNIST: A Novel Image Dataset For Benchmarking Machine Learning Algorithms, ArXiv Preprint, 1708.07747, New York, 6 p.

Zhong, Z., Zheng, L., Kang, G., Li, S. and Yang, Y., 2017, Random Erasing Data Augmentation, Arxiv Preprint 1708.04896, New York, 10 p.

TEŞEKKÜR

Bu çalışma süresince danışmanlığımı yapan, bilgi ve birikimleriyle yol gösteren ve yardımlarını esirgemeyen Sayın Prof. Dr. M. Serdar Korukoğlu'na teşekkürü bir borç bilirim. Ayrıca her zaman maddi ve manevi desteğini yanımda hissettiğim, eğitim hayatım boyunca da bana devamlı yardımcı olan değerli eşim Sümeyra Nur Altan'a teşekkür ederim.

20/08/2019

Emre Altan

ÖZGEÇMİŞ

Emre Altan, 11 Temmuz 1991 tarihinde İzmir'de doğmuştur ve Türkiye Cumhuriyeti vatandaşıdır. 2009 yılında başladığı İstanbul Ticaret Üniveristesi Bilgisayar Mühendisliği lisans programından 2013 yılında mezun olan Emre Altan, yaklaşık 5 senedir özel sektörde faaliyet gösteren yazılım firmalarında yazılım uzmanı olarak çalışmaktadır.