MARMARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ADAPTİF HİBRİT SİNİR AĞLARI

MAHMUT ÇAKAR

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Yüksek Lisans Programı

DANIŞMAN

Doç. Dr. Kazım Yıldız

EŞ-DANIŞMAN

Dr. Öğr. Üyesi Yakup Genç

İSTANBUL, 2022

MARMARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ADAPTİF HİBRİT SİNİR AĞLARI

Mahmut Çakar

523619011

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Yüksek Lisans Programı

DANIŞMAN

Doç. Dr. Kazım Yıldız

EŞ-DANIŞMAN

Dr. Öğr. Üyesi Yakup Genç

İSTANBUL, 2022

MARMARA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Marmara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Öğrencisi Mahmut Çakar'ın "Çoklu Adaptif Hibrit Sinir Ağları" başlıklı tez çalışması, 7 Haziran 2022 tarihinde savunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Doç.Dr. Kazım YILDIZ (Danışman)

Marmara Üniversitesi ..... (İMZA).....

Doç.Dr. Önder DEMİR (Üye)

Marmara Üniversitesi ..... (İMZA).....

Doç. Dr. Tevfik Aytekin (Üye)

Bahçeşehir Üniversitesi ..... (İMZA).....

ONAY

Marmara Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararı ile Mahmut Çakar'ın Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında Yüksek Lisans/Doktora derecesi alması onanmıştır.

Fen Bilimleri Enstitüsü Müdürü

Prof. Dr. Bülent Ekici

ÖNSÖZ

Tez çalışmamın araştırılmasında ve yürütülmesinde her daim destek olan, her fırsatta yardımını esirgemeyen değerli danışmanım Doç.Dr. Kazım Yıldız'a, eş danışmanım Dr. Öğr. Üyesi Yakup Genç'e ve her zaman yanımda olan arkadaşım Filiz Dalıp'a ve FYL-2021-10352 numaralı projeye desteklerinden dolayı Marmara Üniversitesi Bilimsel Araştırma Projeleri Birimi'ne

Teşekkürlerimi sunarım.

HAZİRAN, 2022 **Mahmut Çakar**

İÇİNDEKİLER

SAYFA

ÖNSÖZ **i**

İÇİNDEKİLER **ii**

ÖZET **v**

ABSTRACT **vi**

SEMBOLLER **vii**

KISALTMALAR **viii**

ŞEKİL LİSTESİ **ix**

TABLO LİSTESİ **x**

1. GİRİŞ **1**

1.1. Genel Bakış **1**

1.1.1. Problemin Tanımı 2

1.1.2. Amaç ve Hedef 2

1.1.3. Ana Katkılar 2

1.1.4. Tez Organizasyonu 2

1.2. Derin Öğrenme **3**

1.2.1. Derin Öğrenme ile Makine Öğrenmesi Arasındaki Fark 4

1.2.2. Derin Öğrenme Ne Zaman Kullanılmalı 5

1.2.3. Derin Öğrenmede Karşılaşılabilecek Sorunlar ve Zorluklar 5

1.3. Evrişimsel Sinir Ağları **6**

1.3.1. Katmanlar 6

1.3.1.1. Konvolüsyon 6

1.3.1.2. Ortaklama 7

1.3.1.3. Yığın Normalizasyonu 8

1.3.1.4. Aktivasyon Fonksiyonları 8

1.3.1.5. Düzleştirme 9

1.3.1.5. Tam Bağlamalı Katmanlar 9

1.3.2. Evrişimsel Sinir Ağları Öğrenme Hususları 9

1.3.2.1. İleri ve Geri Yayılım 9

1.3.2.2. Kayıp Fonksiyonları 10
1.3.2.3. Optimizasyon Algoritmaları 10
1.3.3. Evrişimsel Sinir Ağları Kullanılan Terimler 11
1.3.4. Aşırı Uyum (Over-Fitting) 11
1.4. Klasik Evrişimsel Sinir Ağları 12
1.4.1. LeNet 12
1.4.2. AlexNet 12
1.4.3. VGGNet 13
1.4.4. ResNet 14
1.4.5. GoogleNet 14
1.5. Derin Sinir Ağları Araçları ve Kütüphaneleri 15
1.5.1. CUDA (Birleşik Cihaz Mimarisi Hesaplayıcısı) 15
1.5.2. cuDNN (CUDA Derin Sinir Ağları Kütüphanesi) 16
1.5.3. Tensorflow 16
1.5.4. Keras 17
1.5.5. Torch 17
1.5.6. Theano 18
1.6. Literatürde Yapılmış Çalışmalar 18
2. MATERYAL VE YÖNTEM 21
2.1. Hazır Veri Kümeleri 21
2.1.1. MNIST Veri Kümesi 21
2.1.2. Fashion MNIST Veri Kümesi 21
2.1.3. CIFAR10 ve CIFAR100 Veri Kümeleri 22
2.1.3. STL10 Veri Kümesi 23
2.2. Seçim Katmanı 23
2.3. Aday Modüller 25
2.3.1. VGGNet Aday Modülü 25
2.3.2. ResNet Aday Modülü 26
2.3.3. Inception Aday Modülü 27
2.4. Veri Önişleme 28
2.5. Derin Öğrenme Mimarisi 29

2.5.1. Özellik Çıkarma Katmanları 29
2.5.2. Seçim Katmanları 30
2.5.3. Tam Bağlantı Katmanları 30
3. BULGULAR VE TARTIŞMA 32
3.1. Deney Ortamı 32
3.2. Performans Metrikleri 32
3.3. Deney Sonuçları 33
3.4. Diğer çalışmalarla olan karşılaştırmalar 38
SONUÇLAR 40
KAYNAKLAR 42
ÖZGEÇMİŞ

ÖZET

ÇOKLU ADAPTİF HİBRİT SİNİR AĞLARI

Evrişimsel sinir ağları, özellikle bilgisayarla görme problemlerini oldukça başarılı bir şekilde çözmekte ve her geçen gün de yeni gelişen mimariler ile daha iyi sonuçlar vermeye devam etmektedir. Hem geliştirilen yeni mimariler ile gelişmekte ama bu mimarilerin de birbirlerinin karışımıyla oluşturulan hibrit modeller de bazı problemlere oldukça iyi sonuçlar da vermektedir. Bu çalışmada farklı mimarilerden sentezlenen modüllerin karışımıyla bir hibrit model oluşturulmuştur. Ancak farklı mimariler kolektif öğrenmede olduğu gibi her modüle eğitilebilir ağırlık katsayısı eklenmektedir. Böylelikle, düşük ağırlıklı modüllerin elenerek model için en uygun modüllerin seçilmesi hedeflenmiştir. Bu modüller farklı ağ mimarilerinde kullanılan yapılardan oluşmaktadır. Bu modüllerin toplanıp seçilmesinin yapıldığı bir katmandan oluşmaktadır. Başlangıçta eşit olan bu ağırlıklar bir eğitim sürecinden sonra incelenerek, önceden belirlenmiş bir eşik değerin altında ise o modül modelden çıkarılmış veya bir başka önceden belirlenmiş eşik değerinin üstünde ise de o modül korundu ve diğer modüller modelden çıkarılmıştır. Bu öğrenme yönetimi kolektif öğrenme yapısına benzemektedir ancak bu çalışmada katsayılar sonradan modelin seçilmesini ya da elenmesini sağlayacaktır. Modüller seçildikten sonra model eğitilmeye devam edilerek başarımı koruyarak model küçültülmüştür. Farklı modüller kullanılarak dinamik bir hibrit modül oluşturulması amaçlanmıştır.

Hata ayıklama için kullanılan MNIST ve Fashion MNIST veri kümelerinde sırasıyla %99,09 ve %99,02 doğruluk değerleri elde edilmiştir. Kalite testlerinde kullanılan CIFAR10 veri kümesinde %87,86 doğruluğa ulaşmışken, CIFAR100 veri kümesinde de %54,11 doğruluk sonuçlarına ulaşılmıştır.

HAZİRAN, 2022 **Mahmut Çakar**

ABSTRACT

MULTI ADAPTIVE HYBRID NETWORK

Convolutional neural networks, especially with computer vision problems quite successfully and continue to give better results with newly developing architectures day by day. It develops with the developed architectures, but hybrid models created by mixing these architectures with each other also gave very good results for some problems. In this study, a hybrid model was created with a mixture of modules used from different architectures. However, as in collective learning of different architectures, a trainable weight coefficient is added to each module. Thus, it was aimed to select the most suitable modules for the model by eliminating the low-weight modules. These modules consist of structures used in different network architectures. It consists of a selection layer where these modules are collected and selected. These initially equal weights were examined after a training process, and if it was below a predetermined threshold, that module was removed from the model, or if it was above another predetermined threshold, that module was retained, and other modules were excluded from the model. This learning management is similar to the collective learning structure, but in this study, the coefficients will enable the model to be selected or eliminated later. After the modules were eliminated, the model continued to be trained and the model was shrunk while maintaining its performance. It is aimed to create a dynamic hybrid module by using different modules.

MNIST and Fashion MNIST datasets used in debugging tests reached 99.09 % and 99.02 % accuracy respectively. CIFAR10 and CIFAR100 datasets used in benchmark tests also reached 87.86 % and 54.11 % accuracy respectively.

JUNE, 2022 **Mahmut Çakar**

SEMBOLLER

$\boldsymbol{x}$	:	Girdi vektörü
$\boldsymbol{w}$	:	Nöronun ağırlık vektörü
$\boldsymbol{b}$	:	Nöronun sapması
max	:	En fazlası
$\boldsymbol{e}$	:	Euler sayısı
$\sum$	:	Toplam
$\boldsymbol{q}$	:	Tahmin edilen olasılık
$\boldsymbol{p}$	:	Gerçek olasılık
$\boldsymbol{L}$	:	Kayıp fonksiyonu
$\boldsymbol{log}$	:	Doğal logaritma
$\boldsymbol{m_t}$	:	Birinci moment gradyanları
$\boldsymbol{\nabla g_t}$	:	Gradyanlar
$\boldsymbol{t}$	:	Adım sayısı
$\boldsymbol{v_t}$	:	İkinci moment gradyanları
$\boldsymbol{\lambda}$	:	Adım boyutu
$\boldsymbol{W_t}$	:	Ağırlık vektörü
$\boldsymbol{\Omega}$	:	Üssel bozulma oranı

KISALTMALAR

ADAM	**:**	Uyarlanabilir moment tahmini
CIFAR	**:**	Canadian Institute for Advanced Research geliştirdiği veri kümesi
CPU	**:**	Merkezi işlem birimi
cuDNN	**:**	CUDA derin sinir ağları kütüphanesi
GPU	**:**	Grafik işlem birimi
MAHNet	**:**	Multi Adaptive Hybrid Network
MOH	**:**	Mutlak ortalama hatası
ReLU	**:**	Düzeltilmiş lineer birim
OKH	**:**	Ortalama kare hatası
TPU	**:**	Tensör işlem birimi
VGGNet	**:**	Visual Geometry Group tarafından geliştirilen mimari

ŞEKİL LİSTESİ

SAYFA

Şekil 1.1. Basit sinir ağları ve derin öğrenme sinir ağları ……..……..…… 3

Şekil 1.2. Derin öğrenmedeki nöron yapısı ……..…….…….…... 4

Şekil 1.3. Evrişimsel sinir ağları yapısı …..……..…… 6

Şekil 1.4. İki boyutta konvolüsyon işlemi örneği …….……..... 7

Şekil 1.5. Ortaklama işlemi örneği ...…….……... 7

Şekil 1.6. Yığın normalizasyonunun modelin öğrenmesine etkisi …….……..…… 8

Şekil 1.7. LeNet mimarisi ...…… 12

Şekil 1.8. AlexNet mimarisi ...…….... 13

Şekil 1.9. VGGNet mimarisi ...……...…….……...… 13

Şekil 1.10. ResNet mimarisi ...……...……..... 14

Şekil 1.11. GoogleNet mimarisi …...…….….. 15

Şekil 1.12. Inception modülü ...…….…. 15

Şekil 2.1. MNIST veri kümesinden örnek veriler …….…….…… 21

Şekil 2.2. Fashion MNIST veri kümesinden örnek veriler ...…….…….…… 22

Şekil 2.3. CIFAR veri kümelerinden örnek veriler ...…….…….…… 22

Şekil 2.4. STL10 veri kümesinden örnek veriler ..…….…….…… 23

Şekil 2.5. Seçim katmanı ……..…….…….…… 24

Şekil 2.6. VGGNet aday modülü ……..…….……..…..…….…... 26

Şekil 2.7. ResNet aday modülü ……..…….…… 27

Şekil 2.8. Inception aday modülü ……...…….……...… 28

Şekil 2.9. Veri artırımı yöntemleri ……..……..…….……. 29

Şekil 2.10. Model mimarisi ……..…..…..…….…… 30

Şekil 3.1. CIFAR10 veri kümesinde modelin doğruluk ve kayıp grafiği ……...…… 35

TABLO LİSTESİ

SAYFA

Tablo 1.1. Derin öğrenme araçlarının karşılaştırılması 17

Tablo 2.1. CIFAR veri kümesi etiketleri 23

Tablo 3.1. Karmaşıklık matrisi 32

Tablo 3.2. MAHNet ilerleme aşamaları ve yapılan deneyler 33

Tablo 3.3. MAHNet'in veri kümelerinde final sonuçları 34

Tablo 3.4. CIFAR10 veri kümesinde sınıfa göre sonuçlar 36

Tablo 3.5. CIFAR100 veri kümesinde sınıfa göre sonuçlar 36

Tablo 3.6. MAHNet ve diğer modellerin CIFAR veri kümeleri üzerindeki başarım karşılaştırması 39

1. GİRİŞ

Evrişimsel sinir ağları, son yıllarda nesne sınıflandırmada ve nesne tespitinde hem hız olarak hem de doğrulukta oldukça başarılıdır. Ancak modelin mimarisini seçerken kararsız kalınmakta ve mevcut veri kümesinde uygun mimari seçiminde çok fazla vakit kaybedilmektedir. Bu vakit kaybı, farklı modellerin mevcut veri kümesinde tekrar tekrar eğitilmesi zaten normalde bir adet eğitim bile oldukça uzunken, oldukça zahmetlidir. Bu çalışmada oluşturulan model aynı giriş ve çıkış katmanına sahip modüller paralel olarak eğitilip uygun modüllerin belirlenmesi şeklinde eğitilmesi amaçlanmıştır. Böylelikle, model için en uygun modüllerin seçilmesi hedeflenmiştir. Bu modüller farklı ağ mimarilerinde kullanılan yapılardan oluşacaktır. Oxford Üniversitesi Visual Geometry Group'un geliştirdiği VGGNet mimarisinde ikinin kuvvetleri kadar filtre sayısından oluşan ve seri bir şekilde bağlı konvolüsyon katmanları [1], ResNet'de kullanılan konvolüsyon katmanlarında önceki katmanlardan çıkan değerlerin eklenen oluştuğu katmanlar [2] ya da GoogleNET'de kullanılan Inception modüller [3] bu modelde modül olarak kullanılmıştır. En uygun modülün seçilebilmesi için her bir modülün [0,1] aralığında eğitilebilir bir ağırlık değişkeni tanımlanmıştır ve diğer katmanların değişkenleri ile başlangıçta eşit ve toplamları 1 olacaktır. Yeni bir katman oluşturularak bu modüller ağırlıklarla çarpıldıktan sonra diğer modüllerin çıktılarıyla toplanmıştır. Bu ağırlıklar belirli bir epoch sayısından sonra incelendikten sonra, bu katsayılar önceden belirlenmiş bir eşik değerin altında ise o modül modelden çıkarılmıştır. Başlangıç olarak ağırlıklar eşit olacaktır ve sonrasında eğitimden sonra bu değerler incelenerek 0'a yakın olan modül elenecektir çünkü o ağırlık o katmanın çıkış değerlerini azaltacak ve modele katkısı azalacaktır. Bu öğrenme yönetimi kolektif öğrenme yapısına benzemektedir ancak bu çalışmada katsayılar sonradan modelin seçilmesini ya da elenmesini sağlayacaktır. Modüller elendikten sonra model eğitilmeye devam edilerek başarımı koruyarak model küçültülmüştür. Farklı modüller kullanılarak dinamik bir hibrit modül oluşturulması amaçlanmıştır.

1.1. Genel Bakış

Oluşturulan Multi Adaptive Hybrid Network (MAHNet) evrişimsel sinir ağları modeli, farklı mimarilerin dinamik bir şekilde çıkarılmasıyla farklı veri kümesi problemlerine adaptif bir şekilde uyum sağlaması amaçlanan hibrit bir modeldir [4].

1.1.1. Problemin tanımı

Son yıllarda farklı yapılarda evrişimsel sinir ağları mimarileri oluşturulmuştur. Ancak, bazı problemleri çözmek için bu mimariler ile kendi hibrit modelini oluşturmak daha iyi sonuçlar da vermektedir. Bu yüzden bu mimarilerin dinamik bir şekilde oluştuğu model ile adaptif bir model oluşturulması amaçlanmıştır.

1.1.2. Amaç ve hedef

Bu çalışmada planlanan model en uygun modülleri seçebilmek için aynı giriş ve çıkış katmanına sahip modüller paralel olarak eğitilip uygun modüllerin belirlenmesi şeklinde eğitilmiştir. Bu aday modüller, farklı mimarilerinde kullanılan yapılardan sentezlenmiştir. Her modülün başlangıçta eşit olacak şekilde [0,1] aralığında ve toplamları 1 olan eğitilebilir ağırlık katsayıları eklenmiştir.

Seçim katmanında en uygun modülün seçilmesi bu ağırlığın büyüklüğüne göre karar verilmesi amaçlanmıştır. Bu ağırlıklar, belirlenmiş bir eşik değerinden düşükse o modül modelden çıkarılacak ya da bir başka eşik değerinin üstünde ise de o modül tutularak diğer modüller modelden çıkarılacaktır. Çünkü ağırlık o katmanın çıkış değerlerini azaltacak ve modele katkısı azalacaktır. Kolektif öğrenme yapısına benzeyen bu öğrenme metodu modelin seçilmesini ya da elenmesini sağlayacaktır. Modüller elendikten sonra model eğitilmeye devam edilmiş ve model küçültülmüştür. Bu da model seçimini kolaylaştırmasını hedeflenerek ve farklı modülleri birleştirerek hibrit bir model oluşturulması sağlanacaktır. Hibrit modelin oluşmasıyla da adaptif bir modelin oluşması amaçlanmıştır. Farklı modüller kullanılarak dinamik bir hibrit modül oluşturulacaktır.

1.1.3. Ana katkılar

Önerilen derin öğrenme modeliyle yaygın olarak kullanılan hazır veri kümeleri üzerinde eğitim yapılmış ve diğer yapılmış çalışmalardaki yöntemlerle kıyaslandırılmıştır. Python diliyle hazırlanmış model adaptif ve dinamiktir. Ayrıca derin öğrenme modelinde farklı bir kolektif öğrenme anlayışı denenmiştir.

1.1.4. Tez organizasyonu

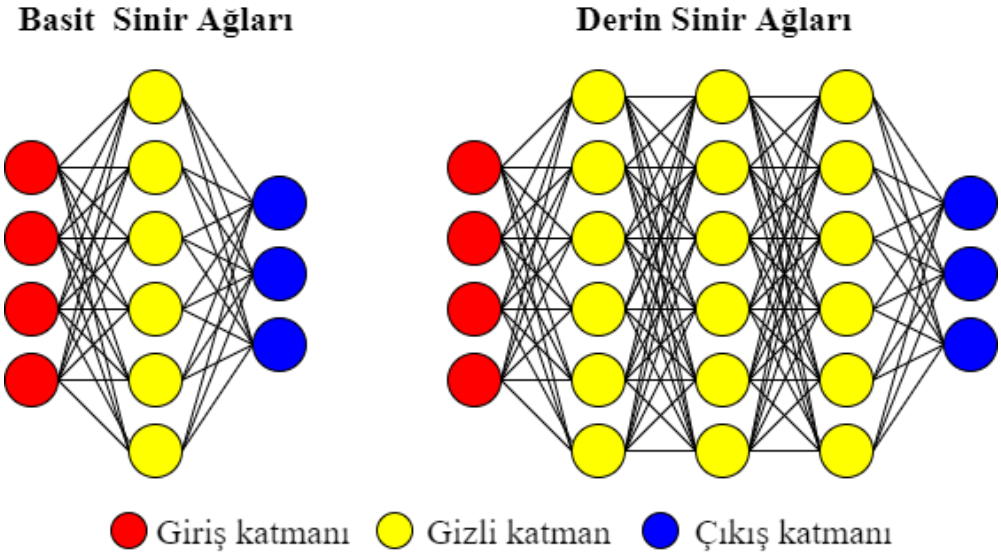
Tez dört bölümden oluşmaktadır. “Giriş” bölümünde Derin öğrenme ile ilgili genel bilgilere, evrişimsel sinir ağlarının yapısına, evrişimsel sinir ağları mimarilerine, derin

öğrenme araçlarına ve literatür çalışmalarına yer verilmiştir. İkinci bölümde "Materyal ve Yöntem" bölümünde kullanılan veri kümelerinde ve önerilen model ve mimarisinden bahsedilmiştir. Üçüncü olarak "Bulgular ve Tartışma" bölümünde deney ortamı, deney performansı ve sonuçları anlatılmıştır. Son bölümde "Sonuçlar" bölümünde ise çalışma genel olarak özetlenerek sonuca bağlanılmış ve gelecekte nasıl çalışmalar yapılacağı yer almaktadır.

1.2. Derin Öğrenme

Derin öğrenme, matematik hesaplamalarla eğitilebilen nöronlardan ve bu nöronların oluşturduğu katmanların ve gizli katmanların bulunduğu bir makine öğrenmesi türüdür. Nesne tanıma, konuşma tanıma, doğal dil işleme gibi birçok karmaşık problemi diğer makine öğrenmesi metotlarına göre daha iyi çözmektedir.

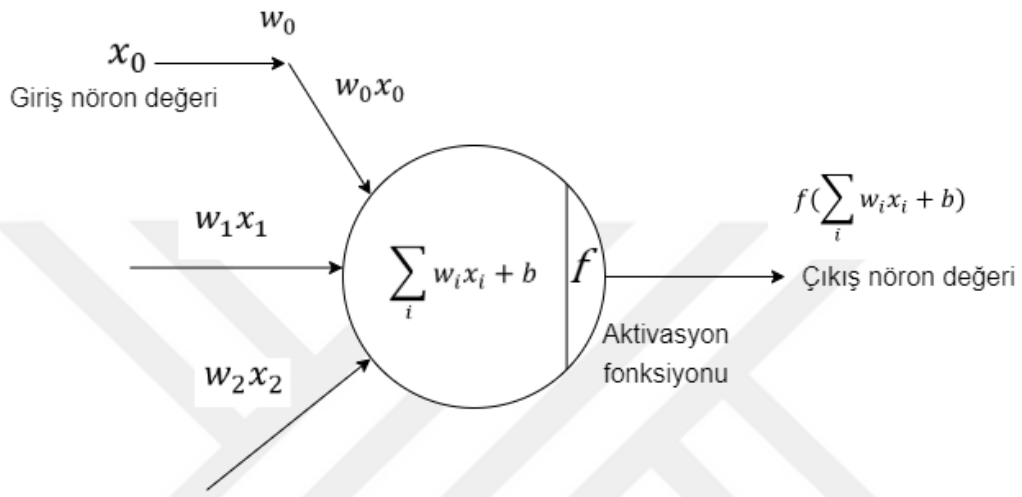
Çok katmanlı yapay sinir ağlarının daha verimli eğitilebileceğini 2006 yılında derin öğrenme kavramını da tanımlayan Hinton'dur [5]. Çok katmanlılardan kasıt görselde Şekil 1.1 de görüldüğü gibi giriş ve çıkış katmanların arasında oluşturulan gizli katmanlardır. Bu gizli katmanlar özellik çıkarmayı sağlamaktadır. Örneğin, bir resimde yüz olduğunu anlamak için bu gizli katmanlar; göz, ağız ve burun gibi anlamları çıkarabilir.



Şekil 1.1. Basit sinir ağları ve derin öğrenme sinir ağları.

Bu katmanlar, insanda bulunan sinir sistemlerinden esinlenerek tanımlanan nöronlardan oluşmaktadır. Bu nöronlar, Şekil 1.2'de de görüldüğü gibi eğitilebilir ağılıklıklar ve

sapmalardan oluşmaktadır. Bu nöronların çıktıları, girdilerin ağırlıklar (x_i) ile nöron ağırlıklarıyla çarpılıp (w_i) ve nöron sapmasının (b) eklenmesiyle oluşan değerlerin toplamından oluşur. Bu çıktıları doğrusal olmayan aktivasyon fonksiyonları uygulanarak karmaşıklık seviyesi artırılır. Böylece görüntü veya ses işleme gibi birçok karmaşık problemi çözmekte kullanılması sağlar. Bu nöronlardaki ağırlıkların doğruluğunu artırmak için geriye yayılım algoritması uygulanır.



Şekil 1.2. Derin öğrenmedeki nöron yapısı [6].

Makine öğrenmesi çok fazla eğitilebilir parametre içerdiğinden dolayı çok daha karmaşık problemleri çözerken, çok fazla eğitilebilir parametrenin eğitilmesinin zorluğundan dolayı ancak grafik işlem birimlerinin (graphics processing units-GPU) güçlenmesi ve birçok kütüphanenin gelişmesiyle mümkün kılmıştır. Grafik işlem birimlerinin, mevcut işlemcilere göre çok daha fazla işlem birimi olduğundan dolayı paralel hesaplamayı daha kısa sürede yapabilmektedir.

1.2.1. Derin Öğrenme ile Makine Öğrenmesi Arasındaki Fark

Derin öğrenme genel olarak makine öğrenmesinin bir alt kümesi olarak tanımlanır. Ancak, daha doğru tanımı derin öğrenme makine öğrenmesi yöntemlerinin evrimleşmesidir. Makine öğrenmesi, bilgisayarların veriye göre öğrenmesi olarak tanımlanabilir. Derin öğrenme de aynı şekilde veriye göre bilgisayar öğrenmesinin gizli katmanlar sayesinde daha karmaşık problemleri çözen bir makine öğrenme yöntemi olarak tanımlanabilir.

Derin öğrenmeyi makine öğrenmesinden ayıran etkenleri ilk ve en göze çarpan özelliği; makine öğrenmesinde kullanılan basit yapıdaki karar ağaçları ya da lineer regresyondan ziyade insan beyninden ilham alınarak oluşturulan karmaşın nöronların çok katmanlı yapısı olarak gösterilebilir. Buna ek olarak da makine öğrenmesi insan müdahalesine daha açıktır ancak derin öğrenme özellik çıkarmayı otomatik olarak yapar ve kendi hatalarından öğrenir. Son olarak da derin öğrenme çok katmanlı ve karmaşık yapısından dolayı daha çok veriye ihtiyaç duymaktadır. Makine öğrenmesi ise çok daha az veriyle öğretilebilir.

1.2.2 Derin Öğrenme Ne Zaman Kullanılmalı

Çok daha karmaşık problemlerde ya da çok fazla verinin olduğu problemlerden derin öğrenme kullanılmalıdır. Derin öğrenmenin çok katmanlı yapısı sayesinde daha karmaşık sorunları çözer. Örneğin, sadece trafik ışığı tespit etmek için kırmızı, sarı ve yeşil filtrelerden geçirilerek yuvarlak objelerin trafik ışığı olduğu varsayılabilir. Ancak bu bir otomatik sürüşlü araba için yeterli olmayabilir. Başka bir aracın yuvarlak far ışıkları ile karıştırabilir ve araba ayrıca başka bir arabayı tespit etmesi gerekirse ve yayaları tespit etmesini gerekirse makine öğrenme yöntemleri eksik kalabilir. Bu yüzden derin öğrenme kullanılması gereklidir.

1.2.3 Derin Öğrenmede Karşılaşılabilecek Sorunlar ve Zorluklar

Derin öğrenme karmaşık problemleri çözmede başarılı olsa da bir derin öğrenme modeli geliştirirken birçok problemle de karşılaşılabilir. Bu problemler şöyle sıralanabilir;

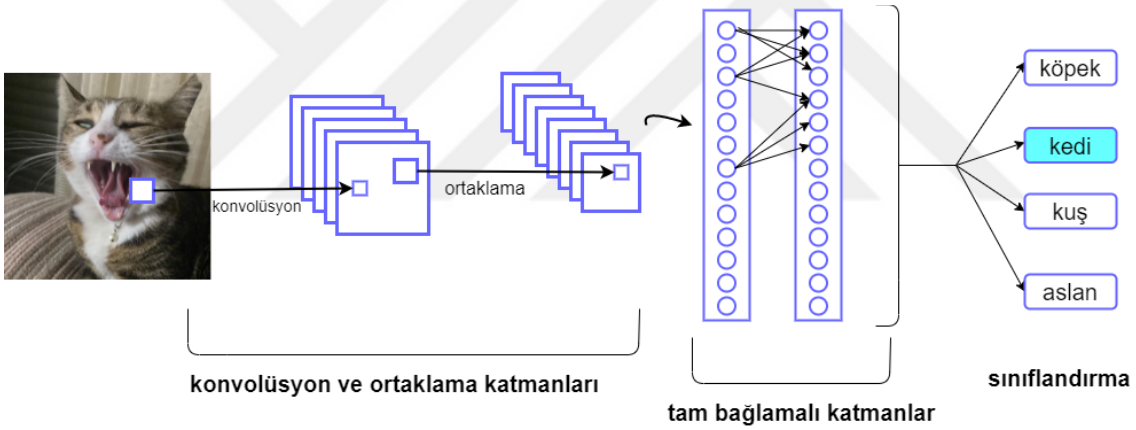
- **Veri açlığı**: Her durumu modele iletmek mümkün değildir. Bunu engellemek için daha genel görüntüler ile verinin çeşitliliğini artırmak gerekmektedir. Bu yüzden olabildiğince fazla veri toplamak ve veri büyütme teknikleri uygulanmalıdır [7].
- **Öğrenmedeki sığlık:** Derin öğrenme problemleri çözmede başarılı olsa da bir insan gibi anlaması günümüzde hala yeterli değildir. Bir objeyi tanıması için bile yüzlerce fotoğrafı onlarca defa modele vermek yeterli olmayabilir [6]. Kedi ve köpeği ayırt edebilmesi, yetişkin bir insan için birkaç fotoğraf yeterli olsa da derin öğrenme için binlerce fotoğrafı modele vermek gerekirken, farklı kedi türlerini öğrenmesi bile oldukça uzun bir süreç gerekmektedir.

- **Opaklığı:** Derin öğrenme modelleri kapalı bir kutuya benzetilebilir. Ne kadar nöronların ağırlıklarını görebilsek de bu nöronların ya da katmanların amaçlarını anlamak zordur. Bu yüzden bir model başarısız olduğunda bunun nedenini analiz etmek ve çözüm bulmak oldukça zordur.

1.3. Evrişimsel Sinir Ağları

Evrişimsel sinir ağları, konvolüsyon işleminin uygulandığı bir derin öğrenme metotudur. Evrişimsel sinir ağları genellikle görüntülerde nesneleri tanıma ve tespitinde kullanılır.

Şekil 1.3'te de görüldüğü gibi görseldeki nesneyi tanımlamak için konvolüsyonlarla kenar ve köşe çıkarma gibi öznitelik çıkarımı yapılır. Ortakalama katmanları sayesinde de boyut küçültülürken sadece önemli değerlerin saklanmasını sağlayan bir örnekleme işlemidir. Son olarak da tam bağlantı katmanları ile sınıf skorlarını optimize etmekte kullanılır [6].



Şekil 1.3. Evrişimsel sinir ağları yapısı.

1.3.1. Katmanlar

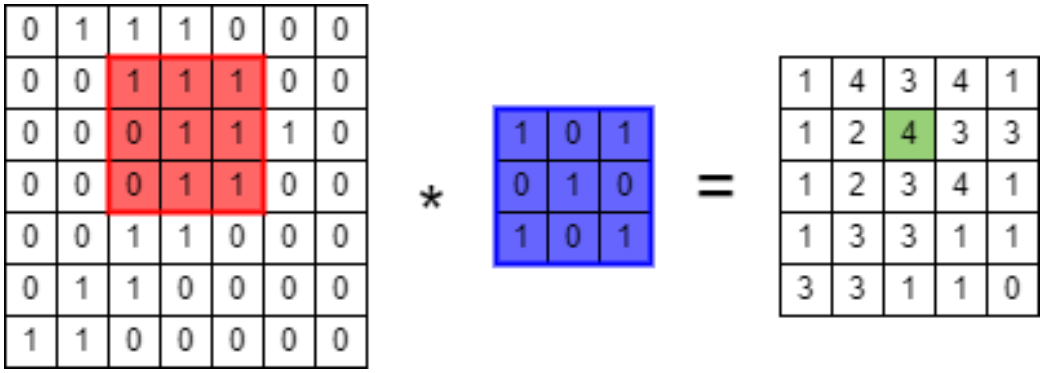
Katmanlar bölümünde evrişimsel sinir ağlarında kullanılan yapılara yer verilmiş ve açıklanmıştır.

1.3.1.1 Konvolüsyon

Filtreleme olarak da tanımlanan konvolüsyon işlemi, iki sinyali birbirinin üzerinde kaydırılarak altında kalan alan olarak tanımlanabilir [8]. İki boyutlu iki ayrık sinyalin konvolüsyonu Denklem 1.1'de görülmektedir. x, y sinyallerin giriş değişkenleridir.

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2].g[x-n_1, y-n_2] \quad (1.1)$$

Bir iki boyutlu görselin bir filtre ile konvolüsyon işlemi Şekil 1.4'te de görüldüğü gibi giriş resmine filtre kaydırılarak değerler çarpımının toplamıyla (1.1) uygulanır ve çıktı üretilir.



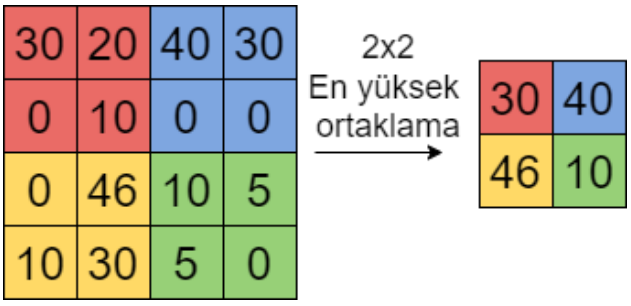
Şekil 1.4. İki boyutta konvolüsyon işlemi örneği.

Konvolüsyon işleminde özenle seçilmesi gereken parametreler;

- **Filtre sayısı;** girişe kaç adet filtre kullanılacağını belirler.
- **Kernel boyutu;** filtrenin boyutunu belirler
- **Adımlar;** filtrenin girişe uygulanırken kaydırılacak adım boyutunu belirler
- **Dolgu;** filtreleme işlemi uygulanırken boyut küçültmesini engellemek için girişin çevre eklenen dolguya denir. Sıfır ya da aynı değerler ile dolgu uygulanabilmektedir.
- **Aktivasyon;** filtrenin çıktıya uygulanacak aktivasyon fonksiyonunu belirler.

1.3.1.2 Ortaklama

Ortaklama katmanı, girdideki değerleri örnekleme yaparak değerli bilgileri çıkarmayı amaçlar [9]. Ortalama ortaklama ile değerlerin ortalaması alınırken, en fazla ortaklama ile de en yüksek değeri alır. Şekil 1.5'te girişe 2x2'lik en fazla ortaklaması uygulanmıştır.

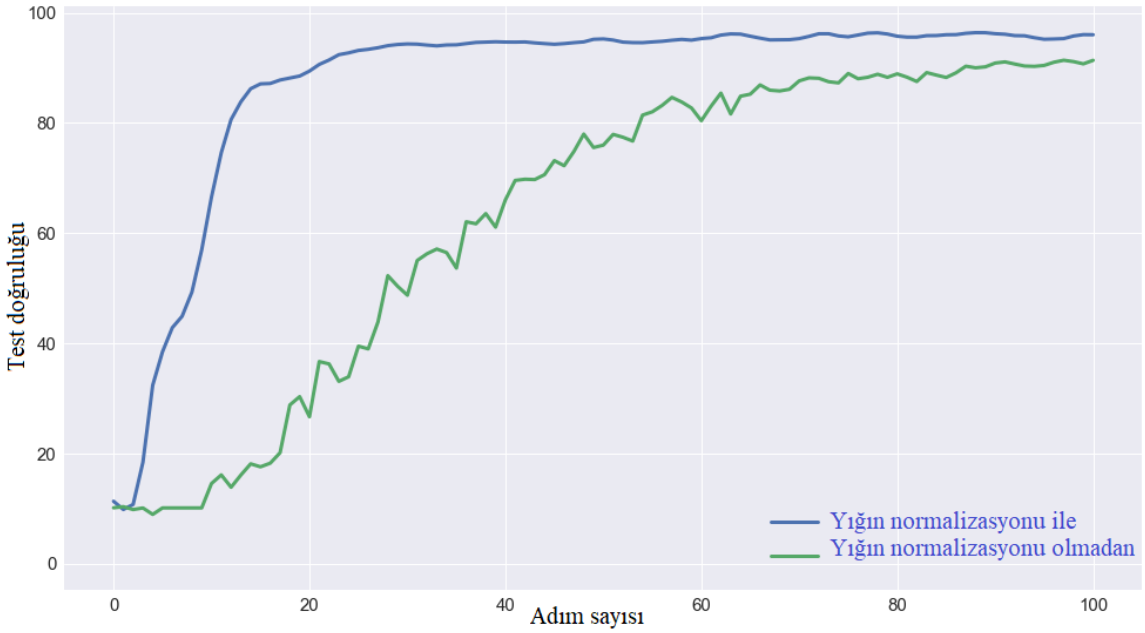


Şekil 1.5. Ortaklama işlemi örneği.

Ortaklama'da da adımlar ve dolgu parametreleri bulunur ve konvolüsyondaki gibi işler. Bunlara ek olarak ortaklama boyutu bulunur ve ne kadar alanda ortaklama işleneceği belirlenir. Bu ortakalama boyutuna göre görsel o oranda küçülür.

1.3.1.3 Yığın Normalizasyonu

Yığın normalizasyonu, ağdaki değerleri daha düzenli hale getiren bir normalizasyon yöntemidir. Ayrıca, eğitim sırasında yok olma gradyanını direnç de verdiğinden görselde de görüldüğü gibi eğitim süresini kısaltır ve daha stabil bir eğitim süreci sağlar. Şekil 1.6'da mavi çizgi ile gösterilen yığın normalizasyonunun kullanılıldığı modelin hem daha hızlı öğrenmesine hem de daha yumuşak bir ilerleme sağladığı görülmektedir.



Şekil 1.6. Yığın normalizasyonunun modelin öğrenmesine etkisi [10].

1.3.1.4 Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, evrişimsel sinir ağlarında çok sık kullanılan doğrusal olmayan fonksiyonlardır. x giriş değerlerini, istenmeyen değerleri sönümlemek ya da belirli aralıklara dönüştürmek için kullanılır [11]. Bu aktivasyon fonksiyonlarından bazıları:

$$ReLU(x) = \max(0, x) \tag{1.2}$$

$$SızdıranReLU(x) = \max(0.1x, x) \tag{1.3}$$

$$sigmoid(x) = \frac{1}{e^{-x}+1} \tag{1.4}$$

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \quad (1.5)$$

Düzeltilmiş Lineer birim (ReLU) aktivasyon fonksiyonu (1.2) negatif değerleri sönümler. Sızdıran ReLU aktivasyon fonksiyonu (1.3) ise negatif değerleri onda bir değerlerine düşürelek tamamen sönümlemez ama yumuşatır. Sigmoid fonksiyon (1.4), değerleri 0 ile 1 arasına sınırlayan ve türevlenebilir bir fonksiyon olması sayesinde geri yayılım metotlarına kolaylık sağlayan doğrusal olmayan aktivasyon fonksiyonudur. Denklem 1.5'te hiperbolik tanjant aktivasyon fonksiyonu (tanh) ise giriş değerinin hiperbolik tanjantı alır ve sigmoid'den farklı olarak -1 ile 1 arasındadır.

1.3.1.5 Düzleştirme

Özellik çıkarma katmanlarından tam bağlamalı katmanlara geçiş yapabilmek düzleştirme katmanı kullanılır. Bu katman sayesinde iki ya da daha fazla boyutlu katmanın çıkışı tek boyuta indirilir [12].

1.3.1.6 Tam Bağlamalı Katmanlar

Katmanlar düzleştirildikten sonra klasik derin öğrenme düğümlerine tam bağlamalı katmanlar denir. Bu katmanlar, evrişimsel sinir ağlarının son kısmında kullanılır ve modelin çıktısını verir [13].

1.3.2 Evrişimsel Sinir Ağları Öğrenme Hususları

Evrişimsel sinir ağları her adımda olması gören sonuca göre kayıp değerleri hesaplayarak geri besleme yöntemiyle ağırlıkları tekrar oluşturur. Bu yüzden kayıp fonksiyonları ve geriye yayılım algoritmalarını kullanan optimize edicileri doğru seçmek önemlidir.

1.3.2.1 İleri ve Geri Yayılım

Giriş değerlerinin katmanlardan ve nöronlardan aktarılarak çıktıya ulaşma sürecine ileri yayılım denilmektedir. Ancak hatayı azaltmak için çıkıştan girişe doğru geri yönde olan gradyan iniş metodu ile parametrelerin güncellenmesine geri yayılım denir [14]. Böylelikle hata azaltılır ve parametrelerin daha doğru sonuçlara ulaşması sağlanır.

1.3.2.2 Kayıp Fonksiyonları

Kayıp fonksiyonlar, olması gereken değerler ile modelin ürettiği sonuçları kıyaslayarak bir kayıp değeri oluşturur. Geriye yayılım yöntemleri bu değeri iyileştirmeyi amaçlar. Bazı kayıp fonksiyonları şu şekildedir:

$$OKH = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (1.6)$$

$$MOH = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (1.7)$$

$$ÇD = -\sum_n p_n \, log \, q_{\,n} \quad (1.8)$$

$$ÇD = -y \, log \, \hat{y} - (1-y) log(1-\hat{y}) \quad (1.9)$$

Denklem 1.6'da ortalama kare hatası (OKH) ile olması gereken değerler y ile modelin hesapladığı ($\hat{y}$) değerlerinin farklarının karesinin ortalaması alınır. Denklem 1.7'de mutlak ortalama hatası (MOH) ile de olması gereken değerler y ile modelin hesapladığı $\hat{y}$ değerlerinin farklarının mutlak değerlerinin ortalaması alınır. Denklem 1.8'de, gerçek olasılık dağılımı p ile bulunan q olasılık dağılımı için beklenen entropi değeri yani çapraz düzensizlik (ÇD) hesaplanır [15]. Kayıp değeri hesaplanırken, $y = 1$ değeri için kayıp $-\log \hat{y}$, $y = 0$ için de $-(1-y)\log(1-\hat{y})$ şeklinde hesaplanır ve Denklem 1.9'daki gibi gösterilebilir.

1.3.2.3 Optimizasyon Algoritmaları

Her kayıp değeri hesaplandıktan sonra en düşük kayıp değerine ulaşılmalıdır ki modelin en iyi çalışabileceği parametreler hesaplanabilsin. Bunu sağlamak için de en düşük kayıba ulaşmasını sağlayan algoritmalara optimizasyon algoritmaları denir [16].

Derin öğrenmede en sık kullanılan optimizasyon algoritmalarından biri Uyarlanabilir Moment tahmini (ADAM) algoritmasıdır [17]. Bazı uyarlanabilir öğrenme oranları hesaplanarak, gradyanların birinci ve ikinci momentlerinin tahminlerinden farklı parametreler elde edilir. ADAM 'ın güncelleme formülü parametre şu şekilde verilir:

$$m_t = \Omega_1 m_{t-1} + (1-\Omega_1) g_t \quad (1.10)$$

$$v_t = \Omega_2 v_{t-1} + (1-\Omega_2)(\, g_t)^2 \quad (1.11)$$

$$m_t^{corrected} = \frac{m_t}{1-(\Omega_1)^t} \quad (1.12)$$

$$v_t^{corrected} = \frac{v_t}{1-(\Omega_2)^t} \tag{1.13}$$

$$W_t = W_{t-1} - \lambda \frac{m_t^{corrected}}{\sqrt{v_t^{corrected}}+\epsilon} \tag{1.14}$$

Burada g_t, t adımdaki gradyanları, m_t birinci moment gradyanları, v_t ikinci moment gradyanları, Ω_1, $\Omega_2 \in [0,1)$ üssel bozulma oranlarını, λ adım boyutu, ϵ sıfıra bölünmeyi önlemek için küçük bir değeri, W_t ağırlık vektörünü (güncellenecek parametre) temsil etmektedir [18].

1.3.3 Evrişimsel Sinir Ağları Kullanılan Terimler

Bir derin öğrenme modeli eğitilirken oluşan dikkate alınması gereken birçok parametrik terim vardır [19].

- **Eğitim Tur Sayısı (epoch):** Tüm veri kümesini modele bir kere geçirilmesine bir eğitim turu denir.
- **Yığın (batch):** Tüm veri kümesini modele tek seferde verilemediği için parçalara bölünerek modele verilir. Bu parçalar yığın adı verilir.
- **Yineleme (iteration):** Bir eğitim turunu tamamlamak için gerekli yığın sayısına denir.
- **Öğrenme Oranı (learning rate):** Geriye yayılımdan sonra parametrelerin ne kadar değiştireceğinin miktarına öğrenme oranı denir. Çok fazla olursa, kayıp değerini düşürmek zorlaşacaktır. Çok az olursa hem eğitim daha uzun sürecektir hem de bazı yerel kayıp değerlerde kalabilir ve istenilen başarıma ulaşmayabilir.

1.3.4 Aşırı Uyum (Over-Fitting)

Aşırı uyum makine öğrenmelerinde çok sık karşılaşılan bir sorundur. Model çok iyi görünmüş gibi görünse de aslında sadece model verilerinde iyi görünür ama yeni verilerde kötü sonuç alacaktır [20]. Aşırı uyum olduğu tespit etmek için veriler eğitim ve test verileri olarak bölünür ve test verileri üzerindeki doğruluk sonuçları ile eğitim verilerinin doğruluk sonuçları karşılaştırılır.

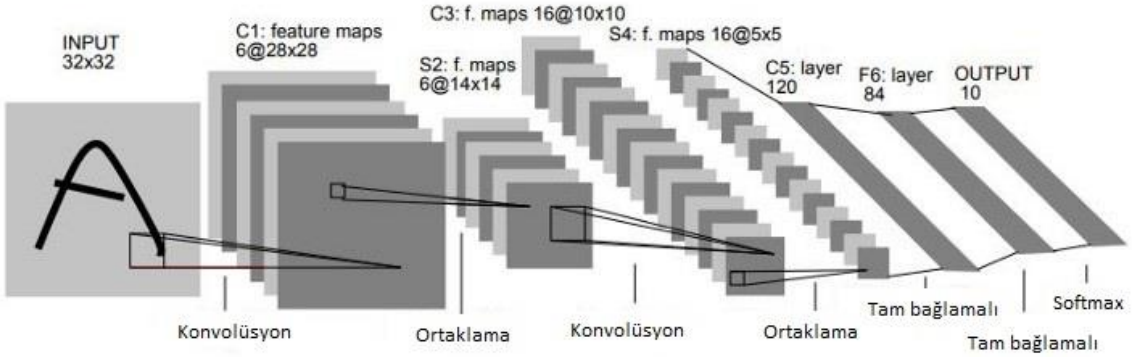
Aşırı uyumu engellemek için en çok kullanılan yöntem verileri büyütmektir. Verileri büyütmek için resimler için gürültüler, kaydırma, eksen döndürme gibi yöntemler uygulanır.

1.4 Klasik Evrişimsel Sinir Ağları

Evrişimsel sinir ağları çoğu zaman deneme yanılma yoluyla gelişmiştir. Bundan dolayı da hem gelişme mantığını incelemek için hem de çalışmada kullanılan klasik ağları anlayabilmek için bu ağlar açıklanmaktadır.

1.4.1 LeNet

Yann LeCun tarafından 1998 yılında geliştirilen LeNet ağı ilk başarılı sonuç veren evrişimsel sinir ağıdır [21]. Posta numaralarının ve banka çeklerinin üzerindeki sayıları tanımlamak için oluşturulmuş ve MNIST veri kümesi kullanılmıştır. Şekil 1.7'de de görüldüğü gibi günümüz ağlarından farkı ortalama ortaklama kullanılmış ve daha küçük bir ağdır. 5x5 filtre boyutunda konvolüsyonlar ve tam bağlamalı katmanlar kullanılmıştır. Çıkışa da softmax aktivasyon fonksiyonu uygulanmıştır.

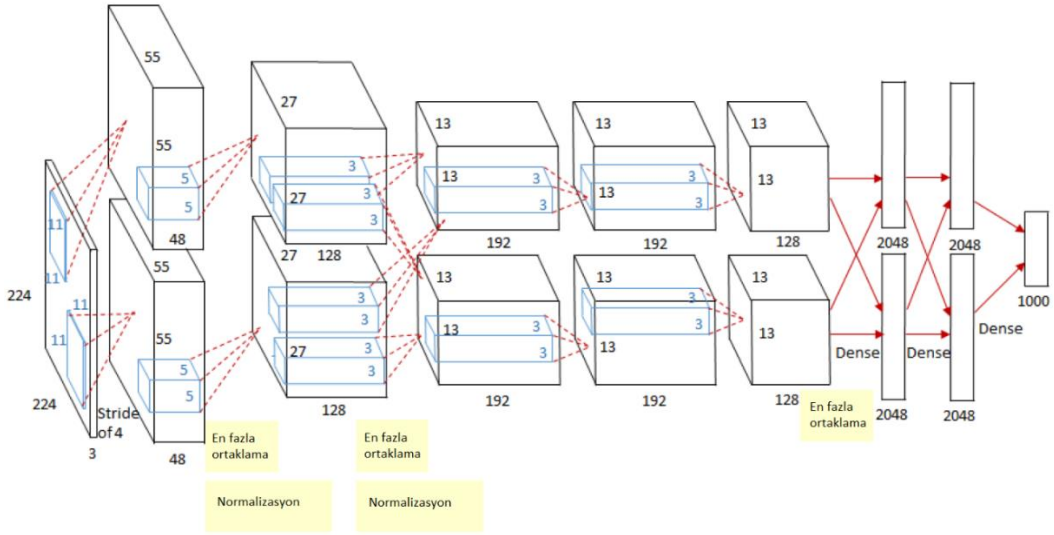


Şekil 1.7. LeNet mimarisi [21].

1.4.2 AlexNet

Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton tarafından 2012 yılında geliştirilen AlexNet [22], o yıllardaki başarım sonuçlarında yüksek bir sıçrama yaparak derin öğrenmenin tekrar popüler hale gelmesini sağlamıştır.

LeNet'den en büyük farklarından biri olan en yüksek ortalama ile tüm veriyi kullanmak yerine ayırt edici değerlerin daha önemli olduğu görülmüştür. Ayrıca Şekil 1.8'de de görüldüğü gibi paralel modüller kullanılmıştır. Ayrıca katmanlar arası ReLU aktivasyon fonksiyonu da kullanılmıştır. 3x3 ve 5x5 filtre boyutları kullanılmıştır.

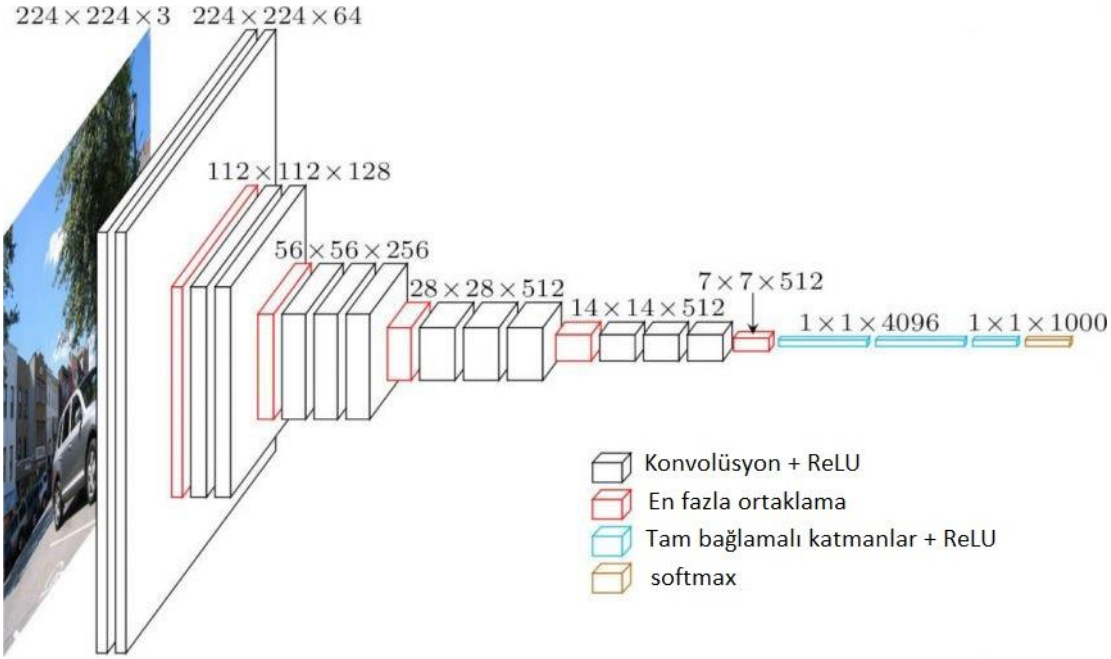


Şekil 1.8. AlexNet mimarisi [23].

1.4.3 VGGNet

Visual Geometry Group adında Oxford Üniversitesinde kurulan bir kuruluşun oluşturduğu bir mimaridir [1].

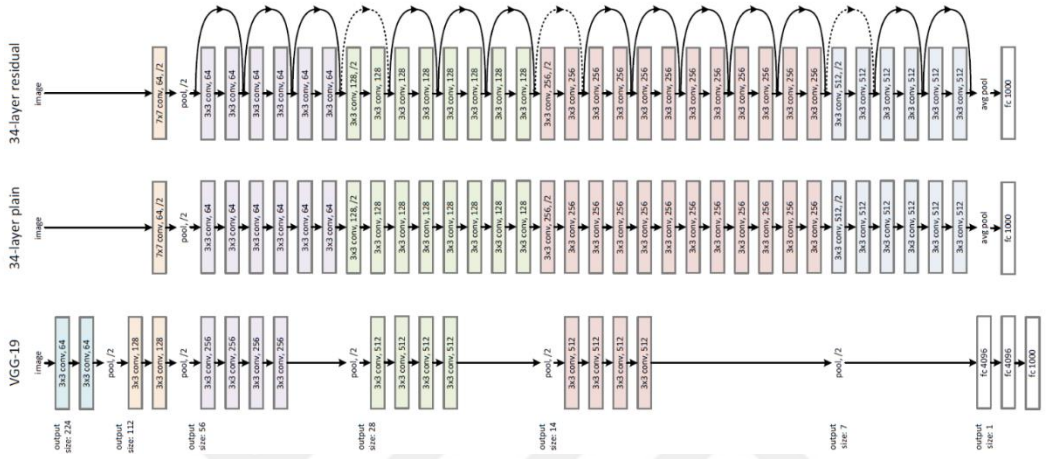
Şekil 1.9'da gösterilen VGG ağları ise her bir katmanda sonucu küçültürken filtre sayısını ikinin kuvvetleri sayısınca artırmaktadır ve kanal sayısı artmaktadır. Kanal sayısını ve derinliği artırdıkça karmaşık artsa da modelin sonuçlarını geliştirmiştir.



Şekil 1.9. VGGNet mimarisi [1].

1.4.4 ResNet

ResNet mimarisinin ise en önemli farkı aktarma işlemleridir. Bu aktarma işlemleri Şekil 1.10'da da görüldüğü gibi bir katman atlayarak sonraki katmana değerleri aktarır [2]. Bu toplam işlemi, sonraki katmanlarda öğrenme düşük de olsa önceki katmandaki değerleri de tutarak modelin iyi performansta eğitilebilmesini amaçlamaktadır.

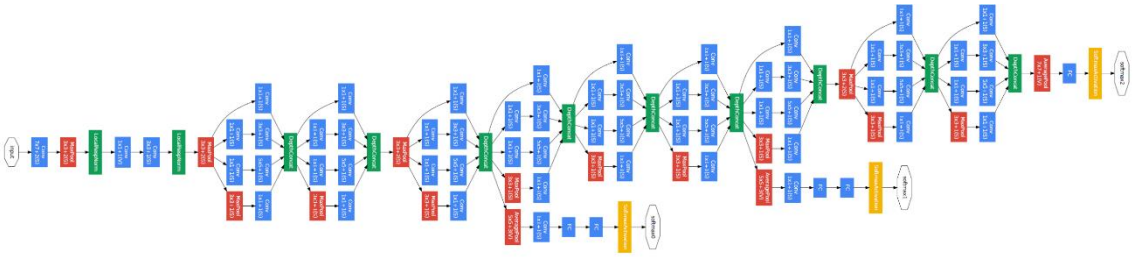


Şekil 1.10. ResNet mimarisi [2].

Ancak, gözlemlenen sonuçlarda bu pek de istenildiği gibi olmadığı ortaya çıkmıştır ve katmanı oldukça artırmak modelin başarısının düşmesine yol açmıştır. Bu sorunun ilk nedeni, katman sayısını artırmak parametre sayısını yani modelin karmaşıklığını artırmaktadır. Bundan dolayı da model aşırı uyum sergilemektedir. Diğer bir neden ise ilk katmanlarda öğrenilen değerlerin sonraki katmanlara aktarılırken değerinin düşmesidir.

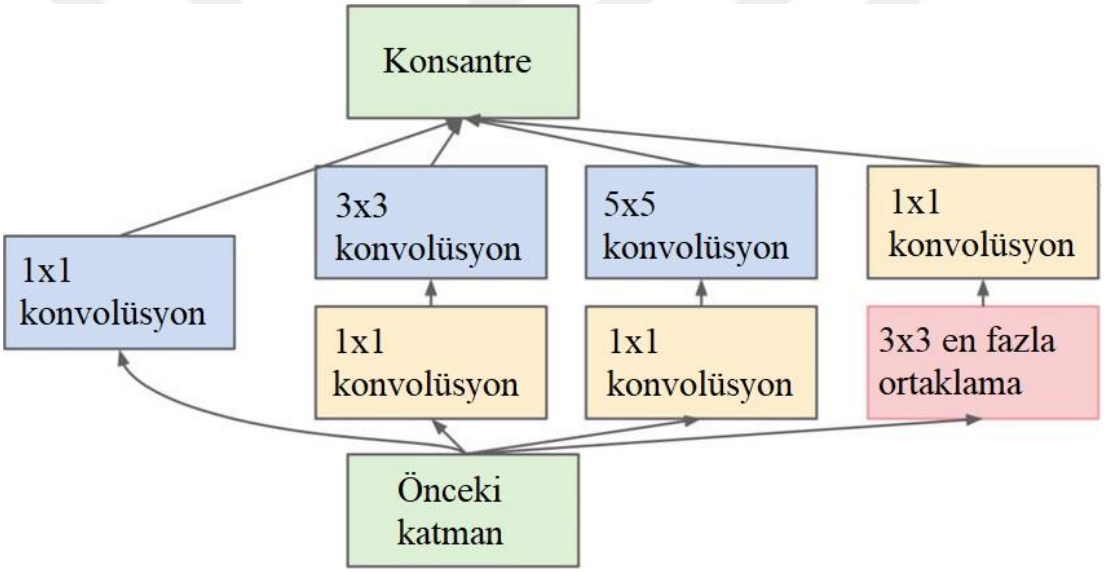
1.4.5 GoogleNet

GoogleNet mimarisi Google ekibi tarafından geliştirilen ve 2014'te yapılan ImageNet yarışmasını kazanmış mimaridir [3]. Bu mimari Inception modüllerinden oluşmaktadır ve Inception mimarisi olarak da anılır. Şekil 1.11'da gösterilen GoogleNet mimarisinde maviler konvolüsyon, kırmızılar havuzlama ve sarılar sınıflandırma katmanlarını göstermektedir.



Şekil 1.11. GoogleNet Mimarisi [3].

GoogleNet seri bir şekilde bağlanmış inception modüllerinden oluşmaktadır. Bu inception modülleri Şekil 1.12'de de görüldüğü gibi 1x1, 3x3 ve 5x5 boyutundaki konvolüsyon katmanlarının paralel şekilde bağlanmasıyla oluşmaktadır. Bu filtreler boyut azaltmak için kullanılır.



Şekil 1.12. Inception modülü [3].

1.5 Derin Sinir Ağları Araçları ve Kütüphaneleri

Derin öğrenmenin son yıllarda hızla büyümesinin nedeni hiç kuşkusuz ücretsiz geliştirilen derin öğrenme araçları ve kütüphaneleridir. Bu araçlar hem topluluklarca yeni metotlar uygulayarak gelişmesinde hem de geri bildirimler sayesinde bu araçların geliştirilmesinde önemlidir.

1.5.1 CUDA (Birleşik Cihaz Mimarisi Hesaplayıcısı)

CUDA, NVIDIA tarafından grafik işleme birimlerinde genel bilgi işlem için geliştirilmiş bir paralel bilgi işlem platformu ve programlama modelidir. CUDA ile geliştiriciler,

GPU'ların gücünden yararlanarak bilgi işlem uygulamalarını önemli ölçüde hızlandırabilir [24].

GPU hızlandırmalı uygulamalarda, iş yükünün sıralı kısmı tek iş parçacıklı performans için optimize edilmiş olan CPU üzerinde çalışırken, uygulamanın yoğun işlem gerektiren kısmı binlerce GPU çekirdeğinde paralel olarak çalışır. CUDA'yı kullanırken, geliştiriciler C, C++, Fortran, Python ve MATLAB gibi popüler dillerde programlama yapar ve birkaç temel anahtar kelime biçiminde uzantılar aracılığıyla paralelliği ifade eder. Bu sayede de GPU çekirdeklerinin işlemci çekirdeklerinden daha yavaş olsa da daha fazla işlem birimi ve CUDA sayesinde paralel olarak işlem yaparak çok daha hızlı hesaplama imkânı sağlar.

1.5.2 cuDNN (CUDA Derin Sinir Ağları Kütüphanesi)

NVIDIA CUDA Derin Sinir Ağı Kütüphanesi (cuDNN), derin sinir ağları için GPU hızlandırmalı bir ilkel kitaplıktır [25]. cuDNN, ileri ve geri evrişim, ortaklama, normalleştirme ve etkinleştirme katmanları gibi standart rutinler için yüksek düzeyde ayarlanmış uygulamalar sağlar.

Dünya çapında derin öğrenme araştırmacıları ve çerçeve geliştiricileri, yüksek performanslı GPU hızlandırma için cuDNN kullanılır. Düşük seviyeli GPU performans ayarına zaman harcamak yerine sinir ağlarını eğitmeye ve yazılım uygulamaları geliştirmeye odaklanmalarını sağlar. cuDNN; Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle, PyTorch ve TensorFlow dahil olmak üzere yaygın olarak kullanılan derin öğrenme kütüphanelerini hızlandırır.

1.5.3 Tensorflow

TensorFlow, makine öğrenimi için geliştirilmiş geliştiricilerin makine öğrenimi destekli uygulamaları kolayca oluşturup dağıtmalarına olanak tanıyan kapsamlı, esnek bir araç, kitaplık ve topluluk kaynakları ekosistemine sahip uçtan uca bir açık kaynak platformdur [26].

CPU ve GPU olmak üzere iki temel sürümü vardır. CPU versiyonunda işlemci üzerinden hesaplamalar yapılırken, GPU versiyonu sayesinde de cuDNN kütüphanesini kullanarak grafik işlem birimleri üzerinde hesaplamalar yapılabilmekte ve makine öğrenme

algoritmalarına gelişmekte olanak sağlamaktadır. Ayrıca bunların dışında Google'ın geliştirdiği tensor işleme birimlerini (TPU) de desteklemektedir.

En çok kullanılan ve daha stabil olan Python dili desteği dışında, Javascript, C++ ve Java dillerine de desteği bulunmaktadır. Ayrıca C#, Haskell, Julia, MATLAB, R, Ruby, Rust, Scala, Go ve Swift dillerine de destek vermeye başlamıştır.

1.5.4 Keras

Yunan mitolojisinden esinlenerek ve yunanca boynuz anlamına gelen Keras, makine öğrenme platformu Tensorflow üzerinde çalışan Python dilinde bir derin öğrenme API (uygulama programlama arayüzü)'sidir. Keras, yazması kolay, esnek yapısı ve endüstri seviyesinde kullanılabilecek kadar güçlü olarak betimlenir [27].

Kasım 2017'de Tensorflow 1.4 sürümüyle Keras API'sini kapsadığını duyurdu [29]. Ayrıca Tensorflow 2.0 sürümüyle de çok daha stabil ve kolay bir şekilde kullanımına olanak sağlamıştır.

Tablo 1.1. Derin öğrenme araçlarının karşılaştırılması [28]

	Diller	**Başlangıç ve Eğitim materyalleri**	**CNN modelleme kapasitesi**	**RNN modelleme kapasitesi**	**Mimari: kullanması kolay ve modüler olma arayüzü**	**Hızı**	**Çoklu GPU desteği**	**Keras uyumluluğu**
Theano	Python, C++	++	++	++	+	++	+	+
Tensorflow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	++	
Neon	Python	+	++	+	+	++	++	
CNTK	C++	+	+	+++	+	++	++	

1.5.5 Torch

Facebook AI Araştırma Laboratuvarı FAIR tarafından geliştirilen Torch ya da çoğu kişi tarafından bilinen adıyla Pytorch, açık kaynaklı bir makine öğrenmesi kütüphanesidir. Bilgisayarla görme ve doğal dil işleme uygulamalarında sıkça kullanılır. Python dilinde daha çok bilinmesine rağmen C++ desteği de bulunmaktadır [30].

1.5.6 Theano

Montreal Üniversitesi Öğrenme Algoritmaları Enstitüsü tarafından 2017 yılında geliştirilen Theano, çok boyutlu matrisleri içeren matematiksel ifadeleri tanımlamanıza, optimize etmenize ve verimli bir şekilde değerlendirmenize izin veren bir Python kütüphanesidir. Matris matris işlemlerinin derlendiği NumPy kütüphanesiyle sıkı entegrasyonu bulunmaktadır. Ayrıca, dinamik bir şekilde C kodu üreterek çalıştığından hesaplamaları hızlı yapmaktadır [31].

2017 yılında yapılan bir araştırmada Tablo 1.1'de de görüldüğü gibi çok fazla başarılı derin öğrenme kütüphaneleri mevcuttur. Ancak genel çerçevede bakıldığında ve bizim çalışmada evrişimsel sinir ağları da kullanıldığından dolayı Tensorflow öne çıkmaktadır. Bu yüzden bu çalışmada da Tensorflow ve Keras kullanılmıştır.

1.6 Literatürde Yapılmış Çalışmalar

Makine öğrenmesi ilk olarak Donald Hebb tarafından 1949 yılında ortaya atmıştır. İlerleyen yıllarda birçok makine öğrenmesi algoritması geliştirilmiş ve 1970'li yıllarda da Evrişimsel sinir ağları kullanılmaya başlanmıştır. Ancak 1989 yılında Yann LeCun'un geliştirdiği çok katmanlı konvolüsyon katmanlarının kullanıldığı Le-Net-5 ağı, kullanışlı bir seviyeye çıkmasını sağlamıştır [21]. İlerleyen yıllarda ise Evrişimsel sinir ağları ivmelenerek gelişmiştir ve birçok mimari ortaya çıkmıştır. Bu mimarilerden en çok dikkat çeken AlexNet [22] 2012 yılında ve hala sıkça kullanılan VGG ağı ise 2014 yılında geliştirilmiştir [1]. VGG ağı ile filtre sayısı artırılmış ve ikinin kuvvetleri şeklindedir. Bir diğer ağ mimarisi olan Resudial Network [2] ile de katmak sonuçları, sonraki katmanın da çıkışına eklenmiştir. Sonrasında da 1x1'lik konvolüsyonlar kullanılarak lineer konvolüsyon kullanılmasının dışına da çıkılmıştır [32]. Bu fikir ile Google, Inception modüllerinden oluşan GoogLeNet'i [3] ve sonrasında da XCeption [33] ağ mimarlarını ortaya koymuştur. Diğer başarılı mimarilere MobileNet [34], DenseNet [35], NASNET [36] ve EfficientNet [37] de eklenebilir. Bu mimariler özellikle obje tanıma ve bulma gibi birçok görüntü işleme problemlerinde oldukça başarılı ve gözde mimarilerdir. Bu çalışmadaki amaç, bu mimarilerinden bazılarını kullanarak kullanılan modüllerin dinamik bir şekilde oluşan bir hibrit model oluşturmaktır.

Evrişimsel sinir ağları ile başka derin öğrenme algoritmaları ile birleştirilip farklı hibrit mimariler de oluşturulmuştur. Örneğin, önceki değerleri de hafızalarında tutan ve Yinelenen Sinir Ağları [38] olarak tanımlanan algoritma ile Evrişimsel sinir ağlarının karışımı hibrit ağlar da mevcuttur. Bir çalışmada, Evrişimsel sinir ağlarının nesne tanıma ve yinelenen sinir ağlarının da hafızasıyla el yazısı okuması yapılmaktadır [39]. Yine bir çalışmada da Evrişimsel sinir ağları ve yine sınıflandırmada kullanılan destek vektör makinesi [40] ile oluşturulan hibrit model ile el yazısı rakamların tanınmasında kullanılmıştır [41]. Bir diğer çalışmada da Evrişimsel sinir ağları ile anahtar nokta çıkaran Scale Invariant Feature Transform algoritması [42] birleştirerek yüzdeki anahtar noktaları çıkararak Evrişimsel sinir ağlarının tanımadaki başarısını birleştirip duygu tespiti yapmaktadır [43]. Farklı mimarilerin Evrişimsel sinir ağları ile birleştirerek hem o mimarilerin kendine özgü çıkarımları kullanılır hem de Evrişimsel sinir ağlarının nesne sınıflanlandırmadaki yüksek başarısı kullanılır. Böylelikle belirli problemlerin çözümünde bu hibrit yapılar daha iyi sonuç vermektedir. Ancak biz bu çalışmada farklı makine öğrenme algoritmalarının birleştirilmesinden ziyade Evrişimsel sinir ağında farklı mimari modülleriyle hibrit model oluşturulacaktır.

Farklı evrişimsel sinir ağları mimarilerinin sentezlenerek kullanıldığı bazı hibrit ağlar da mevcuttur. Inception ile Resnet modüllerinin ilişkisi incelenerek iki adet hibrit modelin oluşturulduğu Inception-v4 çalışmasında sonuçlar diğer ağlardan daha başarılı olduğu gösterilmiştir [44]. Oluşturulan bu hibrit modeller birçok farklı alanda kullanılılarak incelenmiştir. Tıp alanında yapılan bir çalışmada Inception-Resnet modeli kullanılarak oluşturulan hibrit model ile Optik Koherens Göz Tomografisi görüntülerinden diyabetik maküler ödem tespiti yapılmıştır [45]. Bu hibrit modelin sonuçları ise Inception modeli ve Resnet modelleri ile kıyaslanmış ve daha iyi sonuçlar aldığı gözlemlenmiştir. Bir başka çalışmada da MRI görüntülerindeki hareket tespitleri için de Inception-Resnet mimarisi kullanılmıştır [46]. Görsellerden hastalara cevaplar veren bir başka hibrit model de oluşturmuştur [47]. Bir diğer çalışmada ise resimleri renklendirmek için kullanılmıştır [48]. Inception-ResNet-v2 modelinin küçültülmüş hali olan Tiny-Inception-ResNet-v2 modeli ise daha az parametreyle yine yüksek başarımlar bir modeldir ve uydu görüntülerinde tuğla fırınlarının tanımlanması için kullanılmıştır. Bir başka hibrit evrişimsel sini ağları modeli olan CNN-LSTM modeli de hem güç üretimlerinin tahminlenmesinde [49] hem de güç sistemlerinde iletim arızalarının tipi ve konumlarının

belirlenmesinde [50] kullanılmıştır. Bir başka çalışmada, tek boyutlu ve iki boyutlu evrişimsel sinir ağları paralel olarak eğitilerek adaptif bir öğrenme sağlanarak güç elektroniği sistemlerinde arıza teşhisi sağlanmaktadır [51].

Evrişimsel sinir ağlarında farklı modelin bir arada kullanılması çok gözlemlenmese de birçok makine öğrenmesi algoritmasında görülmektedir. Birçok modelin kullanıldığı kolektif öğrenmenin torbalama ve yükseltme adında iki türü mevcuttur. Torbalama metodunda tüm modeller birbiriyle eşit ve oy çoğunluğu ya da ortalama ile sonuca varılır. Ancak yükseltme metodunda daha başarılı modellerin ağırlıkları başlangıçta aynıdır ancak model eğitilirken bu ağırlıklar güncellenir ve daha başarılı olan ağların ağırlıkları daha fazla olur [52]. Böylelikle de daha yorumlanabilir bir model de oluşmuş olur. Bu yaygın kolektif yükseltme algoritmalarından olan AdaBoost (Adaptif Yükseltme), Gradyan Yükseltme ve XGBoost metotları kredi kartı dolandırıcılığı gibi [53] birçok güvenlik sistemlerinde hala kullanılmaktadır [54, 55].

Bu çalışmada da kolektif öğrenmedeki yükseltme metodunu evrişimsel sinir ağlarına sentezleyerek farklı mimariler bir arada eğitilmiştir. Böylelikle daha başarılı mimari modüllerinin ağırlıklarına göre modül seçimi yapılacaktır. Yükseltme modüllerinden farklı olarak bu modüller elenecek ve son bir ağ seçilecektir ve adaptif hibrit bir evrişimsel sinir ağı oluşturulacaktır.

2. MATERYAL VE YÖNTEM

Bu çalışmada güncel ve literatürde yaygın olarak çalışılan veri kümeleri kullanarak geliştirilen MAHNet mimarisinde uygulanmıştır ve test edilmiştir. Çalışmanın bu bölümünde, obje sınıflandırmasında kullanılan veri kümeleri, veri ön işleme süreçleri ve MAHNet modeli hakkında detaylı bilgilere yer verilecektir.

2.1 Hazır Veri Kümeleri

Bu çalışmada hazır veri kümesi olarak MNIST [56], Fashion MNIST [57], CIFAR10 [58], CIFAR100 [58]ve STL10 [59] veri kümeleri kullanılmıştır. Bu veri kümeleri, derin öğrenme mimarilerinde kalite testlerinde kullanıldığından bu çalışmada da tercih edilmişlerdir.

2.1.1 MNIST Veri Kümesi

El yazısı rakamlarından oluşan ve 70000 adet veri içeren tek kanallı bir veri kümesidir. 60000'i eğitim, 10000'i test olarak önceden ayrılmıştır [56] ve örnek verileri Şekil 2.1'de gösterilmiştir. Derin öğrenme problemlerinin başlangıç problemidir ve çoğu model rahatlıkla üstesinden gelebilmektedir. Her bir görsel 28x28 boyutundadır.



Şekil 2.1. MNIST veri kümesinden örnek veriler.

2.1.2 Fashion MNIST Veri Kümesi

Giysilerden oluşan ve 70000 adet veri içeren tek kanallı bir veri kümesidir. 60000'i eğitim, 10000'i test olarak önceden ayrılmıştır [57] ve örnek verileri Şekil 2.2'de gösterilmiştir. On farklı etiket mevcuttur ve bunlar; "T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot" 'dır.



Şekil 2.2. Fashion MNIST veri kümesinden örnek veriler.

2.1.3 CIFAR10 ve CIFAR100 Veri Kümeleri

AlexNet'in de geliştiricisi Alex Krizhevsky ve ekibi tarafından oluşturulan CIFAR veri kümeleri, 80 milyonluk bir veri kümesinin bir alt kümesidir. 32x32 boyutunda üç kanallı yani renkli olan görseller, 60000 görselden oluşmaktadır [58] ve örnek verileri Şekil 2.3'de gösterilmiştir. Bu görseller, 50000 eğitim ve 10000 test görseli olarak ayrılmış ve hem eğitim hem test için her bir sınıf için eşit sayıda görsel seçilmiştir.



Şekil 2.3. CIFAR veri kümelerinden örnek veriler.

10 sınıflı CIFAR10 ve 100 sınıflı CIFAR100 veri kümelerinin etiketleri Tablo 2.1'de gösterilmiştir.

Tablo 2.1. CIFAR veri kümesi etiketleri.

CIFAR-10 Etiketleri							
airplane	automobile	bird	cat	deer	dog	frog	horse
ship	truck						
CIFAR-100 Etiketleri							
apple	aquarium_fish	baby	bear	beaver	bed	bee	beetle
bicycle	bottle	bowl	boy	bridge	bus	butterfly	camel
can	castle	caterpillar	cattle	chair	chimpanzee	clock	cloud
cockroach	couch	crab	crocodile	cup	dinosaur	dolphin	elephant
flatfish	forest	fox	girl	hamster	house	kangaroo	keyboard
lamp	lawn_mower	leopard	lion	lizard	lobster	man	maple_tree
motorcycle	mountain	mouse	mushroom	oak_tree	orange	orchid	otter
palm_tree	pear	pickup_truck	pine_tree	plain	plate	poppy	porcupine
possum	rabbit	raccoon	ray	road	rocket	rose	sea
seal	shark	shrew	skunk	skyscraper	snail	snake	spider
squirrel	streetcar	sunflower	sweet_pepper	table	tank	telephone	television
tiger	tractor	train	trout	tulip	turtle	wardrobe	whale
willow_tree	wolf	woman	worm				

2.1.4 STL10 Veri Kümesi

STL-10 veri kümesi, denetimsiz özellik öğrenme, derin öğrenme, kendi kendine öğrenme algoritmaları geliştirmeye yönelik bir görüntü tanıma veri kümesidir [59]. CIFAR-10 veri kümesinden esinlenilmiştir ancak bazı değişiklikler yapılmıştır ve örnek verileri Şekil 2.4'te gösterilmiştir. Ancak, bu veri kümesinin (96x96) daha yüksek çözünürlüğünün, onu daha ölçeklenebilir denetimsiz öğrenme yöntemleri geliştirmek için zorlu bir kıyaslama yapmasını beklenir.



Şekil 2.4. STL10 veri kümesinden örnek veriler.

2.2. Seçim Katmanı

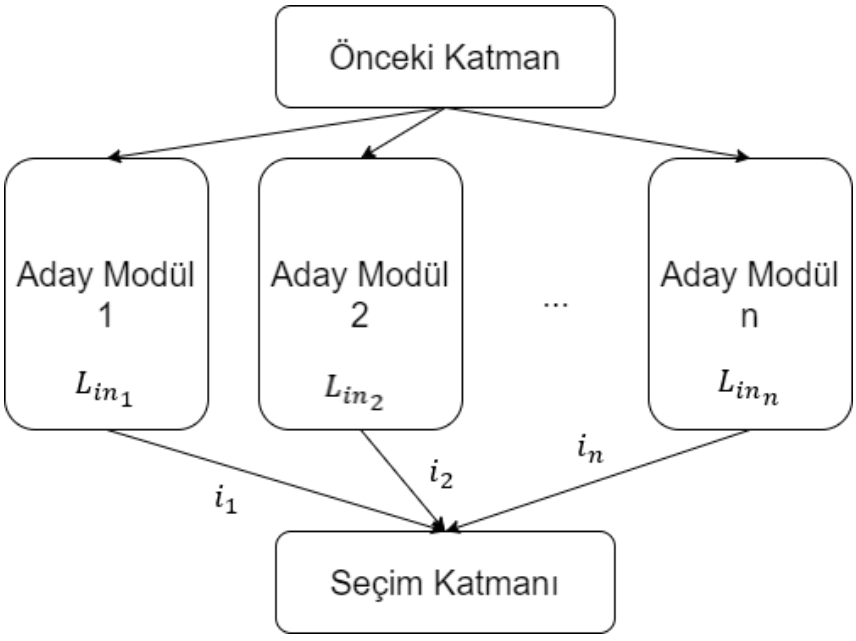
Seçim katmanı, klasik evrişimsel sinir mimarilerinden baz alınarak oluşturulan ve paralel olarak eğitilen aday modüllerden seçim yapmayı sağlar. Böylelikle en iyi modelin

seçilmesi amaçlanmıştır. Seçim katmanı, aday model sayısı kadar eğitilebilir ağırlık bulunmaktadır ve her bir aday modülün bir seçim ağırlığı bulunmaktadır. Bu ağırlıklar 0 ile 1 arasında ve toplamları da 1'dir. Aynı zamanda, modüllerin eşit şartlarda yarışabilmesi için başlangıçta her biri eşit ağırlıkta başlar. Düşük seçim ağırlıkların modele etkisi daha az olacağından, belirli bir eğitim sürecinden sonra modelden çıkarılmıştır ve en iyi modelin oluşması amaçlanmıştır.

$$L_{out} = \sum_k L_{in_k} \frac{i_k}{\sum_n i_n} \tag{2.1}$$

Seçim katmanın çıktısı (L_{out}), Denklem 2.1'de görüldüğü üzere her bir aday modül (L_{in}) ile seçim katmanındaki seçim ağırlığı i ile çarpılır ve bu çarpımlar toplanarak elde edilir. Seçim ağırlığının düşük olması yani sıfıra yaklaşması o modülün modele katkısı azalır. Bundan dolayı modelden çıkarılır. Düşük ağırlığa rağmen modül çıkarılması modelin başarısını başlangıçta düşürse de model kendini toplamaya devam etmektedir.

Denklem 2.1'de görüldüğü gibi, seçim ağırlıklarının toplamının 1 olmasını sağlamak için seçim ağırlıklarının toplamına bölünmüştür. Başlangıçta bunu sağlamak için sigmoid fonksiyon kullanılmıştır ancak ağırlıkların uzaklaşması sağlıklı olmadığı için modelden aday modül çıkarmak kötü sonuçlara neden olmuştur. Şekil 2.5'te seçim katmanının bağlantısı gösterilmiştir ve önceki modül sayısına göre dinamik olarak oluşmaktadır.



Şekil 2.5. Seçim katmanı.

Belirli bir eğitim tur sayısından sonra aday modül eleme işleme sürecine eğitim-adımı olarak tanımlanmıştır. Seçim katmanında tepe ve dip eşik değerleri olmak üzere iki adet eşik değeri tanımlanmıştır. Yapılan testlerde dip eşik değeri için 0,2 ve tepe eşik değeri için 0,8 olarak belirlenmiştir. Her bir eğitim-adımından sonra dip eşik değerin altındaki modüller modelden çıkarılmıştır. Tepe eşik değerinin üstündeki modüllerde ise o modül tutularak diğer aday modüller çıkarılmıştır.

2.3. Aday Modüller

Aday modüller, klasik evrişimsel sinir mimarilerinden oluşturulan modüllerdir. Bu modüller, seçim katmanı sayesinde en iyi modelin seçilmesi için aday konumundadırlar. Aynı seçim katmanına bağlı aday modüller, paralel olarak eğitilir ve diğer aday modüllerle aynı giriş boyutuna ve çıkış boyutuna sahiptirler. Seçim katmanındaki ağırlıklarına göre de aday modüllerden seçim yapılır.

Seçim katmanında modüllerin kolayca seçilip tüm modüller ile işlemlerin kolaylıkla yapılabilmesi gerekmektedir. Bunu sağlamak için de Keras API'si sayesinde bu modeller ayrı bir özel tanımlanmış katman olarak kodu yazılmıştır. Böylelikle seçim katmanının modülleri seçebilmesi kolaylaşmıştır.

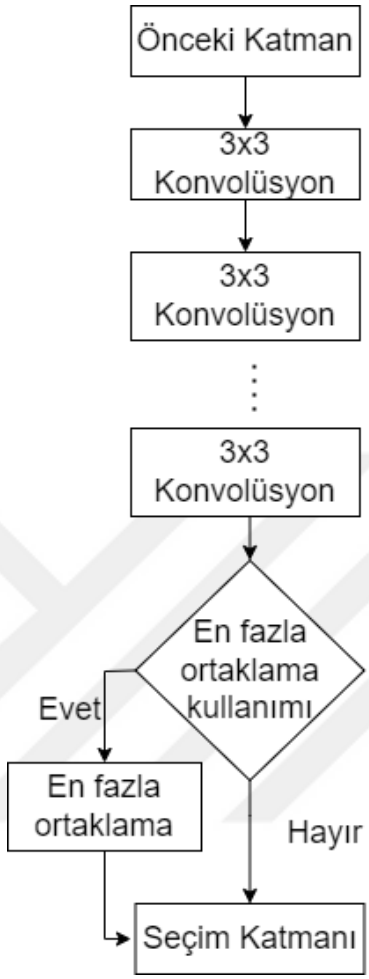
Bu çalışmada VGGNet, ResNet ve Inception mimarilerinden esinlenerek üç farklı aday modül oluşturulmuştur. Ancak, aday modüllerin eşit şartlarda yarışabilmesi için benzer sayıda eğitilebilir parametreye sahip olmaları gerekmektedir. Çünkü, düşük parametreli modüller daha hızlı öğrenip fazla parametreli modüllerin potansiyeline erişemeden öne geçebilmektedir. Bundan dolayı ResNet ve Inception modüllerine ek olarak çoklu versiyonları da eklenmiştir.

Tüm aday modüllere iki adet parametre tanımlanmıştır. İlk olarak, modülün çıktısının filtre sayısının ne kadar olması gerektiğidir. Böylelikle modül buna göre katmanlarını şekillendirir. İkincisi, modüle en fazla ortaklamanın uygulanıp uygulanmayacağıdır. Bu parametreye göre modülün sonuna en fazla ortaklama eklenir.

2.3.1. VGGNet Aday Modülü

VGGNet aday modülü, VGGNet mimarisinde olduğu gibi seri şekilde bağlanmış katmanlardan oluşmaktadır. Şekil 2.6'da da görüldüğü gibi verilen parametre kadar 3x3

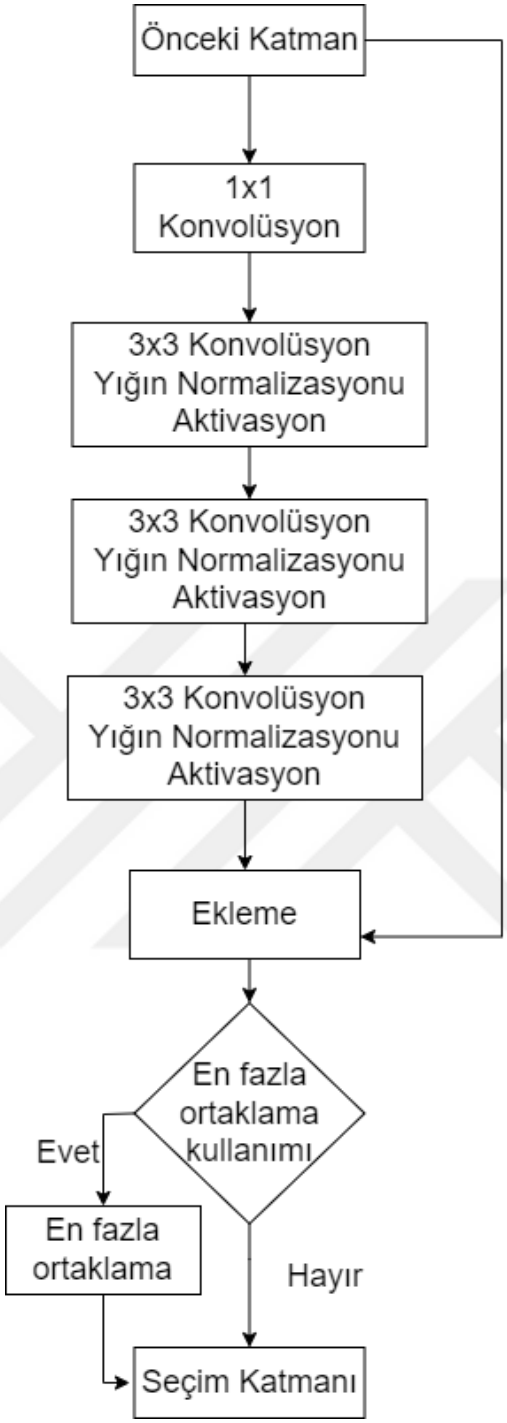
konvolüsyonlar seri şekilde bağlanmıştır ve koşullu olarak sonuna en fazla havuzlama katmanı vardır. Her bir konvolüsyona ReLU aktivasyon fonksiyonu uygulanmıştır.



Şekil 2.6. VGGNet aday modülü.

2.3.2. ResNet Aday Modülü

ResNet aday modülü, Resnet mimarisinde olduğu gibi bilgiyi korumayı amaçlayan katmanlar arası ekleme yapar. Şekil 2.7'de görüldüğü gibi girişe 1x1 konvolüsyon uygulandıktan sonra 3x3 konvolüsyon, yığın normalizasyonu ve ReLU aktivasyon fonksiyonunun olduğu blok üç kere uygulandıktan sonra giriş tekrar eklenmiştir. Koşula göre de çıkışa en fazla havuzlama uygulanır.

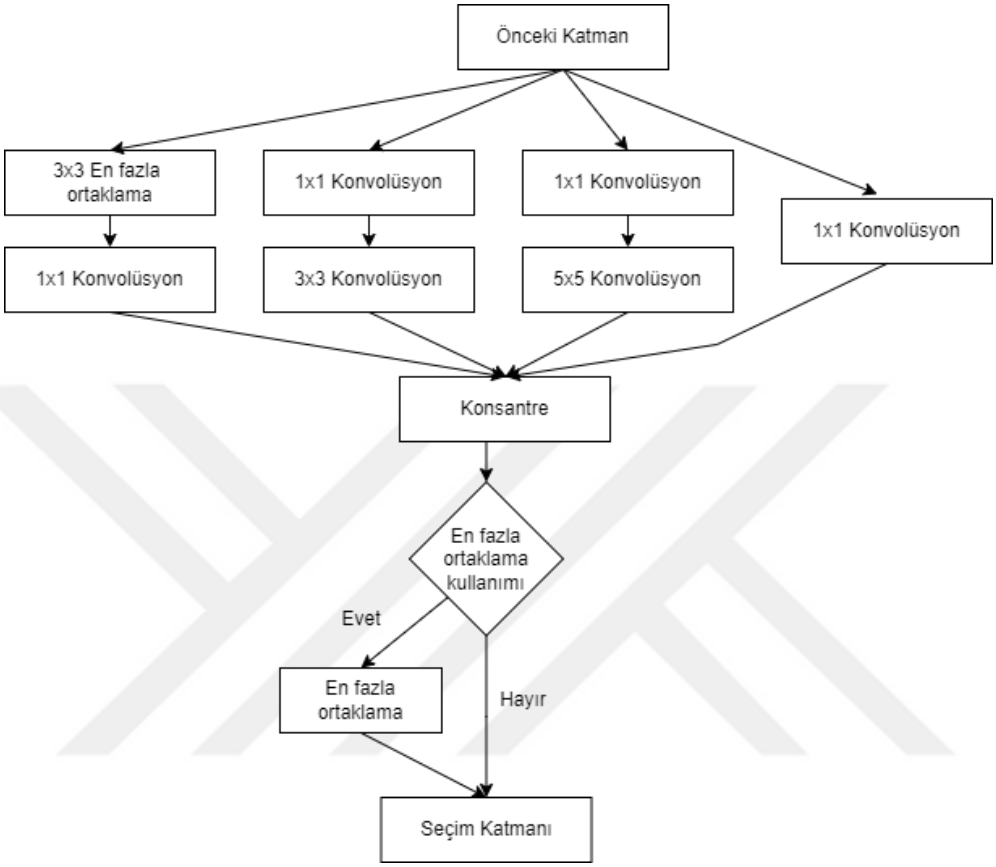


Şekil 2.7. ResNet aday modülü.

2.3.3. Inception Aday Modülü

Inception aday modülü ise GoogleNet'de kullanılan Inception modülü baz alınarak oluşturulmuştur. Şekil 2.8'de görüleceği üzere 1x1, 3x3 ve 5x5 filtre boyutundaki konvolüsyonlar paralel olarak eklendikten sonra konsantre yani çıktıları yanlarına

eklenmiştir. Inception modülünden farklı olarak aday modüllerin çıktılarının eşit olması gerektiğinden filtre sayısı diğerleriyle eşit olacak şekilde ayarlanmıştır. Diğer modüllerde olduğu gibi koşullu en fazla havuzlaması eklenmiştir.



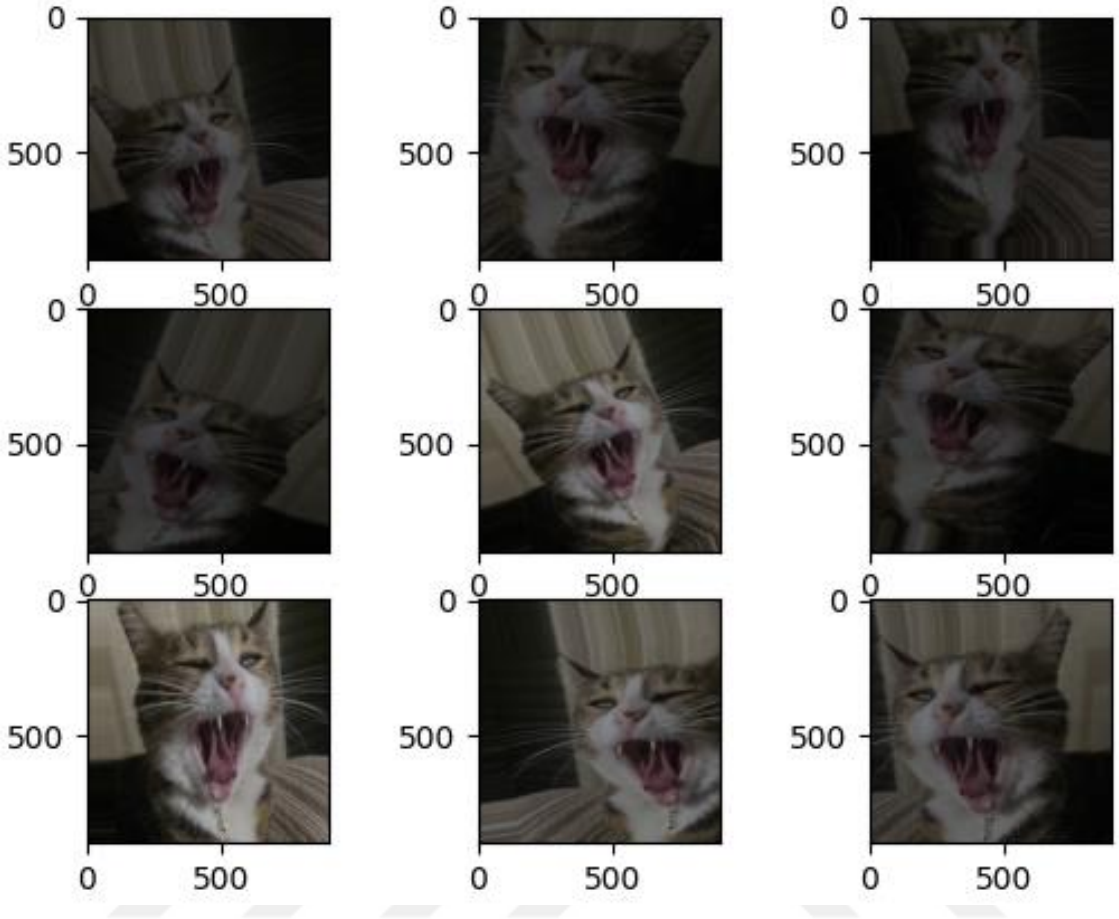
Şekil 2.8. Inception aday modülü.

2.4. Veri Önişleme

Modeldeki aşırı uyumu önlemek için veri artırımı uygulanmıştır. Veri artırımı sayesinde hem veri açlığı engellenir hem de aşırı uyumu önler. Veri artırımı için;

- 20 derece **döndürme**,
- 0,2 ve 0,8 oranında **parlaklık düşürme**,
- 0,2 oranında **yatay** ve **dikey kaydırma,**
- **Yatay döndürme,**
- 0,2 oranın **yakınlaştırma**

metotları uygulanmıştır. Şekil 2.9'da de görüldüğü gibi her mothodun görsel üzerindeki etkisi görülmektedir.



Şekil 2.9. Veri artırımı yöntemleri.

Kullanılan veri kümeleri 8 bit olduğu için değerler 0 ile 255 arasındadır. Optimizasyon metotları hesaplarken normalize edilmiş değerlerin kayıp değeri daha hızlı yakınsar [60]. Bu yüzden veriler 255'e bölünerek 0 ile 1 arasına normalize edilmiştir.

2.5. Derin Öğrenme Mimarisi

Bu bölümde derin öğrenme modeline ait mimari detay yeralmaktadır. Bu mimari; özellik çıkarma katmanları, seçim katmanları ve tam bağlantı katmanlarından oluşmaktadır.

2.5.1 Özellik Çıkarma Katmanları

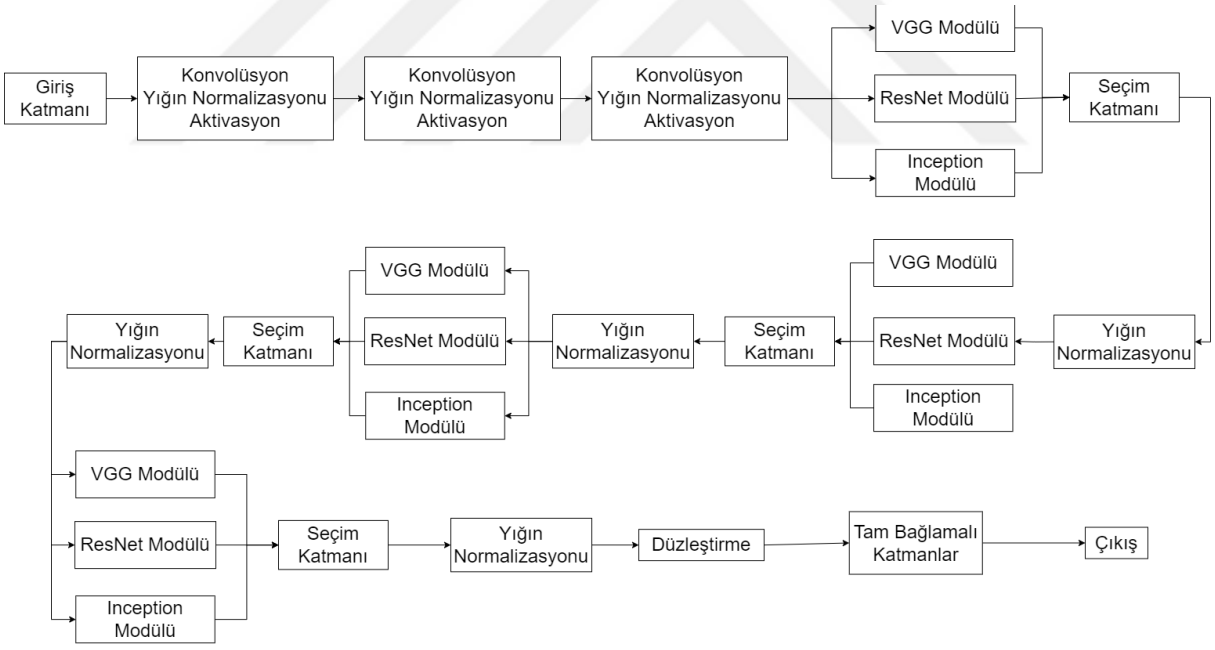
Seçim katmanlarından önce modelde özellik çıkarma katmanları sayesinde de bir öğrenme süreci oluşur. Bu katmanlar da özellik çıkarma kullanılır. Bu katmanlar; konvolüsyon, yığın normalizasyonu ve ReLU aktivasyon fonksiyon bloğundan üç adet içermektedir.

2.5.2 Seçim Katmanları

Seçim katmanı, aday modüllerden sonra oluşturulmuş katmandır. Aday modül sayısı kadar eğitilebilir ağırlık içerir ve başlangıçta eşittir. Bu katmanda aday modüllerin çıktıları ağırlıklar ile çarpıldıktan sonra toplanır ve seçim katmanının çıktısını oluşturur. Bu eğitilebilir ağırlıklar 0 ile 1 arasında ve toplamları 1 olduğundan bir süre sonra bazıları artacak bazıları azalacaktır. Ağırlığı az olan modüllerin modele katkısı azalacak ve önemsizleşecektir. Böylelikle bu modüller, modelden çıkarılacak yani seçim yapılacaktır. Modelde birden fazla seçim katmanı uygulanabilir. Her seçim katmanından sonra da yığın normalizasyonu uygulanmıştır.

2.5.3 Tam Bağlantı Katmanları

Şekil 2.10'da görüldüğü gibi seçim katmanlarından sonra model düzleştirilir yani tek boyuta indirilir. Bu katmanlarda boyut küçültülür ve sınıflandırma yapılır. Çıktıya Softmax aktivasyon fonksiyonu uygulanarak sınıflandırma tamamlanır.



Şekil 2.10. Model mimarisi

MAHNet modeli bu üç temel yapıdan oluşur. Özellik çıkarma katmanları ile öznitelikler temel seviyede çıkarılır. Özellik çıkarma katmanları; 3x3 boyutunda konvolüsyon, yığın normalizasyonu ve ReLU aktivasyon fonksiyonu bloğundan üç adet içerir. Sonrasında ise

dört adet seçim katmanlarıyla aday modüller seçilir ve tam bağlamalı katmanlar ile de sınıflandırma yapılır.

3. BULGULAR VE TARTIŞMA

Bu bölümde önerilen modelin uygulandığı deney ortamından, deneylerin değerlendirilme ölçütlerinden, önerilen MAHNet modelinin kullanılan veri kümeleri üzerinde test edilmesinden, elde edilen deney sonuçlarından ve modelin başarı durumundan bahsedilmektedir.

3.1. Deney Ortamı

Deneyler için kullanılan bilgisayar Windows 10 OS, AMD Ryzen 1700 3.0 GHz işlemci, 16 GB RAM, 512GB SSD ve NVIDIA GTX 1080 grafik kartı özelliklerini içermektedir. Geliştirme ortamı olarak PyCharm IDE, programlama dili olarak Python 3.7 tercih edilmiş ve derin öğrenme modeli için Tensorflow, Keras, Pandas kütüphaneleri kullanılarak deneysel ortam hazırlanmıştır.

3.2. Performans Metrikleri

Modelin performansını ölçmek için karıştırma matrisi kullanılarak elde edilen Doğruluk, Kesinlik, Duyarlılık ve F ölçümü gibi değerler kullanılmıştır. Tablo 3.1'deki karmaşıklık matrisi performans ölçümlerini içermektedir [61]. Pozitif tahminlerin doğru değerde olmasına TP, yanlış olmasına FP denir. Negatif tahminlerinse doğru olduğu yanlış tahminlere FN, doğru tahminlere de TN denir.

Tablo 3.1. Karmaşıklık matrisi.

	Pozitif Tahmin	**Negatif Tahmin**
Gerçek Pozitif	Doğru Pozitif (TP)	Yanlış Negatif (FN)
Gerçek Negatif	Yanlış Pozitif (FP)	Doğru Negatif (TN)

Denklem 3.1'de, tüm doğru tahminlerin sayısının tüm veri kümesine bölünmesiyle doğruluk hesaplanır.

$$\text{Doğruluk} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

Denklem 3.2'de de doğru pozitif tahminlerin sayısının tüm pozitif tahminlerin sayısına bölünmesiyle kesinlik hesaplanır.

$$\text{Kesinlik} = \frac{TP}{TP+FP} \tag{3.2}$$

Gerçek pozitif tahminlerin sayısının veri kümesindeki gerçek pozitiflerin sayısına bölünmesiyle duyarlılık bulunur.

$$\text{Duyarlılık} = \frac{TP}{TP+FN} \tag{3.3}$$

Duyarlılık ve keskinlik arasındaki dengeyi gösteren, ikisinin harmonik ortalamasıyla F puanı hesaplanır.

$$\text{F Puanı} = 2 \text{ x } \frac{Kesinlik\ x\ Duyarlılık}{Kesinlik\ +Duyarlılık} \tag{3.4}$$

3.3. Deney Sonuçları

Deneyler sonucu olgunlaştırılan MAHNet modelinin başarısı veri kümelerinde sınandıktan sonra literatürde aynı veri kümeleri üzerinde yapılan çalışmalarda kullanılan diğer derin öğrenme modellerinin doğruluk değerleriyle kıyaslanmıştır. Önerilen MAHNet modeli, Tablo 3.2'de yapılan deneylerle kat edilen aşamalar gösterilmiştir.

Tablo 3.2. MAHNet ilerleme aşamaları ve yapılan deneyler

Deney No	Açıklama	Veri Kümesi	Doğruluk (%)	Kayıp
1	İlk Deney	CIFAR10	68,73	1,37
2	CIFAR100 deneyi	CIFAR100	36,17	3,73
3	ADAM optimize edicinin kullanılması	CIFAR100	50,17	2,06
4	Veri artıcının uygulanması	CIFAR10	76,15	0,70
5	Veri artıcının uygulanması	CIFAR100	41,59	2,31
6	Seçim katmanında softmax uygulanması	CIFAR100	16,87	4,36
7	Özellik çıkarma katmanlarının eklenmesi	CIFAR100	46,85	2,34
8	Özellik çıkarma katmanlarının eklenmesi	CIFAR10	85,97	0,49
9	Adım sayısını artırarak uzun deney yapılması	CIFAR10	87,86	0,39
10	Adım sayısını artırarak uzun deney yapılması	CIFAR100	54,11	1,79

Hatalı ya da yarıda kesilen deneyler dışında toplam 61 adet deney yapılmıştır. Ancak kalite testinin yapıldığı veri kümelerinin ve önemli adımların içerdiği 10 adet deney incelenmiştir. İlk olarak veri artırımının yapılmadığı ve stokastik gradyan inişinin kullanıldığı deneyde düşük ama vaat edilmiştir. Sonrasında ADAM optimize edicinin

daha iyi sonuçlar vermesinden dolayı kullanılmasına karar verilmiştir. Sonuçlarda aşırı uyum gözlemlendiği için veri artırımı uygulanmıştır. Sonraki aşamalarda seçim katmanında softmax uygulanması denenmiştir ancak modül ağırlıklarının uzaklaşmasında engel yaşadığı için modül çıkarmada başarı oranı düşük sonuçlar elde edilmiştir. İlerleyen deneylerde de aday modüllerin öncesine özellik çıkarma katmanları eklenerek aday modüllerindeki yük azaltılmıştır. Son olarak da adım sayısını artırarak modelin en fazla çıkabileceği değerler gözlemlenmiştir.

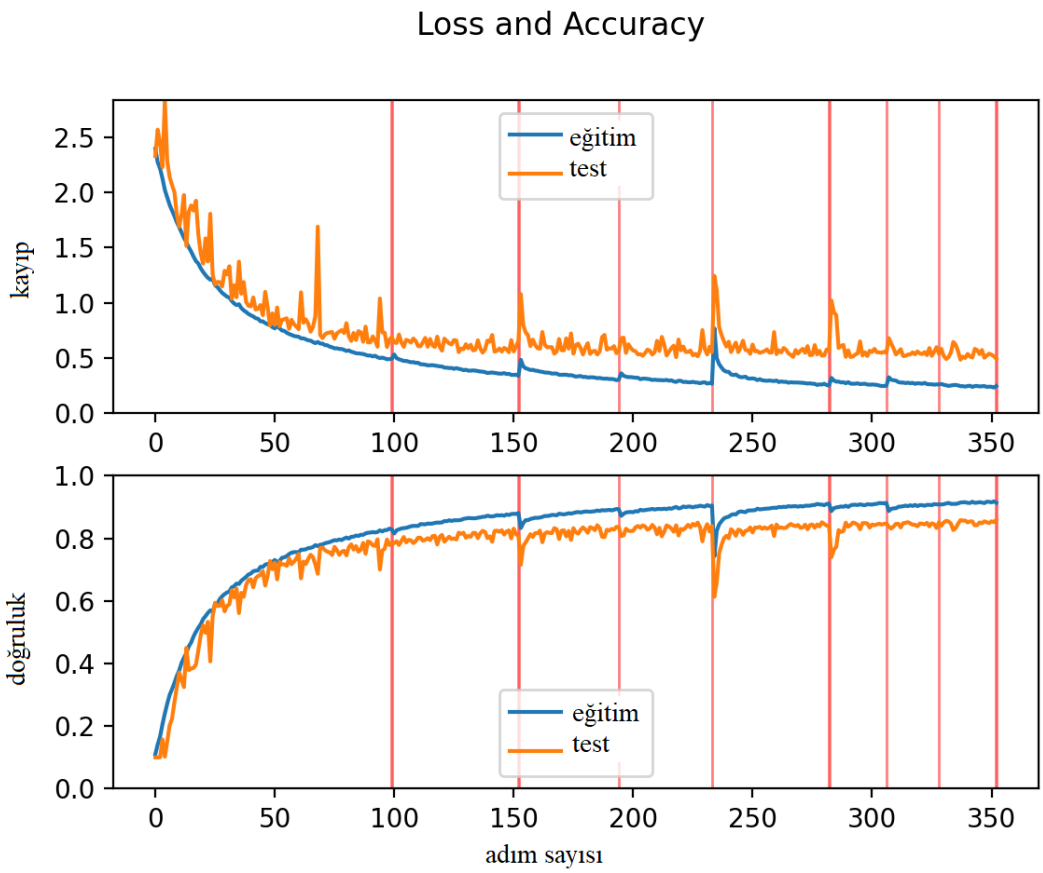
Deneyler sonucu olgunlaştırılan MAHNet modelinin başarısı mevcut veri kümelerinde sınanmıştır ve Tablo 3.3'te gösterilmiştir.

Tablo 3.3. MAHNet'in veri kümelerinde final sonuçları

Veri Kümesi	Başlangıç Parametre Sayısı	Son Parametre Sayısı	Doğruluk (%)	Kayıp
CIFAR10	1,74 Milyon	0,67 Milyon	87,86	0,39
CIFAR100	1,84 Milyon	0,77 Milyon	54,11	1,79
STL10	1,74 Milyon	0,66 Milyon	87,69	0,40
MNIST	1,73 Milyon	0,66 Milyon	99,09	0,03
Fashion MNIST	1,73 Milyon	0,66 Milyon	99,02	0,03

Parametre sayısı, modüller çıkarıldığı için azalmıştır. MNIST ve Fashion MNIST veri kümeleri için başarılı sonuçlar ortaya çıkmıştır. STL10 ve CIFAR10 veri kümeleri içinse diğer modellerle yarışabilecek seviyedir. CIFAR100 veri kümesi içinse daha düşük doğruluk sonuçları elde edilmiştir.

Şekil 3.1'de CIFAR10 veri kümesinde modelin epoch başına doğruluk ve kayıp değerleri gösterilmiştir.



Şekil 3.1. CIFAR10 veri kümesinde modelin doğruluk ve kayıp grafiği.

Modelin sürekli gelişimi görülmektedir. Kırmızı çizgilerle belirtilmiş eğitim adımlarında aday modüllerin seçimi yapılmıştır ve beklenildiği gibi kayıplar oluşmuştur. Ancak, model kritik kayıplar olmadığı gözlemlenmiş ve sonrasında modelin toparladığı gözlemlenmiştir.

CIFAR10 veri kümelerindeki sonuçlarda her sınıf için dengeli sonuçların alındığını Tablo 3.4 göstermektedir.

Tablo 3.4. CIFAR10 veri kümesinde sınıfa göre sonuçlar

Sınıf	Kesinlik	Duyarlılık	F skoru
airplane	0,886	0,890	0,888
automobile	0,932	0,940	0,936
bird	0,869	0,824	0,846
cat	0,787	0,753	0,770
deer	0,865	0,880	0,873
dog	0,874	0,783	0,826
frog	0,869	0,940	0,903
horse	0,910	0,915	0,913
ship	0,916	0,920	0,918
truck	0,874	0,941	0,906
ortalama	**0,878**	**0,879**	**0,878**

Tablo 3.5'in gösterdiği üzere CIFAR100 veri kümesinde ise bazı sınıflarda iyi başarım bazılarında ise diğerlerine göre başarısız görünse de yüz sınıflı veri kümesinde bunun normal olduğunu varsayılabilir.

Tablo 3.5. CIFAR100 veri kümesinde sınıfa göre sonuçlar

Sınıf	Kesinlik	Duyarlılık	F skoru
apple	0,817	0,760	0,788
aquarium_fish	0,630	0,580	0,604
baby	0,447	0,420	0,433
bear	0,512	0,220	0,308
beaver	0,375	0,300	0,333
bed	0,487	0,550	0,516
bee	0,604	0,580	0,592
beetle	0,653	0,620	0,636
bicycle	0,478	0,760	0,587
bottle	0,679	0,570	0,620
bowl	0,282	0,220	0,247
boy	0,398	0,330	0,361
bridge	0,537	0,650	0,588
bus	0,413	0,520	0,460
butterfly	0,381	0,530	0,444
camel	0,543	0,510	0,526
can	0,559	0,620	0,588
castle	0,622	0,740	0,676
caterpillar	0,511	0,470	0,490
cattle	0,512	0,430	0,467
chair	0,672	0,780	0,722
chimpanzee	0,753	0,580	0,655
clock	0,362	0,510	0,423
cloud	0,690	0,690	0,690
cockroach	0,817	0,670	0,736

couch	0,442	0,380	0,409
crab	0,302	0,520	0,382
crocodile	0,404	0,400	0,402
cup	0,633	0,620	0,626
dinosaur	0,581	0,500	0,538
dolphin	0,620	0,490	0,547
elephant	0,663	0,570	0,613
flatfish	0,495	0,480	0,487
forest	0,427	0,530	0,473
fox	0,683	0,560	0,615
girl	0,419	0,310	0,356
hamster	0,662	0,530	0,589
house	0,529	0,550	0,539
kangaroo	0,373	0,380	0,376
keyboard	0,529	0,720	0,610
lamp	0,570	0,450	0,503
lawn_mower	0,755	0,710	0,732
leopard	0,306	0,640	0,414
lion	0,627	0,520	0,568
lizard	0,309	0,290	0,299
lobster	0,398	0,370	0,383
man	0,415	0,340	0,374
maple_tree	0,573	0,590	0,581
motorcycle	0,613	0,840	0,709
mountain	0,663	0,670	0,667
mouse	0,388	0,310	0,344
mushroom	0,509	0,550	0,529
oak_tree	0,556	0,690	0,616
orange	0,709	0,830	0,765
orchid	0,704	0,570	0,630
otter	0,378	0,170	0,234
palm_tree	0,861	0,680	0,760
pear	0,700	0,490	0,576
pickup_truck	0,682	0,580	0,627
pine_tree	0,571	0,520	0,545
plain	0,783	0,720	0,750
plate	0,496	0,630	0,555
poppy	0,705	0,620	0,660
porcupine	0,461	0,590	0,518
possum	0,415	0,390	0,402
rabbit	0,533	0,400	0,457
raccoon	0,500	0,620	0,554
ray	0,485	0,470	0,477
road	0,809	0,760	0,784
rocket	0,728	0,670	0,698
rose	0,663	0,610	0,635
sea	0,670	0,710	0,689
seal	0,359	0,230	0,280

shark	0,480	0,360	0,411
shrew	0,366	0,450	0,404
skunk	0,802	0,730	0,764
skyscraper	0,711	0,810	0,757
snail	0,594	0,410	0,485
snake	0,280	0,470	0,351
spider	0,550	0,550	0,550
squirrel	0,433	0,260	0,325
streetcar	0,475	0,570	0,518
sunflower	0,871	0,810	0,839
sweet_pepper	0,527	0,490	0,508
table	0,471	0,410	0,439
tank	0,667	0,660	0,663
telephone	0,424	0,610	0,500
television	0,452	0,560	0,500
tiger	0,529	0,730	0,613
tractor	0,587	0,610	0,598
train	0,580	0,470	0,519
trout	0,616	0,690	0,651
tulip	0,551	0,540	0,545
Turtle	0,380	0,300	0,335
wardrobe	0,700	0,840	0,764
whale	0,617	0,580	0,598
willow_tree	0,632	0,430	0,512
wolf	0,582	0,530	0,555
woman	0,352	0,250	0,292
worm	0,439	0,610	0,510
ortalama	**0,551**	**0,541**	**0,539**

Tablo 3.4 ve 3.5'in gösteriği üzere her bir sınıf için kesinlik, duyarlılık ve F skorları dengeli bir şekilde dağıldığı ve aşırı uyumun olmadığı gözlemlenmiştir.

3.4. Diğer Çalışmalar ile Karşılaştırma

Deneyler sonucu olgunlaştırılan MAHNet modelinin başarısı veri kümelerinde sınandıktan sonra literatürde aynı veri kümeleri üzerinde yapılan çalışmalarda kullanılan diğer derin öğrenme modellerinin doğruluk değerleriyle kıyaslanmıştır. Önerilen MAHNet modeli, doğruluk değeri bakımından diğer çalışmalarda kullanılan yöntemlerle Tablo 3.6'te karşılaştırılmıştır.

Tablo 3.6. MAHNet ve diğer modellerin CIFAR veri kümeleri üzerindeki başarım karşılaştırması

Model adı	Parametre Sayısı	CIFAR10 Doğruluk (%)	CIFAR100 Doğruluk (%)
MAHNet (Önerilen Model)	0,67 Milyon	87,86	54,11
AlexNet (DFA) [62]	-	62,70	48,03
AlexNet (FA) [62]	-	60,45	19,49
Sign-symmetry [63]	-	80,98	47,75
CLTV [64]	1,3 Milyon	86,99	60,11
ResNet20+UnsharpMaskLayer [65]	-	-	60,36
VisionNyströmformer [66]	0,53 Milyon	65,06	-
WaveMix [67]	9,2 Milyon	85,21	-
FLSCNN	-	75,86	-
Resnet-156 [2]	0,85 Milyon	93,03	74,84
Densenet (k=12) [35]	1,0 Milyon	93,00	72,45
RoR-110+SD [68]	1.7 Milyon	94,92	76,40
Highway [69]	-	92,68	67,61
FractalNet [70]	30 Milyon	95,41	77,15
ResNet-164 [60]	2,5 Milyon	94,07	74,84
ResNet in ResNet [71]	10,3 Milyon	94,99	77,10
WRN28-10 [72]	36,5 Milyon	95,83	79,50
multi-resnet [73]	145 Milyon	96,27	80,40
ELU [74]	-	93,45	75,72

MAHNet modeli başlangıçta 1,7 milyon parametre ile başladıktan sonra modüllerin elenmesiyle 0,67 milyon parametreye düşmüştür. CIFAR10 veri kümesinde %87,86 doğruluğa ulaşılmışken, CIFAR100 veri kümesinde de %54,11 doğruluğa ulaşılmıştır. CIFAR10 veriseti için diğer modellerle yarışabilecek bir model oluşmuştur. CIFAR100 için ise daha düşük ama başarılı sayılabilecek bir seviyededir.

SONUÇLAR

Bu tez çalışmasında, farklı evrişimsel sinir ağları mimarilerinin eşit şartlarda yarışarak dinamik bir şekilde oluşturulan ve her bir veri kümesine adaptif hibrit model oluşturulmuştur. Farklı mimarilerden oluşturulan aday modüllerle eğitilir ve seçim katmanı sayesinde seçim yapılır. Seçim katmanlarında öncesinde bağlı aday modülü kadar eğitilebilir ağırlıklar bulunur ve başlangıç değerleri eşittir. Modül ağırlıkları toplamları 1 olacak şekilde belirlenmiştir. Belirli eğitim sürecinin sonunda düşük ağırlıklı modüller çıkarttırılarak model küçültülür ve modele uyumlu adaptif hibrit model oluşturulur.

Modeli, yapılan deneylerde optimum sürece almak vakit almıştır. Modül ağırlıklarının uzaklaşması modülün seçilmesinde daha optimum değerler alınmasını sağlamıştır. Düşük ağırlıklı modülün katkısı daha az olduğundan modülü çıkardıktan sonra modelin toparlanması daha hızlı olacaktır. Bu yüzden seçim katmanlarında başlangıçta softmax kullanılsa da sonrasında toplamına bölünmesi daha iyi sonuçlar aldığı gözlemlenmiştir.

Modelde az modül kullanıldığı için daha hızlı iyi sonuçlar alsa da mevcut mimarilerle yarışabilir sonuçlara ulaşamamıştır. CIFAR10 veri kümesinde 87,86 doğruluk skorunu, CIFAR100 veri kümesinde de 54,11 doğruluk sonucuna ulaşmıştır. Bunun nedeni az modülün olmasından dolayı ve daha az parametre içermesinden dolayı modelin karmaşıklığı çözmekte zorlanmasıdır. Fazla modülün kullanılmasında ise hem uzun eğitim süreçlerinden dolayı hem de fazla parametreden dolayı oluşan aşırı uyum problemini çözmede zorluk yaşamaktadır. Veri çoğaltma teknikleri de aşırı uyumu çözse de yeterli başarımları ulaşamamasına neden olmuştur.

Önerilen MAHNet modelinin farklı veri kümelerine uyum sağlayan hibrit modelleri oluşturulmuştur. Buradan yola çıkarak farklı çalışmalarla farklı hibrit mimari modellerin de çıkarılması hedeflenmektedir. Ayrıca diğer mimarilerin de modele entegre edilmesi amaçlanacaktır.

Sonraki çalışmalarda modelin büyüklüğü artırılıp daha başarılı modellerin oluşması amaçlanacaktır. Böylelikle büyük veri kümelerinde daha başarılı sonuçlar alınması sağlanacaktır. Ayrıca farklı mimarilerden oluşturulan yeni modüller eklenerek çeşitlilik

de artırılacaktır. Böylelikle farklı modeller ile çalışılarak modelin başarımına katkısı olacağı öngörülmektedir.

KAYNAKLAR

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

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

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

[4] M. Cakar, K. Yildiz, and Y. Genc, "Multi Adaptive Hybrid Networks (MAHNet): Ensemble Learning in Convolutional Neural Network," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 2021: IEEE, pp. 1-4.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[6] L. C. Yan, B. Yoshua, and H. Geoffrey, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[7] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[8] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[9] F. Xie, Q. Gao, C. Jin, and F. Zhao, "Hyperspectral image classification based on superpixel pooling convolutional neural network with transfer learning," *Remote Sensing*, vol. 13, no. 5, p. 930, 2021.

[10] C. Rawles. "tf.keras and TensorFlow: Batch Normalization to train deep neural networks faster." https://towardsdatascience.com/how-to-use-batch-normalization-with-tensorflow-and-tf-keras-to-train-deep-neural-networks-faster-60ba4d054b73 (accessed 05.02, 2022).

[11] "Layer activation functions." https://keras.io/api/layers/activations/ (accessed 02.04.2022.

[12] E. Jeczmionek and P. A. Kowalski, "Flattening Layer Pruning in Convolutional Neural Networks," *Symmetry*, vol. 13, no. 7, p. 1147, 2021.

[13] D. Jiang, G. Hu, G. Qi, and N. Mazur, "A fully convolutional neural network-based regression approach for effective chemical composition analysis using near-infrared spectroscopy in cloud," *Journal of Artificial Intelligence and Technology*, vol. 1, no. 1, pp. 74-82, 2021.

[14] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, 1988, vol. 1, pp. 21-28.

[15] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[16] E. Ünal, "Derin Öğrenme-Giriş," June 01, 2020. [Online]. Available: https://enginunal.medium.com/derin-%C3%B6%C4%9Frenme-giri%C5%9F-dad1a98241d0.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] A. E. Çil, "DDoS Saldırılarının Tespiti ve Sınıflandırılması İçin Derin Öğrenme Modeli," Yüksek Lisans, Marmara Üniversitesi, 2021.
[19] "Model training APIs." https://keras.io/api/models/model_training_apis/ (accessed 01.04.2022.
[20] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*, 2020: PMLR, pp. 8093-8104.
[21] Y. LeCun *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, 1995, vol. 60: Perth, Australia, pp. 53-60.
[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097-1105, 2012.
[23] A. BOUGUETTAYA, A. KECHIDA, and A. M. TABERKIT, "A survey on lightweight CNN-based object detection algorithms for platforms with limited computational resources," *International Journal of Informatics and Applied Mathematics*, vol. 2, no. 2, pp. 28-44, 2019.
[24] NVIDIA. "CUDA Toolkit." https://developer.nvidia.com/cuda-toolkit (accessed 30.01.2022.
[25] NVIDIA. "NVIDIA cuDNN." https://developer.nvidia.com/cudnn (accessed 30.01.2022.
[26] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2015. [Online]. Available: https://www.tensorflow.org/.
[27] F. a. o. Chollet. "Keras." https://keras.io (accessed 30.01.2022.
[28] M. Rubashkin. "A Review of Available Tools." https://www.svds.com/getting-started-deep-learning/?utm_campaign=KDNuggets%20Blog&utm_source=KDNuggets (accessed 30.01.2022.
[29] Tensorflow. "TensorFlow 1.4.0." https://github.com/tensorflow/tensorflow/releases/tag/v1.4.0 (accessed 30.01.2022.
[30] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026-8037, 2019.
[31] R. Al-Rfou *et al.*, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, p. arXiv: 1605.02688, 2016.
[32] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251-1258.
[34] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700-4708.

[36] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697-8710.
[37] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019: PMLR, pp. 6105-6114.
[38] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64-67, 2001.
[39] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Improving cnn-rnn hybrid networks for handwriting recognition," in *2018 16th international conference on frontiers in handwriting recognition (ICFHR)*, 2018: IEEE, pp. 80-85.
[40] W. S. Noble, "What is a support vector machine?," *Nature biotechnology*, vol. 24, no. 12, pp. 1565-1567, 2006.
[41] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318-1325, 2012.
[42] G. Lowe, "Sift-the scale invariant feature transform," *Int. J*, vol. 2, no. 91-110, p. 2, 2004.
[43] T. Connie, M. Al-Shabi, W. P. Cheah, and M. Goh, "Facial expression recognition using a hybrid CNN–SIFT aggregator," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, 2017: Springer, pp. 139-149.
[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.
[45] R. M. Kamble *et al.*, "Automated diabetic macular edema (DME) analysis using fine tuning with inception-resnet-v2 on OCT images," in *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 2018: IEEE, pp. 442-446.
[46] K. Pawar, Z. Chen, N. J. Shah, and G. F. Egan, "Suppressing motion artefacts in MRI using an Inception-ResNet network with motion simulation augmentation," *NMR in Biomedicine*, p. e4225, 2019.
[47] Y. Zhou, X. Kang, and F. Ren, "Employing Inception-Resnet-v2 and Bi-LSTM for Medical Domain Visual Question Answering," in *CLEF (Working Notes)*, 2018.
[48] F. Baldassarre, D. G. Morín, and L. Rodés-Guirao, "Deep koalarization: Image colorization using cnns and inception-resnet-v2," *arXiv preprint arXiv:1712.03400*, 2017.
[49] A. Agga, A. Abbou, M. Labbadi, Y. El Houm, and I. H. O. Ali, "CNN-LSTM: An efficient hybrid deep learning architecture for predicting short-term photovoltaic power production," *Electric Power Systems Research*, vol. 208, p. 107908, 2022.
[50] A. Moradzadeh, H. Teimourzadeh, B. Mohammadi-Ivatloo, and K. Pourhossein, "Hybrid CNN-LSTM approaches for identification of type and locations of transmission line faults," *International Journal of Electrical Power & Energy Systems*, vol. 135, p. 107563, 2022.
[51] Q. Sun, X. Yu, H. Li, and J. Fan, "Adaptive feature extraction and fault diagnosis for three-phase inverter based on hybrid-CNN models under variable operating conditions," *Complex & Intelligent Systems*, vol. 8, no. 1, pp. 29-42, 2022.
[52] J. R. Quinlan, "Bagging, boosting, and C4. 5," in *Aaai/iaai, Vol. 1*, 1996, pp. 725-730.

[53] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, "Credit card fraud detection using AdaBoost and majority voting," *IEEE access,* vol. 6, pp. 14277-14284, 2018.
[54] Y. Zhou, T. A. Mazzuchi, and S. Sarkani, "M-adaboost-a based ensemble system for network intrusion detection," *Expert Systems with Applications,* vol. 162, p. 113864, 2020.
[55] F. Wang, D. Jiang, H. Wen, and H. Song, "Adaboost-based security level classification of mobile intelligent terminals," *The Journal of Supercomputing,* vol. 75, no. 11, pp. 7460-7478, 2019.
[56] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 141-142, 2012.
[57] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747,* 2017.
[58] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[59] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics,* 2011: JMLR Workshop and Conference Proceedings, pp. 215-223.
[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning,* 2015: PMLR, pp. 448-456.
[61] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters,* vol. 27, no. 8, pp. 861-874, 2006.
[62] M. B. Webster and J. Choi, "Learning the Connections in Direct Feedback Alignment," 2020.
[63] Q. Liao, J. Leibo, and T. Poggio, "How important is weight symmetry in backpropagation?," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30, no. 1.
[64] P. Jeevan, "Convolutional Xformers for Vision," *arXiv preprint arXiv:2201.10271,* 2022.
[65] J. Carranza-Rojas, S. Calderon-Ramirez, A. Mora-Fallas, M. Granados-Menani, and J. Torrents-Barrena, "Unsharp masking layer: injecting prior knowledge in convolutional networks for image classification," in *International Conference on Artificial Neural Networks*, 2019: Springer, pp. 3-16.
[66] P. Jeevan and A. Sethi, "Vision Xformers: Efficient Attention for Image Classification," *arXiv preprint arXiv:2107.02239,* 2021.
[67] A. Sethi, "WaveMix: Multi-Resolution Token Mixing for Images," 2021.
[68] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 28, no. 6, pp. 1303-1314, 2017.
[69] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.
[70] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648,* 2016.

[71] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029,* 2016.
[72] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146,* 2016.
[73] M. Abdi and S. Nahavandi, "Multi-residual networks: Improving the speed and accuracy of residual networks," *arXiv preprint arXiv:1609.05672,* 2016.
[74] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289,* 2015.

ÖZGEÇMİŞ

Adı Soyadı : Mahmut Çakar

Yabancı Dili : İngilizce

Öğrenim Durumu

Derece	Bölüm/Program	Üniversite/Lise	Mezuniyet Yılı
Lise	Anadolu Öğretmen Lisesi	Kahramanmaraş Anadolu Öğretmen Lisesi	Haziran 2010
Lisans	Elektronik ve Haberleşme Mühendisliği	İstanbul Teknik Üniversitesi	Haziran 2017

İş Deneyimi

Yıl	Firma/Kurum	Görevi
2017	Wipro Ltd.	Test Otomasyon Mühendisi
2018	Ekol Lojistik	Yazılım Mühendisi
2020	Digiturk	Yazılım Mühendisi

Bilimsel Eserler:

1. Çakar, M., Yıldız, K., & Demir, Ö. (2020, October). Creating Cover Photos (Thumbnail) for Movies and TV Series with Convolutional Neural Network. In 2020 Innovations in Intelligent Systems and Applications Conference (ASYU) (pp. 1-5). IEEE.
2. Çakar, M., Yıldız, K., & Demir, Ö. Thumbnail Selection with Convolutional Neural Network Based on Emotion Detection. International Journal of Advances in Engineering and Pure Sciences, 33, 88-93.
3. Gönül, P. D. C. E. Y., & Kayalar, E. (2021, May). Effects of COVID-19 Pandemic on Distance Workers. In II. International COVID-19 and Current Issues Congress.
4. Çakar, M., Yıldız, K., & Genç, Y. (2021, December). Multi Adaptive Hybrid Networks (MAHNet): Ensemble Learning in Convolutional Neural Network. In 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) (pp. 1-4). IEEE.