T.C.

AKDENİZ ÜNİVERSİTESİ

ENDÜSTRİYEL UYGULAMALARIN DERİN ÖĞRENME İLE YÜKSEK HIZLARDA GERÇEKLENMESİ İÇİN OPTMİZE EDİLMİŞ AĞ YAPILARININ GELİŞTİRİLMESİ

Hakan AKTAŞ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

ARALIK 2020

ANTALYA

T.C.

AKDENİZ ÜNİVERSİTESİ

ENDÜSTRİYEL UYGULAMALARIN DERİN ÖĞRENME İLE YÜKSEK HIZLARDA GERÇEKLENMESİ İÇİN OPTMİZE EDİLMİŞ AĞ YAPILARININ GELİŞTİRİLMESİ

Hakan AKTAŞ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

ARALIK 2020

ANTALYA

T.C.

AKDENİZ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL UYGULAMALARIN DERİN ÖĞRENME İLE YÜKSEK HIZLARDA GERÇEKLENMESİ İÇİN OPTMİZE EDİLMİŞ AĞ YAPILARININ GELİŞTİRİLMESİ

Hakan AKTAŞ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

Bu tez Akdeniz Üniversitesi Bilimsel Araştırmalar Projeleri Koordinasyon Birimi tarafından **FDK-2019-4879 nolu proje** ile ve TÜBİTAK tarafından **BİDEB/2211-C/1649B031900774 nolu proje** ile desteklenmiştir.

ARALIK 2020

T.C.
AKDENİZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENDÜSTRİYEL UYGULAMALARIN DERİN ÖĞRENME İLE YÜKSEK HIZLARDA GERÇEKLENMESİ İÇİN OPTMİZE EDİLMİŞ AĞ YAPILARININ GELİŞTİRİLMESİ

Hakan AKTAŞ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

DOKTORA TEZİ

Bu tez 14/12/2020 tarihinde jüri tarafından Oybirliği/~~Oyçokluğu~~ ile kabul edilmiştir.

Doç. Dr. Övünç POLAT (Danışman) **e-imzalıdır**

Prof. Dr. Şükrü ÖZEN **e-imzalıdır**

Prof. Dr. Ömer Halil ÇOLAK **e-imzalıdır**

Prof. Dr. Selçuk ÇÖMLEKÇİ **e-imzalıdır**

Doç. Dr. Mesud KAHRİMAN **e-imzalıdır**

ÖZET

ENDÜSTRİYEL UYGULAMALARIN DERİN ÖĞRENME İLE YÜKSEK HIZLARDA GERÇEKLENMESİ İÇİN OPTMİZE EDİLMİŞ AĞ YAPILARININ GELİŞTİRİLMESİ

Hakan AKTAŞ

Doktora Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Övünç POLAT

Aralık 2020; 82 sayfa

Derin öğrenmenin son yıllardaki hızlı yükselişi ile birlikte birçok endüstriyel problem derin öğrenme ile yapılabilir hale gelmiştir. Endüstrinin her alanında olduğu gibi tarım alanında da bitki hastalığı sınıflandırma, meyve sınıflandırma, tohum sınıflandırma ve tohum ayıklama gibi endüstriyel problemler bulunmaktadır. Bu tez çalışmasında tohum sınıflandırma ve tohum ayıklama problemi evrişimli sinir ağ yapıları ile gerçeklenmiştir. Tohum sınıflandırma işleminde veri seti olarak sarı buğday, pirinç, kırmızı mercimek ve yeşil mercimek tohumları kullanılmış ve sınıflandırma işlemi bu tohumlar arasında yapılmıştır. Tohum ayıklama problemi için gerekli veri seti kara buğday tohumları ve bu tohumların içinden çıkan istenmeyen nesnelerden üretilmiştir. Tüm bu veri setlerini üretebilmek için probleme özgü endüstriyel bir deney düzeneği kurulmuştur. Yine önerilen algoritma sayesinde farklı türlerde (doğrudan ölçekleme ve şablona uygun şekilde ölçekleme) ve boyutlarda veri setlerinin otomatik bir şekilde oluşturulması sağlanmıştır. Bu tez çalışmasında tohum ayıklama problemi için 227x227, 157x157, 77x77 ve 37x37 boyutlarında ve farklı iki türde veri seti (doğrudan ölçekleme ve şablona uygun şekilde ölçekleme) olmak üzere toplamda sekiz farklı veri seti oluşturulmuştur. Oluşturulan bu sınıflandırma ve ayıklama veri setleri literatürdeki AlexNet ve MobileNet-V2 ağ yapıları ile test edilmiş ve en yüksek test doğruluğu tohum sınıflandırma için %100 (AlexNet ile) tohum ayıklama için %99 (MobileNet-V2 ile) olarak hesaplanmıştır.

Tohum ayıklama işlemi endüstride yüksek hızlarda görüntü işleme yöntemleri ile yapılmaktadır. Ayıklama işleminin derin öğrenme ile yüksek hızlarda yapılabilmesi için iki evrişim katmanından oluşan (ilk evrişim katmanındaki filtre sayısı 64, ikinci evrişim katmanındaki filtre sayısı 96), işlem yükü düşük özel bir evrişimli sinir ağı yapısı (giriş data boyutu 227x227) önerilmiştir. Önerilen bu özel evrişimli sinir ağı yapısı 227x227 boyutundaki veri seti ile test edildikten sonra en yüksek test doğruluğu %99 olarak hesaplanmıştır. Doğruluktan ödün vermeden (minimum %98 doğruluk) önerilen bu ağ yapısındaki işlem yükünü daha da azaltmak için iki aşamalı ölçekle ve budama yöntemi önerilmiştir. Önerilen yöntemin özel evrişimli sinir ağı üzerinde denenmesi sonucu giriş data boyutu 77x77, birinci evrişim katmanındaki filtre sayısı 16 ve ikinci evrişim katmanındaki filtre sayısı 24 olan özel ağ yapısı en optimum ağ yapısı olarak hesaplanmıştır. Bu optimum ağ yapısına ait toplam kayan nokta işlem sayısının (FLOPs), ilk önerilen ağ yapısına (giriş veri boyutu 227x227) oranla 46.072.682 / 863.530 = 53,35 kat daha küçük olduğu yine MobileNet-V2 ağ yapısına göre 1.690 kat daha küçük olduğu hesaplanmıştır.

Önerilen ağ yapıları üzerindeki optimizasyon işlemleri genetik algoritma ile de denenmiş ve genetik algoritma ile elde optimum (minimum FLOPs) ağ yapısının FLOPs değerinin, özel ağ yapısına (giriş data boyutu 227x227) oranla 46.072.682/715.453 = 64,39 kat daha küçük olduğu hesaplanmıştır. Genetik algoritma ve önerilen ölçekle ve budama yöntemi ile yapılan optimizasyon sonuçları detaylı bir şekilde karşılaştırılmış; tüm sonuçlar gerçek zamanlı sistemler ve gömülü sistemler açısından detaylı bir şekilde tartışılmıştır. Ölçekle ve budama yöntemi ile yapılan optimizasyonda elde edilen özel ağ yapısının (giriş data boyutu 77x77) hafızada kapladığı alan 0,96MB olarak hesaplanmıştır. Bu değer MobileNet-V2 ağ yapısına göre 12,9MB/0,96MB = 13,43 kat ve AlexNet yapısına göre 236MB/0,96MB = 245,83 kat daha küçüktür. Aynı oranlar toplam parametre sayısı için de geçerli olup; bu açıdan bakıldığında optimize edilmiş ağ yapılarının, literatürde yaygın olarak bilinen mimarilere kıyasla daha donanım dostu olduğu söylenebilir.

Sonuç olarak probleme-özel evrişimli sinir ağ modellerinin önerilmesi ve optimize edilmesi alanında literatürde geliştirilmeye açık konuların olduğu gösterilmiştir. Bu çalışmada elde edilen sonuçlar göstermektedir ki yüksek hız gerektiren uygulamalarda literatürdeki bilinen ağ yapılarını kullanmak yerine probleme özgü ağ yapılarının önerilmesi işlem yükünü azaltmakta ve sistemi hızlandırmaktadır.

ANAHTAR KELİMELER: Derin Öğrenme, Evrişimli Sinir Ağlar, Genetik Algoritma, Gerçek Zamanlı Sistemler, Optimizasyon Teknikleri, Tohum Ayıklama, Uygulamaya Özel CNN Modelleri

JÜRİ: Doç. Dr. Övünç POLAT

Prof. Dr. Şükrü ÖZEN

Prof. Dr. Ömer Halil ÇOLAK

Prof. Dr. Selçuk ÇÖMLEKÇİ

Doç. Dr. Mesud KAHRİMAN

ABSTRACT

DEVELOPMENT OF OPTIMIZED NETWORK ARCHITECTURES FOR HIGH SPEED INDUSTRIAL APPLICATIONS USING DEEP LEARNING

Hakan AKTAŞ

PhD Thesis in Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Övünç POLAT

December 2020; 82 pages

Through the rapid rise of deep learning in recent years, many industrial problems have become feasible by the use of deep learning. As in every field of industry, there are industrial problems including plant disease classification, fruit classification, seed classification and seed sorting in agriculture as well. In this thesis, the seed classification and seed sorting problems has been implemented with convolutional neural network structures. For the process of seed classification, yellow wheat, rice, red lentil and green lentil seeds were used as the data set and the classification process was executed on these seeds. The data set required for the seed sorting problem has been obtained from buckwheat seeds and intruding objects that come out of these seeds. In order to make this realization, an industrial experimental setup which is specific to the problem has been established. Again, the proposed algorithm, allowed different types of datasets (direct scaling and scaling according to the template) and data sizes to be created automatically. In this thesis, for seed sorting problem a total of eight different data sets were generated, with two different data types (scaling directly and scaling according to the template) and four different image sizes (227x227, 157x157, 77x77 and 37x37). All of these classification and sorting data sets were tested with AlexNet and MobileNet-V2 network structures in the literature and the highest test accuracy was calculated as %100 (with AlexNet) for seed classification and %99 (with MobileNet-V2) for seed sorting.

In industry, seed sorting can be executed by image processing methods at high speeds. In order to perform seed sorting applications with deep learning at high speeds a special convolutional neural network structure (input data size 227x227) consisting of two convolution layers (in first and second convolution layers, there are 64 filters and 96 filters respectively) with a low processing load is proposed. After testing this custom convolutional neural network structure with 227x227 data set, the highest test accuracy was calculated as 99%. In order to further reduce the processing load on this proposed custom network structure without sacrificing accuracy (minimum 98% accuracy), a two-step scale and prune method have been proposed. As a result after testing this proposed method on custom convolutional neural network providing that input data size is 77x77, number of filters in the first and the second convolution layers have been calculated as 16 and 24 respectively for a optimal network structure. It is also calculated that the total number of floating-point operations (FLOPs) of this network structure is 46,072,682 / 863,530 = 53.35 times smaller than the initially proposed custom network structure (input data size 227x227) and 1,690 times smaller than the MobileNet-V2 network structure.

Optimization processes on the proposed network structures were also tested with the genetic algorithm, and it was calculated that the FLOPs value of the optimum (minimum FLOPs) network structure obtained with the genetic algorithm is 46,072,682 / 715,453 = 64.39 times smaller than the custom network structure (input data size 227x227). Optimization results with the genetic algorithm and the proposed scaling and pruning method were compared in details and all results were discussed in detail in terms of real time systems and embedded systems. The required memory for the custom network structure (input data size 77x77) obtained from scale and prune method was calculated as 0.96MB. This value is 12.9MB / 0.96MB = 13.43 times smaller than the MobileNet-V2 network structure and 236MB / 0.96MB = 245.83 times smaller than the AlexNet structure. The same rates are valid for the total number of parameters; from this point of view, it can be said that optimized network structures are more hardware friendly compare to literally known architectures.

As a result, it has been shown that in the literature there are issues that are likely be improved in the field of proposing and optimizing problem-specific custom convolutional neural network structures. The results obtained in this study show that problem-specific custom network structures proposed for high speed applications instead of using the known network structures in the literature reduces the processing load and speeds up the system.

KEYWORDS: Deep Learning, Convolutional Neural Networks, Deep Learning, Genetic Algorithm, Real Time Systems, Optimization Techniques, Seed Sorting, Application Specific CNN Models

COMMITTEE: Assoc. Prof. Dr. Övünç POLAT

Prof. Dr. Şükrü ÖZEN

Prof. Dr. Ömer Halil ÇOLAK

Prof. Dr. Selçuk ÇÖMLEKÇİ

Assoc. Prof. Dr. Mesut KAHRİMAN

ÖNSÖZ

Doktora dönemim boyunca yardımlarını ve desteklerini benden esirgemeyen, aldığım kararlarda beni her zaman destekleyen, bu konuda çalışmak için beni cesaretlendiren danışman hocam Doç. Dr. Övünç POLAT'a teşekkürü bir borç bilirim. Tezime sağladığı katkılardan dolayı Prof. Dr. Ömer Halil ÇOLAK ve Doç. Dr. Mesut KAHRİMAN hocama, yüksek lisans dönemimde danışman hocam olan, doktora çalışmalarımın temellerini atan Dr. Refik SEVER'e ve doktorada bir dönem beraber çalıştığımız Prof. Dr. Bekir Taner SAN hocama ve sekiz yıldır Akdeniz Üniversitesi Elektrik-Elektronik Mühendisliği bölümünde beraber çalıştığımız çok kıymetli hocalarıma da teşekkür ve saygılarımı sunarım.

Doktora tez dönemimde çalışmalarımda beni motive eden, 2211 Yurt İçi Lisansüstü Burs Programı ile beni destekleyen, ülkemizin en güzide kurumlarından biri olan TÜBİTAK'a teşekkürlerimi bir borç bilirim.

Maddi manevi her zaman yanımda olan, tüm zorlukları aşmamda beni destekleyen hayat arkadaşım Özge AKTAŞ'a, bu uzun yolculukta benimle her zaman gurur duyan canım annem Türkan AKTAŞ'a , beni her zaman destekleyen canım aileme ve son olarak ne kadar yorgun olsam da bir gülümsemesiyle bana dünyaları veren biricik oğlum Alperen AKTAŞ'a sonsuz ve en samimi teşekkürlerimi sunuyorum.

İÇİNDEKİLER

AKADEMİK BEYAN

Doktora Tezi olarak sunduğum "Endüstriyel Uygulamaların Derin Öğrenme ile Yüksek Hızlarda Gerçeklenmesi için Optimize Edilmiş Ağ Yapılarının Geliştirilmesi" adlı bu çalışmanın, akademik kurallar ve etik değerlere uygun olarak yazıldığını belirtir, bu tez çalışmasında bana ait olmayan tüm bilgilerin kaynağını gösterdiğimi beyan ederim.

14/12/2020

Hakan AKTAŞ

SİMGELER VE KISALTMALAR

Simgeler

fps : Frame per second

K : Kilo

KB : Kilobayt

M : Mega

MB : Megabayt

ms : milisaniye

sn : saniye

% : yüzde

Tez yazımında ondalık ayracı olarak nokta (.) kullanılmıştır.

Kısaltmalar

ACT : Aktivasyon Katmanı

BN : Batch Normalization Layer

CNN : Convolutional Neural Network

Conv : Convolution

CPU : Central Processing Unit

DO : Dropout Layer

FC : Fully-Connected Layer

FLOPs : The Number of Floating Point Operations

FLOPS : The Number of Floating Point Operations Per Second

FPGA : Field Programmable Gate Array

GPU : Graphic Processing Unit

GHz : GigaHertz

GFLOPs : GigaFLOPs

HSV : Hue Saturation Value

NCM : Nine Color Model

Pool : Pooling

RELU : Rectified Linear Unit

RGB : Red Green Blue

SGDM : Stochastic Gradient Descent with Momentum

TBK : Tamamen Bağlı Katman

TFLOPs : TeraFLOPs

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

İnsanlar tarafından gerçekleştirilen zihinsel görevlerin makineler tarafından otomatik olarak gerçekleştirilmesi olarak tanımlanan yapay zekanın gelişimi, son yıllarda büyük bir ivme kazanmıştır. Bu ivmelenmenin en büyük sebebi, büyük veri ve derin öğrenme alanında başarılı çalışmalara imza atılmış olmasıdır. Derin öğrenme, makine öğrenmesinin ve doğal olarak yapay zekanın bir alt kümesidir. Kendi başına öğrenmeye ve geliştirmeye dayalı bir alandır. Derin öğrenmenin en sık kullanıldığı alanlar ise görüntü tanıma ve sınıflandırma problemleridir. Evrişimli sinir ağları, sınıflandırma için geliştirilmiş özel bir derin sinir ağı modelidir. Evrişimli sinir ağlarındaki gelişmeler ile günümüzdeki en zor sınıflandırma problemleri bile bu ağ yapıları ile yüksek doğruluklarda gerçeklenebilmektedir. Evrişimli sinir ağ yapıları ile yapılan bu yüksek doğruluktaki sınıflandırma işlemleri literatürde ve endüstride kendine çok geniş bir yer bulmuş olup; günümüzdeki birçok sınıflandırma probleminin evrişimli sinir ağ yapıları ile gerçeklendiğinden bahsedebiliriz.

Nesne sınıflandırma, endüstride birçok alanda kullanılan ve derin öğrenmenin temel uygulama alanlarından biridir. Literatürde nesneleri sınıflandırmak için kullanılan birçok algoritma ve teknik vardır. Makine öğrenmesi algoritmaları bunlardan bir tanesidir. Ancak her yeni sınıflandırma problemi için sınıflandırılmak istenilen nesneye ait özniteliklerin bazı algoritmalar ile çıkarılması ve devamında bu özniteliklerin makine öğrenmesi algoritmaları ile sınıflandırılması gerekmektedir. Bu da makine öğrenmesi algoritmalarını zaman maliyeti yüksek ve uygulamada verimsiz kılmaktadır. Günümüz sınıflandırma problemleri daha çok derin öğrenme ile yapılmaktadır. Derin öğrenme, makine öğrenmesinin özelleşmiş bir alt kümesi olup; en bilindik derin öğrenme yapıları evrişimli sinir ağlarıdır (CNN-Convolutional Neural Network). CNN yapıları, görüntünün içeriğini anlamak için en iyi öğrenme algoritmalarından biridir ve günümüzde, nesne sınıflandırma (Ruvalcaba-Cardenas vd. 2019), nesne tanıma (Mureşan ve Oltean 2018), yüz algılama (Mehta vd. 2018), konuşma tanıma (Nassif vd. 2019), hiper spektral görüntülerin süper çözünürlükte görüntülere dönüştürülmesi (Hu vd. 2020), plaka tanıma (Elihos vd. 2019), hastalık tanıma (Abade vd. 2020) vb. gibi birçok uygulamada yaygın olarak kullanılmaktadır.

Sınıflandırma işlemi tarım alanında da yaygın olarak kullanılmaktadır. Son yıllardaki gelişmeyle birlikte tarımsal sınıflandırma problemleri için derin öğrenmenin kullanılması, bu sorunları daha uygulanabilir hale getirmekte ve sonuçlar yüksek doğrulukta çalışmaktadır. Tarım endüstrideki sınıflandırma işlemlerinden bir tanesi de tohum ayıklama problemidir. Tohum ayıklama işlemi endüstride çok yaygın bir şekilde kullanılmakta olup; modern dünyamızda çiftçinin topladığı tohum mahsullerinin neredeyse tamamı paketlenmeden önce ayıklama makineleri üzerinde ayıklanmaktadır. Ayıklama makineleri ve teknolojileri yaklaşık 15 yıllık bir geçmişe sahiptir. Geleneksel tohum ayıklama makineleri görüntü işleme yöntemleri ile çalışmaktadır. Ayıklama işlemi renk histogramındaki farklılık üzerinden yapılmaktadır. Bu makinelerdeki en büyük sorun hatalı ve hatasız ürünlerin renk histogramlarının birbirlerine benzer olmaları durumunda hatasız ürünün hatalı olarak kabul edilmesidir. Yine bu makinelerde farklı kamera teknolojileri kullanılarak NIR kameralar ile daha yüksek doğrulukta ayıklama işlemi yapılmaktadır. Kullanılan kameranın değişmesi ile doğruluk oranının artmasına rağmen tohum ayıklama problemindeki en büyük sorunlardan biri ayıklanmak istenen her yeni tohum için ayrı bir araştırma ve geliştirme yapılması

gerekmektedir. Derin öğrenme sayesinde ayıklanmak istenen her yeni tohum için yeni bir algoritma geliştirmek yerine iyi tohum ve istenmeyen nesnelerin sisteme tanıtılması yeterli olacaktır. Sisteme tanıtılması istenilen bu yeni tohumlar için öncelikle veri setlerinin oluşturulması gerekmektedir. Veri setlerinin oluşturulması derin öğrenme uygulamalarında birincil işlem olarak göze çarpmaktadır. Ağın yüksek doğruluklarda çalışabilmesi için yeterli miktarda verinin toplanması gerekmektedir. Tohum ayıklama işlemi kendisine hem literatürde hem de endüstride geniş bir yer bulmaktadır. Probleme literatür açısından bakıldığında veri setleri basit yöntemlerle (tohumlar masa gibi düz bir zeminin üzerine serilerek görüntüleri çekilmektedir) oluşturulmaktadır. Ancak bu probleme endüstriyel açıdan bakıldığında veri setlerinin endüstriyel ortamlara uygun olarak oluşturulması gerektiği şüphe gerektirmeyecektir. Bu tez çalışmasının diğer bir hedefi de tohum ayıklama gibi geleneksel ayıklama problemleri için genelleştirilmiş bir veri seti deney düzeneğinin geliştirilmesi ve farklı tiplerde veri seti oluşturan otomatik bir yazılımın geliştirilmesidir.

Bu tez çalışmasında kara buğday tohumlarının ayıklanması evrişimli sinir ağ yapıları ile gerçeklenmiştir. Tohum ayıklama işlemi endüstride çok yüksek hızlarda yapılmakta olup; evrişimli sinir ağ yapıları derin ağlardan oluştuğu için doğası gereği işlem yükü çok fazladır. Literatürdeki gelişmelere bakıldığında CNN yapılarını hızlandırmak için budama (Han vd. 2015) ve optimizasyon işlemleri (Jiang vd. 2018) önerilmekte, bu sayede mevcut ağ yapıları daha hızlı hale getirilmektedir. Öte yandan daha az işlem yüküne sahip ShuffleNet (Zhang vd. 2018), EfficientNet (Tan ve Le 2019) gibi yeni ağ yapıları da geliştirilmekte ve literatürde kendilerine yer bulmaktadırlar. Tüm bu gelişmeler yaşanırken eğitim ve test aşamaları ImageNet (Deng vd. 2010) gibi çok büyük veri setleri üzerinde yapılmaktadır. Peki endüstriyel uygulamalarda ne kadar derin ağlara ihtiyaç duyulmakta? Çünkü literatürdeki geliştirilen ağ yapıları çok büyük veri setlerini yüksek doğruluklarda sınıflandırabilecek kapasitedeki derin ağlardır. Ancak endüstriyel uygulamalarda uygulamanın çeşidine göre çok büyük veri setlerine ve çıkış katmanında çok fazla sınıfa ihtiyaç duyulmamaktadır. Örneğin hatalı ürün ayıklama işlemi derin öğrenme ile yapılmak istenirse çıkış katmanında iki sınıf olacaktır. Hatalı ve hatasız ürünlere ait yeterli sayıda veri seti oluşturulduğu takdirde bu veri setleri literatürdeki ağ yapıları ile eğitilip test edildiğinde sınıflandırma işlemi yüksek doğrulukta gerçekleşecektir. Başka bir deyişle, endüstriyel bir problem için literatürdeki ağ yapıları kullanıldığında sınıflandırma işlem sonucu yüksek olacaktır. Ancak bu uygulamalar yüksek hızlarda yapılmak istenirse gerçekten bu kadar derin ağlara ihtiyaç var mı? Literatürdeki bilindik ağ modellerini kullanmak yerine uygulamaya özel geliştirilmiş, yüksek doğrulukta sınıflandıran ve düşük işlem yüküne sahip (FLOPs) özel ağ modelleri geliştirilebilir mi? Bu soru aslında tezin ana fikrini oluşturmaktadır. Bu tez çalışmasında tohum ayıklama işlemini, düşük işlem yüküne sahip evrişimli sinir ağ yapıları ile yapabilmek için probleme özgün özel ağ yapıları önerilmiş ve devamında bu ağ yapıları farklı optimizasyon teknikleri ile optimize edilmiştir.

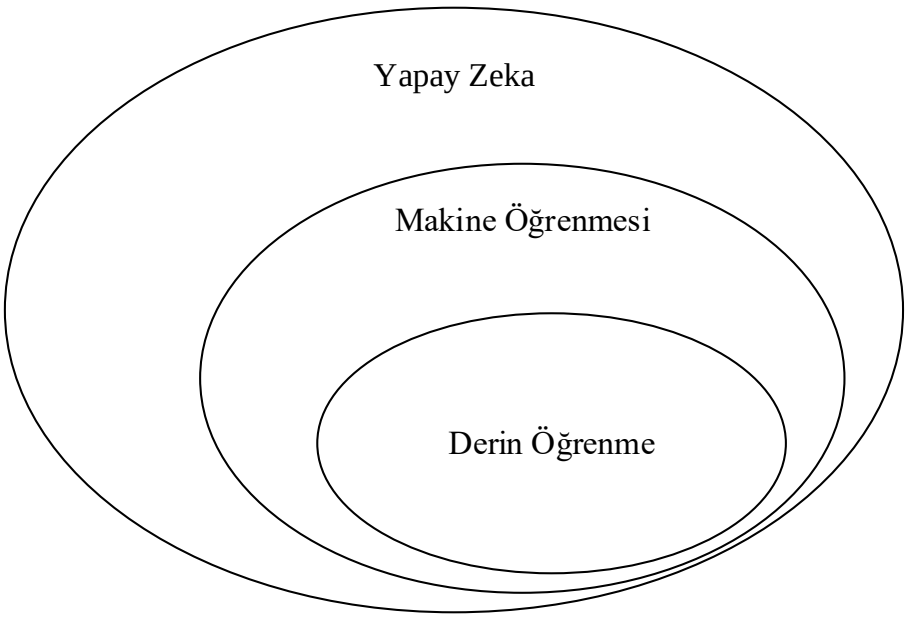
Bu tez çalışmasının birinci bölümünde problemin kısa tanımı ve tezin ana kapsamından bahsedilmiştir. İkinci bölümde tezdeki teorik bilgilere, literatürde CNN yapılarındaki gelişmelere, tohum sınıflandırma ve ayıklama üzerine yapılmış çalışmalara detaylı bir şekilde yer verilmiştir. Yine bu bölümde tezin katkısından ve çalışmanın öneminden bahsedilmiştir. Üçüncü bölümde tohum sınıflandırma ve

ayıklama için kullanılan deney düzeneği, geliştirilen otomatik veri seti oluşturma algoritması, önerilen özel ağ modeli ve iki aşamalı ölçekle ve budama metodu detaylı bir şekilde anlatılmıştır. Yine evrişimli sinir ağlarının işlem yükü ve parametre sayısı ile karmaşıklık seviyesinin tanımlanması için tüm katmanlardaki FLOPs ve parametre hesabı detaylı bir şekilde anlatılmıştır. Dördüncü bölümde ise deneysel gerçeklemeler yapılmıştır. Tohum sınıflandırma ve ayıklama problemi ilk olarak eğitilmiş ağ yapıları ile denenmiştir. Bir sonraki aşamada bu veri setleri önerilen özel ağ modeli ile test edilmiştir. Yine önerilen özel ağ modelini optimize etmek için ilk olarak önerilen ölçekle ve budama metodu devamında ise genetik algoritma ile optimizasyon yapılmıştır. Tüm deney sonuçları gerçek zamanlı ve gömülü sistemler açısından detaylı bir şekilde incelenmiş ve tartışılmıştır. Beşinci bölümde ise sonuç ve gelecek çalışmalardan bahsedilmiştir.

2. KAYNAK TARAMASI

2.1. Yapay Zeka

Yapay zeka, 1950'lerde, bilgisayar bilimi öncülerinin, bilgisayarlara düşünme yeteneğinin nasıl gerçekleştirilebileceğini sormaya başlamasıyla doğmuştur. Bu soru, bugün hala sonuçları araştırılmakta olunan bir sorudur. Yapay zeka için kısa bir tanım şu şekildedir: normalde insanlar tarafından gerçekleştirilen entelektüel görevleri otomatikleştirme çabası. Bu nedenle, yapay zeka, makine öğrenimi ve derin öğrenmeyi kapsayan ancak aynı zamanda herhangi bir öğrenmeyi içermeyen daha birçok yaklaşımı da içeren genel bir alandır. Örneğin, ilk satranç programları, yalnızca programcılar tarafından hazırlanmış kodlanmış kuralları içeriyordu ve makine öğrenimi olarak nitelendirilmiyordu. Oldukça uzun bir süre boyunca, birçok uzman, programcıların bilgiyi manipüle etmek için yeterince geniş bir dizi açık kural oluşturarak insan düzeyinde yapay zekaya ulaşılabileceğine inanıyordu. Bu yaklaşım sembolik yapay zeka olarak bilinmektedir ve 1950'lerden 1980'lerin sonlarına kadar yapay zekada baskın paradigmaydı (Chollet 2018). En yüksek popülaritesine 1980'lerin uzman sistem patlaması sonrasında ulaştı. Sembolik yapay zeka, satranç oynamak gibi iyi tanımlanmış mantıksal problemleri çözmek için uygun olsa da görüntü sınıflandırması, konuşma tanıma ve dil çevirisi gibi daha karmaşık, bulanık problemleri çözmek için açık kuralları bulmanın zor olduğu ortaya çıkmıştır. Makine öğrenmesi sembolik yapay zekanın yerini almak için yeni bir yaklaşım olarak ortaya çıkmıştır.



Şekil 2.1. Yapay zeka, makine öğrenmesi, derin öğrenme

2.2. Makine Öğrenmesi

Kısaca, makine öğrenmesi verileri içeren bir modelleme tekniğidir. Başka bir tanımla, makine öğrenmesi verilerin modelini ortaya çıkaran bir tekniktir (Kim 2017). Burada "veri" tam olarak belge, ses, görüntü gibi bilgiler anlamına gelir. "Model" ise makine öğrenmesinin son ürünüdür. Makine öğrenmesinde modelleme sürecinde

kullanılan verilere "eğitim" verileri denmektedir. Şekil 2.2 makine öğrenmesi sürecini göstermektedir (Kim 2017).

Şekil 2.2. Makine öğrenmesi blok diyagram gösterimi

Çeşitli alanlardaki problemleri çözmek için birçok farklı makine öğrenmesi tekniği geliştirilmiştir. Bu makine öğrenmesi teknikleri eğitim yöntemine bağlı olarak üç türe ayrılmaktadır.

- Denetimli öğrenme
- Denetimsiz öğrenme
- Pekiştirmeli öğrenme

2.2.1. Denetimli öğrenme

Denetimli öğrenme açık ara en yaygın olarak kullanılan makine öğrenmesi tipidir. Denetimli öğrenmede, her eğitim veri seti, girdi ve doğru çıktı çiftlerinden oluşmalıdır. Doğru çıktı, modele verilen girdi için modelin üretmesi gereken değerdir. Denetimli öğrenmede öğrenme, aynı girdi için doğru çıktı ile modelden elde edilen çıktı arasındaki farkı azaltmak için bir modelin revizyonları dizisidir. Bir model mükemmel bir şekilde eğitilmişse, eğitim verilerinden gelen girdiye karşılık gelen doğru bir çıktı üretecektir.

2.2.2. Denetimsiz öğrenme

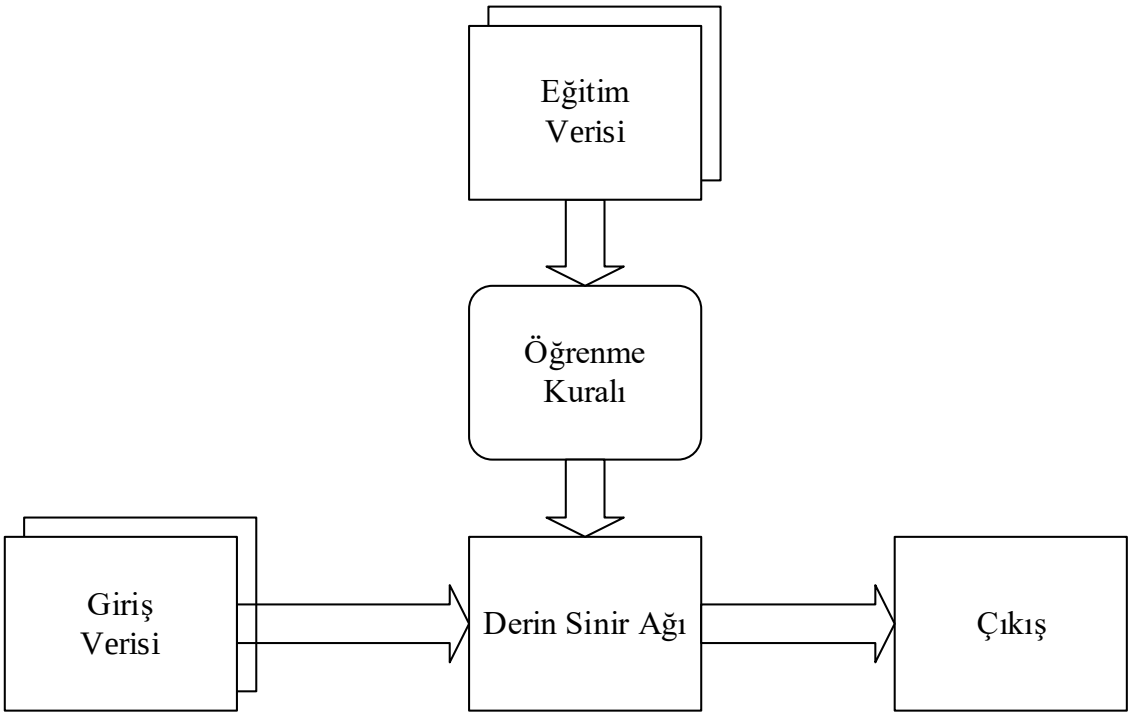
Denetimli öğrenmenin tersine, denetimsiz öğrenmede eğitim verileri, yalnızca doğru çıktıları olmayan girdileri içermektedir. İlk bakışta, doğru çıktılar olmadan nasıl eğitim yapılacağını anlamak zor görünebilir. Ancak, bu türden birçok yöntem halihazırda geliştirilmiştir. Denetimsiz öğrenme, genellikle verilerin özelliklerini araştırmak ve verileri önceden işlemek için kullanılır. Bu kavram, problemleri sadece yapı ve niteliklere göre sıralayan ve bunların nasıl çözüleceğini öğrenmeyen bir öğrenciye benzer çünkü bilinen doğru çıktılar yoktur.

2.2.3. Pekiştirmeli öğrenme

Pekiştirmeli öğrenme, eğitim verileri olarak girdi kümelerini, bazı çıktıları ve notları kullanır. Genellikle kontrol ve oyun oynama gibi optimal etkileşim gerektiğinde kullanılır. Pekiştirmeli öğrenmede, bir temsilci çevresi hakkında bilgi alır ve bazı ödülleri en üst düzeye çıkaracak eylemleri seçmeyi öğrenir. Örneğin, bir video oyunu ekranına bakan ve puanını en üst düzeye çıkarmak için oyun eylemlerini çıkaran bir sinir ağı, pekiştirmeli öğrenme yoluyla eğitilebilir. Şu anda, pekiştirmeli öğrenme çoğunlukla bir araştırma alanıdır ve oyunların ötesinde henüz önemli pratik başarıları olmamıştır.

2.3. Derin Öğrenme

Derin öğrenme, derin sinir ağını kullanan bir Makine Öğrenimi tekniğidir. Derin sinir ağı, iki veya daha fazla gizli katman içeren çok katmanlı sinir ağıdır. Şekil 2.3 Derin Öğrenme kavramını ve bunun makine öğrenmesi ile ilişkisini göstermektedir.



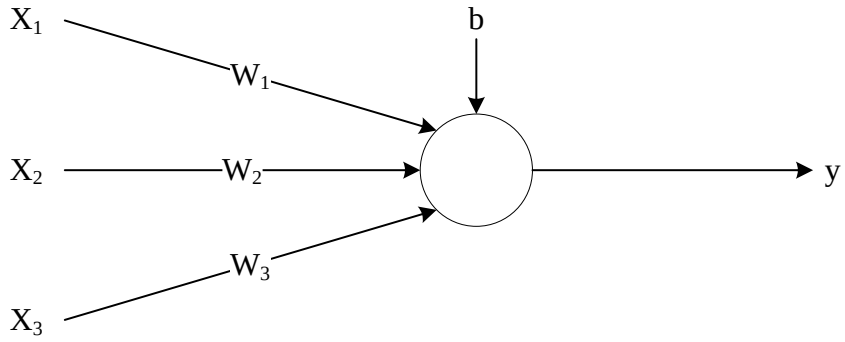
Şekil 2.3. Derin öğrenme kavramı ve makine öğrenmesi ile ilişkisi

Derin sinir ağı, makine öğrenmesinin son ürünü olarak görülmektedir. Öğrenme kuralı, eğitim verilerinden modeli (derin sinir ağı) oluşturan algoritma olarak tanımlanmaktadır.

2.4. Sinir Ağı Temelleri

2.4.1. Sinir düğüm yapısı

Sinir ağı, beynin mekanizmasından ilham alınarak geliştirilmiştir. Beyin, çok sayıda nöronun bağlantılarından oluştuğundan, sinir ağı, beynin nöronlarına karşılık gelen elemanlar olan düğümlerin bağlantılarından oluşur. Sinir ağı, ağırlık değerini kullanarak beynin en önemli mekanizması olan nöronların ilişkisini taklit eder. Sinir ağının mekanizmasını daha iyi anlamak için bir düğüm yapısı Şekil 2.4'de gösterilmiştir.



Şekil 2.4. 3 girişli bir düğüm örneği

Şekildeki daire ve ok sırasıyla düğümü ve sinyal akışını gösterir. x1, x2 ve x3 giriş sinyalleridir. w1, w2 ve w3 karşılık gelen sinyallerin ağırlıklarıdır. Son olarak, b, bilginin depolanmasıyla ilişkili bir başka faktör olan bias değeridir. Başka bir deyişle, sinir ağının bilgisi, ağırlık ve bias değeri şeklinde depolanır.

Dışarıdan gelen giriş sinyali, düğüme ulaşmadan önce ağırlık ile çarpılır. Ağırlıklandırılmış sinyaller düğümde toplandığında, bu değerler ağırlıklı toplam olarak eklenir. Bu örnek için ağırlıklandırılmış toplam, denklem (2.1)'deki gibi hesaplanır:

$$v = (w_1 * x_1) + (w_2 * x_2) + (w_3 * x_3) + b \tag{2.1}$$

w, 3x1'lik ve x, 1x3'lük bir matris olmak üzere; bu hesaplama aynı zamanda matris formatında denklem (2.2)'deki gibi yazılabilir.

$$v = wx + b \tag{2.2}$$

Son olarak, düğüm ağırlıklı toplamı aktivasyon işlevine girer ve çıktısını verir. Aktivasyon fonksiyonu denklem (2.3)'deki gibi tanımlanıp, düğümün davranışını belirler.

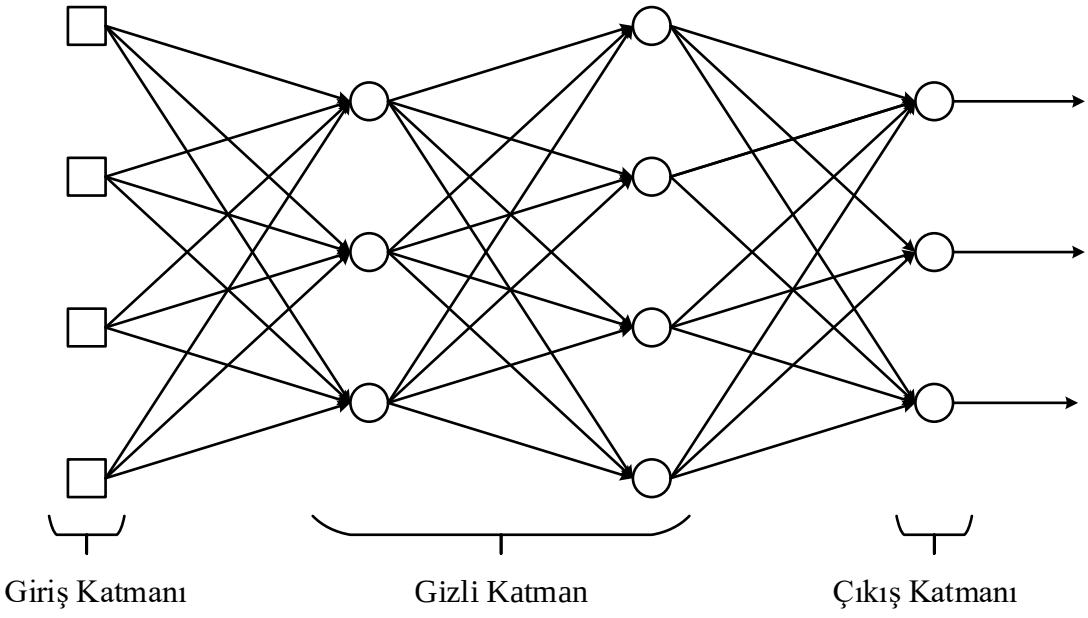
$$y = \varphi(v) \tag{2.3}$$

Bu denklemdeki $\varphi(v)$ aktivasyon fonksiyonudur. Sinir ağında birçok aktivasyon işlevi vardır.

2.4.2. Sinir ağı katmanları

Beyin devasa bir nöron ağından oluştuğu gibi, sinir ağı da bir düğüm ağıdır. Düğümlerin nasıl bağlandığına bağlı olarak çeşitli sinir ağları oluşturulabilir. En yaygın

kullanılan sinir ağı türlerinden biri Şekil 2.5’ de gösterildiği gibi çok katmanlı düğüm yapısıdır.

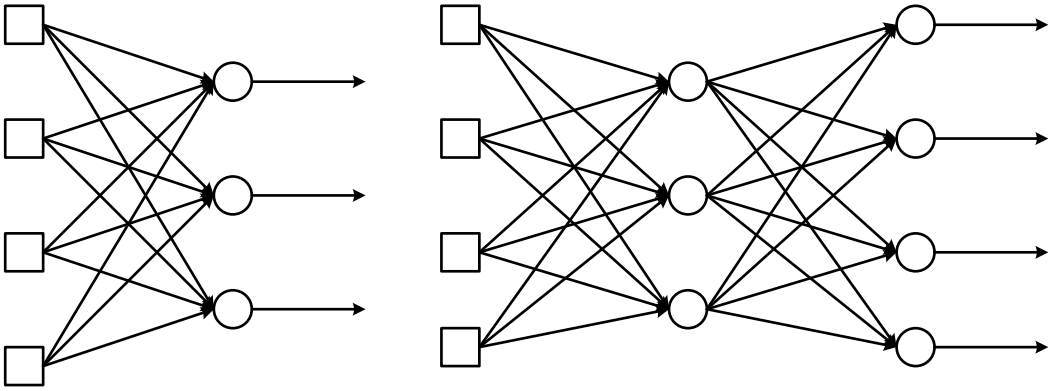


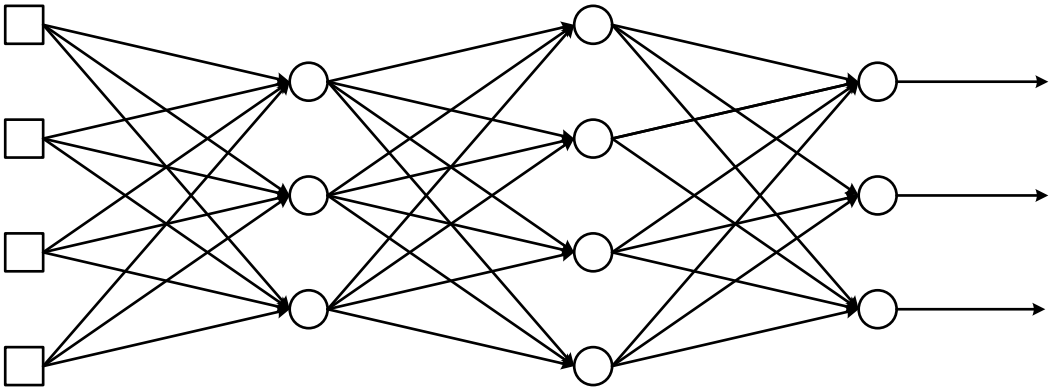
Şekil 2.5. Çok katmanlı düğüm yapısı

Şekil 2.5’deki kare düğümler giriş katmanı olarak adlandırılmaktadır. Giriş katmanının düğümleri, yalnızca giriş sinyallerini sonraki düğümlere ileten geçit görevi görür. Bu nedenle, ağırlıklı toplamı ve aktivasyon fonksiyonunu hesaplamazlar. Karelerle gösterilmelerinin ve diğer dairesel düğümlerden ayrılmalarının nedeni budur. Bunun aksine, en sağdaki düğüm grubuna çıkış katmanı denir. Bu düğümlerden gelen çıktı, sinir ağının nihai sonucu olur. Giriş ve çıkış katmanları arasındaki katmanlara gizli katman denir. Sinir ağının dışından erişilemedikleri için onlara bu isim verilmiştir.

Sinir ağı, basit bir mimariden giderek daha karmaşık bir yapıya doğru geliştirilmiştir. Başlangıçta, sinir ağı öncüleri, tek katmanlı sinir ağları olarak adlandırılan yalnızca giriş ve çıkış katmanlarına sahip çok basit bir mimariye sahipti. Gizli katmanlar tek katmanlı bir sinir ağına eklendiğinde, bu çok katmanlı bir sinir ağını oluşturur. Bu nedenle, çok katmanlı sinir ağı bir giriş katmanı, gizli katman ve çıktı katmanından oluşur. Tek bir gizli katmana sahip sinir ağına sığ sinir ağı denir. İki veya daha fazla gizli katman içeren çok katmanlı bir sinir ağına derin sinir ağı denir. Pratik uygulamalarda kullanılan çağdaş sinir ağlarının çoğu derin sinir ağlarıdır.

Çok katmanlı sinir ağını bu iki türe göre sınıflandırmamızın nedeni, tarihsel gelişimi ile ilgilidir. Sinir ağı, tek katmanlı sinir ağı olarak başladı ve sığ sinir ağına, ardından derin sinir ağına dönüştü. Derin sinir ağı, sığ sinir ağının gelişmesinin üzerinden yirmi yıl geçtikten sonra, 2000'lerin ortalarına kadar ciddi bir şekilde önemsenmedi. Bu nedenle, uzun bir süre için, çok katmanlı sinir ağı yalnızca tek gizli katmanlı sinir ağı anlamına geliyordu (Kim 2017). Birden fazla gizli katmanı ayırt etme ihtiyacı ortaya çıktığında, derin sinir ağına ayrı bir isim verildi (Şekil 2.6).





Şekil 2.6. Sinir ağı tipleri

Katmanlı sinir ağında, sinyal giriş katmanına girer, gizli katmanlardan geçer ve çıktı katmanından çıkar. Bu işlem sırasında sinyal katman katman ilerler. Diğer bir deyişle, bir katmandaki düğümler sinyali eşzamanlı olarak alır ve işlenen sinyali aynı anda bir sonraki katmana gönderir.

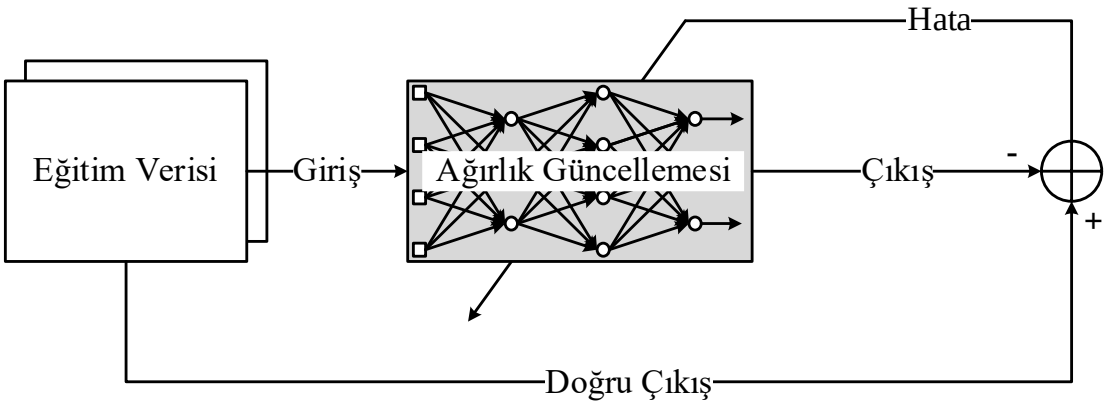
2.4.3. Bir sinir ağının denetimli öğrenimi

Sinir ağının denetimli öğrenimini açıklamak gerekirse; bu işlem aşağıdaki adımlarla ilerler:

1. Ağırlıklar için uygun başlangıç değerleri belirlenir.
2. Eğitim verilerindeki giriş değerleri sinir ağına giriş olarak verilir. Sinir ağının çıkışı gözlemlenir ve hata değeri hesaplanır.
3. Hatayı azaltmak için ağırlıklar güncellenir.
4. Eğitim verilerindeki her bir değer için adım 2'den itibaren işlemler tekrarlanır.

Bu adımlar temelde makine öğrenmesi türlerinden olan denetimli öğrenme süreciyle aynıdır. Tek fark, model değişikliği yerine sinir ağı için ağırlıktaki

değişiklikler haline gelmesidir. Şekil 2.7'de, sinir ağının denetimli öğrenme kavramı gösterilmektedir.



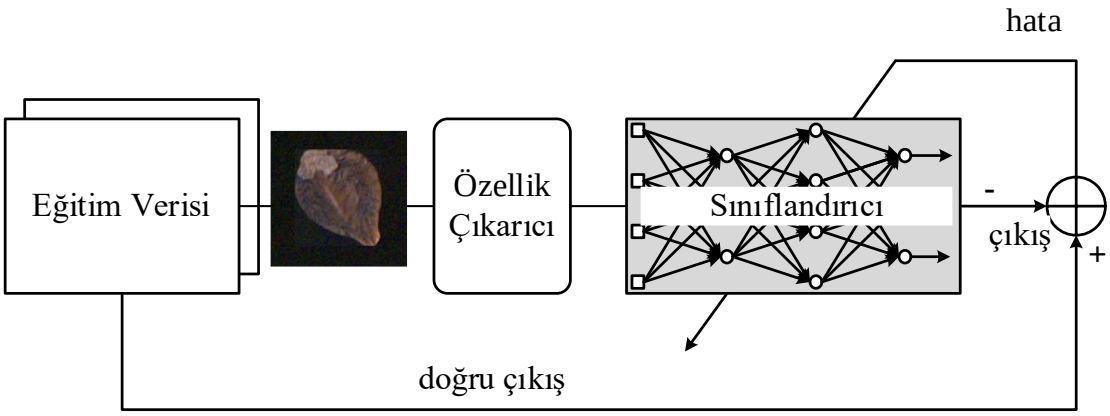
Şekil 2.7. Denetimli öğrenme konsepti

2.5. Evrişimli Sinir Ağı (CNN)

Evrişimli sinir ağı (Convulutional Neural Network - CNN) daha çok görüntü tanıma uygulamaları için özelleştirilmiş bir derin sinir ağı modelidir. Bu teknik, derin katmanların gelişiminin görüntü işleme için ne kadar önemli olduğunu örneklemektedir. Aslında CNN, 1980'lerde ve 1990'larda geliştirilmiş eski bir tekniktir (Cun vd. 1989). Ancak, karmaşık görüntülere sahip gerçek uygulamalar açısından pratik olmadığı için bir süredir göz ardı edilmiştir. 2012'de dramatik bir şekilde tekrar uygulanmaya başlandıktan sonra (Krizhevsky vd. 2012), CNN çoğu bilgisayarla görme alanında kullanılmaya başlandı. CNN yalnızca birçok gizli katmana sahip derin bir sinir ağı değildir. Beynin görsel korteksinin görüntüleri nasıl işlediğini ve tanıdığını taklit eden derin bir ağdır.

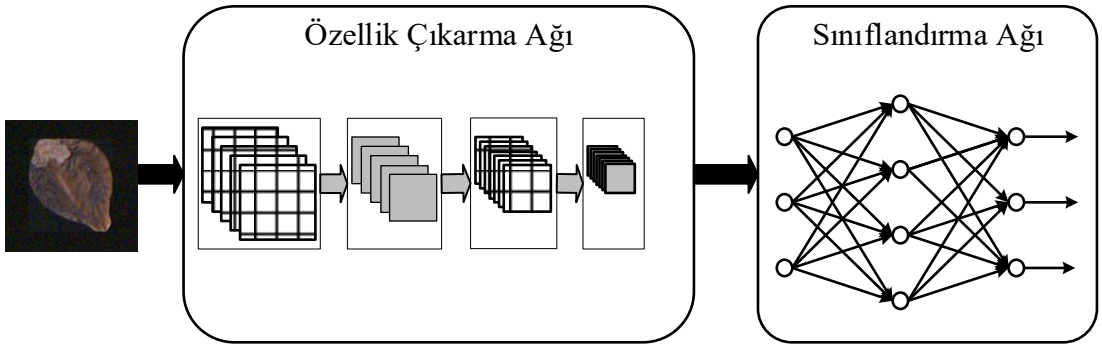
Temel olarak görüntü tanıma bir sınıflandırma problemidir. Örneğin, bir resmin görüntüsünün kedi mi yoksa köpek mi olduğunu anlamak, resmi bir kedi veya köpek sınıfı olarak sınıflandırmakla aynıdır. Aynı şey harf tanıma için de geçerlidir; Bir görüntüdeki harfi tanımak, görüntüyü harf sınıflarından birine sınıflandırmakla aynıdır. Bu nedenle, CNN'in çıktı katmanı genellikle çok sınıflı sınıflandırma sinir ağını kullanır. Ancak, görüntü tanıma için orijinal görüntülerin doğrudan kullanılması, tanıma yönteminden bağımsız olarak kötü sonuçlara yol açabilir. Bunun için bazı durumlarda sınıflandırma için orijinal görüntü ile birlikte bu görüntüye ait özniteliklerin de beraber kullanılması gerekebilir. Bu nedenle, görüntü özelliği çıkarımı için çeşitli teknikler geliştirilmiştir (Bay vd. 2008; Dalal ve Triggs 2005; Lowe 2004).

CNN'den önce özellik çıkarma işlemi, alanındaki uzman kişiler tarafından ilgili özellik çıkarma fonksiyonları kullanılarak yapılmaktaydı. Bu işlem, tutarsız bir performans seviyesi ortaya koyarken önemli miktarda maliyet ve zaman gerektiriyordu. Bu özellik çıkarıcılar Makine Öğreniminden bağımsız bir işlem olup bir nevi ön-işlem (pre-processing) olarak tanımlanmaktadır. Şekil 2.8 bu süreci göstermektedir.



Şekil 2.8. Makine öğreniminden bağımsız olan özellik çıkarma ile sınıflandırma işlemi

CNN yapısı ise Şekil 2.8'den farklı olarak özellik çıkarıcıyı eğitim sürecine dahil eder. CNN'in özellik çıkarıcısı, ağırlıkları eğitim sürecinde belirlenen özel sinir ağlarından (evrişim katmanları) oluşur. CNN'in avantajlarından biri de özellik çıkarma işleminin ağın kendisi tarafından otomatik olarak gerçekleştirilmesidir. Şekil 2.9, CNN mimarisini göstermektedir.



Şekil 2.9. Tipik CNN mimarisi

Şekil 2.9'a göre giriş görüntüsü, özellik çıkarma ağına (evrişim katmanlarına) girer. Evrişim katmanlarından çıkarılan özellik sinyalleri sınıflandırma sinir ağına girer. Sınıflandırma sinir ağı daha sonra görüntünün özelliklerine göre çalışır ve çıktıyı üretir.

Özellik çıkarma sinir ağı, evrişimsel katman yığınlarından ve katman çiftlerini bir araya toplayarak oluşur. Evrişim katmanı, adından da anlaşılacağı gibi, görüntüyü evrişim işlemini kullanarak dönüştürür. Dijital filtrelerin bir koleksiyonu olarak düşünülebilir. Havuzlama katmanı, komşu pikselleri tek bir pikselde birleştirir. Bu nedenle, havuzlama katmanı görüntünün boyutunu azaltır. CNN'in temel kaygısı görüntü olduğu için; Evrişim ve havuzlama katmanlarının işlemleri kavramsal olarak iki boyutlu bir düzlemdedir. Bu, CNN ile diğer sinir ağları arasındaki farklardan biridir.

Özet olarak, CNN, özellik çıkarma ağının ve sınıflandırma ağının seri bağlantısından oluşur. Eğitim süreci ile her iki katmanın ağırlıkları belirlenir. Özellik çıkarma katmanı, birçok evrişim ve havuz katmanı çiftlerinden oluşur. Evrişim katmanı

görüntüleri evrişim işlemi aracılığıyla özniteliklere dönüştürür ve havuzlama katmanı görüntünün boyutunu azaltır. Sınıflandırma ağı genellikle sıradan çok sınıflı sınıflandırma sinir ağını kullanır.

Sinir ağları, bir giriş görüntüsünü ya da özellik vektörünü kabul eder ve bunu genellikle doğrusal olmayan aktivasyon fonksiyonlarını kullanarak bir dizi gizli katman aracılığıyla gerçekleştirir. Her bir gizli katman ayrıca, her bir nöronun bir önceki katmandaki tüm nöronlara tamamen bağlı olduğu bir dizi nörondan oluşur. Bir sinir ağının son katmanı (yani "çıktı katmanı") da tamamen bağlı katmandır ve ağın nihai çıkış sınıflandırmalarını temsil eder.

Bir CNN oluşturmak için kullanılan birçok katman türü vardır, bunlardan karşılaşma olasılığı en yüksek olanlar aşağıdakilerdir:

- Evrişim Katmanı (CONV- Convolution Layer)
- Aktivasyon Katmanı (ACT veya RELU)
- Havuzlama Katmanı (POOL – Pooling Layer)
- Tamamen-Bağlı Katman (FC-Fully-Connected Layer)
- Yığın Normalleştirme Katmanı (BN- Batch Normalization Layer)
- Düşürme Katmanı (DO-Dropout Layer)

Bu katmanların belirli bir şekilde ve sırayla bir araya gelmesi tipik bir CNN yapısını verir. Bir CNN'i tanımlamak için aşağıdaki gibi genellikle basit metin diyagramları kullanılır:

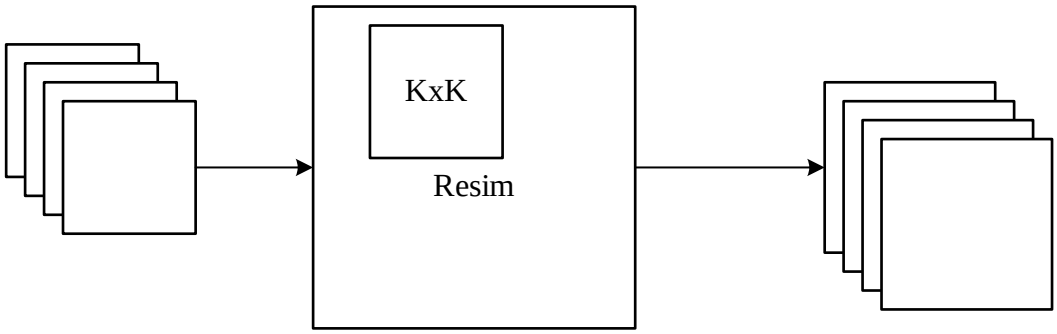
INPUT => CONV => RELU => FC => SOFTMAX

Yukarıdaki gösterimde, girdiyi kabul eden bir INPUT katmanı, devamında öznitelik çıkarım için bir evrişim katmanı (CONV), ardından bir aktivasyon katmanı (RELU), ardından bir tam-bağlı katman (FC) ve son olarak, çıktı sınıflandırma olasılıklarını elde etmek için bir softmax sınıflandırıcı katman tanımlanmıştır. Yukardaki yapı en temel ve basit bir CNN ağ yapısı olarak tanımlanabilir.

2.5.1. Evrişim katmanı

Evrişim katmanı, CNN'in en temel yapı taşıdır. Evrişim katmanı parametreleri, her filtrenin (kernel) bir genişliğe ve yüksekliğe sahip ve çoğunlukla kare olan bir dizi *K* öğrenilebilir filtrelerden oluşur. Bu filtreler boyut olarak küçüktür (uzamsal boyutları bakımından) ancak derinlik (katmandaki filtre sayısı) olarak çok büyük olabilirler.

CNN'de girişler için derinlik, görüntüdeki kanalların sayısıdır (örneğin, RGB görüntülerle çalışırken her bir kanal için derinlik 3 olacaktır). Ağda daha derin olan birimler için, derinlik, önceki katmanda uygulanan filtre sayısı olacaktır. Bu kavramı daha net hale getirmek için, *KxK* boyuttaki filtrelerin her birinin giriş bölgesi boyunca kaydığını, eleman bazında çarpımı hesapladığını, topladığını ve ardından çıktı değerini Şekil 2.10'daki gibi 2 boyutlu bir aktivasyon haritasında depoladığını düşünebiliriz.



(a) Resime uygulanmak için bekleyen N adet filtre

(b) Her filtre giriş hacmi ile konvüle edildi

(c) Her konvülüsyon işleminin sonucunda "aktivasyon haritası" denen bir 2B çıkış oluşur ve

Şekil 2.10. Evrişim işlemi **a)** Bir CNN'deki her bir evrişimli katmanda, giriş hacmine uygulanan K çekirdekleri; **b)** K çekirdeğinin her birinin giriş hacmi ile dönüştürülmesi; **c)** Her çekirdek, aktivasyon haritası adı verilen bir 2B (iki boyutlu) çıktı üretir

Her çıkış resmindeki her değer, giriş resminin yalnızca küçük bir bölgesine "bakan" bir nöronun çıkışıdır. Bu şekilde ağ, giriş hacminde belirli bir uzamsal konumda belirli bir özellik türünü gördüklerinde etkinleştirilen filtreleri "öğrenir". Ağın daha alt katmanlarında, kenar benzeri veya köşe benzeri bölgeler gördüklerinde filtreler etkinleşebilir. Daha sonra, ağın daha derin katmanlarında, yüzün bölümleri, bir köpeğin pençesi, bir arabanın kaputu gibi üst düzey özelliklerin varlığında filtreler etkinleşebilir.

Evirişim katmanının en temel işlemi filtreler ile konvolüsyon işlemi olup; farklı CNN yapılarında farklı sayılarda ve farklı boyutlarda filtreler kullanılmaktadır. Konvolüsyon katmanlarında genellikle 3x3, 5x5, 7x7 ve 11x11 boyutlarındaki filtreler kullanılmaktadır. Bilindik bazı filtreler şu şekildedir:

- Yataydaki kenarları bulan filtre:

$$Fh = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

- Dikeydeki kenarları bulan filtre:

$$Fv = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

- Parlaklık büyük ölçüde değiştiğinde kenarları bulan filtre:

$$Fi = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

- Görüntüdeki kenarları bulanıklaştıran filtre:

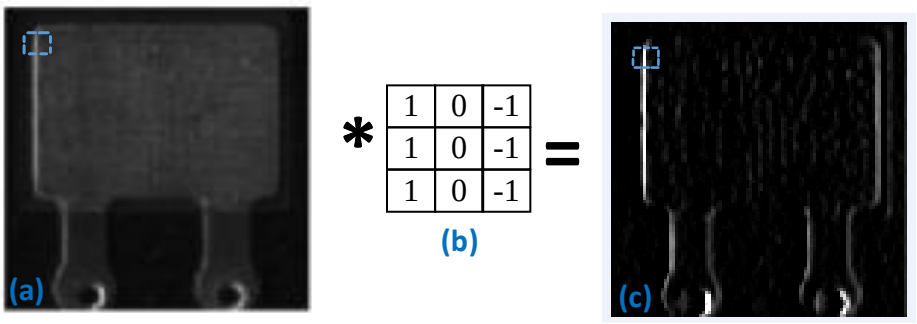
$$Fb = -1/9\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Evirişim (konvolüsyon) işlemi genel olarak * işareti ile gösterilir. Giriş matrisi (CNN yapılarında görüntü matrisi) ile filtre matrisine evrişimsel işlem uygulanması sonucu çıkış elde edilir. Bu işlemin genel tanımı denklem (2.4)'deki gibidir:

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} * \begin{pmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 \end{pmatrix} = \sum_{i=1}^{9} a_i k_i \tag{2.4}$$

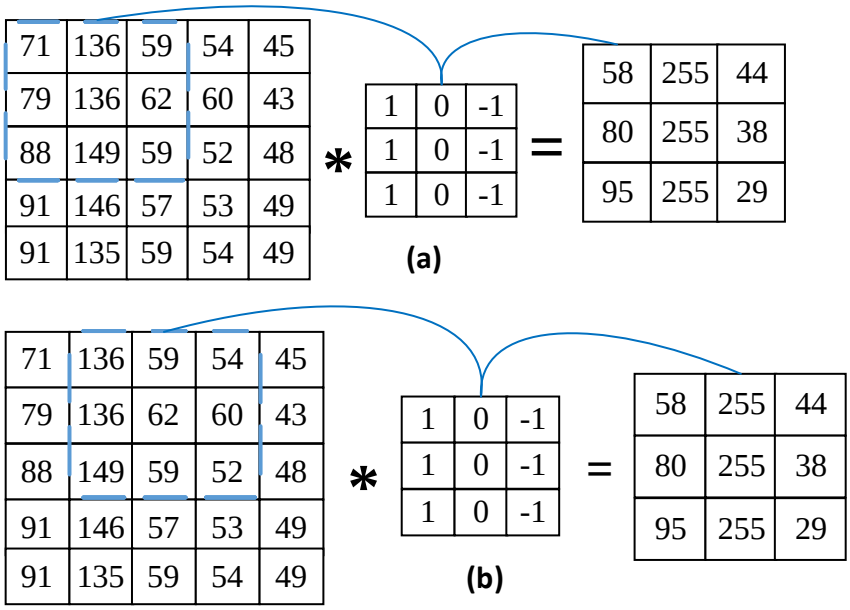
Yukarıdaki denklemde A matrisi ve K matrisi her ne kadar 3x3 boyutunda bir matris olsa da CNN uygulamalarında A giriş matrisi bir görüntü olup ağın yapısına göre farklı boyutlarda (227x227, 224x224) olabilmektedir. Örneğin AlexNet yapısındaki ilk erişim katmanında giriş görüntü boyutu 227x227x3, filtre boyutu ise 11x11'dir.

Filtreleme işlemini daha iyi anlayabilmek için: Şekil 2.11a'daki 100x100 boyutundaki gri formattaki görüntü ile dikey kenar bulma filtresi 'Fv' konvüle edildiğinde bu işlemin sonucu Şekil 2.11c'deki gibi olmuştur.



Şekil 2.11. Konvolüsyon işlemi **a)** Gri formattaki giriş görüntüsü; **b)** Dikey kenar bulma filtresi; **c)** Konvolüsyon sonucu

Şekil 2.11'i daha iyi anlayabilmek için şekilde mavi kutucuklar ile işaretlenmiş alanların detaylı gösterimi Şekil 2.12'deki gibidir. Şekildeki mavi kutucukta bulunan piksel değerleri ile Fv filtresinin evrişim işlemini detaylı bir şekilde inceleyecek olursak 3x3'lük filtre ilgili ilk matrisin çarpımların toplamı sonucu 58 (71x1+136x0+59x(-1)+79x1+136x0+62x(-1)+88x1+149x0+59x(-1)=58) (Şekil 2.12a), aynı filtrenin ilgili ikinci matris ile çarpımların toplamı sonucu da 255 olarak hesaplanmıştır (Şekil 2.12b). İlgili filtrenin tüm görüntü üzerinde gezdirilmesi ve her bir turda çarpımların toplamı sonucu çıkış değeri hesaplanmaktadır.



Şekil 2.12. Evrişim işleminin detaylı incelenmesi **a)** İlk piksel gurubunun konvüle edilmesi; **b)** İkinci piksel grubunun konvüle edilmesi

Bir evrişim katmanında, giriş hacmini giriş görüntü boyutuyla ilişkilendirip ve katmandaki filtrelerle yapılan konvülüsyon işlemleri yukarıdaki gibi anlatıldıktan sonra evrişim katmanlarının çıkış hacmini açıklamak gerekir. Bir çıkış hacminin boyutunu kontrol eden üç parametre vardır: derinlik (depth), adım (stride) ve sıfır-dolgu (zero-padding) boyutu.

Derinlik (Depth *N*): Bir çıktı hacminin derinliği, CONV katmanındaki giriş hacminin yerel bir bölgesine bağlanan nöronların (yani filtrelerin) sayısını kontrol eder. Her filtre belirli bir kenar, imge veya renk ile aktive olan bir aktivasyon haritası üretir. Verilen bir CONV katmanı için, etkinleştirme haritasının derinliği *N* ya da aynı katmanda eğitilmekte olan filtre sayısı olacaktır.

Adım Büyüklüğü (Stride *S*): Evrişim işlemi, küçük bir matrisi büyük bir matris boyunca "kaydırmak", her koordinatta durmak, eleman bazında çarpma ve toplamı hesaplamak ve ardından çıktıyı depolamak olarak tanımlanmaktadır. Bu açıklama, bir görüntü boyunca soldan sağa ve yukarıdan aşağıya kayan bir pencereye benzer. *KxK* filtrelerinin her biri bölgeyle birleştirilir ve çıktı 3 boyutlu bir hacimde depolanır. CONV katmanları oluşturulurken normalde S = 1 veya S = 2 olan adım boyutları kullanılır.

Daha küçük adımlar, örtüşen alıcı alanlara ve daha büyük çıktı hacimlerine yol açacaktır. Tersine, daha büyük adımlar daha az örtüşen alıcı alanlara ve daha küçük çıktı hacimlerine neden olacaktır. Evrişimsel adım kavramını daha somut hale getirmek için Şekil 2.13'ü göz önünde bulundurursak, burada 5x5 girişli bir görüntü (Şekil 2.13a) ve 3x3 Laplacian kerneli (Şekil 2.13b) bulunmaktadır.

95	242	186	152	39
39	14	220	153	180
5	247	212	54	46
46	77	133	110	74
156	35	74	93	116

(a)

0	1	0
1	-4	1
0	1	0

(b)

Şekil 2.13. Adım kavramının anlaşılması için örnek girdiler **a)** 5x5 giriş resmi; **b)** Evrişim işlemini gerçekleştireceğimiz kernel

S = 1 kullanılırsa, çekirdek soldan sağa ve yukarıdan aşağıya kayar, her seferinde bir piksel, aşağıdaki çıktıyı üretir (Şekil 2.14a). Bununla birlikte, aynı işlem, S = 2'lik bir adımla uygulanacak olursa; bir seferde iki piksel (x ekseni boyunca iki piksel ve y ekseni boyunca iki piksel) atlayarak daha küçük çıkış hacmi elde edilir (Şekil 2.14b).

692	-315	-6
-680	-194	305
153	-59	-86

(a)

692	-6
153	-86

(b)

Şekil 2.14. Adım kavramının anlaşılması için örnek çıktılar **a)** 1x1 adımda evrişim çıktısı; **b)** 2x2 adımda evrişim çıktısı

Böylece, basitçe kernel adımını değiştirek, giriş hacimlerinin uzamsal boyutlarını azaltmak için evrişim katmanlarının nasıl kullanıldığı görülmektedir. Evrişimli katmanlar ve havuzlama katmanları, uzamsal girdi boyutunu azaltmak için birincil yöntemlerdir.

Sıfır Doldurma (Zero-padding): Bir evrişim uygularken orijinal görüntü boyutunu korumak için görüntünün kenarlarının "doldurulması" gerekir. Bu kural CNN için de geçerlidir. Sıfır-doldurma (zero-padding) kullanarak; çıktı hacmi boyutu girdi hacmi boyutuna eşleşecek şekilde, girdi sınırları boyunca dolduralabilir. Uygulanan dolgu miktarı *P* parametresi tarafından kontrol edilir.

Bu teknik, üst üste birden çok CONV filtresi uygulayan derin CNN mimarilerinde kritik bir tekniktir. Sıfır dolguyu görselleştirmek için, S = 1 adımıyla 5x5 girişli bir görüntüye 3x3 Laplacian kerneli uygulanması işlemi örnek olarak gösterilebilir (Şekil 2.13a).

Şekil 2.15a'da, evrişim işleminin doğası gereği çıktı hacminin giriş hacminden (5x5) nasıl daha küçük olduğu (3x3) görülmektedir. Bunun yerine P = 1 olarak ayarlanırsa, 7x7'lik bir hacim oluşturmak için girdi hacmini sıfırlarla (Şekil 2.15b) doldurup ve ardından evrişim işlemi uygulandığında orijinal giriş hacmi boyutuyla eşleşen bir çıktı hacmi boyutu olan 5x5'i sağlanmaktadır (Şekil 2.15c).

692	-315	-6
-680	-194	305
153	-59	-86

(a)

0	0	0	0	0	0	0
0	95	242	186	152	39	0
0	39	14	220	153	180	0
0	5	247	212	54	46	0
0	46	77	133	110	74	0
0	156	35	74	93	116	0
0	0	0	0	0	0	0

(b)

-99	-673	-130	-230	176
-42	692	-315	-6	-482
312	-680	-194	305	124
54	153	-59	-86	-24
-543	167	-35	-72	-297

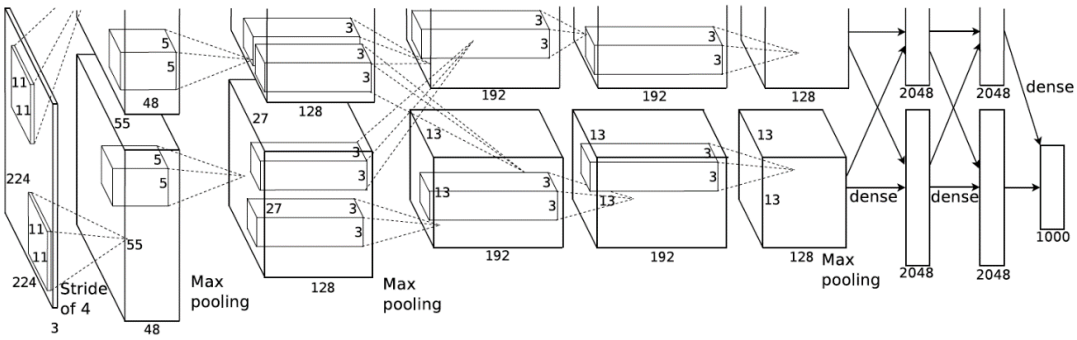
(c)

Şekil 2.15. a) Bir 5x5 çıktısına 3x3 evrişim uygulamanın çıktısı (yani, uzamsal boyutlar azalır); **b)** P = 1 ile orijinal girişe sıfır dolgu uygulanarak uzamsal boyutların 7x7'ye çıkarılması; **c)** 3x3 evrişimi sıfır doldurulmuş girdiye uyguladıktan sonra, çıktı hacmi 5x5 olan orijinal girdi hacmi boyutuyla eşleşir, bu nedenle sıfır dolgu uzaysal boyutları korumamıza yardımcı olur

Tüm bu parametreler bir araya geldiğinde, bir çıktı hacminin boyutu girdi hacmi boyutunun bir fonksiyonu olarak hesaplanabilir (H, giriş görüntülerinin kare olduğu varsayılarak, alıcı alan boyutu K, adım S, ve sıfır dolgu P miktarı). Geçerli bir CONV katmanı oluşturmak için aşağıdaki denklemin bir tam sayı olduğundan emin olunması gerekir:

$$\left(\frac{H - K + 2P}{S}\right) + 1 \tag{2.5}$$

Örnek olarak, 2012 yılında yapılan ImageNet sınıflandırma yarışmasını kazanan ve derin öğrenmenin görüntü sınıflandırma uygulamalarında kullanılmasındaki büyük artıştan sorumlu olan AlexNet mimarisinin ilk katmanı verilebilir (Krizhevsky vd. 2012).



Şekil 2.16. Krizhevsky vd. tarafından geliştirilen orijinal AlexNet mimari diyagramı (Krizhevsky vd. 2012)

İlk katmanın giriş görüntü boyutunun 224x224 piksel olarak belirlenmiştir. Ancak yukarıdaki denklemi 11x11 filtre kullanarak, 4 adım değeri ve sıfır doldurma P=0 olmadan uygulanırsa muhtemelen doğru olmayacak ve tamsayı olmayan bir sonuç elde edilecektir.

((224-11+2(0))/4)+1 = 54.25

Bu sonuç hataya neden olmaktadır ve Krizhevsky vd. (2012) 227x227 giriş görüntüsü kullanarak bu hatanın önüne geçmişlerdir.

((227-11+2(0))/4)+1 = 55

Evrişim katmanı özetlenecek olursa:

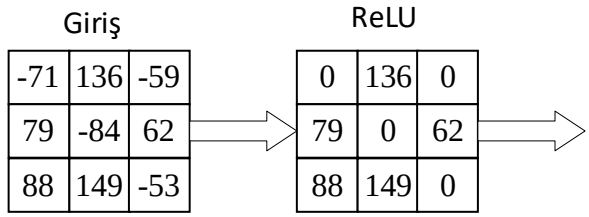
- $W_{in}xH_{in}xD_{in}$ şeklinde bir giriş kabul eder (giriş boyutu normalde kare olmaktadır, bu sebeple $W_{in} = H_{in}$ olarak kabul edilmektedir).
- 4 parametre gerektirmektedir
  - Filtre sayısı N (aynı zamanda çıkış hacminin derinliğidir)
  - Alıcı boyutu K (N adet KxK kare kernel evrişim için kullanılır)
  - Adım büyüklüğü S
  - Sıfır doldurma boyutu P
- Evrişim katmanının çıkış boyutları olan $W_{out}, H_{out}, D_{out}$ denklem (2.5) kullanılarak şu şekilde ifade edilebilir.
  - $W_{out} = \left(\frac{W_{in}-K+2P}{S}\right) + 1$
  - $H_{out} = \left(\frac{H_{in}-K+2P}{S}\right) + 1$
  - $D_{out} = N$

2.5.2. Aktivasyon katmanı

Bir CNN'deki evrişim katmanından sonra, ReLU veya ELU gibi doğrusal olmayan bir aktivasyon fonksiyonu uygulanır. ReLU aktivasyon fonksiyonu en çok kullanılan olduğu için ağ diyagramlarında tipik olarak aktivasyon katmanlarını RELU olarak gösterilmektedir (Rosebrock 2017).

Aktivasyon katmanları teknik olarak "katmanlar" değildir (bir aktivasyon katmanında hiçbir parametrenin/ağırlığın öğrenilmemesi nedeniyle). Aktivasyon katmanı her zaman bir evrişim katmanın hemen ardından kullanıldığı için bazen ağ mimarisi diyagramında gösterilmezler. Bu durumda, her CONV katmanından sonra hangi aktivasyon işlevinin kullanıldığının belirtilmesi gerekmektedir. Örnek olarak, aşağıdaki ağ mimarisi göz önünde bulundurulursa: INPUT => CONV => RELU => FC. Bu diyagramı daha kısa hale getirmek için, aktivasyon katmanının her zaman bir evrişim katmanını takip ettiği varsayıldığı için RELU bileşeni kaldırılabilir: INPUT => CONV => FC.

Bir aktivasyon katmanı, $W_{in}xH_{in}xD_{in}$ giriş hacmini kabul eder ve ardından verilen aktivasyon fonksiyonunu uygular **(**Şekil 2.17). Aktivasyon fonksiyonu eleman bazında uygulandığından, bir aktivasyon katmanının çıktısı her zaman girdi boyutuyla aynıdır.



Şekil 2.17. ReLU aktivasyonundan geçen bir giriş hacmi örneği, max (0; x)

2.5.3. Havuzlama katmanı

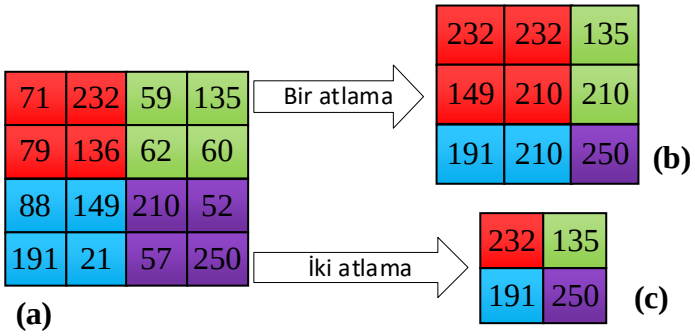
Bir giriş hacminin boyutunu küçültmenin iki yöntemi vardır. Evrişim katmanında adım büyüklüğünü 1'den büyük olarak ayarlamak veya havuzlama katmanını kullanmak. Bir CNN mimarisinde ardışık CONV katmanları arasına POOL katmanları eklemek yaygın bir kullanımdır:

INPUT => CONV => RELU => POOL => CONV => RELU => POOL => FC

Havuzlama katmanının birincil işlevi, giriş hacminin uzamsal boyutunu (yani, genişlik ve yüksekliği) aşamalı olarak azaltmaktır. Bunu yapmak, parametre ve hesaplama miktarını azaltmayı sağlar. Havuzlama ayrıca aşırı öğrenmeyi (overfitting) kontrol etmeye yardımcı olur (Sharma ve Mehra 2019).

Havuzlama katmanları, bir girdinin derinlik dilimlerinin her biri üzerinde bağımsız olarak maksimum veya ortalama işlevini kullanarak çalışır. Maksimum havuz oluşturma, uzamsal boyutu azaltmak için genellikle CNN mimarisinin ortasında yapılırken, ortalama havuzlama FC katmanlarını kullanmaktan tamamen kaçınmak istediğimiz durumlarda ağın son katmanı olarak kullanılır. En yaygın POOL katmanı türü maksimum havuzlamadır, ancak bu eğilim daha farklı mikro mimarilerin ortaya çıkmasıyla değişmektedir.

Havuzlama katmanında daha çok 2x2'lik havuz boyutu kullanılır, ancak daha büyük giriş resimleri (> 200 piksel) kullanan daha derin CNN'ler, ağ mimarisinin ilk katmanlarında 3x3 havuz boyutunu kullanabilir. Ayrıca adım büyüklüğü S = 1 veya S = 2 en çok kullanılan adım büyüklüğüdür. Şekil 2.18, 2x2 havuz boyutu ve S = 1 adımda maksimum havuzlama uygulama örneğini göstermektedir. Her 2x2 blok için, sadece en büyük değer korunur, tek bir adım atlanır (kayan bir pencere gibi) ve bu işlem tüm resim pikselleri işlenene kadar tekrar uygulanır. Böylece 3x3 boyutunda bir çıktı hacmi elde edilmiştir. Adımları artırarak çıktı hacminin boyutu daha da azaltılabilir. Aynı girdiye S = 2 uygulanırsa (Şekil 2.18c) havuzlama katmanı, genişliği ve yüksekliği iki kat azaltarak önceki katmandaki aktivasyonlarının %75'ini etkili bir şekilde atmayı sağlar.



Şekil 2.18. Farklı atlama sayıları ile örnek bir atlama işlemi **a)** 4x4 giriş hacmi; **b)** S = 1 adımıyla maksimum 2x2 havuzlama uygulaması; **c)** S = 2 ile maks. 2x2 havuzlama uygulaması

Özetle, havuzlama katmanları $W_{in}xH_{in}xD_{in}$ boyutunda bir girdi hacmini kabul eder. Daha sonra iki parametreye ihtiyaç duyarlar:

- Alıcı alan boyutu, K (aynı zamanda "havuz boyutu")
- Adım büyüklüğü, S

POOL işleminin uygulanmasından sonra $W_{out}xH_{out}xD_{out}$ boyutunda bir çıkış hacmi elde edilir. $D_{out} = D_{in}$ olmak üzere diğer parametreler denklem 2.6 yardımıyla şu şekilde hesaplanır.

$$H_{out} = W_{out} = \frac{H_{in} - K}{S} + 1 \tag{2.6}$$

Pratikte 2 tip havuzlama çeşidi kullanılmaktadır.

- Tip #1: K = 3; S = 2, örtüşen havuzlama olarak adlandırılır ve normalde büyük uzamsal boyutlara sahip görüntülere uygulanır.
- Tip #2: K = 2; S = 2, örtüşmeyen havuzlama olarak adlandırılır. Bu, en yaygın havuzlama türüdür ve daha küçük uzamsal boyutlara sahip görüntülere uygulanır.

Daha küçük girdi görüntülerini (32-64 piksel aralığında) kabul eden ağ mimarileri için ayrıca K = 2; S = 1'i de kullanılmaktadır.

2.5.4. Tamamen bağlı katmanlar

Tamamen bağlı katmanlarındaki (FC-Fully Connected) nöronlar, ileri beslemeli sinir ağları için standart olduğu gibi, önceki katmandaki tüm aktivasyonlara tam olarak bağlıdır. FC katmanları her zaman ağın sonuna yerleştirilir (Örnek olarak Bir CONV katmanının ardından bir FC katmanı ve ardından başka bir CONV katmanı uygulanmaz).

Aşağıdaki basitleştirilmiş mimaride gösterildiği gibi, softmax sınıflandırıcısını uygulamadan önce bir veya iki FC katmanı kullanmak yaygındır:

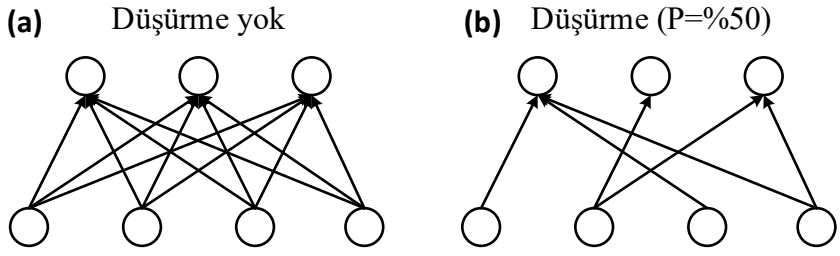
INPUT => CONV => RELU => POOL => CONV => RELU => POOL => FC => FC

Bu gösterimde, her bir sınıf için nihai çıktı olasılıklarını hesaplayacak softmax sınıflandırıcısından önce iki FC katmanı uygulanmaktadır.

2.5.5. Düşürme katmanı

Son katman türü düşürme (DO-dropout) katmanıdır. DO aslında test doğruluğunu artırarak aşırı öğrenmeyi önlemeye yardımcı olmayı amaçlayan bir düzenleme biçimidir. Eğitim setindeki her bir "batch" için, *p* olasılığı olan DO katmanları, ağ mimarisindeki önceki katmandan sonraki katmana girişlerin bağlantısını rastgele keser (Rosebrock 2017).

Şekil 2.19, belirli bir mini parti için *p=0,5* olasılığıyla iki FC katmanı arasındaki bağlantıları rastgele ayıran bu kavramı görselleştirmektedir. Mini "batch" için ileri ve geri geçiş hesaplanır. Sonra bırakılan bağlantılar yeniden bağlanır ve başka bir bağlantı kümesi bağlantıyı kesmek için örneklenir.



Şekil 2.19. Düşürme katmanına örnek **a)** Herhangi bir kesinti olmaksızın tamamen birbirine bağlı bir sinir ağının iki katmanı; **b)** Bağlantıların %50'sini bıraktıktan sonra aynı iki katman

Düşürme uygulanmasının nedeni, eğitim zamanında ağ mimarisini açıkça değiştirerek aşırı öğrenmeyi azaltmaktır. Bağlantıların rastgele kesilmesi, ağdaki hiçbir

düğümün, belirli bir modelle sunulduğunda "etkinleştirmekten" sorumlu olmamasını sağlar. Bunun yerine, düşürme, benzer girdilerle sunulduğunda etkinleşecek birden fazla, yedekli düğüm olmasını sağlar. Bu da modelin genelleşmesine yardımcı olur.

Son FC katmanının softmax sınıflandırıcı olduğu varsayılan bir mimarinin FC katmanları arasına düşürme katmanlarının yerleştirilmesi en yaygın kullanımdır:

... CONV => RELU => POOL => FC => DO => FC => DO => FC

2.6. İlgili Çalışmalar

CNN yapıları, her katmanında bir dizi evrişim filtreleriyle çoklu dönüşümlerin gerçekleştirildiği ileri beslemeli, çok katmanlı hiyerarşik ağ yapılarıdır (LeCun vd. 2010). CNN'in en çekici özelliği, verilerdeki mekansal veya zamansal korelasyondan yararlanma yeteneğidir. CNN topolojisi, evrişimli katmanlar, doğrusal olmayan işlem birimleri ve alt örnekleme katmanlarının kombinasyonundan oluşan bir dizi çoklu öğrenme aşamalarına bölünmüştür (Jarrett vd. 2009). Otomatik öznitelik çıkarma yeteneği ile CNN, ayrı bir öznitelik çıkarıcıya olan ihtiyacı azaltmaktadır. Böylelikle, ayrıntılı öznitelik ve ön işlemlere gerek olmaksızın CNN, ham piksellerden görüntün içeriğini çok iyi bir şekilde öğrenebilmektedir.

LeNet-5, geliştirilen ilk "ünlü" CNN mimarisidir (LeCun vd. 1998). Bu çalışmada el yazısıyla yazılmış rakamların tanınması için 7 katmanlı, 32x32 giriş veri boyutuna sahip ve 60.000 öğrenebilir parametre içeren bir evrişimli sinir ağı mimarisi önerilmiştir. LeNet-5, sonraki yıllarda CNN modellerinin geliştirilmesinde araştırmacılar için büyük bir ilham kaynağı olmuştur. Devam eden süreçte 2012 yılında AlexNet'in (Krizhevsky vd. 2012) ImageNet Büyük Ölçekli Görsel Tanıma Yarışmasında (Anonymous 1) elde ettiği başarısıyla, evrişimli sinir ağı (CNN) mimarileri sınıflandırma uygulamalarında çok daha popüler hale geldi. Bir yıl sonra AlexNet'in gelişmiş versiyonu olan ZFNet önerilmiştir (Zeiler ve Fergus 2014). Önerilen yapı AlexNet yapısına benzemekteydi ancak en belirgin fark ilk katmanda öznitelik kaybı yaşamamak için 11x11 filtre yerine 7x7 filtrelerin kullanılması olmuştur. AlexNet'in literatüre girmesinden sonra araştırmacılar ilk başta böylesi bir yapıyı bilgisayarlı görü uygulamalarında kullanabileceklerinden tam olarak emin olamamışlardı. Ancak ZFNet'in CNN yapılarının katman seviyesinde tam olarak nasıl çalıştığını, öznitelikleri görselleştirerek anlatması sonucu CNN yapıları araştırmacılar tarafından daha çok kabul görmeye başladı. Derin ağların boyutlarını artırarak performanslarının daha da artırılması fikri ilk olarak Visual Geometry Group (VGG) çalışanları tarafından denenmiş olup; kendilerinden önce önerilen AlexNet yapısına kıyasla daha fazla katman ve parametre içeren (13 evrişimli ve 3 tam bağlantılı katman ve 138M parametre) VGG-16 modelini önermişlerdir (Simonyan ve Zisserman 2018). Bu ağ yapısında AlexNet'e kıyasla daha küçük boyutlu filtreler (2x2 ve 3x3) kullanılmıştır. Top5 doğruluğu AlexNet yapısında %84,60 iken, bu ağ yapısında %91,90 olarak hesaplanmıştır.

CNN'lere yeni bir çoklu görev kavramı getirmek üzere Inception Ağ Yapısı önerilmiştir (Szegedy vd. 2015). CNN'lerdeki hesaplama maliyetlerini düşürmeyi amaçlayan bu mimari, kapsamlı derin ağlar oluşturmak yerine, tek bir katmanda birden

çok konvolüsyonu istifleyebileceğimiz bir modeli önermiştir. Bu model aynı zamanda küçük boyutlu katmanlar oluşturmak için (boyut azaltmak için) 1x1 filtrelerin kullanımını da literatüre kazandırmıştır. Basit bir deyişle, bir Inception Ağ yapısı 3x3 evrişim , 5x5 evrişim ve havuzlamayı aynı anda yapmakta, bunu yaparken bu işlemlerden önce ya da sonra 1x1'lik evrişim işlemlerinin uygulanmasını (evrişimden önce ve havuzlamadan sonra) ve çıkışta bu işlem sonuçlarının uç uca eklemeyi gerçeklemektedir. Inception ağlarının mimarisindeki bazı değişikliklerle birlikte GoogLeNet (Szegedy vd. 2015), Inception v3 (Szegedy vd. 2016) ve Xception (Chollet 2017) vb. gibi aynı ilkelere dayanan diğer birçok CNN mimarisinin yolunu açmıştır.

CNN yapılarının bu denli gelişmesiyle birlikte, CNN yapıları mobil cihazlarda da kendine uygulamada ve geliştirmede yer bulmuşlardır. 2017 yılında mobil cihazlar üzerinde gömülü görsel işlemlerin gerçeklenmesi için MobileNet yapısı önerilmiştir (Howard vd. 2017). Bu yapıda derinlemesine ayrılabilir evrişim yapıları parametre sayısını ve işlem sayısını azaltmak için efektif bir şekilde kullanılmıştır. Bu sayede ağın gerçekleme süresi ve test doğruluğu arasında hassas bir denge kurulmaya çalışılmıştır. Devam eden süreçte mobil cihazlar gibi sınırlı kaynaklara sahip donanımların üzerinde CNN yapılarının efektif bir şekilde gerçeklenmesi için ShuffleNet yapısı önerilmiştir (Zhang vd. 2018). Bu yapıda test doğruluğundan ödün vermeyecek şekilde, nokta tabanlı grup evrişimi ve kanal karıştırması olmak üzere iki yeni işlem bloğu, işlem yükünü azaltmak için önerilmiştir. ImageNet üzerinde yapılan testler sonucunda MobileNet'e göre yaklaşık 7,8%'lik daha düşük top-1 hatasına sahip olduğu gösterilmiştir.

Derin öğrenme mimarileri hızla gelişmeye devam ederken; derin ve geniş mimarilerde gözlemlenen iki ana sorun göze çarpmaktadır: yüksek hesaplama maliyeti ve bellek gereksinimi. Bellek gereksinimi mobil cihazlar, ARM işlemcili sistemler ve FPGA gibi donanımsal gerçeklemelerde kısıtlı kaynaklardan dolayı sorun teşkil edebilmektedir. Yine derin öğrenme ile gerçeklenen uygulama gerçek zamanlı çalıştırılmak istenirse yüksek hesaplama maliyeti ve çıkarım süresi sistemin sınırlarını zorlayabilmektedir. Geleneksel evrişim işlemi, çıkarım süresini artıran düşük bellek ve zaman kısıtlaması uygulamalarında CNN yapısının gerçeklenmesini sınırlayan çok sayıda çarpma işlemi gerektirmektedir (Shakeel vd. 2019). Otonom araçlar, robotik, sağlık hizmetleri ve mobil uygulamalar gibi birçok gerçek dünya uygulaması, hesaplama açısından sınırlı kaynaklara sahip platformlarda gerçekleştirilmesi gereken görevleri zamanında gerçekleştirir. Literatürde CNN'lerin verimliliğini artırmak için yapılmış birçok çalışma mevcuttur. En genel yaklaşım CNN yapılarındaki ağırlıkların sayısını azaltmak için (dolaylı olarak parametre sayısını ve işlem yükünü azaltmak) budama (Chin vd. 2018; Guo vd. 2016; Li vd. 2016; Luo vd. 2017), seyreltme (Changpinyo vd. 2017; Han vd. 2016; Wen vd. 2016) ve kuantalama yöntemleri (Banner vd. 2018; Hubara vd. 2016; Li vd. 2017) önerilmiştir. Budama, eğitimden sonra ağdaki ağırlıkların sayısının azaltılması işlemidir. Çoğu durumda, ağa ait ağırlıkların büyük bölümünün test doğruluğunu minimum düzeyde azaltmasıyla kaldırılabilmesi gerçeği, standart CNN yapılarının aşırı sayıda parametreye sahip olmasını göstermektedir (Molchanov vd. 2016). Budama işlemi, ağırlıkların ve kayan noktalı işlemlerin (FLOP) sayısını azaltmada etkili olsa da genellikle, hafızaya erişim maliyetlerini artırdığı için çıkarım süresini aynı oranda azalttığından söz edilemez. Bir diğer yöntem ise tam-bağlı evrişim operatörlerini, seyrek evrişim operatörleri ile

eğitimden önce değiştirerek CNN yapısının verimliliğinin artırılmasıdır. Kanalları gruplara ayıran ve yalnızca gruplanmış birleştirmeye izin veren gruplanmış evrişim operatörü buna örnek olarak verilebilir (Krizhevsky vd. 2012).

Gruplanmış evrişimlerle evrişim katmanındaki öznitelik sayısı kadar geniş ağlar kurabilir. Belli bir sayıda filtre grubu bloğu oluşturulur ve bu gruplar aynı anda işlem yapacak şekilde çoğaltılır. Bu sayede her filtre yalnızca kendi filtre grubundaki öznitelik haritaları üzerinde işlem yapacak, bu da işlem yükünü önemli ölçüde azaltacaktır. Filtre grupları aynı anda çalışabilmesi için kopyaladığından daha fazla sayıda hesaplama ile sonuçlandığı düşünülse de tüm filtreleri tek bir kümede toplayıp ve gruplanmış evrişimler kavramını kullanılmazsa, hesaplamaların sayısı katlanarak artacaktır. Kayan nokta işlem sayısını (FLOPs) azaltmak ve CNN yapılarını iyileştirmek için önerilmiş bir diğer yöntem ise Derinliğe Göre Ayarlanabilir Evrişimler (Depth-wise seperable convolutions) yöntemidir. Bu yöntem ilk olarak (Sifre 2014) tarafından önerilmiştir ve bazı çalışmalarda (Howard vd. 2017; Ioffe ve Szegedy 2015) kullanılmıştır. Bu yöntem geleneksel evrişim işleminin yaptığı işlemi iki aşamalı bir evrişim ile yapmaktadır. İlk aşamada evrişim işlemi derinliğe göre yapılmaktadır. İkinci aşamadaki evrişim işlemi ise noktasal olarak yapılmaktadır. Tüm filtreler ve özniteliklerin evrişim işlemleri bu şekilde yapıldığında; N filtre sayısı ve Dk çıkış öznitelik boyutu olmak üzere FLOPs değeri (1/N)+(1/(Dk)^2) oranında değişmektedir (Sifre 2014). Mobil cihazların işlem kapasiteleri düşük olduğu düşünülürse bu yöntem MobileNet ve ShuffleNet gibi mobil platformlar için geliştirilmiş ağ yapılarında efektif bir şekilde kullanılmıştır. Bu yöntem ile FLOPs sayısında azalma olmasına rağmen aynı şeyi çıkarım süresi için bahsetmek mümkün değildir. Derinliğe Göre Ayarlanabilir Evrişimler, iki aşamalı evrişim işlemi yapmakta bu da hafıza işlemlerini daha da artırmaktadır. Hafıza işlemlerindeki bu artış, hesaplama süresini doğal olarak çıkarım süresini de büyütmektedir.

Evrişimli sinir ağı mimarileri genel olarak alanında uzman kişiler tarafından elle geliştirilmektedir. Mimari yapı kurulurken birçok seçenek mimariyi geliştiren kişinin tahmin ve önsezilerine bırakılmıştır. Otomatikleştirilmiş mimari arama, modern derin öğrenme araştırmalarının öncü konularından birisi haline gelmiştir. Otomatik mimari arama, aslında bir hiperparametre tarama optimizasyonu olup; grid arama veya Bayes optimizasyonu gibi çeşitli yaklaşımlarla çözülebilen genel bir problemdir (Shahriari vd. 2016). Evrişimli sinir ağları işlem yükünün ve hiperparametre sayısının çok fazla olması ve optimizasyonda farklılıklar göstermesi gibi birçok komplikasyonu da beraberinde getirmektedir. Bu yüzden CNN yapılarında hiperparametre optimizasyonu için özel yaklaşımlar gerekmektedir. Otomatikleştirilmiş mimari arama yapıları alan bilgisinin gerekli olup olmadığına göre iki ana grupta toplanabilir (Sun vd. 2020). İlk grupta "otomatik + manuel olarak ayarlanan" CNN mimari tasarımları bulunmaktadır. Bu uzmanlığa dayalı manuel ayarlamanın hala gerekli olduğu tasarım grubudur. Bu kategorideki örnek çalışmalar: genetik CNN yöntemini (Xie ve Yuille 2017), hiyerarşik temsil yöntemi (Hiyerarşik Evrim) (Liu vd. 2017), verimli mimari arama yöntemi (EAS) (Cai vd. 2017), blok tasarım yöntemi (Blok-QNN-S) (Zhong vd. 2017) ve gelişmiş nöral mimari arama yöntemi (NSANet) (Zoph vd. 2017). İkinci grup ise "otomatik" CNN mimari tasarımlarıdır. Kullanıcıların bu yapıyı kullanırken manuel olarak ayarlamasını gerektirmeyen yapılardır. Bu mimariye örnek çalışmalar: büyük ölçekli evrim yöntemi (Büyük Ölçekli Evrim) (Real vd. 2017), Kartezyen genetik programlama yöntemi (CGP-CNN) (Suganuma vd. 2017), sinir mimarisi arama yöntemi

(NAS) (Zoph ve Le 2016) ve metamodelleme method (MetaQNN) (Baker vd. 2016). İki metot karşılaştırıldığında, "otomatik + manuel ayarlama" metodunun "otomatik" tasarımlardan daha iyi performans gösterilmesi beklenmektedir. Örneğin, "otomatik + manuel" metodunu kullanan NASNet ve "otomatik" metodunu kullanan Büyük Ölçekli Evrim Algoritmasının CIFAR10 veri seti üzerinde uygulanması sonucu sırasıyla %96,27 ve %94,60 sonuçları elde edilmiştir.

Otomatikleştirilmiş mimari arama yöntemlerinde son yıllarda en çok kullanılan algoritmalardan bir tanesi de genetik algoritmadır. Genetik algoritma evrimsel algoritmalar grubundandır. Evrimsel algoritma (Back 1996), biyolojik evrimden esinlenen popülasyon tabanlı meta-sezgisel optimizasyon paradigmaları sınıfıdır. Tipik evrimsel algoritmalar arasında genetik algoritmalar (GA'lar) (Davis 1991), genetik programlama (Banzhaf 1997), evrimsel strateji (Janis 1976) gibi algoritmalar bulunmaktadır. GA'ların biyo-esinlenmiş operatörler yani: mutasyon, çapraz geçiş ve seçim (Mitchell 1998) kullanarak yüksek kaliteli optimal çözümler üretebildikleri de kabul edilmiştir. Genetik algoritmalar (Beheshti vd. 2013) gibi popülasyon tabanlı algoritmalar, geniş arama alanları ve patolojik hedef fonksiyonları (Bianchi vd. 2009) ile özellikle optimizasyon problemlerinde (Sun vd. 2017) iyi performans gösterirler. CNN yapılarındaki hiperparametreleri optimize etmek için sezgi ötesi kullanımıyla ilgili güncel araştırmalar bulunmaktadır. Bu araştırmalarda belirli bir problemde filtre boyutu veya evrişimsel blok dizisi gibi ağların farklı özelliklerini ayarlamak için bir sezgi ötesi yöntem kullanılmaktadır. Başka bir çalışmada derinlik dahil atlama blokları ve Max-Pool katmanlarının sayısını optimize etmek için genetik algoritma kullanmıştır (Sun vd. 2020). Ancak konvolüsyonların filtre boyutları bu çalışmada sabit kalmıştır. Yine farklı araştırmalarda, yalnızca ağın yapısını değil, aynı zamanda öğrenme parametrelerini de optimize etmek için daha basit CNN mimarileri ve genetik algoritmaları kullanmıştır (Aszemi ve Dominic 2019).

Literatürde endüstriyel tarım sistemleri için geliştirilmiş birçok derin öğrenme ile sınıflandırma uygulaması bulunmaktadır. Karlekar ve Seal (2020) yaygın olarak bilinen 16 soya fasulyesi yaprak hastalığı semptomunu sınıflandıran SoyNet adlı bir CNN mimarisi önermişlerdir. Önerilen bu ağı kullanarak test doğruluğu %98,14 olarak gerçeklenmiştir. Bal peteği hücrelerini tanımlayan ve daha sonra bu hücreleri içeriklerine göre sınıflandıran başka bir çalışmada, on üç farklı evrişimli sinir ağı mimarisi test edilmiştir (Alves vd. 2020). Bu on üç farklı ağı kullanarak, en yüksek f1 puanı %94,3 olarak hesaplamışlardır. Steinbrener vd. (2019) hiperspektral görüntüler kullanarak meyvelerin CNN yapıları ile sınıflandırmasını gerçeklemiştir. Bu çalışma sonucunda sınıflandırma test doğruluğu RGB verisine göre %88,15'ten %92,23'e çıkmıştır.

Tarım sektöründe üzerinde çalışılan konulardan bir tanesi de tohumların kalitesine göre sınıflandırılması ve ayıklanmasıdır. Literatürdeki çalışmalara bakıldığında: Kolza tohumunun renklerinin daha hassas bir şekilde tespit edilmesi için HSV ve NCM (Nine Color Model) renk uzayını birlikte kullanarak; tek bir tohum RGB renk uzayında %83,96 doğrulukta hesaplanırken bu oran önerilen modelde %97,72'ye çıkmıştır (Li vd. 2008). Farklı hata sınıfına ait tohumları sınıflandırabilmek için renk, şekil bilgisi ve SVM algoritması kullanmıştır (Kiratiratanapruk ve Sinthupinyo 2011). Önerilen sistem 10.000 örnek resim üzerinde uygulanmış ve test doğruluğu sağlıklı tohumlar için %95,6 hatalı tohumlar için %80,6 olarak gerçeklenmiştir. Mısır

tohumlarını kalitesine göre sınıflandırmak için RGB görüntü yerine hiperspektral görüntü kullanılmıştır (Zhang vd. 2012). Mısır tohumlarının sınıflandırılması için en küçük kareler destekli vektör makinesi (LS-SVM), dört farklı ana bileşen (PC) kombinasyonunu kullanan geri yayılım sinir ağı (BPNN), çekirdek ana bileşenleri (KPC'ler) ve dokusal özellikleri sırasıyla kullanmıştır. En yüksek doğruluk CA-GLCM-LS-SVM modelinde %98,89 olarak gerçeklenmiştir. Parnian ve Javidan (2014) k-means algoritmalarını kullanarak buğday tohumlarının kalitesine göre sınıflandırmasını gerçeklemiştir. Kama, Rosa ve Kanada tohumlarını sınıflandırmak için makine öğrenmesi algoritmalarını kullanmışlardır. Sınıflandırmada kullanılan tohum özellikleri: alan, çevre, kompaktlık, çekirdek uzunluğu, çekirdek genişliği, asimetri katsayısı ve çekirdek uzunluğudur. Sınıflandırmada kullanılan fonksiyonda Bayes, Meta ve Lazy yöntemlerini kullanmışlardır. En yüksek sınıflandırma sonucu Çok Katmanlı Algılayıcı (Multilayer Perceptron) ve Lojistik ile yapılan sınıflandırma sonucu olup; %95,2 olarak gerçeklenmiştir. Kurtulmuş vd. (2016) 8 farklı biber tohumunu sınıflandırmak için bir yapay sinir ağı modeli önermişlerdir. Önerilen ağ, giriş katmanı, gizli katman ve çıktı katmanı dahil olmak üzere üç katmana sahiptir. Gizli katmanda 30 nöron vardır. Önerilen ağ, derin evrişimli bir sinir ağı olmayıp %84,94 sınıflandırma doğruluğuna sahiptir. Benzer bir çalışmada, Çin lahana tohumlarını sınıflandırmak için; öznitelik çıkartma fonksiyonları ile tohumun özelliklerini çıkarmış ve daha sonra bu özellikleri sınıflandırmak için bir geri yayılım sinir ağı modelini (BNNP) önerilmiştir (Huang ve Cheng 2017). Bu çalışmaya göre BNNP'lerin kullanılması ile iyi tohumlar ve iyi olmayan tohumlar için sınıflandırma doğruluğu sırasıyla %91,53 ve %88,95 olarak hesaplanmıştır. Başka bir çalışmada, diploit ve haploit mısır tohumlarını sınıflandırmak için bir custom CNN mimarisi önerilmiştir (Veeramani vd. 2018). Yine sonuçları bu yöntemle Kurtulmuş vd. (2016)'nin çalışması ile karşılaştırmak için bir destek vektör makinesi (SVM) ve bir görüntü analizi boru hattını önermişlerdir (Veeramani vd. 2018). Önerilen CNN mimarisinin sınıflandırma doğruluğu %96,80 ve SVM doğruluğu ise %87,60 olarak hesaplanmıştır. Makine öğrenimi algoritmalarını ve derin öğrenme algoritmalarını karşılaştırmak için Huang vd. (2019) tarafından başka bir çalışma yapılmıştır. Bu çalışmada derin öğrenme (GoogleNet) ve makine öğrenme teknikleri (SURF + SVM) ile kusurlu ve iyi mısır tohumlarını sınıflandırmışlardır. Deneysel sonuçlar, GoogleNet'in %95 ve SURF + SVM'nin ise %79,2 doğruluğa sahip olduğunu göstermiştir. Veeramani vd. (2018) ve Huang vd. (2019)'nin çalışmalarına göre, derin öğrenme algoritmalarının makine öğrenme tekniklerine göre daha yüksek doğruluğa sahip olduğu gösterilmiştir. Bu sonuçlardan, gelecekteki sınıflandırma çözümlerinin derin öğrenme algoritmalarına dayalı olacağı söylenebilir. Son güncel çalışmada Eldem (2020) Hussain vd. (2015)'nin çalışmasını genişleterek Kama, Rosa ve Canadian tohumlarını sınıflandırmak için derin yapay sinir ağı modelini önermiş ve %100 test başarısı elde etmiştir. Heo vd. (2018) tohumları tespit etmek, izlemek ve sınıflandırmak için ResNet-18 evrişimli sinir ağ modelini kullanmışlardır. Sistemi yüksek hızlarda (500 fps) gerçek zamanlı çalıştırabilmek için dört CPU ve bir GPU kullanmışlardır. GPU üzerinde koşturulan algoritmanın test doğruluğu %99'dan fazla olarak hesaplanmıştır.

Derin öğrenme uygulamaları gerçek zamanlı veya çevrimdışı olarak (kayıtlı verilerin analizi) yapılmaktadır. Çevrimdışı uygulamalar genellikle depolanan verileri analiz etmek için yapılmakta olup; bu tarz uygulamalarda ağın çıkarım süresi çok önem arz etmemektedir. Ancak sahada çalışan gerçek zamanlı bir sistem geliştirmek için evrişimli sinir ağlarının çıkarım süresi en önemli kriter olarak göze çarpmaktadır.

Çıkarım süresini etkileyen birçok parametre olmakla birlikte bu parametrelerden bir tanesi de FLOPs'dur (Saniyedeki Kayan Nokta İşlemleri). Kayan Nokta İşlemleri sayısını azaltmak, CNN modellerini optimize etmek ve CNN modellerini hızlandırmak için geliştirilen farklı teknikler literatürde detaylı bir şekilde ele alınmıştır. Probleme özel CNN modellerinin geliştirilmesi ve gerçek zamanlı çalıştırılması açısından bakıldığında; CNN mimarisini geliştirenler genellikle iki ana yöntem kullanmaktadırlar: ilki literatürdeki mevcut evrişimli sinir ağı modellerinin uygulamaya özel optimize edilmesi, ikincisi ise uzman kişiler tarafından uygulamaya özel ağ modellerinin önerilmesi (literatür taramasında detaylı bir şekilde bahsedildiği gibi bunun üzerine yapılan çalışmalar artık daha çok otomatikleştirilmiş mimari arama teknikleri üzerine evrilmektedir). Literatürdeki evrişimli sinir ağı modellerinin uygulamaya özel optimize edilmesine örnek olarak: Kamal vd. (2019) bitki hastalığı sınıflandırması işlemini derin öğrenme ile gerçek zamanlı bir şekilde yapabilmek için MobileNet mimarisini küçülterek, VGG modelinden 29 kat ve MobileNet modelinden 6 kat daha az parametreye sahip bir ağ yapısı önermişlerdir. Yine başka bir çalışmada Li vd. (2019) VGG mimarisinden esinlenerek kandaki kanser hücrelerini derin öğrenme ile gerçek zamanlı bulmak için düşük çıkarım süresine sahip kendi özel ağ modellerini önermişlerdir. Algoritmanın NVIDIA P100 GPU üzerinde koşturulması sonucu toplu iş boyutu = 1 (batch size = 1) iken tek bir görüntünün sınıflandırması işlemi 3.6ms sürmüştür. Bir diğer başlık olan uzman kişiler tarafından uygulamaya özel ağ modellerinin önerilmesi de literatürde kendine geniş bir yer bulmaktadır. Örneğin Knoll vd. (2018) organik tarım uygulamalarında sınıflandırma için özel bir ağ yapısı önerip ve bu ağı sonraki çalışmalarında sistemi gerçek zamanlı hale getirmek için kullanmışlardır (Knoll vd. 2019). Başka bir çalışmada Bircanoğlu vd. (2018) derin öğrenme ile gerçek zamanlı çalışan bir atık madde ayıklama sistemi geliştirmek için çıkarım süresi düşük özel CNN mimarisi önermişlerdir.

2.6.1. Tezin katkısı ve çalışmanın önemi

Tohum sınıflandırma ve tohum ayıklama akademide ve endüstride kendine geniş yer bulan endüstriyel tarım uygulamalardan bir tanesidir. Literatür taramasında tohum sınıflandırma ve tohum ayıklama üzerine yapılmış çalışmaların çoğuna detaylıca yer verilmiştir. Literatürdeki çalışmaları ve bu tezde yapılan çalışmayı karşılaştıracak olursak; tezin literatüre katkısını ve önemini aşağıdaki gibi özetleyebiliriz:

Endüstriyel test düzeneği ile daha gerçekçi veri setlerinin oluşturulması: Literatürdeki çalışmaların büyük çoğunluğunda hazır veri setleri ya da sabit bir düzlem üzerine serilmiş tohumların görüntüleri çekilmiş ve bu görüntüler üzerinden veri setleri oluşturulmuştur. Bu tez çalışmasında ise uygulamanın gelecekte endüstride de aktif bir şekilde gerçeklenebilmesi için endüstriyel bir test düzeneği oluşturulmuş ve tüm veri setleri bu test düzeneğinden elde edilen kayıtlı görüntüler ile oluşturulmuştur. Bu açıdan bakıldığında daha gerçekçi veriler ile çalışıldığı söylenebilir. Nitekim deney düzeneği üzerinden elde edilen görüntüler tohumlar serbest düşme yaparken çekilen görüntüler olup; tohumlar serbest düşme esnasında kendi etraflarında dönmektedirler. Bu da tohumların düşerken her açıdan farklı şekilde görüntülenmesini sağlamaktadır.

Veri Setlerini oluşturmak için otomatik bir algoritma önerilmesi: Literatürdeki birçok çalışmada probleme özgü veri setleri basit yöntemler ile oluşturulmuş olup; bu veri setlerinin nasıl oluşturulduğu hakkında detaylı bir bilgiye literatürde

rastlanmamıştır. Bu tez çalışmasında özellikle ayıklama problemleri için (tohum ayıklama, meyve ayıklama, maden ayıklama vb.) yazılımsal olarak otomatik ve hızlı bir şekilde veri seti nasıl oluşturulur detaylı bir şekilde açıklanmıştır. Literatürdeki diğer çalışmalar ile karşılaştırıldığında veri setinin nasıl oluşturulması gerektiği hakkında daha önce bu kadar detaylı bir çalışmaya rastlanmamıştır. Bu açıdan bakıldığında bu çalışmanın ayıklama problemi üzerine çalışan araştırmacılar için kapsamlı bir kaynak olacağı söylenebilir.

Farklı ölçekleme yöntemleri uygulanarak farklı türde veri setlerinin oluşturulması: Literatürde birçok CNN yapısı olup; giriş veri boyutu CNN mimarisine göre değişebilmektedir. Bu durumda ya veri oluşturulurken bu giriş ağ boyutuna göre oluşturulacak ya da kayıtlı veri setleri bu giriş veri boyutuna göre ölçeklenecektir. Yapılan detaylı literatür taramasında ayıklama problemleri için veri setlerinin hangi oranda ve ne şekilde ölçekleneceği üzerine herhangi bir çalışmaya rastlanmamıştır. Bu tez çalışmasında tohum veri setleri oluşturulurken farklı ölçekleme yöntemleri ile farklı türlerde veri setleri oluşturulmuştur. Oluşturulan bu veri setleri CNN modelleri üzerinde test edilmiş ve tohum ayıklama problemi için en uygun ölçekleme yöntemi detaylı bir şekilde anlatılmıştır.

Sınıflandırmanın derin öğrenme ile yapılması ve sonuçların detaylıca incelenmesi: Literatürdeki tohum sınıflandırma ve tohum ayıklama çalışmalarının büyük bir çoğunluğu öznitelik çıkarıcı fonksiyonlar ve makine öğrenmesi algoritmaları ile yapılmıştır. Derin öğrenme ile tohum sınıflandırma ve ayıklama üzerine bazı çalışmalar olsa da bu çalışmalarda sadece mevcut CNN yapıları denenmiş ve test sonuçlarına değinilmiştir. Bu çalışmada tohum sınıflandırma ve ayıklama problemi hem mevcut CNN ağ yapıları ile denenmiş hem de sistemin gelecekte gerçek zamanlı hale gelebilmesi ve donanım dostu olabilmesi açısından problem tüm detayları ile incelenmiştir.

Probleme özgü evrişimli sinir ağ modelinin önerilmesi: Bu çalışmada tohum ayıklamanın derin öğrenme ile gerçeklenmesi sadece mevcut CNN yapıları ile denenmemiş aynı zamanda problemine özgü bir CNN modeli önerilmiştir. Önerilen model iki evrişim ve bir softmax katmanından oluşan bir ağ yapısı olup; düşük işlem yüküne sahip, donanım dostu ve yüksek doğrulukta çalışması hedeflenmiştir.

Önerilen ağı optimize etmek için iki aşmalı bir yöntem önerilmesi: Alanında uzman kişiler tarafından probleme özgü özel CNN yapılarının önerilmesi literatürde kendi geniş bir yer bulmaktadır. Yapılan literatür taramasına göre hem uygulamaya özgü ağ modeli önerilmesi hem de önerilen ağ modelinin optimize edilmesi üzerine yapılan bir çalışmaya literatürde rastlanmamıştır. Bu çalışmada önerilen özel ağ modelini optimize etmek için iki aşamalı Ölçekle ve Buda metodu önerilmiştir ve metot verimli sonuçlar vermiştir. Bu açıdan bakıldığında bu tez çalışması, literatürdeki diğer çalışmalara göre özgün bir çalışmadır.

Genetik algoritma ile önerilen ağın optimize edilmesi ve optimizasyonun genelleştirilmesi: Önerilen özel ağ yapısının farklı parametrelere (FLOPs ve/veya parametre sayısı) göre optimize edilmesi için genetik algoritma kullanılmıştır ve başarılı sonuçlar elde edilmiştir. Elde edilen sonuçlar bir önceki Ölçekle ve Budama metodu ile karşılaştırılmış ve iki yöntemin de yakın sonuçlar verdiği gösterilmiştir. Bu açıdan

bakıldığında probleme özgü, kullanıcının belirleyeceği parametrelere göre optimize olmuş ağ yapılarının genetik algoritmaları ile yapılmasının ne kadar önünün açık olduğu önemli bir şekilde vurgulanmıştır. Ve bu açıdan gelecek çalışmalara ışık tutar niteliktedir.

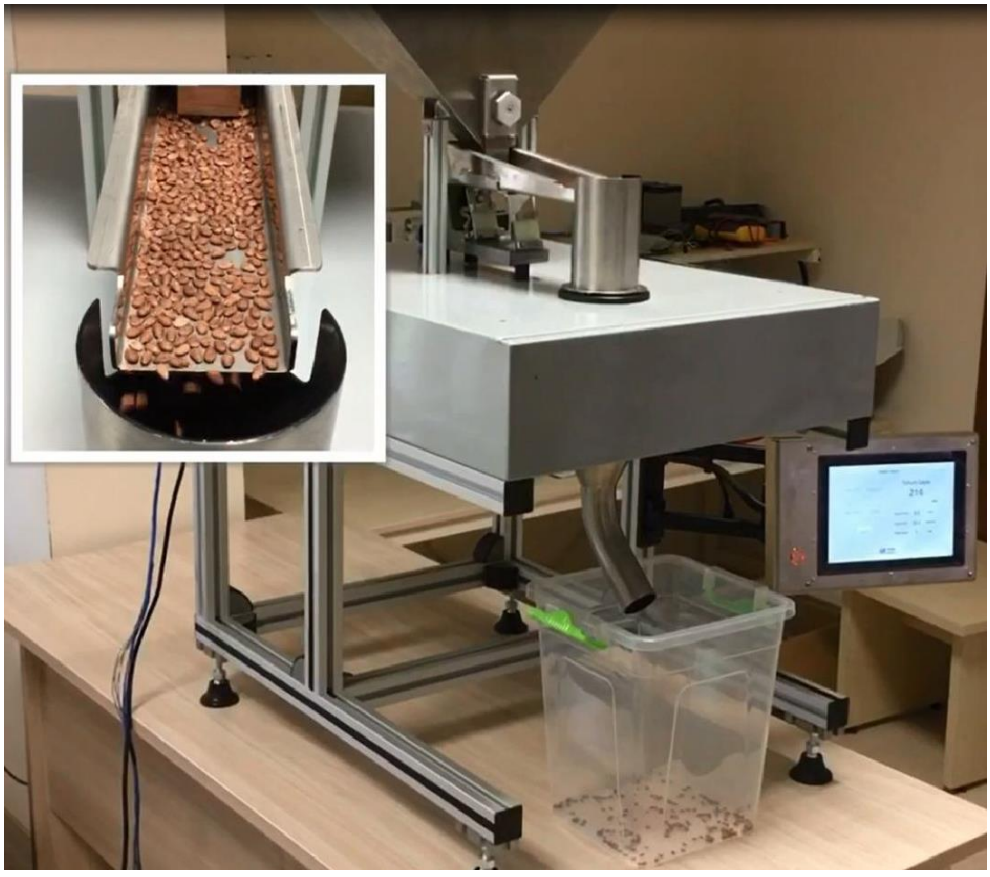
Optimize edilen ağın işlem yükünün küçük ve donanım dostu olması: Yapılan literatür araştırmasına göre, gelecekte derin öğrenmenin düşük işlem ve bellek kapasitesine sahip donanımlar ile gerçeklenmesinin önünün ne kadar açık olduğu ve bu alanda birçok yeni çalışmanın yapıldığı görülmüştür. Nitekim tezin çıktılarına bakıldığında optimizasyon sonucunda ağın işlem yükünün azaldığı ve parametre sayısının azaldığı görülmüştür. Bu da tez çalışmasında yapılan çalışmaların daha da genişletilerek gelecekte mobil ve FPGA gibi alanlarda uygulanabilir ve geliştirilebilir olduğunu göstermektedir.

3. MATERYAL VE METOT

Bu tez çalışmasının asıl amacı, derin öğrenme uygulamalarının endüstride kullanılabilir seviyede hızlı ve yüksek doğrulukta çalışan bir ağ yapısının önerilmesi ve bu ağ yapısının donanım dostu olmasıdır. Uygulama alanı olarak tohum sınıflandırma ve tohum ayıklama problemleri seçilmiştir. Her iki probleminde derin öğrenme ile gerçeklenebilmesi için öncelikle veri setlerinin oluşturulması gerekmektedir. Veri setlerinin oluşturulması için de öncelikle bir deney düzeneği geliştirilmiştir.

3.1. Deney Düzeneği

Literatürde tohum sınıflandırma ve tohum ayıklama üzerine farklı çalışmalar yapılmış olup; bu çalışmalarda kullanılan veri setleri, tohumların düz bir zemin üzerinde durağan halde görüntülerinin çekilmesi ile oluşturulmuştur. Bu çalışmada literatürdeki çalışmalardan farklı olarak tohumların serbest düşme yaparken görüntüleri çekilecek ve bu çekilen görüntüler ile veri setleri oluşturulacaktır. Böylesi bir veri seti oluşturmak için Şekil 3.1'deki gibi bir deney düzeneği geliştirilmiştir.



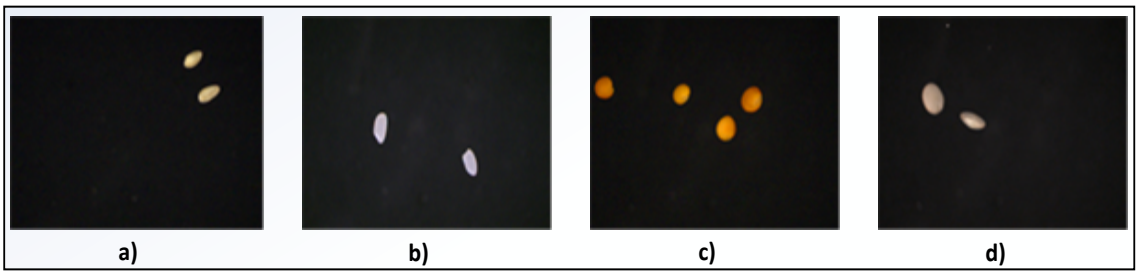
Şekil 3.1. Deney düzeneği

Bu deney düzeneğinde üst kısımda bir tohum haznesi bulunup; tohumlar bu hazneye dökülmektedir. Haznenin ucunda bir oluk ve oluğun alt kısmında da vibrasyon motoru bulunmaktadır. Tohumların oluk üzerinde ilerlemesi için, vibrasyon motoru çalıştırılmakta bu sayede tohumlar vibrasyon motoru üzerinde birbirinden ayrılarak

ilerlemektedir. Tohumlar, oluğun uç noktasına geldiklerinde ise tohumlar serbest düşme yaparak aşağıya doğru düşmektedirler. Yine deney düzeneğinin içinde Basler 107649 acA1440-73gc 1440 x 1080, 73 fps, color 1/3" CMOS kamera ve MBJ imaging DTL-ic-1010 Diffuse Flat Dome Ön aydınlatma sistemi bulunmaktadır. Kamera ve aydınlatma birbirlerine bir konektör ile bağlı olup; aydınlatma tetikleme sinyalini kamera üzerinden almaktadır. Bu sayede kameranın pozlama süresi boyunca aydınlatma aktif olmakta ve daha net görüntüler elde edilmektedir.

3.2. Veri Setlerinin Oluşturulması

Veri setlerinin oluşturulması için Şekil 3.1'deki deney düzeneği kullanılmış olup; veri setlerinin yetirince büyük olması için çok fazla sayıda görüntü kaydedilmiştir. Kayıtların alınması sürecinde hangi sınıfa ait kayıt alınacaksa öncelikle bu sınıfa ait tohumlar hazneye yerleştirilmiştir. Devamında ise vibrasyon motoru sürekli modda çalıştırılarak, haznedeki tüm tohumların düşmesi sağlanmış ve düşme zamanında da görüntüler çekilip kaydedilmiştir. Görüntüyü gerçek zamanlı kaydedebilmek için kullanılan Basler marka kameranın ara yüz programı olan Pylon Viewer yazılımı kullanılmıştır. Bu sayede bir sonraki işlem olan Nesne Tespit Algoritması (Blob Detection Method) ve devamında uygulanacak olan ölçeklendirme işlemleri için veriler bilgisayar hafızasında kaydedilmiştir. Şekil 3.2'de tohumlar düşerken çekilmiş dört farklı tohuma ait örnek görüntüler gösterilmektedir.

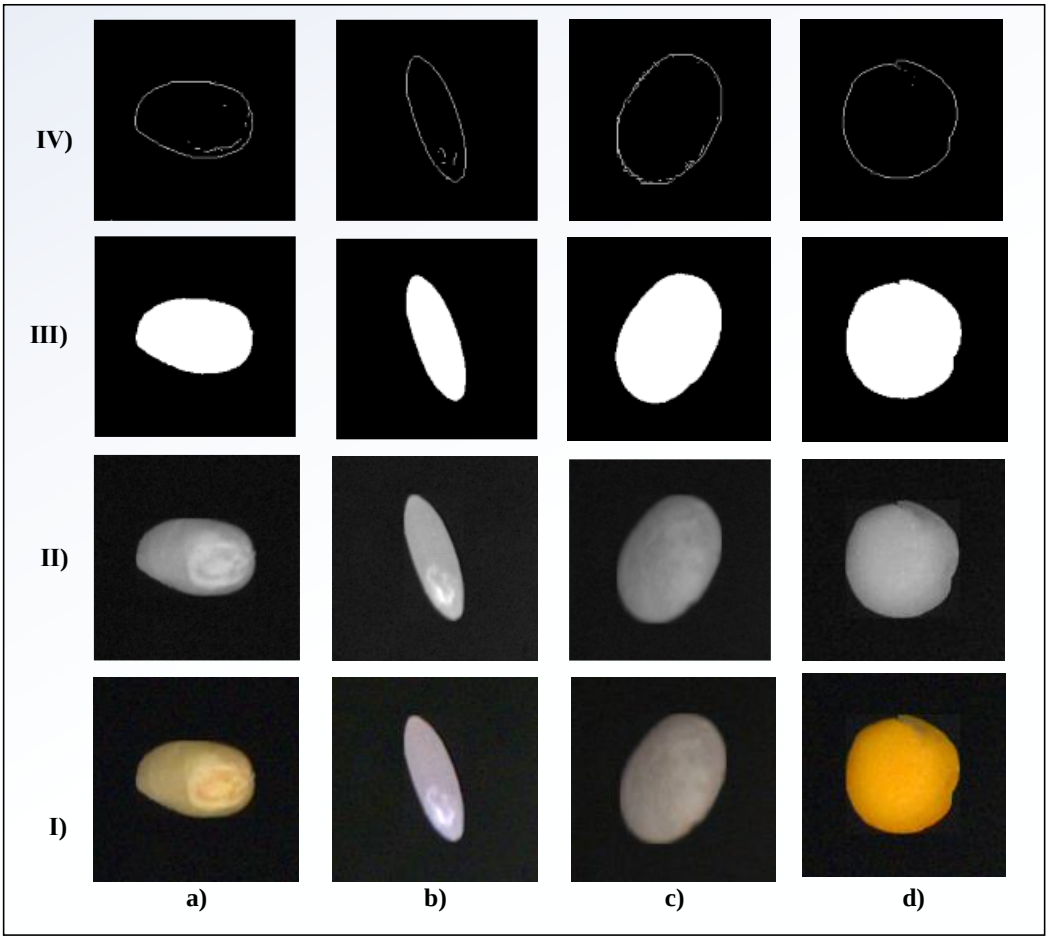


Şekil 3.2. Kayıtlı örnek tohum görüntüleri **a)** Buğday tohumu; **b)** Pirinç tohumu; **c)** Kırmızı mercimek tohumu; **e)** Yeşil mercimek tohumu

3.2.1. Tohum sınıflandırma problemi veri seti

Derin öğrenme birçok alanda kullanılmakta olup; bunlardan bir tanesi de tohum sektörüdür. Bu tez çalışmasındaki ana amaç derin öğrenmenin endüstriyel uygulamalarda yüksek hızlarda ve doğrulukta gerçeklenmesi için özel ağ modellerinin önerilmesi ve bu modellerin optimize edilmesidir. Böylesi bir gerçeklemeyi yapabilmek için tohum sektöründen farklı problemler seçilmiştir. Bu problemlerden ilki tohum sınıflandırma problemidir. Tohum sınıflandırma problemi tohum ayıklama problemine göre daha kolay bir problem olup; bu problemi biraz daha enteresan ve zor hale getirmek için bu probleme ait 4 farklı veri seti oluşturulmuştur. Bu veri setleri sırasıyla: RGB veri seti, Gri formatta veri seti, İkili Görüntü veri seti ve kenar bilgisi içeren veri setidir (Aktas H. ve Polat O. 2020). Veri setleri oluşturulurken dört farklı tohum türü kullanılmıştır. Bunlar: buğday tohumu, pirinç tohumu, kırmızı mercimek tohumu ve yeşil mercimek tohumudur. Şekil 3.1'deki deney düzeneği kullanılarak buğdaya ait 3153 görüntü, kırmızı mercimeğe ait 3969 görüntü, pirince ait 3736 görüntü yeşil

mercimeğe ait 3507 görüntü kaydedilmiştir. Dört farklı tohum için oluşturulan dört farklı veri seti Şekil 3.3‘deki gibidir.



Şekil 3.3. Dört farklı tohum için dört farklı veri toplamda 16 farklı veri setine ait örnek görüntüler **a)** Buğday tohumu; **b)** Pirinç tohumu; **c)** Kırmızı mercimek tohumu; **d)** Yeşil mercimek tohumu; **I)** RGB Görüntü; **II)** Gri formatta görüntü; **III)** İkili görüntü; **IV)** Kenar bilgisine sahip görüntü

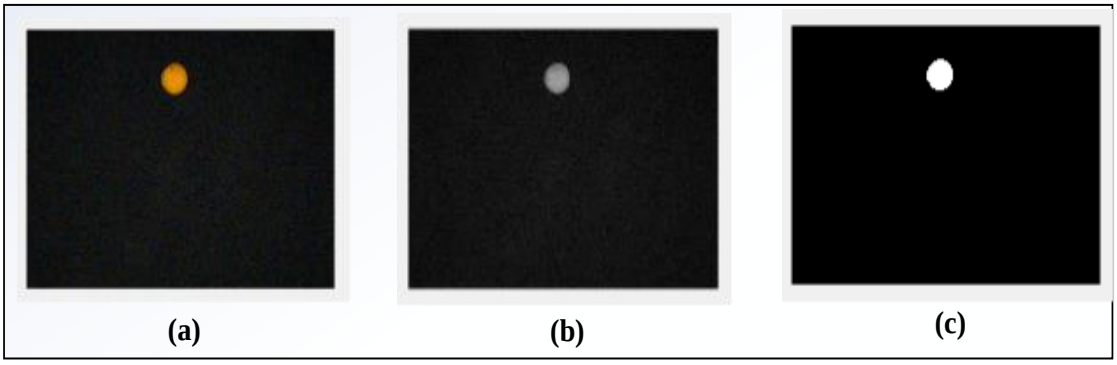
3.2.2. Tohum ayıklama problemi veri seti

Bu tez çalışmasında, sağlam kara buğday tohumu ve sağlam kara buğday tohumları içinde istenmeyen nesneleri içeren bir veri seti ayıklama problemi için oluşturulmuştur. Veri setine ait bazı örnek görüntüler Şekil 3.4‘deki gibidir. Şekil 3.4a’da sağlam kara buğday tohumları, Şekil 3.4b’de ise çer çöp diye ifade edebileceğimiz istenmeyen nesneler görünmektedir. Veri seti oluşturulurken daha önceden makinelerden ayıklanmış kara buğday ve çer çöp diye adlandırılan nesneler deney düzeneğine sırası ile konulmuş ve her iki sınıf için de sırası ile görüntü kayıtları yapılmıştır. Bu sayede görüntüler üzerinde etiketleme sorunu ortadan kaldırılmıştır. Şöyle ki bir klasörde sadece sağlam kara buğday tohumları diğer klasörde ise sadece çer çöp diye ifade edebileceğimiz nesneler kaydedilmiştir. Bu şekilde sağlam kara buğday için toplam 3246 görüntü, çer çöp için ise toplam 2625 görüntü kaydedilmiştir.

Şekil 3.4. Tohum ayıklama veri seti için örnek görüntüler **a)** Sağlam kara buğday; **b)** Kara buğday tohumlarının içinden çıkan çer çöp diye tarif edilen istenmeyen maddeler

3.2.3. Nesne tespit algoritmasının (blob detection algorithm) kayıtlı veriler üzerinde uygulanması

Şekil 3.5'deki görüntülerin boyutları 1440 x 1080 olup; bu görütülerdeki tohumları tek tek tespit edebilmek için Nesne Tespit Algoritmasının kullanılması gerekmektedir. Nesne Tespit algoritması endüstride kendine çok geniş kullanım alanı bulmaktadır. Özellikle ayıklama (sorting) ve hata tespiti (inspection) uygulamalarında yoğun olarak kullanılmaktadır. Şekil 3.5a'daki RGB Görüntüye Nesne Tespit Algoritmasının uygulanabilmesi için görüntü öncelikle gri formata sonra da ikili formata çevrilmesi gerekmektedir. Bu çevrimlerin sonucu olan görüntüler aşağıdaki gibidir (Şekil 3.5b ve Şekil 3.5c).



Şekil 3.5. 1440 x 1080 boyutundaki örnek kayıtlı görüntüler **a)** RGB formatındaki kırmızı mercimek tohumu görüntüsü; **b)** Aynı tohuma ait gri formattaki görüntü; **c)** Aynı tohuma ait ikili formattaki görüntü

İkili formattaki görüntüye Nesne Tespit algoritmasının uygulanması sonucu algoritma çıktı olarak nesneye ait temel bilgileri verecektir. Uygulanan yönteme ait MATLAB algoritması aşağıdaki gibidir:

for j ← 1 **to** number_of_images **do**

image ← **read**(image(j))

gray_image ← **rgb2gray**(image)

binary_image ← **threshold** (gray_image)

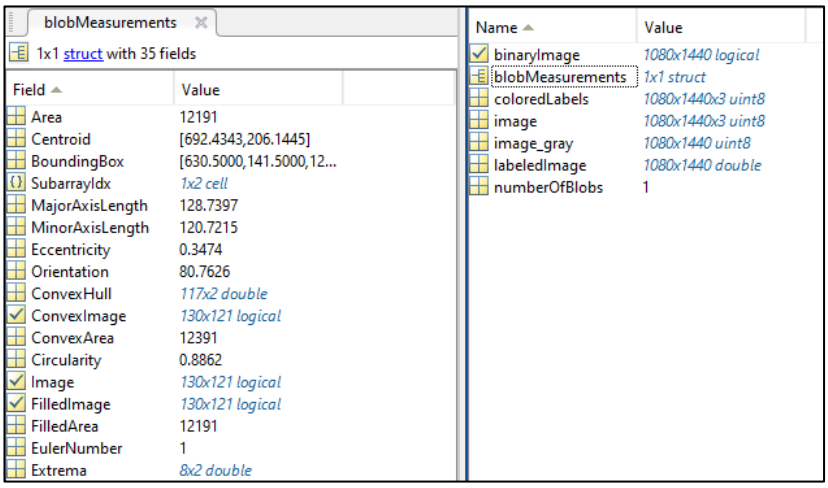
labeled_image ← bwlabel(binary_image,8)

coloredLabels = **label2rgb** (labeled_image, 'hsv', 'k', 'shuffle');

blobMeasurements = **regionprops** (labeled_image, image_gray, 'all');

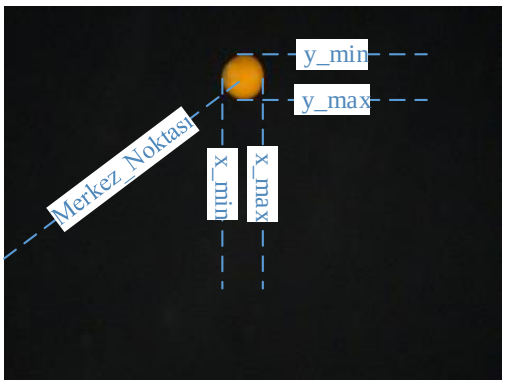
end for

Nesnelere ait temel bilgileri "blobMeasurements" yapısı içermekte olup; bu yapının içine detaylı bir şekilde bakılırsa Şekil 3.6 gibi değerleri içermektedir.



Şekil 3.6. Nesnelere ait temel bilgilerin MATLAB çalışma klasöründeki görünümü

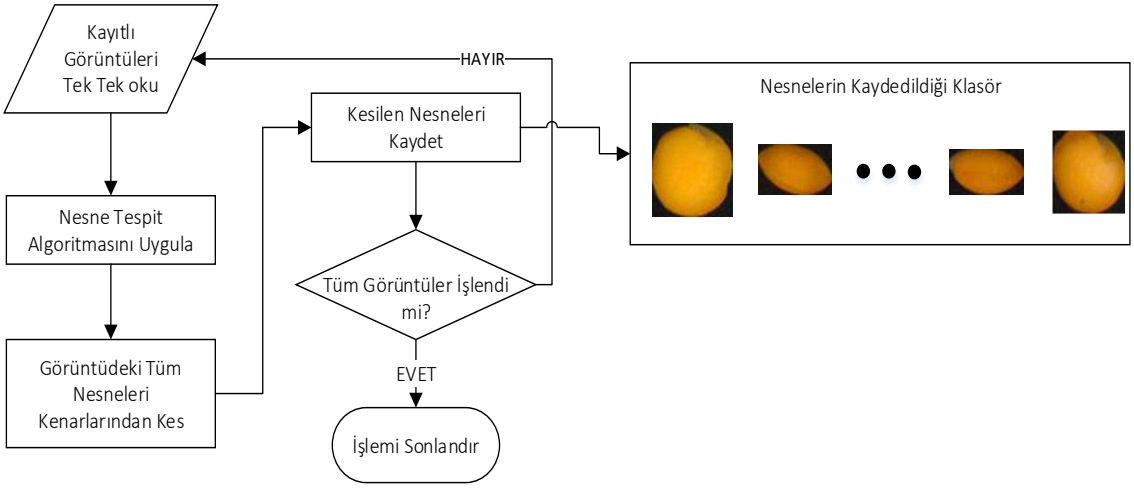
Görüntünün içindeki tohumları görüntüden kesmek için "blobMeasurements. Centroid" ve "blobMeasurements.Extrama" parametrelerinin kullanılması yeterli olacaktır. Bu parametrelerin Şekil 3.5a üzerindeki gösterimi Şekil 3.7'deki gibi olacaktır.



Şekil 3.7. Nesneye ait parametrelerinin görüntü üzerinde gösterilmesi

Bu işlemin uygulanmasından sonra 'x_min, x_max, y_min ve y_max noktası tespit edilen tohumlar sınırlarından kesilerek yeni görüntü olarak kaydedilirler. Nesne Tespit Algoritmasının kayıtlı görüntüler üzerinde uygulanması ve algoritmanın kayıtlı

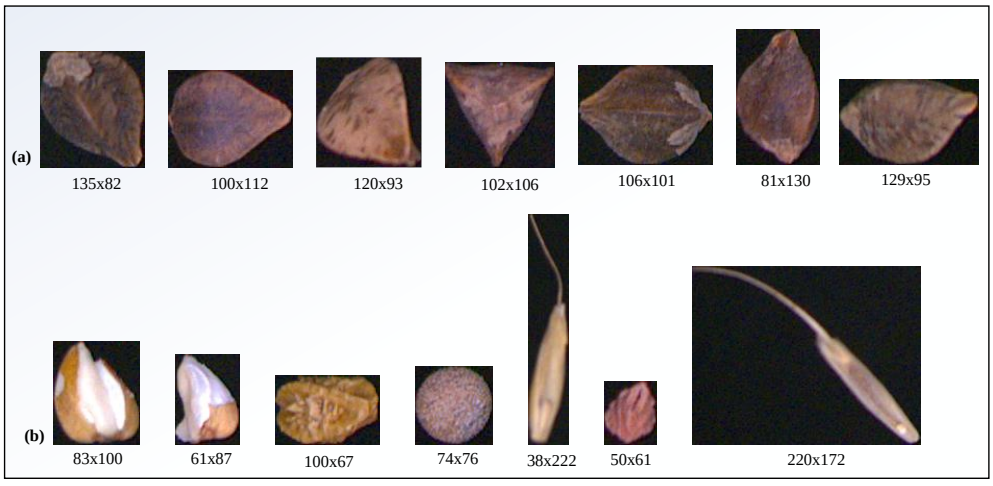
görüntülerdeki (1440 x 1080) tüm nesneleri bulması işlemi Şekil 3.8'deki gibi özetlenmektedir.



Şekil 3.8. Nesne tespit algoritmasının kayıtlı görüntülere uygulanması

3.2.4. Farklı ölçeklendirme yöntemlerinin kullanılması

Kayıtlı görüntüler üzerine Nesne Tespit Algoritması uygulandıktan sonra bu görüntüler üzerindeki nesneler (tohumlar) kenarlarından kesilerek bir klasörde kaydedilirler. Kaydedilen bu nesne dataları hem tohumların boyutlarının farklı olmasından hem de düşerken x-z ekseninde hareket etmelerinden dolayı farklı datalar farklı boyutlarda olabilmektedir. Bu durum çer çöp datası için çok daha göze çarpmaktadır. Şekil 3.9'da bu durumu daha iyi anlamamızı sağlayan kayıtlı nesne datalarını görebiliriz.

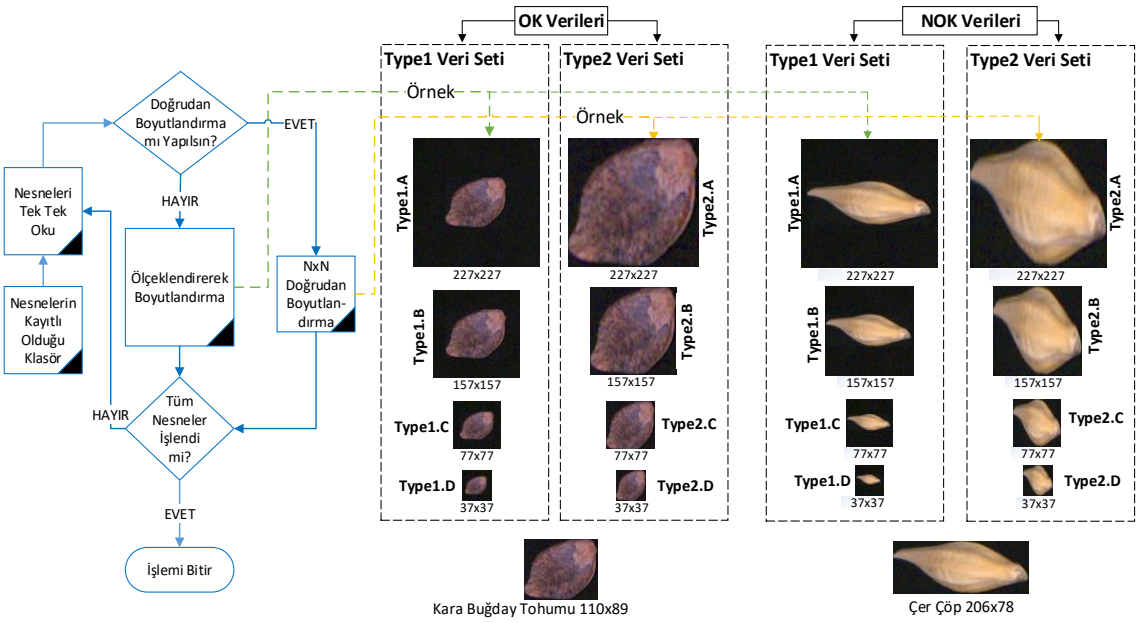


Şekil 3.9. Tohumların serbest düşme esnasında farklı şekillerde görüntülenmesi **a)** Kara buğday tohumları; **b)** Kara buğday tohumlarının içinde çıkan çer çöp maddeleri

Şekil 3.9'u inceleyecek olursak tohum dataları farklı boyutlardadır. Bu kayıtlı datalar bir sonraki aşamada evrişimli sinir ağı modelinde eğitim ve test veri setlerinde kullanılacaktır. Evrişimli sinir ağ modellerinin giriş data boyutları kullanılan modele

göre farklılık göstermekte olup; bazı ağ modellerinin giriş data boyutları şu şekildedir: AlexNet=227x227x3, MobileNet-V2=224x224x3, Inceptionv-3 =299x299x3, VGG-16=224x224x3. Şekil 3.9'daki görüntülerin bu ağ modellerinde eğitim ve test aşamasında kullanılabilmesi için öncelikle kayıtlı bu verilerin ağın girişine uygun boyutlara getirilmesi gerekmektedir. Bu tez çalışmasında görüntüyü boyutlandırmak için iki farklı yöntem kullanılmıştır.

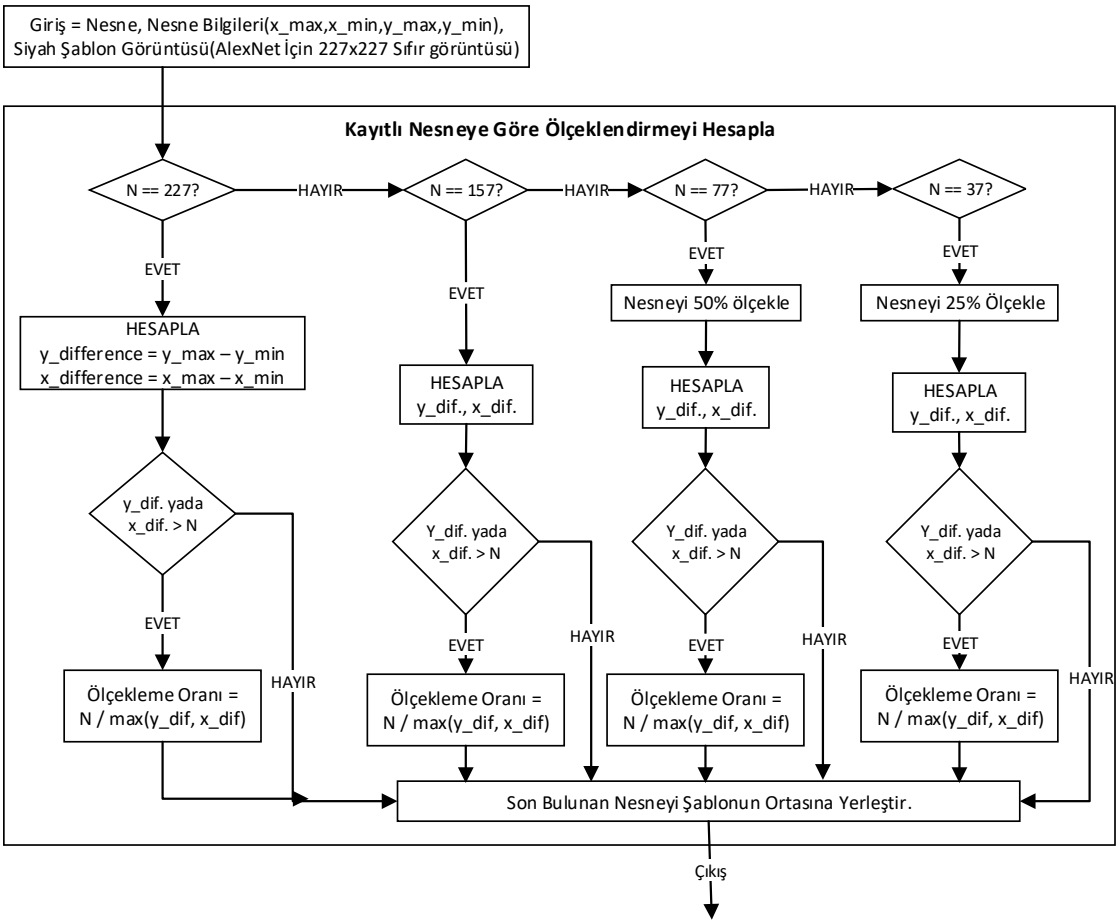
1. Doğrudan boyutlandırma (Type2 Veri Seti)
2. Ölçeklendirerek boyutlandırma (Type1 Veri Seti)



Şekil 3.10. İki farklı ölçeklendirme işlemi ve örnek uygulamaları

Şekil 3.10'da gösterildiği gibi boyutlandırma için kullanılan bu yöntemlerden birincisi standart kaba ölçeklendirme işlemi olup; ikinci yöntem ise nesnenin x/y oranını koruyacak şekilde boyutlandırılmasıdır. Yapılan bu ikinci işlemin blok diyagramla gösterimi Şekil 3.11'deki gibidir. Bu tez çalışmasında tohum ayıklama problemi için Type1 ve Type2 olmak üzere iki farklı veri türü üretilmiştir. Şekil 3.9'da belirtilen OK verileri sağlam kara buğday tohumlarını, NOK verileri de çer çöp nesnelerini içermektedir. Şekil 3.10'a dikkatlice bakılırsa 227x227, 157x157, 77x77 ve 37x37 olmak üzere toplam dört farklı boyutta veri seti hem OK hem de NOK için üretilmiştir. Bu boyutların hepsi anlamlı şekilde seçilmiştir. Örneğin 227x227 data boyutu bir sonraki bölümde veri setinin AlexNet ile eğitilip test edilmesi için seçilmiştir. Yine 157x157 data boyutu ise tamamen probleme özgü olarak seçilmiştir. Şöyle ki OK verileri analiz edildiğinde en büyük boyutlu kara buğday tohumları x ekseninde ve y ekseninde yaklaşık olarak 150 piksel değerindedir. 1440 x 1080 boyutundaki görüntüden buğday tohumunu kestikten sonra herhangi bir ölçekleme yapmadan uygun bir şablona (siyah görüntüye) yerleştirilmek istenirse bu şablonun değeri 157x157 olarak seçilebilmektedir. Bu sayede tohuma herhangi bir ölçekleme yapmadan sığabileceği minimum boyuttaki şablona yerleştirilir. Diğer 77x77 ve 37x37 boyutları

ise ileride anlatılacak olan ölçekle ve budama metodunda kullanılması için geliştirilmiş veri setleridir.



Şekil 3.11. Ölçeklendirerek boyutlandırma işleminin detaylı gösterimi

3.3. Probleme Uygun Eğitilmiş Ağ Yapılarının Seçilmesi

AlexNet'in elde ettiği başarıdan bu yana literatürde birçok CNN ağ yapısı önerilmiştir. Önerilen ağ yapıları doğruluk (accuracy), işlem yükü (FLOPs) veya toplam parametre sayısının bir önceki ağ yapısına göre daha üstün olmasına göre sürekli gelişim göstermektedir. Önerilen bu ağ yapıları ImageNet gibi çok büyük veri setlerinde testleri yapıldıktan sonra literatürdeki diğer ağ yapıları ile karşılaştırılmaktadır. ImageNet veri seti gibi veri setleri içlerinde milyonlarca görüntü ve yüzbinlerce sınıf içermektedir. Nitekim bu veri setleri oluşturulurken doğal ortamlardan ve dış mekanlardan görüntüler çekilerek veri setine eklenmiştir. Dış ortam görüntülerin en büyük sorunu ise arka plandaki karmaşıklık ve kontrast değerinin sürekli değişken olmasıdır. Bu da ImageNet gibi veri setlerini yüksek doğruluklarda gerçeklemek için zor bir problem haline getirmektedir. Bu tez çalışmasında ise tohumların sınıflandırılması ve ayıklanması problemleri derin öğrenme ile gerçeklenecektir. Şekil 3.1'deki deney düzeneğinden de görüldüğü gibi veri setleri tamamen kontrollü bir

ortamda (aydınlık seviyesi, tohum görüntülerinin sabit çözünürlükte olması) geliştirilmektedir. Bu da gerçeklenmek istenen problemi ImageNet gibi veri setlerine göre daha kolay hale getirmektedir. Nitekim bu tez çalışmasında öncelikli hedef doğruluktan en fazla %1 ödün vererek, bu problemlerin yüksek hızlarda yapılması ve derin öğrenme ile sınıflandırma işleminin gerçek zamanlı sistemlere uygun hale getirilmesidir. Literatürde birçok ağ modeli olup; ağın çıkarım süresinin (inference time) hızlı olabilmesi için hem ağın işlem yükünün düşük olması hem de hafıza işlemlerinin az olması gerekmektedir. Yine kullanılan donanım da (GPU,CPU,Mobil GPU) çıkarım süresini etkilemekle birlikte CNN yapısının GPU'da koşturulması için kullanılan kütüphane de hızı etkilemektedir. Bu tez çalışmasında tohumların derin öğrenme ile sınıflandırılması ve ayıklanması işlemlerini literatürdeki eğitilmiş ağ yapıları ile yapabilmek için en bilindik ve GPU çıkarım süresi en düşük (Bianco vd. 2018) olduğu bilenen AlexNet ile bu yapıya göre biraz daha yavaş olan MobileNet ağ yapısı seçilmiştir.

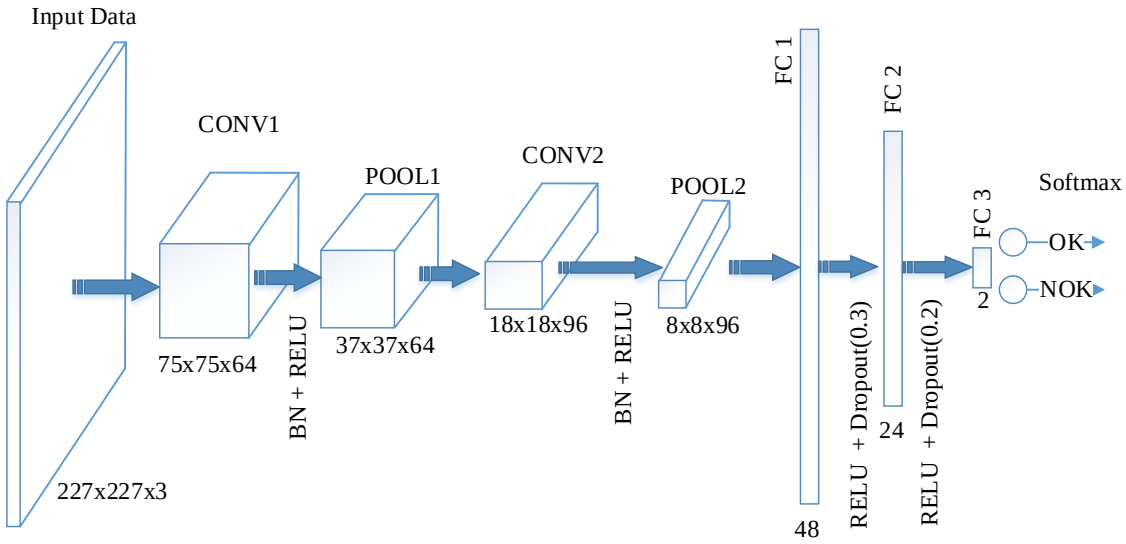
AlexNet ve MobileNet-V2 eğitilmiş ağ yapıları daha önceden oluşturulan tohum sınıflandırma ve tohum ayıklama veri setlerinin eğitilmesinde ve test edilmesinde kullanılacaktır. Tez çalışmasındaki eğitim ve test işlemleri Matlab R2019b platformunda gerçeklenmiş olup; AlexNet ve MobileNet-V2 ağ yapıları Matlab Deep Learning Toolbox'ında eklenti olarak yüklenebilmektedir. Bu ağ yapıları daha önceden ImageNet veri seti ile eğitilmiş olup ağların çıkış katmanında 1000 sınıf bulunmaktadır. Bu ağ yapıları uygulamaya özgü olarak kullanılmak istenildiğinde (Örneğin tohum ayıklama probleminde OK ve NOK olmak üzere toplam iki adet sınıf bulunmaktadır.) çıkış katmanındaki sınıf sayısı güncellenerek kullanılabilmektedir. Ağların ImageNet veri seti ile önceden eğitilmiş olması yüksek doğruluk oranına daha az epocla ulaşılmasını sağlamaktadır. Bu da eğitim süresini önemli ölçüde azaltmaktadır.

3.4. Uygulamaya Özel Ağ Modelinin Önerilmesi

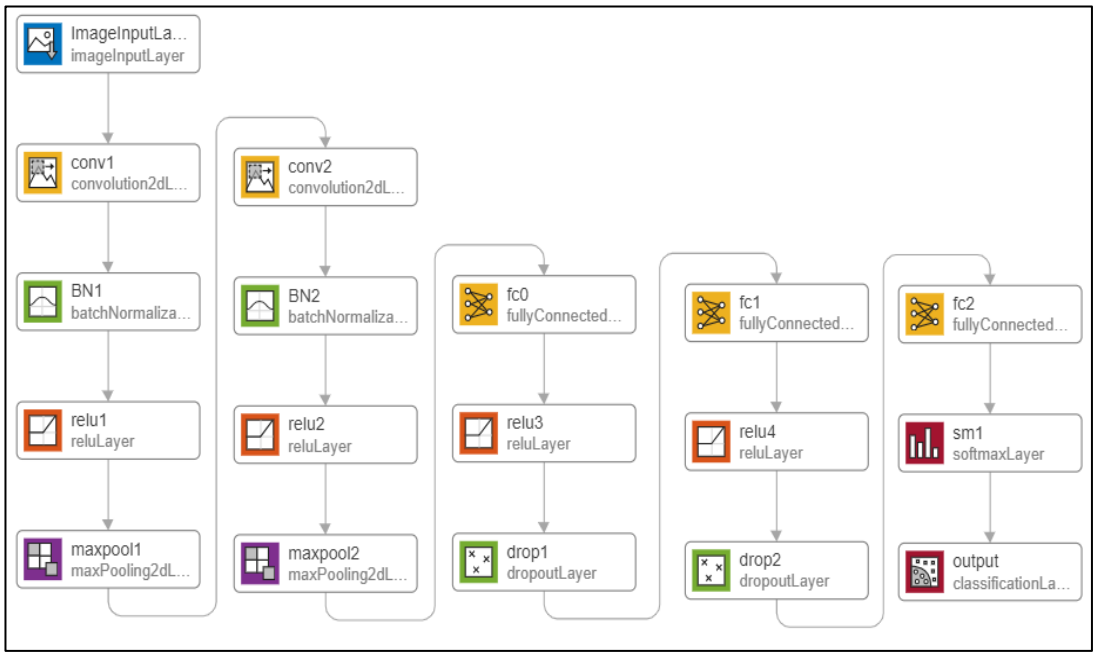
CNN yapılarında işlem yükünün %90'dan fazlası evrişim katmanındaki çarpma işlemlerinden kaynaklanmaktadır (Nakahara ve Sasao 2015). Bu katmanda yapılacak ufak değişiklikler işlem sayısını önemli ölçüde değiştirecektir. Konvolüsyon katmanındaki işlem yükü daha önce denklem 3.3'deki gibi ifade edilmiştir. Cin, Cout, Hin, Hout ve K parametrelerinden herhangi birinin değiştirilmesiyle işlem yükü otomatik olarak değişecektir. Bu açıdan bakıldığında evrişim katmanındaki filtre sayısı, filtre boyutu ve data boyutu ne kadar küçük ise işlem yükü de o kadar az olacaktır. Evrişim katmanındaki işlem yükü göze alındığında, bu tez çalışmasında doğruluktan ödün vermeden (maksimum %1 kayıp) düşük işlem yüküne sahip literatürdeki diğer ağ yapılarına göre daha az katmana sahip özel bir ağ yapısı önerilmiştir.

Tipik olarak CNN mimarileri, evrişimli katmanlar ve çıkış katmanlardan oluşurlar. Evrişimli bir katman, evrişim (Convolution) + Doğrultulmuş Doğrusal Birim (ReLU) + Yığın Normalleştirme (BN) içerir. Ana blok evrişim bloğudur ve diğer ReLU

ve BN blokları isteğe bağlı olabilir (Rosebrock 2017). Çıkış katmanında ise tamamen bağlı katmanlar ve sınıflandırıcı katmanlar bulunur. Bu çalışmada, sınıflandırma problemi çok karmaşık olmadığı için derin bir ağ yerine 2 evrişimli katman ve çıkış katmanından oluşan özel bir ağ modeli önerilmiştir. Giriş görüntü boyutu 227x227 için önerilen Custom227 ağ yapısının gösterimi ise Şekil 3.12'deki gibidir Aynı ağın Matlab Deep Network Designer Toolbox ile tasarlanmış hali Şekil 3.13'deki gibidir.



Şekil 3.12. Tohum sınıflandırma problemi için önerilmiş özel ağ yapısı (Custom227)



Şekil 3.13. Custom227 ağ yapısının Matlab Deep Network Designer ile tasarlanması

3.4.1. Uygulamaya özel ağ modelinin farklı parametreler için incelenmesi

Şekil 3.12'deki uygulamaya özel ağ yapısının giriş görüntü boyutu 227x227 bilinçli olarak seçilmiştir. Buradaki amaç AlexNet ile eğitilip test edilen veri seti devamında uygulamaya özel model ile eğitilecek ve veri seti üzerinde hiçbir ölçekleme yapılmadan AlexNet ile uygulamaya özel modelin başarısı karşılaştırılacaktır. Yine bu tez çalışmasında dört farklı boyutta (227x227, 157x157, 77x77, 37x37) tohum ayıklama veri seti geliştirilmiş ve bunun sebepleri detaylı bir şekilde açıklanmıştır. Bu dört farklı veri setini test edebilmek için dört farklı boyutta özel ağ yapısı önerilmiştir. Önerilen bu özel ağ yapıları, ağ yapılarının katmanları, evrişim katmanlarındaki filtre sayıları ve filtre boyutları, atlama miktarı ve dolguluma türü Çizelge 3.1'de detaylı bir şekilde verilmiştir. Custom227 ağını inceleyecek olursak: bu ağın conv1 katmanında 5x5 boyutlarında 64 adet filtre vardır. Denklem 3.5'e göre bu katmanın çıkışındaki datanın boyutları şu şekilde hesaplanacaktır: (227-5)/3)+1 = 75. Toplam 64 tane filtre olduğu için havuzlama katmanında 75x75 boyutunda 64 adet data bulunacaktır. Yani kısa gösterimle 75x75x64 olarak gösterilmektedir. Havuzlama 1 katmanının çıkışı ise Denklem 2.6'ya göre şu şekilde hesaplanır: ((75-2)/2) +1)= 37. Tüm bu işlemler ağın çıkış katmanına doğru tekrarlandıkça datanın katmanlar üzerinde ilerlerken boyutunun giderek küçüldüğünü en son Tamamen Bağlı Katman çıkışında ise data boyutunun 1x1'e indiğini görmekteyiz. Burada dikkat edilmesi gereken husus özel bir ağ modeli önerildiğinde TBK katmanlarından önce data boyutunun filtre boyutu ve atlama miktarına göre kabul edilebilir en küçük seviyeye kadar küçülmesi gerekmektedir. Örneğin conv 2 katmanının girişindeki datanın boyutu 4x4 iken böyle bir dataya conv katmanında 5x5'lik bir filtre uygulanmasından bahsedilemez. Böyle bir durumda ağ oluşturulurken yazılım derleme hatası verecektir. Bu açıdan bakıldığında uygulamaya özel ağı oluştururken bu parametrelerin seçilmesi kullanıcı tarafından dikkatli bir şekilde yapılması gerekmektedir.

Çizelge 3.1. Dört farklı boyuttaki veri seti için önerilmiş dört farklı özel ağ modeli ve katmanlardaki data boyutları. TBK_1 (Tamamen Bağlı Katman 1), Custom227 (giriş data boyutu 227x227 olan özel ağ modeli), Custom157 (giriş data boyutu 157x157 olan özel ağ modeli), Custom77 (giriş data boyutu 77x77 olan özel ağ modeli), Custom37 (giriş data boyutu 37x37 olan özel ağ modeli)

Katmanlar	**Filtre Boyutu**	**Atlama**	**Dolgulama**	**Custom227* Ağının Katman Çıkış Boyutları**	**Custom157* Ağının Katman Çıkış Boyutları**	**Custom77* Ağının Katman Çıkış Boyutları**	**Custom37* Ağının Katman Çıkış Boyutları**
Giriş Katmanı				227x227x3	157x157x3	77x77x3	37x37x3
Conv_1	5x5	3	0	75x75x64	51x51x64	25x25x64	11x11x64
Havuzlama_1	2x2	2	0	37x37x64	25x25x64	12x12x64	5x5x64
Conv_2	3x3	2	0	18x18x96	12x12x96	5x5x96	2x2x96
Havuzlama_2	2x2	2	0	9x9x96	6x6x96	2x2x96	1x1x96
TBK_1	-	-	-	1x1x48	1x1x48	1x1x48	1x1x48

Çizelge 3.1'in devamı

TBK_2	-	-	-	1x1x24	1x1x24	1x1x24	1x1x24
TBK_3	-	-	-	1x1x2	1x1x2	1x1x2	1x1x2
Softmax	-	-	-	1x1x2	1x1x2	1x1x2	1x1x2

Özel ağ yapıları Çizelge 3.1'de boyut hesabı için detaylı bir şekilde incelenmiş olup; aynı ağ yapılarını işlem yükü ve kullanılan parametre sayısı açısından değerlendirecek olursak sonuçlar Çizelge 3.2 ve Çizelge 3.3'deki gibi olacaktır. Tüm bu değerler daha önce belirtilen denklemler ile hesaplanmıştır.

Çizelge 3.2. Dört farklı özel ağ modelindeki her katmana ait FLOPs değeri

Katmanlar	**Custom227**	**Custom157**	**Custom77**	**Custom37**
	FLOPs	FLOPs	FLOPs	FLOPs
Conv_1	27.360.000	13.152.256	3.040.000	700.416
Havuzlama_1	360.000	173.056	40.000	9.216
Conv_2	17.947.008	7.976.448	1.384.800	221.568
Havuzlama_2	31.104	13.824	2.400	384
TBK_1	373.296	165936	18.480	4.656
TBK_2	1.176	1.176	1.176	1.176
TBK_3	98	98	98	98
Softmax	0	0	0	0
TOPLAM	46.072.682	21.482.794	4.486.954	937.514

Çizelge 3.3. Dört farklı özel ağ modelindeki her katmana ait parametre sayısı

Katmanlar	**Custom227**	**Custom157**	**Custom77**	**Custom37**
	Parametre Sayısı	Parametre Sayısı	Parametre Sayısı	Parametre Sayısı
Conv_1	4.864	4.864	4.864	4.864
Havuzlama_1	0	0	0	0
Conv_2	55.392	55.392	83.040	55.392
Havuzlama_2	0	0	0	0
TBK_1	17.915.952	7.962.672	884.784	221.232
TBK_2	27.672	27.672	27.672	27.672
TBK_3	98	98	98	98
Softmax	10	10	10	10
TOPLAM	18.003.988	8.050.708	1.000.468	309.268

3.5. Evrişimli Sinir Ağ Yapılarında FLOPs ve Parametre Sayılarının Hesaplanması

Literatürde önerilmiş bir çok evrişim sinir ağ yapısı bulunmaktadır. Bu ağ yapıları birbiri ile karşılaştırılırken FLOPs ve parametre sayısı gibi bazı parametreler kullanılmaktadır. Bu parametrelerin nasıl hesaplandığını daha iyi anlamak için evrişimli sinir ağlarına ait tipik katmanların (evrişim, havuzlama, tamamen bağlantılı katmanlar) tek tek incelenmesi ve buradaki FLOPs ve parametre hesaplarının denklem ile gösterilmesi gerekmektedir. Tüm bu işlemlerden önce bazı tanımların yapılması gerekmektedir. Girişi data boyutu HinxWin, çıkış data boyutu HoutxWout, evrişim katmanındaki filtre sayısı N, evrişim katmanındaki atlama sayısı Sc, evrişim katmanındaki filtre boyutu KcxKc, havuzlama katmanındaki atlama sayısı Sp, havuzlama katmanındaki filtre boyutu Kp, mevcut katman için (evrişim_1 ya da havuzlama_2) giriş kanal derinliği Cin, mevcut katman için (evrişim_1 ya da havuzlama_2) çıkış kanal derinliği Cout ve ekleme sayısı(padding) P olmak üzere FLOPs ve parametre hesabı için gerekli tüm denklemler aşağıdaki gibidir (Anonymous 2; Mallick and Nayak 2018). Yine bu hesaplamaların yapılması için bazı yardımcı denklemlerinde tanımlanması gerekmektedir. Şekil 3.12'deki önerilmiş özel evrişim yapısı incelendiğinde girişi data boyutu 227x227 iken birinci evrişim katmanının çıkışında data boyutu 75x75, derinliği ise 64'tür. Yine havuzlama katmanlarının giriş ve çıkışları farklıdır. Data çıkış boyutlarının hesaplanması için gerekli olan denklemler FLOPs ve parametre hesaplarındaki denklemlerde yardımcı denklem olarak kullanılacaktır.

Hout = Wout olmak üzere denklem 2.5 Kc parametresi ile yazılacak olursa ; evrişim katmanlarının çıkış data boyutu hesabı denklem (3.1)'deki gibidir.

$$H_{out} = W_{out} = \frac{H_{in} - K_c + 2P}{S} + 1 \tag{3.1}$$

Denklem 2.6 Kp parametresiyle yazılacak olursa; havuzlama katmanlarının çıkış data boyutu hesabı denklem (3.2)'deki gibidir.

$$H_{out} = W_{out} = \frac{H_{in} - Kp}{S} + 1 \tag{3.2}$$

Yukarıdaki yardımcı denklemlerin kullanılması ile tüm FLOPs değerleri şu şekilde hesaplanır. Evrişim katmanlarındaki FLOPs hesabı denklem (3.3)'deki gibidir:

$$FLOPs_{evrisim} = (KxKxC_{in}xC_{out} + C_{out})xH_{out}xW_{out} \tag{3.3}$$

Havuzlama katmanlarındaki FLOPs hesabı denklem (3.4)'deki gibidir:

$$FLOPs_{havuzlama} = H_{in}xW_{in}xC_{in} \tag{3.4}$$

Tamamen bağlantılı katmanlardaki FLOPs hesabı denklem (3.5)'deki gibidir:

$$FLOPs_{tbk} = C_{in}xC_{out} + C_{out} \quad (3.5)$$

Evrişim katmanlarındaki toplam parametre sayısı (Number of paramteres) denklem (3.6)'daki gibidir. Buradaki +1 sayısı evrişim katmanlarındaki filtrelerden gelen polarlama(bias) sayısını ifade etmektedir. Yani evrişim katmanında toplam 96 filtre varsa toplam 96 polarlama vardır.

$$Nop\_c = (KxKxC_{in} + 1)xC_{out} \quad (3.6)$$

Havuzlama katmanında eğitim sırasında güncellenen herhangi bir yapı olmadığı için havuzlama katmanında sıfır parametre vardır. Tamamen bağlantılı katmanlardaki parametre sayısı ise denklem (3.7)'deki gibi ifade edilir.

$$Nop\_tbk = (C_{in}xC_{out} + 1)xC_{out} \quad (3.7)$$

Şekil 3.12'deki Custom227 ağ yapısı detaylı bir şekilde incelenirse FLOPs ve parametre sayılarına ilişkin hesaplamalar aşağıdaki gibi olacaktır.

- Conv1 katmanındaki Hout,Wout, FLOPs ve paramatre sayısı denklem (3.1), (3.3) ve (3.6)'ya göre şu şekilde hesaplanmaktadır:
  Hout = Wout = (227-5)/3+1)=75
  FLOPs = (5x5x3x64 + 64)x75x75
  = 27.360.000
  Parametre Sayısı = ((5x5x3)+1)x64
  = 4.864

- Havuzlama_1 katmanındaki Hout,Wout, FLOPs sayısı denklem (3.2) ve (3.4)'e göre şu şekilde hesaplanmaktadır:
  Hout = Wout = (75-2)/2 = 37 (sonuç 36,5 olmasına rağmen dolguluma(padding) olduğu için her zaman küsürlü sayının bir üst değeri alınır)
  FLOPs= 75x75x64 = 360.000
  Parametre Sayısı = 0

- Conv2 katmanındaki Hout,Wout, FLOPs ve paramatre sayısı denklem (3.1), (3.3) ve (3.6)'ya göre şu şekilde hesaplanmaktadır:
  Hout = Wout = (37-3)/2+1)=18
  FLOPs = (3x3x64x96 + 96)x18x18 = 17.947.008
  Parametre Sayısı = ((3x3x64)+1)x96 = 55.392

- Havuzlama_2 Katmanındaki FLOPs sayısı denklem (3.2) ve (3.4)'e göre şu şekilde hesaplanmaktadır:
  Hout = Wout = (18-2)/2 = 8
  FLOPs= (18)x(18)x96 = 31.104
  Parametre Sayısı = 0

- TBK_1 Katmanındaki FLOPs ve paramatre sayısı denklem (3.5) ve (3.7)'ye göre şu şekilde hesaplanmaktadır:
  Cin = Aktivasyon_Sayısı = HoutxWoutxCout = 9x9x96 = 7.776
  FLOPs= 7.776x48+48 = 373.296
  Parametre Sayısı = (7.776x48+1)x48=17.915.952

- TBK_2 Katmanındaki FLOPs ve paramatre sayısı denklem (3.5) ve (3.7)'ye göre şu şekilde hesaplanmaktadır:
  FLOPs = 48x24+24 = 1.176
  Parametre Sayısı = (48x24+1)x24 = 27.672

- TBK_3 Katmanındaki FLOPs ve paramatre sayısı denklem (3.5) ve (3.7)'ye göre şu şekilde hesaplanmaktadır:
  FLOPs = 48x2+2 = 98
  Parametre Sayısı = (2x24+1)x2 = 98

- Softmax Katmanındaki paramatre sayısı denklem (3.7)'ye göre şu şekilde hesaplanmaktadır:
  FLOPs = 0
  Parametre Sayısı = (CinxCout+1)xCout = (2x2+1)x2 = 10

3.6. Ölçekle ve Budama Metodunun Önerilmesi

Çizelge 3.2'de dört adet özel ağ modeli bulunup bu modellerin giriş data boyutu hariç tüm parametreleri (filtre boyutu, atlama sayısı, filtre sayısı ve havuzlama maskesi) aynıdır. Giriş data boyutunu ne kadar çok azaltırsak bu da çözünürlüğü o kadar çok azaltacaktır. Çözünürlüğün azalması aynı zamanda ağın doğruluk oranını etkileyecektir. Nitekim Çizelge 3.1'de önerilen özel ağ modellerinin giriş data boyutları daha önce detaylı bir şekilde anlatıldığı üzere bilinçli bir şekilde 227x227,157x157,77x77ve37x37 olarak seçilmiştir. Buğday tohumları 227x227 ve 157x157 boyutlarına tam sığacaklarından herhangi bir ölçekleme yapılmayacaktır. Bu da veri kaybı olmadan iki veri setinin de (227x227, 157x157) tam çözünürlükte olması demektir. 77x77 data boyutu 157x157 data boyutuna göre 4 kat daha küçük bir çözünürlük demektir. Yine 37x37 data boyutu da 157x157 data boyutuna göre 16 kat daha küçük bir çözünürlük demektir. Sadece giriş görüntü boyutunun değişimi açısından tablo yorumlanacak olursa: giriş data boyutu ortalama 2 kat azalınca (157/77=2,03) FLOPs 4 kattan daha fazla azalmaktadır (21.482.794/4.486.954 =4,78). Bu sonuçtan hareketle önereceğimiz özel ağ modelinin giriş data boyutunun ne kadar önemli olduğu şüphe gerektirmeyecektir. Ancak ağın giriş data boyutunun çözünürlüğe ve dolayısıyla doğruluğa etkisini göz ardı etmemek gerekmektedir.

Çizelge 3.4'de ise Custom227'de farklı parametrelerin FLOPs'a etkisi detaylı bir şekilde incelenmiştir. İlk sütundaki parametreler Şekil 3.12'de önerilen özel modele ait parametreleri içermektedir. N1 birinci evrişim katmanındaki filtre sayısı, N2 ikinci

evrişim katmanındaki fitlre sayısı, K1 birinci evrişim katmanındaki filtrelerin boyutu, K2 ikinci evrişim katmanındaki filtrelerin boyutu, S1 birinci evrişim katmanındaki atlama miktarı, S2 ikinci evrişim katmanındaki atlama miktarı olmak üzere; ikinci sütunda, birinci sütundaki tüm parametreler aynı kalmak üzere N1 ve N2 sayısı yarıya indirilmiştir. Üçüncü sütunda, K1 ve K2 parametreleri iki katına çıkarılmıştır. Dördüncü sütunda ise K1, K2 iki katına çıkarılmış olup N1 ve N2 yarıya indirilmiştir. Son sütunda ise K1, K2, S1 ve S2 dört parametre de iki katına çıkarılmıştır.

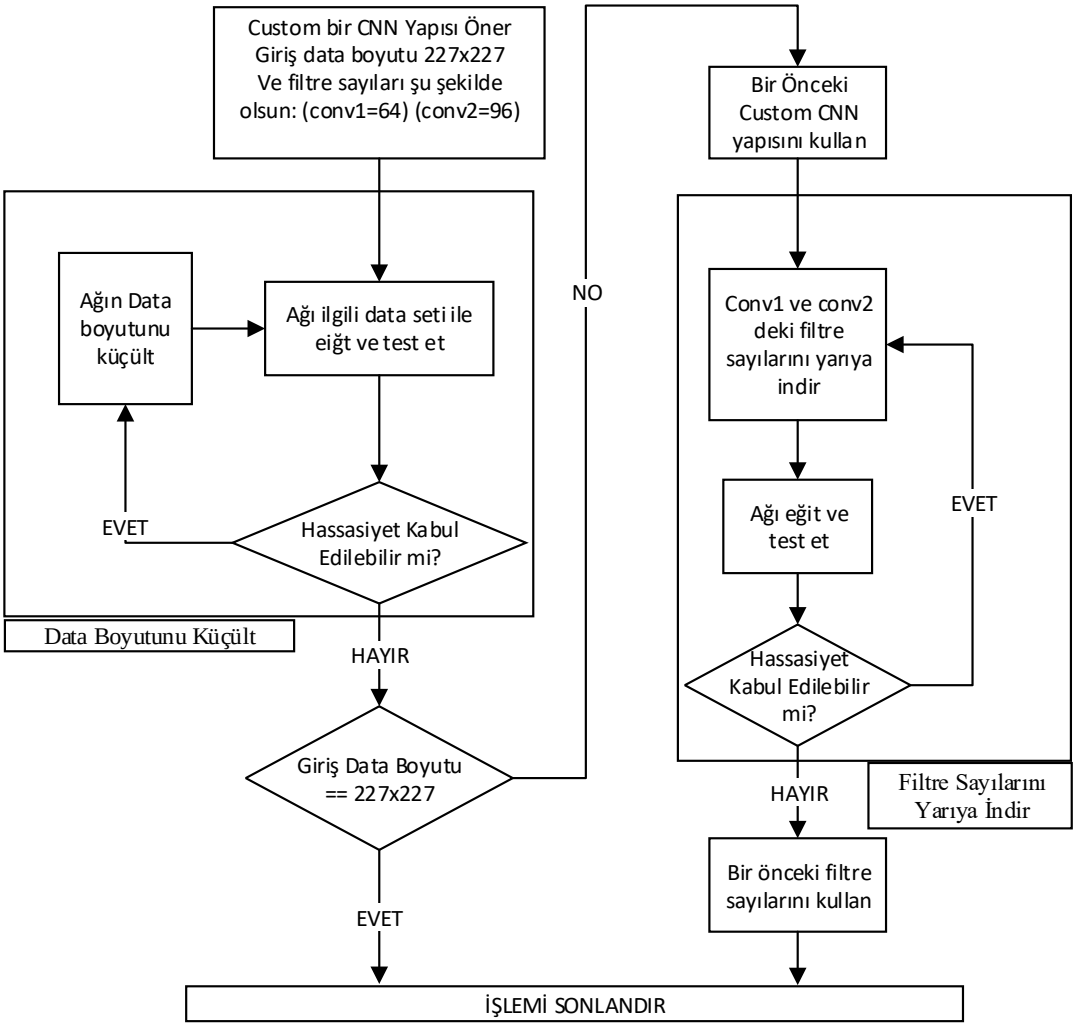
Çizelge 3.4. Farklı parametrelerin değişmesi sonucu FLOPs değişimi (N1=Conv1'deki filtre sayısı, K1= Conv1'deki filtrenin boyutu, N2=Conv2'deki filtre sayısı, K2= Conv2'deki filtrenin boyutu, S1= Conv1'deki atlama sayısı, S2= Conv2'deki atlama sayısı, Sp=Havuzlama katmanlarındaki atlama sayısı)

Katmanlar	**Custom227**				
	FLOPs (N1=64, N2=96, K1=5, K2=3, S1=3, S2=2)	FLOPs (N1=32, N2=48, K1=5, K2=3 S1=3, S2=2)	FLOPs (N1=64, N2=96, K1=10, K2=6, S1=3, S2=2)	FLOPs (N1=32, N2=48, K1=10, K2=6, S1=3, S2=2)	FLOPs (N1=32, N2=48, K1=10, K2=6, S1=6, S2=4)
Conv 1	27.360.000	13.680.000	102.657.856	51.328.928	13.186.208
Havuzlama_1	360.000	180.000	341.056	170.528	43.808
Conv 2	17.947.008	4.494.528	56.647.680	14.168.064	498.096
Havuzlama2	31.104	15.552	24.576	12.288	432
TBK_1	373.296	186.672	294.960	147.504	2352
TBK_2	1.176	1.176	1.176	1.176	1.176
TBK_3	98	98	98	98	98
Softmax	0	0	0	0	0
TOPLAM	46.072.682	18.558.026	159.967.402	65.828.586	16.671.786

Çizelge 3.4'de ki veriler eşliğinde filtre sayısının, filtre boyutunun ve atlama sayısının FLOPs üzerinde ne kadar etkili olduğu açık bir şekilde gösterilmektedir. Çizelge 3.4'deki sonuçlar Custom227 yapısına özgü olup aynı parametrelerin diğer Custom157, Custom77 ve Custom37 ağlarına doğrudan uygulanması söz konusu değildir. Filtre boyutunun büyük olması ve özellikle atlama değerlerinin büyük olması data boyutunun hızlı bir şekilde daralmasına ve TBK katmanlarına gelmeden boyut sorunundan dolayı derleme hatasına sebebiyet verecektir. Çizelge 3.4'de kullanılıp diğer ağ yapılarında da kullanılabilecek tek parametre evrişim ağlarındaki filtre sayısıdır. Yani Çizelge 3.4'e göre N1 ve N2 sayısıdır.

Çizelge 3.2'deki giriş data boyutunun FLOPs a etkisi ve Çizelge 3.4'deki filtre sayılarının FLOPs'a etkisi göz önünde bulundurularak bu tez çalışmasında iki aşamalı Ölçekle ve Budama yöntemi önerilmiştir. Bu sayede hassasiyetten ödün vermeden mevcut problem için en uygun data boyutu ve filtre sayısı yaklaşık değerlerde hesaplanmış olacaktır. Önerilen yöntem Şekil 3.14'deki gibidir. Bu yönteme göre ilk

önce uygulamaya özel bir CNN yapısı önerilmekte (bu çalışmada önerilen ağ yapısı başlangıç koşulunda 227x227 giriş data boyutuna ve (conv1=64) (conv2=96) filtre sayılarına sahiptir).



Şekil 3.14. Önerilen ölçekle ve budama yöntemi

Önerilen ağ yapısı uygun veri seti ile eğitildikten sonra eğer hassasiyet kabul edilebilir ise bir sonraki turda özel CNN modelinin giriş data boyutu 157x157'ye düşürülmekte ve bu ağ yapısı da 157x157'lik bir veri seti ile eğitilip test edilmektedir. Test sonucu hassasiyet kabul edilebilir bir seviyede ise bir sonraki turda giriş data boyutu ve veri seti 77x77 olarak güncellenmektedir. Bu işlem dördüncü ve son tur olan 37x37'lik data boyutunda sonlanmaktadır. Özel CNN giriş data boyutun karar verildikten sonra bir sonraki işlem ise özel CNN yapısındaki filtre sayılarının her turda yarıya indirilmesidir. Tüm bu işlemler gerçeklendikten sonra maksimum 1% hassasiyet kaybı olacak şekilde uygun özel CNN giriş data boyutuna ve evrişim katmanlarındaki filtre sayılarına karar verilmektedir.

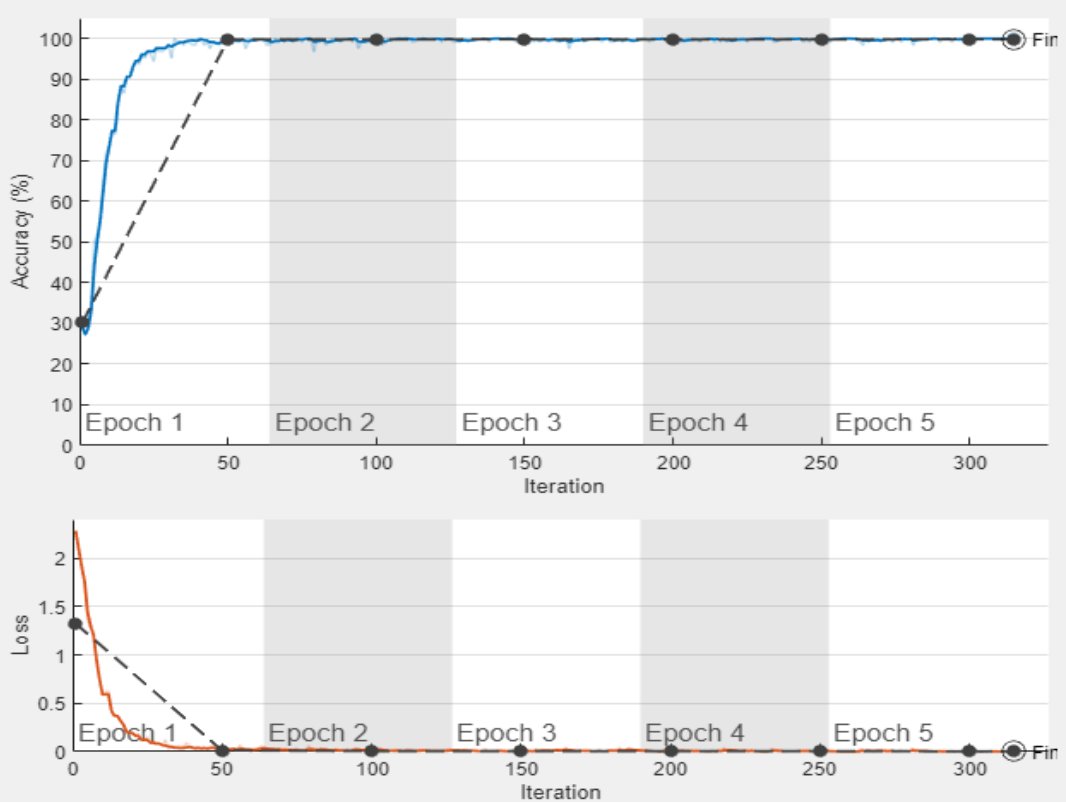
4. BULGULAR VE TARTIŞMA

4.1. Eğitilmiş Ağ Modelleri ile Veri Setlerinin Test Edilmesi

Bu tez çalışmasında eğitilmiş ağlarla veri setlerini test etmek için AlexNet ve MobileNet_v2 yapıları kullanılmıştır. Eğitim ve test aşamalarında kullanılan donanım şu şekildedir: Intel i7, 3.6 GHz, 32 GB DDR3 1600 MHz RAM, NVIDIA RTX 2080 GPU, Windows 10 Pro masaüstü bilgisayar kullanılmıştır. Tüm gerçeklemeler Matlab R2019b yazılımı ile yapılmıştır. Yine ağların eğitilmesi ve test edilmesi sürecinde kullanılan parametreler ise şu şekildedir: Solver Name: 'sgdm', / stochastic gradient descent with momentum (SGDM) optimizer. SGDM için ayarlanabilir Momentum: 0.9000, InitialLearnRate:1.0000e-04, MiniBatchSize: 64.

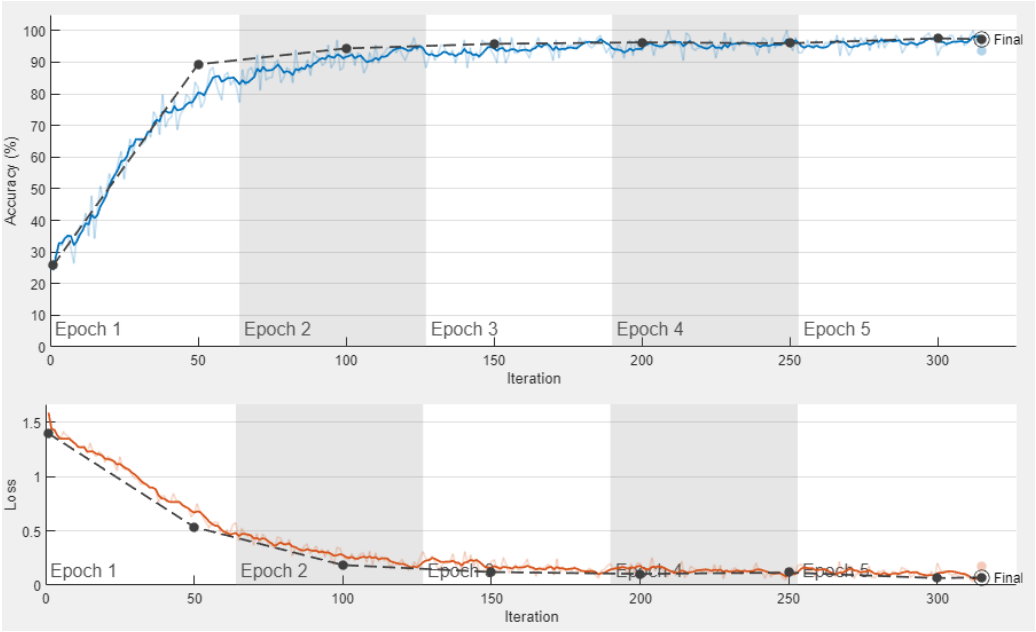
4.1.1. Tohum sınıflandırma veri setinin test edilmesi

Tohum sınıflandırma problemi için daha önce Şekil 3.3 ‘de gösterildiği üzere dört farklı tohum için dört farklı piksel bilgisine sahip toplamda 16 çeşit veri seti üretilmiştir. Veri setleri oluşturulurken datanın %70’i eğitim datası, %15’i doğrulama datası ve %15’i de test datası olarak hazırlanmıştır. İlk testler RGB veri seti (2100 eğitim verisi, 450 doğrulama verisi ve 450 test verisi) üzerinde yapılmış olup; bu veri seti için kullanılan parametreler yukarıda belirtilmiştir. Parametrelerin yukarıdaki gibi seçilip ve epoch(tur)=5 için ağın eğitilme işlemi Şekil 4.1‘deki gibidir ve bu eğitim işlemi toplamda 1 dakika 26 saniye sürmüştür.



Şekil 4.1. RGB tohum sınıflandırma veri setinin AlexNet ile eğitilmesi

Bir sonraki aşamada gri formattaki veri seti (2100 eğitim verisi, 450 doğrulama verisi ve 450 test verisi) aynı parametreler ile eğitilmiş olup; eğitim süreci Şekil 4.2'deki gibidir. İki şekil arasındaki en büyük fark RGB veri seti renk bilgisi içerdiği için doğruluk hızlı bir şekilde (yaklaşık 3. Epochtan sonra) %100'e yaklaşmıştır. Gri formattaki veri setinde ise ilk 5 epochta doğrulukta ciddi dalgalanmalar gözükmektedir. Bunun en büyük sebebi eğitimde kullanılan eğitilmiş AlexNet ağ yapısı ImageNet veri setindeki renkli görüntüler ile eğitilmiş olup; gri formattaki veri setinde renk bilgisinin eksik olmasıdır. Aynı şekilde Loss grafiğinde de dalgalanmalar göze çarpmaktadır.



Şekil 4.2. Gri format tohum sınıflandırma veri setinin AlexNet ile eğitilmesi

Derin öğrenme ile ağın eğitilmesi ve test edilmesinde veri setinin büyüklüğü çok önemli olup, teoride veri seti ne kadar büyük ise test doğruluğu o kadar yüksek olacaktır. Bunu daha iyi anlamak için veri seti ilk başta her bir tohum sınıfı için eğitim verisi = 2100, doğrulama verisi = 450 ve test verisi = 450 olarak seçilmiştir. Bir sonraki aşamada veri setleri her bir tohum sınıfı için eğitim verisi = 700, doğrulama verisi = 150 ve test verisi = 150 son aşamada ise her bir tohum sınıfı için eğitim verisi = 70, doğrulama verisi = 15 ve test verisi = 15 olarak seçilmiştir. Yine ağın yüksek test doğruluğunda çalışabilmesi için ağın yeterince uzun süre eğitilmesi gerekmektedir. Bu parametreye de epoch(tur) denilmektedir. Farklı epoch sayıları ile yapılan ağın test doğrulukları farklı sonuçlar verecek olup; RGB ve gri formattaki veri setlerini eğitirken epoch sayıları 50, 20 ve 5 olarak seçilmiştir. Çok daha zor problemlerde bu sayıların çok daha büyük olması gerekebilmektedir. Bu olay bir sonraki ikili ve kenar bilgisi veri setlerinde daha iyi anlaşılacaktır. RGB ve Gri formattaki veri setleri farklı epochlar ve farklı sayılarda eğitim verisi, doğrulama verisi ve test verisi için eğitilip test edildikten sonra Çizelge 4.1'deki sonuçlar elde edilmiştir. RGB veri seti sınıflandırma için kolay bir problem olup veri seti sayılarında on kattan fazla bir azalma olmasına rağmen aynı epoch sayılarında (epoch = 50 ve epoch = 20) test doğruluğu her zaman %100 olarak hesaplanmıştır. Ancak aynı durum gri formattaki veri seti için söz konusu değildir. Gri formattaki veri seti hem veri sayısından hem de epoch sayısından doğrudan

etkilenmektedir. Örneğin eğitim verisi = 2100, doğrulama verisi = 450 ve test verisi = 450 iken epoch =50'de test doğruluğu %99,54 iken epoch = 20 seçilirse test doğruluğu %98,97'ye düşmektedir. Bu da bize zor problemler için veri seti sayısının ve epoch sayısının en kadar önemli olduğunu göstermektedir.

Çizelge 4.1. RGB ve gri formattaki veri setinin farklı veri sayıları ve epoch'lara göre test sonucu karşılaştırmaları

70% Eğitim, 15% Doğrulama, 15%Test			
		RGB	**Gri Format**
Veri Sayısı	Epochs	Test Doğruluğu	Test Doğruluğu
E = 2100x4	50	%100	%99,54
D = 450x4	20	%100	%98,97
T = 450x4	5	%100	%97,00
E = 700x4	50	%100	%98,17
D = 150x4	20	%100	%96,67
T = 150x4	5	%99,83	%92,83
E = 70x4	50	%100	%93,33
D = 15x4	20	%100	%73,33
T = 15x4	5	%90	%46,67

Benzer işlemler ikili görüntü ve kenar bilgisi veri setlerine de uygulandıktan sonra Çizelge 4.2'deki sonuçlar elde edilmiştir. Çizelge 4.1'de doğruluk sonuçları çok yüksek olduğu için epoch sayıları 5,20 ve 50 olarak seçilmiştir. Çizelge 4.2'de ise epoch sayıları, 5, 20 ve N olarak seçilmiştir. N sayısının hesaplanması için erken durdurma yöntemi kullanılmıştır. Buna göre eğitim parametresi olan max epoch çok büyük bir sayı seçilip (bu çalışmada 1000) doğrulama doğruluk oranı (validation accuracy) 6 kere son maksimum doğruluk oranından küçük çıkarsa eğitim otomatik olarak durdurulacaktır. Kullanılan bu fonksiyon doğrultusunda N değeri farklı değerlerde hesaplanmıştır. Çizelge 4.2'ye bakıldığında veri seti azaldıkça N değerinin arttığı (99, 212, 975) gözlenmektedir.

Çizelge 4.2. İkili görüntü ve kenar bilgisine sahip veri setlerinin sınıflandırma sonuçları

70% Eğitim, 15% Doğrulama, 15%Test				
	İkili Görüntü		**Kenar Bilgisi**	
Veri Sayısı	Epochs	Test Doğruluğu	Epochs	Test Doğruluğu
E = 2100x4	103	%78,93	99	%93,09
D = 450x4	50	%74,59	50	%90,81

Çizelge 4.2'nin devamı

T = 450x4	20	%67,83	20	%72,62
E = 700x4	222	%78,17	212	%90
D = 150x4	50	%67	50	%65
T = 150x4	20	%55,83	20	%59,67
E = 70x4	490	%66,67	975	%86,67
D = 15x4	50	%40	50	%25
T = 15x4	20	%25	20	%25

Çizelge 4.2'ye göre kenar bilgisi veri formatı, ikili görüntü veri formatına göre çok daha iyi sonuçlar vermiştir. Kenar bilgisinde saklanan veri RGB veya gri formattaki görüntüye göre çok daha küçük olmasına rağmen sınıflandırma sonuçları oldukça yüksek çıkmıştır. Çizelge 4.2'den hareketle kenar bilgisinin tohum sınıflandırma probleminde çok önemli bir bilgi olduğu söylenebilir.

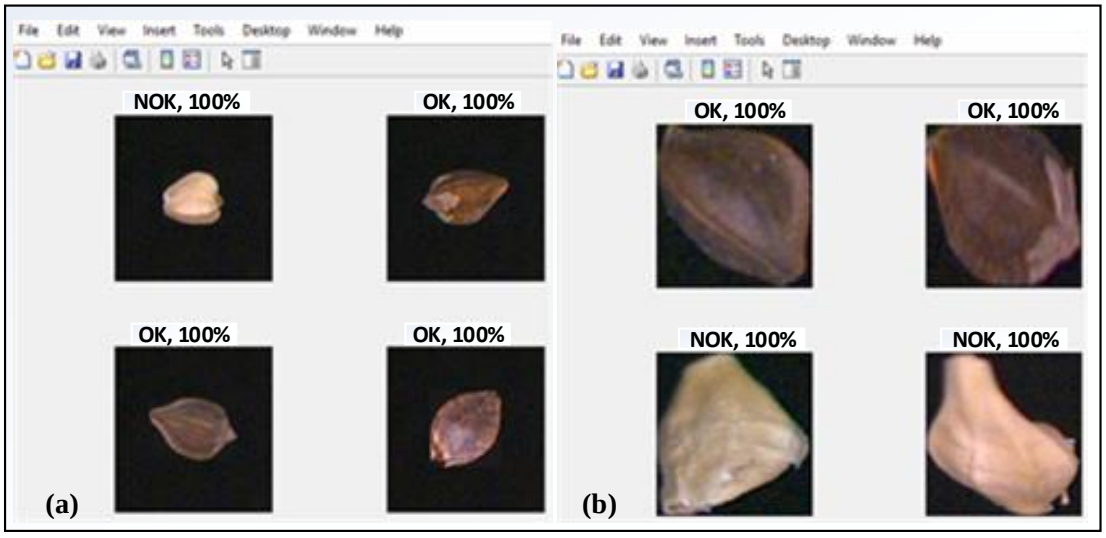
4.1.2. Tohum ayıklama veri setinin test edilmesi

Tohum sınıflandırma veri setlerinin eğitilmiş ağ yapıları ile eğitilip test edilmesi detaylı bir şekilde incelenmiş olup; Çizelge 4.1 ve Çizelge 4.2'den çıkarılacak en kapsamlı sonuç renk bilgisinin ne kadar önemli olduğudur. Tohum sınıflandırma problemi tohum ayıklama problemine göre daha kolay bir problemdir. Bu yüzden tohum ayıklama probleminin eğitilmiş ağlar ile eğitilip test edilmesi sadece RGB veri seti ile gerçeklenecektir. Bu tez çalışmasında tohum ayıklama veri seti için iki farklı boyutlandırma (Doğrudan boyutlandırma (Type2 Veri Seti) ve Ölçeklendirerek boyutlandırma (Type2 Veri Seti)) yöntemi ile iki farklı veri seti oluşturulmuştur. Bu iki veri setinin AlexNet ve MobilNet-V2 ağ yapıları ile eğitilmesinin ve test edilmesinin sonuçları Çizelge 4.3'deki gibidir.

Çizelge 4.3. Type1.A ve Type2.A veri setlerinin AlexNet ve MobileNet-V2 ile test edilmesi (E=Eğitim, D=Doğrulama, T=Test)

		Type1.A Veri Seti			**Type2.A Veri Seti**		
	CNN Yapısı	Epoch	Eğitim Süresi	Test Doğruluğu	Epoch	Eğitim Süresi	Test Doğruluğu
E = 1200x2 D = 300x2 T = 300x2	**Alexnet**	100	7dk 19sn	%98,87	100	6dk 36sn	%98,33
	MobileNet-V2	100	72dk 22sn	%99.00	100	69dk 51sn	%98,50
E = 400x2 D = 100x2 T = 100x2	**Alexnet**	200	3dk 45sn	%97,63	200	3dk 15sn	%97,50
	MobileNet-V2	200	36dk 57sn	%98,50	200	28dk 32sn	%98,17

Çizelge 4.3'e göre MobileNet-V2 ağ yapısının test sonuçları AlexNet'den daha yüksek çıkmıştır. Yine Type1 veri setinin test sonuçları Type2 veri setine göre daha yüksek sonuçlar içermektedir. Type1 ve Type2 data setleri özel ağ modellerinin test edilmesi bölümünde daha detaylı bir şekilde ele alınacaktır. Çizelge 4.3'e göre AlexNet ve Mobilenet-V2 ağ yapıları için Type1 veri setini kullanmanın daha yüksek sonuçlar vereceği söylenebilir. Çizelge 4.3'de ilk satırdaki AlexNete ait eğitim sonuçları görselleştirmek için Şekil 4.3a ve Şekil 4.3b de Type1 ve Type2 veri setlerine ait test sonuçları gösterilmektedir. Şekillerden de görüldüğü gibi doğrudan ölçekleme nesnenin morfolojisini fazlasıyla bozmakta bu da test sonuçlarını etkilemektedir. Nitekim Type1.A veri setinin test sonucu %98,87 iken Type2.A veri setinin test sonucu %98,33 olarak bulunmuştur.



Şekil 4.3. Test sonuçlarının görselleştirilmesi **a)**Type1.A veri seti (Eğitim=1200x2, Doğrulama=300x2, Test=300x2) AlexNet ile 100 epoch eğitildikten sonraki test sonucu; **b)** Type2.A veri seti (Eğitim=1200x2, Doğrulama=300x2,Test=300x2) AlexNet ile 100 epoch eğitildikten sonraki test sonucu

4.2. Uygulamaya Özel Modelin Veri Setleri Üzerinde Test Edilmesi

Bu tez çalışmasında tohum sınıflandırma ve tohum ayıklama işlemlerini düşük işlem yüküne sahip özel ağ yapıları ile gerçekleyebilmek için Şekil 3.12'deki gibi uygulamaya özel bir ağ yapısı önerilmiştir. Nihai hedef tohum ayıklama ve sınıflandırma problemi test doğruluğundan %1 sapma ile daha küçük ağlarla gerçeklenebilir mi bunun araştırılması ve bu optimum ağın elde edilmesidir. Önerilen ağın bu işlemlere uygun olup olmadığını test etmek için öncelikle önerilen ağın test doğruluğu sonuçlarına bakılması gerekmektedir. Çizelge 4.1 ve Çizelge 4.2'ye göre tohum sınıflandırma (%100) ve tohum ayıklama (%99) işlemleri eğitilmiş ağ yapıları ile yüksek doğruluklarda gerçeklemiştir. Önerilen ağ yapısının tohum sınıflandırma ve tohum ayıklama işlemlerine uygun olup olmadığına karar vermek için 227x227 RGB tohum sınıflandırma veri seti ve Type1.A tohum ayıklama veri setleri ile ayrı ayrı test edilmiştir. Yapılan test sonuçları Çizelge 4.4'deki gibidir. Çizelge 4.4'e göre Type1.A veri seti için en yüksek doğruluk oranı 99% olarak hesaplanmıştır. Bu doğruluk oranı Çizelge 4.3'deki MobileNet-V2 doğruluk oranı ile aynıdır. Yine tohum sınıflandırma

veri seti için doğruluk %100 hesaplanmıştır. Bu sonuçlara istinaden önerilen Custom227 modelinin, tohum ayıklama ve tohum sınıflandırma işlemlerinde AlexNet ve MobileNet-V2 ağlarına alternatif olarak kullanılabileceği söylenebilir. Eğer ki test sonuçları beklenen değerden (%98)'den küçük çıksaydı bu sefer başlangıç için daha derin (3 evrişim katmanı) ya da daha fazla filtre sayısına sahip olan bir ağ yapısının önerilmesi gerekebilirdi. Önerilen afğ bu iki problem için de istenilen sonuçları vermiş olup; daha derin ağlar ile testlerin yapılmasına ihtiyaç duyulmamıştır. Ancak daha zor problemler için daha derin ağların gerekebileceğini göz ardı etmemek gerekir.

Çizelge 4.4. Type1.A ve tohum sınıflandırma veri setlerinin özel ağ modeli ile test edilmesi

		Type1.A Veri Seti			**Tohum Sınıflandırma RGB Veri Seti**		
	CNN Yapısı	Epoch	Eğitim Süresi	Test Doğruluğu	Epoch	Eğitim Süresi	Test Doğruluğu
E = 1200x2 D = 300x2 T = 300x2	**Custom227**	100	17dk 19sn	%99,00	100	12dk 40sn	%100
	Custom227	50	8dk 34sn	%98,50	50	6dk 20sn	%100
E = 400x2 D = 100x2 T = 100x2	**Custom227**	300	5dk 42 sn	%98,50	300	11dk 13sn	%100
	Custom227	150	2dk 54sn	%97,00	150	3dk 24sn	%100

4.3. Ölçekle ve Budama Yönteminin Uygulanması

Bu tez çalışmasında tohum sınıflandırma ve ayıklama işlemlerini yüksek hızlarda yapabilmek için Şekil 3.12'deki gibi düşük işlem yüküne sahip özel ağ yapısı önerilmiştir. Önerilen özel ağ modeli tohum sınıflandırma ve tohum ayıklama problemlerinde kullanılacak olup; Çizelge 4.1'e göre tohum sınıflandırma problemi en yüksek %100 doğrulukta ve Çizelge 4.3'e göre tohum ayıklama problemi en yüksek %99 doğrulukta gerçeklenmektedir. Bu yüzden önerilen özel ağ yapısının hızlı olması kadar Çizelge 4.1 ve Çizelge 4.3'deki doğruluk değerlerinden en fazla %1 sapması beklenmektedir.

4.3.1. Tohum sınıflandırma problemi üzerinde uygulanması

Bu tez çalışmasında tohum sınıflandırma problemi için dört farklı veri seti oluşturulmuştur. En yüksek doğruluk RGB veri seti üzerinde elde edilmiştir. RGB veri setinin Custom227 ile test edilmesinde test doğruluğu %100 olarak hesaplanmıştır. RGB veri seti kullanılarak önerilen Custom227 modeli üzerinde ölçekle ve budama yöntemi uygulandığında ölçekle aşamasında kullanılacak data boyutları sırasıyla 227, 157, 77 ve 37 olacaktır. Her bir data boyutunda test süresini sınırlamak için erken durdurma yöntemi kullanılmıştır. Bu yüzden her bir data boyutu için epoch sayıları farklı olacak şekilde hesaplanmıştır. Tüm sonuçlar Çizelge 4.5'de kaydedilmiştir. Çizelge 4.5'e bakacak olursak veri boyutu 37x37 dolayısıyla kullanılan ağ yapısı

Custom37 olmasına rağmen test doğruluğu %99,83 olarak hesaplanmıştır. Bu da maksimum %1 sapma kriterini sağlamaktadır.

Çizelge 4.5. Tohum sınıflandırma probleminin (RGB veri seti için) data boyutunu küçült modülü ile test edilmesi

CNN Yapısı	Veri Seti	Epoch	Eğitim Süresi	Test Doğruluğu	FLOPs
Custom227	RGB	13	1dk 40sn	%100	46.072.682
Custom157	RGB	13	1dk 30sn	%100	21.482.794
Custom77	RGB	36	3dk 17sn	%99,92	4.486.954
Custom37	RGB	45	4dk 24sn	%99,83	937.514

Ölçekle ve budama yöntemi RGB tohum sınıflandırma verisi ile test edildiğinde ilk turdaki çıkış Custom37 olarak hesaplanmıştır. İkinci turda Custom37 yapısına ait filtre sayıları sürekli yarıya indirilecek olup; filtre sayılarının sürekli yarıya indirilmesindeki test sonuçları Çizelge 4.6'daki gibidir. Bu sonuçlara göre ikinci turda filtre sayıları sürekli yarıya indirildiğinde algoritma sınır koşul olan (conv1:2) (conv2:3)'e kadar çalışmıştır. Ancak maksimum %1'lik sapma Custom37 (conv1:4) (conv2:6) parametrelerinde hesaplanmıştır. Bu sonuçlara göre RGB veri seti ile tohum sınıflandırma problemine ölçekle ve budama yöntemi uygulandığında en optimum model Custom37 (conv1:4) (conv2:6) olarak hesaplanmıştır.

Çizelge 4.6. Tohum sınıflandırma probleminin(RGB veri seti için) filtre sayılarını yarıya indir modülü ile test edilmesi

CNN Yapısı	Filtre Sayıları	Epoch	Eğitim Süresi	Test Doğruluğu	FLOPs
Custom37	(conv1:64) (conv2:96)	45	4dk 24sn	%99,83	937.514
Custom37	(conv1:32) (conv2:48)	25	2dk 3sn	%99,83	414.122
Custom37	(conv1:16) (conv2:24)	40	3dk 20sn	%99,83	193.898
Custom37	(conv1:8) (conv2:12)	59	4dk 44sn	%99,75	94.154
Custom37	(conv1:4) (conv2:6)	115	9dk 12sn	%99,33	46.874
Custom37	(conv1:2) (conv2:3)	98	8dk 11sn	%98	23.882

4.3.2. Tohum ayıklama problemi üzerinde uygulanması

Tohum ayıklama probleminde maksimum test doğruluğu MobileNet-V2 ve Custom227 ile %99 olarak hesaplanmıştır. Bu çalışmada %1 doğruluk sapması ile özel modeller ölçeklenecek ve budanacak olup; kabul edilebilir test doğruluğu %98 olarak belirlenmiştir. Bu eşik değer eşliğinde Şekil 3.12'deki yöntemin ilk aşaması olan Data Boyutunu Küçült modülü çalıştırıldığında test sonuçları Çizelge 4.7'deki gibidir. Çizelge 4.7'ye göre işlem dört turda tamamlanmış olup; hedef CNN yapısı Custom77 olarak hesaplanmıştır. Nitekim Custom77'nin test doğruluğu 98.67% olarak hesaplanmıştır. Bu da 0.33% bir sapma demektir ve önerilen yöntem için kabul edilebilir bir sapmadır.

Çizelge 4.7. Tohum ayıklama probleminin data boyutunu küçült modülü ile test edilmesi

CNN Yapısı	Veri Seti	Epoch	Eğitim Süresi	Test Doğruluğu	FLOPs
Custom227	Type1.A	100	17dk 19sn	%99,00	46.072.682
Custom157	Type1.B	100	11dk 22sn	%99,00	21.482.794
Custom77	Type1.C	150	10dk 54sn	%98,67	4.486.954
Custom37	Type1.D	200	12dk 1sn	%97,83	937.514

Data boyutu küçültülüp; Custom77 ağ yapısına karar verildikten sonra bir sonraki aşamada "Filtre Sayılarını Yarıya İndir" modülü çalıştırılmıştır. Bu modülün Type1.C veri seti üzerinde çalıştırılması sonucu elde edilen sonuçlar Çizelge 4.8'deki gibidir.

Çizelge 4.8. Tohum ayıklama probleminin filtre sayılarını yarıya indir modülü ile test edilmesi

CNN Yapısı	Filtre Sayıları	Epoch	Eğitim Süresi	Test Doğruluğu	FLOPs
Custom77	(conv1:64) (conv2:96)	150	10dk 54sn	98,67	4.486.954
Custom77	(conv1:32) (conv2:48)	200	12dk 57sn	98,50	1.898.538
Custom77	(conv1:16) (conv2:24)	250	16dk 33sn	98,16	863.530
Custom77	(conv1:8) (conv2:12)	300	19dk 52sn	97,83	410.826

"Filtre Sayılarını Yarıya İndir" modülünün çalışması yine dört turda tamamlanmış olup maksimum 1% hassasiyet kaybı için filtre sayılarının sonuçları:

conv1 katmanında 16 filtre conv2 katmanında 24 filtre olarak hesaplanmıştır. Tüm bu veriler göz önünde bulundurulduğunda tohum ayıklama problemi için ilk önerilen Custom227 (conv1:64) (conv2:96) ağ yapısında FLOPs 46.072.682 iken "Ölçekle ve Budama" metodunun uygulanması sonucu elde edilen Custom77 (conv1:16) (conv2:24) ağ yapısında FLOPs 863.530 olarak hesaplanmıştır. Bu da FLOP değerini yani yapılması gereken toplam kayan nokta işlem sayısını 53,2 kat azalması demektir.

4.4. Genetik Algoritma ile Özel Ağ Modelinin Optimize Edilmesi

Ölçekle ve Budama Yönteminin test sonuçları Çizelge 4.5, Çizelge 4.6, Çizelge 4.7 ve Çizelge 4.8'deki gibi olup tohum sınıflandırma problemi için yöntem sınır koşullarına kadar çalıştırılmıştır. Tohum ayıklama problemi ise tohum sınıflandırmaya göre daha zor bir problem olduğu için Ölçekle ve Budama Yöntemine karşı daha hassas sonuçlar vermiştir (boyut ve filtre sayısı azaldıkça doğrulukta değişmiştir). Ölçekle ve Budama Yöntemi ile önerilen özel ağ modeli optimize edilmeye çalışılmıştır. Optimizasyon işlemi önerilen ölçekle ve budama Yöntemi ile yapılacağı gibi literatürdeki algoritmalar ile de yapılabilmektedir. Özellikle son yıllarda genetik algoritma ağların hiper parametrelerinin optimize edilmesine kendine geniş bir yer bulmuştur. Tohum sınıflandırma probleminde boyutta ve filtre sayısındaki küçülmelere rağmen test doğruluğu çok değişmemiştir. Bu yüzden genetik algoritma ile optimizasyon işlemleri için tohum ayıklama veri setleri kullanılacaktır. Bu çalışmada tohum ayıklama problemi için 4 farklı boyutta veri seti (Type1.A (227x227), Type1.B (157x157), Type1.C (77x77), Type1.D (37x37)) ve bu 4 farklı veri setinin gerçeklenmesi için 4 farklı özel ağ modeli (Custom227, Custom157, Custom77 ve Custom37) önerilmiştir. Doğruluktan en fazla %1 ödün vererek optimum ağ yapısı Ölçekle ve Budama yöntemi ile test edilmiş ve sonuç Custom77 (conv1:16) (conv2:24) olarak hesaplanmıştır. Genetik algoritma yüksek başarımlı bir algoritma olup; Custom ağ için en uygun parametrelerin buldurulması işlemi genetik algoritma ile test edilmiştir. Genetik algoritma doğası gereği bir fonksiyonu optimize (minimize ya da maksimize) etmeye çalışmaktadır. Bu yüzden öncelikle optimize edilmeye çalışılan fonksiyonu çok iyi tanımlamak gerekmektedir. Bu çalışmada optimize edilmeye çalışılan fonksiyon, özel ağların kayan nokta işlem sayısı (FLOPs) olup; FLOPs için tüm hesaplamalar Çizelge 3.2 ve Çizelge 3.3'de detaylı bir şekilde verilmiştir. Çizelge 3.2 ve Çizelge 3.3'de tüm katmalardaki FLOPs değerleri tek tek gösterilmiş olup; en yüksek işlem yükü evrişim katmanlarındadır. Evrişim katmanlarındaki FLOPs hesabı denklem 3.3'deki gibi açıklanmıştır. Bu denklemde C_{in}, evrişim katmanının giriş kanal derinliği, C_{out} evrişim katmanının çıkış kanal derinliği, *Sc* evrişim katmanındaki atlama miktarı, *K* filtre boyutu, H_{out} çıkış data yüksekliği ve W_{out} çıkış data genişliği olarak tanımlanmıştır. C_{in} ilk evrişim katmanı için veri setindeki datanın derinliğini göstermekte olup; RGB bir veri setinin derinliği üçtür. C_{out} evrişim katmanındaki filtre sayısıyla aynı değeri göstermektedir. Yani evrişim katmanındaki filtre sayısı N1 ise C_{out}=N1 'dir. Veri seti *NxN* boyutunda datalardan oluşmak üzere $H_{out} = W_{out}$ denklem 3.1'deki gibi tanımlanmıştır. Denklem 3.3 detaylı bir şekilde incelendiğinde evrişim katmanındaki *FLOPs K, Cout, Sc* ile doğrudan bağlantılıdır. H_{in} ve C_{in} bir önceki katmanlardan gelen değerlerden hesaplandığı için bu değerlere dolaylı olarak bağlantılıdır. Tüm bu bağlantılardan hareketle evrişim katamındaki FLOPs değeri optimize edilmek istenirse *K, Cout ve Sc* değerlerinin genetik algoritmada parametre olarak tanımlanabilir.

Havuzlama katmanı için FLOPs değeri analiz edilirse; havuzlama katmanındaki FLOPs değeri Denklem 3.4'e göre hesaplanmaktadır. $H_{in} = W_{in}$ olup; havuzlama katmanın giriş data boyutu yani bir önceki evrişim katmanının çıkış data boyutudur. C_{in} ise havuzlama katmanındaki kanal derinliğini yani bir önceki evrişim katmanının çıkış kanal derinliğini göstermektedir. Bu denkleme göre havuzlama katmanından genetik algoritmaya seçilecek bir parametre bulunmamaktadır.

Son olarak tamamen bağlı katmanlardaki işlem yükünü inceleyecek olursak; çizelge 3.2'ye göre tamamen bağlı katmanlardaki en büyük işlem yükü TBK1'dedir. Diğer TBK2 ve TBK3'teki FLOPs değerleri ihmal edilebilecek seviyededir. TB1'deki FLOPs hesabı Denklem 3.5'deki gibi ifade edilip çarpım sonucunun büyük bir kısmını (custom model için) aktivasyon sayısı oluşturmaktadır. Aktivasyon sayısı da bir önceki katmanın çıkış data boyutu olan Hout_Havuzlama2=Wout_Havuzlama2 ve derinliği olan Cout_TBK1'e göre değişmekte olup; tüm bu parametreler de dolaylı olarak bir önceki evrişim katmanındaki parametrelere bağlıdır. Bu sonuçlara göre TBK katmamalarında parametre olarak seçilebilecek tek değer Cout parametresidir. Optimizasyon işlemi FLOPs değeri üzerinde yapılacak olup; Cout değerinin FLOPs'a etkisi çok küçük olduğu için özel modellerin genetik algoritma ile optimize edilmesinde Cout-TBK parametre olarak seçilmemiştir.

Yukarıda detaylı bir şekilde anlatıldığı üzere bu çalışmada özel ağlardaki FLOPs değerinin optimize edilmesi için veri seti data boyutu HvsxWvs (Hws=Wvs), K, Cout ve Sc değerleri parametre olarak seçilmiştir. Bu çalışmada dört farklı boyutta veri seti ve dört farklı özel ağ modeli ile gerçeklemeler yapılmış olup; veri seti data boyutunu (HvsxWvs) dikkate alarak K, Cout ve Sc değerlerinin dikkatli bir şekilde seçilmesi gerekecektir. Nitekim Çizelge 3.1'e bakacak olursak Custom37 en küçük ağ yapısı olup; bu ağ yapısında Conv1 katmanındaki filtre boyutu 5x5 atlama = 3, Havuzlama_1 katmanında filtre boyutu 2x2 ve atlama = 2, yapısında Conv2 katmanındaki filtre boyutu 3x3 atlama =2 ve , Havuzlama_2 katmanında filtre boyutu 2x2 ve atlama = 2 olarak seçilmiştir. Bu değerlerden herhangi biri bir üst basamaktan seçilirse (örneğin Conv1 deki filtre boyutu 7x7 ya da Havuzlama_1 katmanında filtre boyutu 3x3) ağ derleme hatası verip çalışmayacaktır. Tüm bu kriterler göz önünde bulundurulup özel ağ yapısının genetik algoritma ile test edilmesi işlemi sadece custom77 üzerinde gerçeklenmiştir. Nitekim Ölçekle ve Budama yöntemine göre doğruluktan maksimum %1 sapma kriteri için en iyi değer custom77 modeli için hesaplanmıştır. Giriş veri boyutunda herhangi bir değişiklik olmayacağından dolayı Hvs parametresi parametre kümesinden çıkartılıp K, Cout ve Sc parametrelerinin alabileceği değerler aşağıdaki gibi tanımlanmıştır:

İlk evrişim katmanındaki filtre sayısı Cout1 ve ikinci evrişim katmanındaki filtre sayısı Cout2 olmak üzere Cout1 ve Cout2 nin alacağı değerler:

$$C_{out1} = x_1, \qquad x_1 \in [8,64]$$

$$C_{out2} = x_1, \qquad x_2 \in [8,96]$$

İlk evrişim katmanındaki filtre boyutu K1 ve ikinci evrişim katmanındaki filtre boyutu K2 olmak üzere K1 ve K2'nin alacağı değerler:

$$K_1 = \begin{cases} 3\,, & x_3 = 1 \\ 5\,, & x_3 = 2\,, \\ 7\,, & x_3 = 3 \end{cases} \qquad x_3 \in [1,3]$$

$$K_2 = \begin{cases} 3,, & x_4 = 1 \\ 5\,, & x_4 = 2 \end{cases} \qquad x_4 \in [1,2]$$

İlk evrişim katmanındaki atlama sayısı Sc1 ve ikinci evrişim katmanındaki atlama sayısı Sc2 olmak üzere Sc1 ve Sc2'nin alacağı değerler:

$$S_{c1} = \begin{cases} 2, & x_5 = 1 \\ 3, & x_5 = 2 \end{cases} \qquad x_5 \in [1,2]$$

$$S_{c2} = \begin{cases} 2, & x_6 = 1 \\ 3, & x_6 = 2 \end{cases} \qquad x_6 \in [1,2]$$

Optimize edilmeye çalışılan fonksiyon FLOPs hesabı olup genetik algoritmadan beklenen maksimum %1 sapma değeri için minimum FLOPs değerlerine sahip parametrelerin hesaplanmasıdır. Maksimum doğruluk %99 olup; doğruluk %98 altında ise işlem başarısız olarak kabul edilmektedir. Çıkış fonksiyonu bu kriteri sağlaması için çıkış fonksiyonu y aşağıdaki gibi tanımlanmıştır. Bu sayede hassasiyet %98'in altında olunca y değeri olması gerekenden 100 kat daha büyük hesaplanacaktır. Bu da algoritmanın %98 ve üzeri değerleri sağlayan parametreleri bulmasını sağlayacaktır.

```
for j ← 1 to Number_of_Iteration do
   if  Accuracy < 98 do
      Accuracy ← 0,01
   else
      Accuracy ← 1
   end if
    y ← Total_FLOPs / Accuracy
end for
```

Yukarıda parametreler ve y çıkış fonksiyonu detaylı bir şekilde anlatılmış olup; Custom77 ağ yapısının genetik algoritma ile optimizasyon sonuçları Çizelge 4.9'daki gibi gösterilmiştir. Genetik algoritma üç kere koşturulmuş olup her bir koşturma yaklaşık 6 saat sürmüştür. Tüm koşturma sonuçları Çizelge 4.9'da her bir parametre için detaylı bir şekilde gösterilmiştir. Koşturma süresinin çok uzun olmaması için veri seti olabildiğince küçük tutulmaya çalışmıştır. Bu koşturma sırasında kullanılan veri setlerinde eğitim verisi 700 adet, doğrulama verisi 150 adet ve test verisi 150 adet seçilmiştir. Çizelge 4.9'daki ilk satırdaki test doğruluğu eğitim=700, doğrulama=150 ve test=150 veri setlerine ait doğruluk sonuçlarıdır. Yine genetik ile hesaplama sırasında epoch 500 ile sınırlandırılmış olup; ilk satırdaki değerler hesaplanırken epoch 1000 yapılmıştır. Elde edilen verilen ne kadar kararlı ve olduğunu test edebilmek Çizelge 4.8'deki veri seti kullanılmıştır. Bu yeni veri setine ait test doğruluğu Çizelge 4.9'da ikinci satırdaki test doğruluğudur. Çizelge 4.8'deki veri seti daha büyük bir veri seti olmasından dolayı test doğruluğunda artış olması beklenen bir durumdur. Bu da bize

genetik ile hesaplanan parametrelerin test doğruluğunda ne kadar başarılı olduğunu göstermektedir.

Çizelge 4.9. Custom77 modelinin genetik algoritma ile optimize edilmesi

	1.Koşturma Sonucu	**2.Koşturma Sonucu**	**3.Koşturma Sonucu**
Test Doğruluğu (E=700,D=150,T=150)	%98	%98,33	%98,33
Test Doğruluğu (Çizelge 4.8'deki veri seti)	%98,16	%98,50	%98,33
N1	11	9	13
N2	68	46	59
K1	5	5	5
K2	3	3	3
Sc1	3	2	3
Sc2	2	3	2

Parametre değerleri Çizelge 4.9'daki gibi hesaplandıktan sonra bu değerlere ait ağ yapılarının FLOPs değerleri de detaylı bir şekilde Çizelge 4.10'da gösterilmiştir. Çizelge 4.10'daki sonuçlar ile Çizelge 4.8'deki sonuçları karşılaştıracak olursak; sonuçların birbirine yakın değerler olduğundan söz edilebiliriz. Nitekim Çizelge 4.8'deki en optimum ağ yapısındaki FLOPs değeri 863.530'dir. Çizelge 4.10'da hesaplanan en küçük FLOPs değeri 715.453 ve bu değere karşılık test doğruluğu = %98,16'dır. Çizelge 4.10'daki FLOPs değeri Çizelge 4.8'deki değerden daha küçüktür. Yine Çizelge 4.8'deki test doğruluğu ile Çizelge 4.9'daki test doğruluğu aynı olup; buradan hareketle aynı test doğruluğunun, işlem yükü daha küçük bir ağ yapısı ile sağlandığı söylenebilir.

Çizelge 4.10. Genetik algoritma ile optimize edilen ağların FLOPs değerleri

Katmanlar	**1.Koşturma Sonucu**	**2.Koşturma Sonucu**	**3.Koşturma Sonucu**
Conv_1	522.500	936.396	617.500
Havuzlama_1	6.875	12.321	8.125
Conv_2	170.000	94.300	174.050
Havuzlama_2	1.700	1.150	1475
TBK_1	13.104	8.880	11376
TBK_2	1.176	1.176	1.176
TBK_3	98	98	98
Softmax	0	0	0
TOPLAM (FLOPs)	715.453	1.054.321	813.800

4.4.1. Farklı veri setleri için genetik algoritma ile özel ağların optimize edilmesi

But tez çalışmasında tohum ayıklama veri setleri önerilen ağ yapıları ile test edilmiş devamında ise ölçekle ve budama yöntemi ve genetik algoritma ile önerilen ağ yapıları optimize edilmiştir. Ölçekle ve budama yöntemi ile önerilen özel ağın boyutunu ve filtre sayısını küçülterek hassasiyetten ödün vermeden en optimum ağ yapısı custom77 (conv1: 16, conv2: 24, K1:5, Sc1:3, K2:3, Sc2:2) olarak hesaplamıştır. Önerilen özel ağın genetik algoritması ile optimize edilmesi sonucu FLOPs değerine göre en optimum ağ yapısı custom77 (conv1: 11, conv2: 68, K1:5, Sc1:3, K2:3, Sc2:2) olarak hesaplanmıştır.

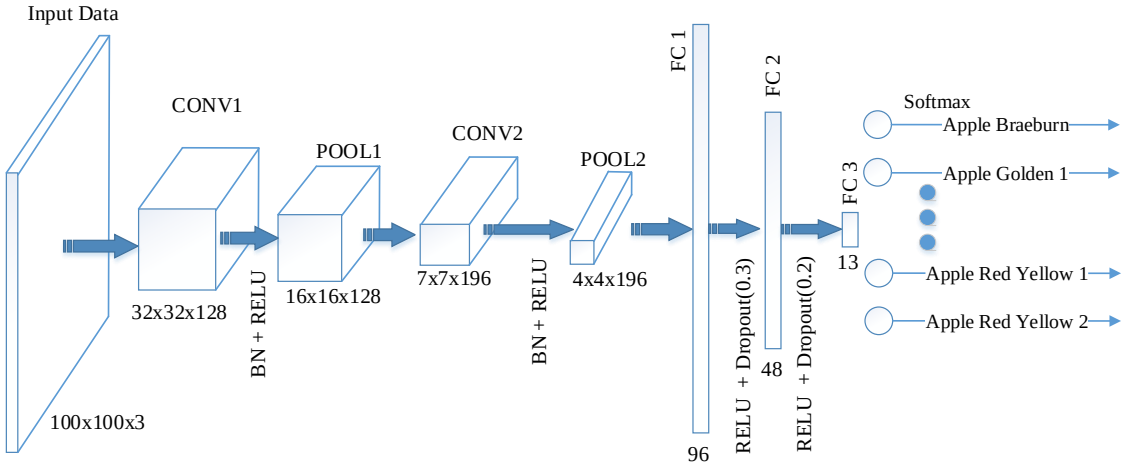
Önerilen özel ağ yapısının literatürdeki farklı veri setleri ile test edilmesi ve devamında genetik algoritma ile optimize edilmesi için son yıllarda endüstriyel tarım alanında popüler olan fruit360 veri seti uygulama alanı olarak seçilmiştir. Fruit 360 (Mureşan ve Oltean 2018) veri setinde 131 meyve sınıfına ait toplamda 90483 görüntü bulunmaktadır. Tüm görüntüler 100x100 formatında kaydedilmiştir. Tüm bu veri seti kullanıldığında ağın eğitilmesi, test edilmesi daha da önemlisi genetik algoritma ile optimize edilmesi haftalar süreceği için problemi daha da özelleştirip; bu veri setindeki sadece elmaların sınıflandırılması ve devamında optimize edilmesi amaçlanmıştır. Fruit360 veri setinde toplam 13 farklı elma türüne ait veri seti bulunmaktadır. Tüm bu değerler Çizelge 4.11'deki gibidir.

Çizelge 4.11. Fruit360 veri setindeki elma sınıflarına ait değerler (Mureşan ve Oltean 2018)

Elma Sınıfı	Eğitim verisi	Test verisi
Apple Braeburn	492	164
Apple Golden 1	492	164
Apple Golden 2	492	164
Apple Golden 3	481	161
Apple Granny Smith	492	164
Apple Red 1	492	164
Apple Red 2	492	164
Apple Red 3	429	144
Apple Red Delicious	490	166
Apple Red Yellow 1	492	164
Apple Red Yellow 2	672	219

Çizelge 4.11'deki veri setinin özel ağ yapısı ile test edilebilmesi için Şekil 3.12'deki ağ yapısı şekil 4.4'deki gibi revize edilmiştir. Şekil3.12'de önerilen ağ yapısının giriş data boyutu 227x227 dir. Bu veri setinin data boyutu 100x100 olup; ağın girişi buna göre değiştirilmiştir. Yine filtre sayıları Şekil 3.12'ye göre iki katına çıkarılmıştır. Tamamen bağlantılı katmanlardaki nöron sayısı da iki katına çıkarılarak TBK_1=96 ve TBK_2=48 olarak güncellenmiştir. Çıkışta 13 sınıf olduğu için çıkış

katmanındaki nöron sayısı da 13 olarak güncellenmiştir. Çizelge 4.11'deki veri seti Şekil 4.4'deki gibi önerilen özel ağ yapısı üzerinde test edilmiştir. Ağın doğruluk sonucu %99,62 olarak hesaplanmıştır. Bu yetirince yüksek bir doğruluk olduğu için Çizelge 4.11'deki veri seti literatürdeki ağ yapıları ile test edilmemiştir. Ağın eğitilmesi işlemi 500 epoch ile yapılmış olup; bu işlem toplamda 12dk 44sn sürmüştür. Bir sonraki aşamada doğruluktan maksimum %1 ödün verecek şekilde genetik algoritma Şekil 4.4'deki özel ağ üzerinde uygulanmıştır.



Şekil 4.4. Fruit360 veri setindeki elmaları sınıflandırmak için önerilmiş özel ağ (Şekil 3.12'deki yapının bu veri setine uygun hale getirilmesi)

Genetik algoritmanın uygulanması için seçilen parametreler ise aşağıdaki gibidir:

İlk evrişim katmanındaki filtre sayısı Cout1 ve ikinci evrişim katmanındaki filtre sayısı Cout2 olmak üzere Cout1 ve Cout2 nin alacağı değerler:

$$C_{out1} = x_1, \qquad x_1 \in [8,128]$$

$$C_{out2} = x_1, \qquad x_2 \in [8,196]$$

İlk evrişim katmanındaki filtre boyutu K1 ve ikinci evrişim katmanındaki filtre boyutu K2 olmak üzere K1 ve K2'nin alacağı değerler:

$$K_1 = \begin{cases} 3, & x_3 = 1 \\ 5, & x_3 = 2, \\ 7, & x_3 = 3 \end{cases} \qquad x_3 \in [1,3]$$

$$K_2 = \begin{cases} 3,, & x_4 = 1 \\ 5, & x_4 = 2 \\ 7, & x_4 = 3 \end{cases} \qquad x_4 \in [1,3]$$

İlk evrişim katmanındaki atlama sayısı Sc1 ve ikinci evrişim katmanındaki atlama sayısı Sc2 olmak üzere Sc1 ve Sc2'nin alacağı değerler:

$$S_{c1} = \begin{cases} 2, & x_5 = 1 \\ 3, & x_5 = 2 \end{cases} \qquad x_5 \in [1,2]$$

$$S_{c2} = \begin{cases} 2, & x_6 = 1 \\ 3, & x_6 = 2 \end{cases} \qquad x_6 \in [1,2]$$

Tüm parametreler yukarıdaki gibi belirlendikten sonra algoritmanın koşturması yaklaşık iki gün sürmüş olup; elde edilen optimize ağ yapısı şu şekildedir: custom100 (conv1: 59, conv2: 108, K1:5, Sc1:2, K2:5, Sc2:3) Bulunan bu optimum ağ yapısı veri seti ile eğitilip test edildiğinde test doğruluğu %98,82 olarak hesaplanmıştır. Ağın eğitimi 500 epoch için yapılmış olup bu işlem 15dk 25sn sürmüştür. Genetik algoritma ile hesaplama yapılırken epoch sayısı 500 olarak seçilmiştir ve genetik algoritmanın çalışması yaklaşık iki gün sürmüştür. Eğer ki epoch sayısı 1000 seçilmiş olsaydı işlem yaklaşık dört gün sürecekti. Bu sebeplerden dolayı ilk başta yeterince büyük bir değer olan 500 değeri epoch için seçilmiştir. Tüm bu hesaplamalar yapıldıktan sonra Şekil 4.4'deki ağ yapısı 1000 epoch ile tekrar eğitilmiş ve ağ doğruluğu %99,95 olarak hesaplanmıştır. Yine genetik algoritma ile bulunan ağ yapısı 1200 epoch eğitilerek test doğruluğu %99,91 olarak hesaplanmıştır. Tüm bu test sonuçları ve ağlara ait FLOPs ve parametre sayıları Çizelge 4.12'deki gibidir.

Çizelge 4.12. Farklı epoch değerleri için şekil 4.4'deki Custom100 ve Custom100 ağının genetik ile optimize edilmesi sonucu elde edilen ağa ait test doğrulukları

Ağ Yapısı	Epoch	Test Süresi	Test Doğruluğu	FLOPs	Parametere Sayısı
Custom100 (Şekil 4.4)	500	12dk 44sn	%99,62	21.816.200	4.327.800
Custom100 Genetik	500	15dk 25sn	%98,82	18.724.027	2.431.208
Custom100 (Şekil 4.4)	1000	12dk 44sn	%99,95	21.816.200	4.327.800
Custom100 Genetik	1200	37dk 17sn	%99,91	18.724.027	2.431.208

Çizelge 4.12 deki değerleri Duong vd. (2020)'nin çalışmasındaki sonuçlar ile karşılaştıracak olursak; karşılaştırma sonuçları Çizelge 4.13'deki gibidir. Çizelge 4.13'e göre en yüksek test doğruluğu Duong vd. (2020)'nin çalışmasındaki EfficientNet-B0 ağ yapısı elde edilen %99,97 lik sonuçtur. Önerilen ağ yapısının test doğruluğu bu sonuçtan sadece sırasıyla %0,02 düşüktür. Yine genetik algoritma ile optimize edilen ağ yapısın test doğruluğu bu sonuçtan %0,06 düşüktür. Test doğruluğu açısından bakıldığında sonuçların başarılı olduğu söylenebilir. Ancak FLOPs ve Parametre sayısı açısından kıyaslandığında önerilen Custom100 ve Genetik ile Optimize edilen

Custom100 yapısındaki FLOPs ve parametre sayısının bu çalışmadaki sonuçlara göre çok daha başarılı olduğu söylenebilir. Tüm bu sonuçlardan hareketle endüstriyel problemlerin derin öğrenme ile gerçeklenmesinde uygulamaya özel ağ yapılarının önerilmesinin literatürde ve uygulamada kendine geniş bir yer bulduğundan bahsedilebilir. Bu sayede doğruluktan ödün vermeden düşük FLOPs değerleri ve parametre sayıları ile çok daha yüksek hızlarda çalışan ve donanım dostu olan ağ yapılarının geliştirilmesi ve bunların Genetik algoritma gibi algoritmalarla optimize edilerek daha verimli hale getirilmeleri mümkündür.

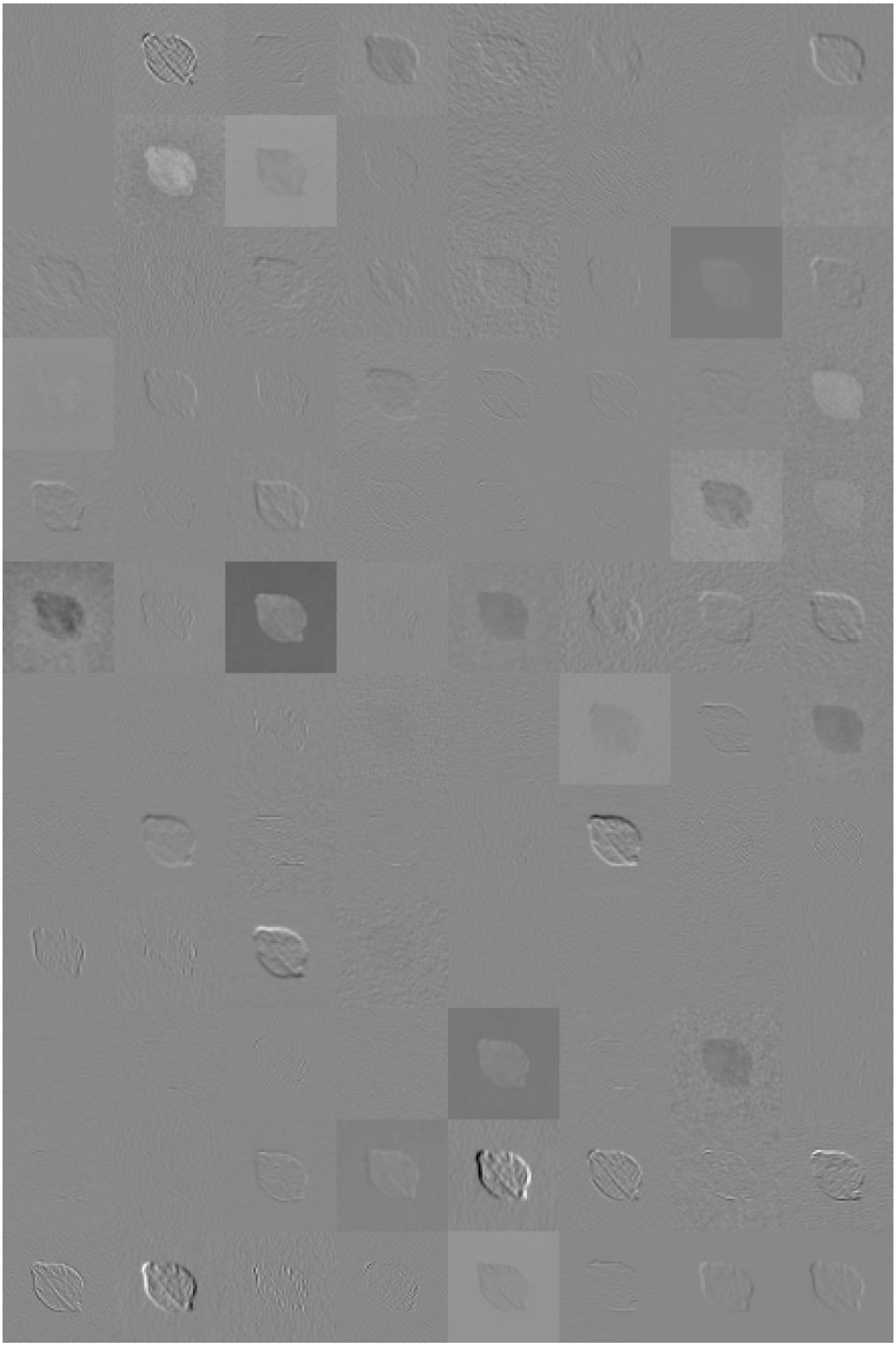
Çizelge 4.13. Sonuçların literatür ile karşılaştırılması

Ağ Yapısı	Test Doğruluğu	FLOPs	Parametere Sayısı
Custom100	%99,95	21.816.200	4.327.800
Custom100 Genetik	%99,91	18.724.027	2.431.208
EfficientNet-B0 (Fruit360 veri setindeki tüm sınıflar kullanılmıştır) (Duong vd. 2020)	%99,97	390.000.000	5.300.000

4.5. Veriyi Görselleştirme

Bu tez çalışmasında tohum ayıklama problemi ilk olarak literatürdeki AlexNet ve MobileNet-V2 ağ yapıları ile denenmiş ve en yüksek test doğruluğu %99 olarak hesaplanmıştır. Tezin ana amacı probleme özgü özel ağ yapılarının önerilmesi ve devamında FLOPs değerine göre optimize edilmesi olup; optimizasyon için en uygun parametrenin evrişim katmanındaki filtreler olduğu daha önceki matematiksel hesaplamalarda detaylı bir şekilde anlatılmıştır. Filtrelerin görsel olarak ne anlama geldiğini anlayabilmek ve filtre sayılarını azalttığımız halde neden test doğruluğunun aynı oranda değişmediğini daha iyi anlayabilmek için evrişim katmanlarındaki filtre çıkışlarına bakmak gerekmektedir. Örneğin Tohum ayıklama veri seti AlexNet yapısı ile eğitilip test edildiğinde test doğruluğu %98,87 olarak hesaplanmıştır. AlexNet ağ yapısında 5 adet evrişim katmanı olup; bu katmanlardan ilkinde 96 adet fitre bulunmaktadır. AlexNet ağ yapısı eğitildikten sonra Şekil 4.5'deki tohum görüntüsü ağa giriş olarak verildiğinde bu görüntünün ilk evrişim katmanındaki çıkışları Şekil 4.6'daki gibi olacaktır.

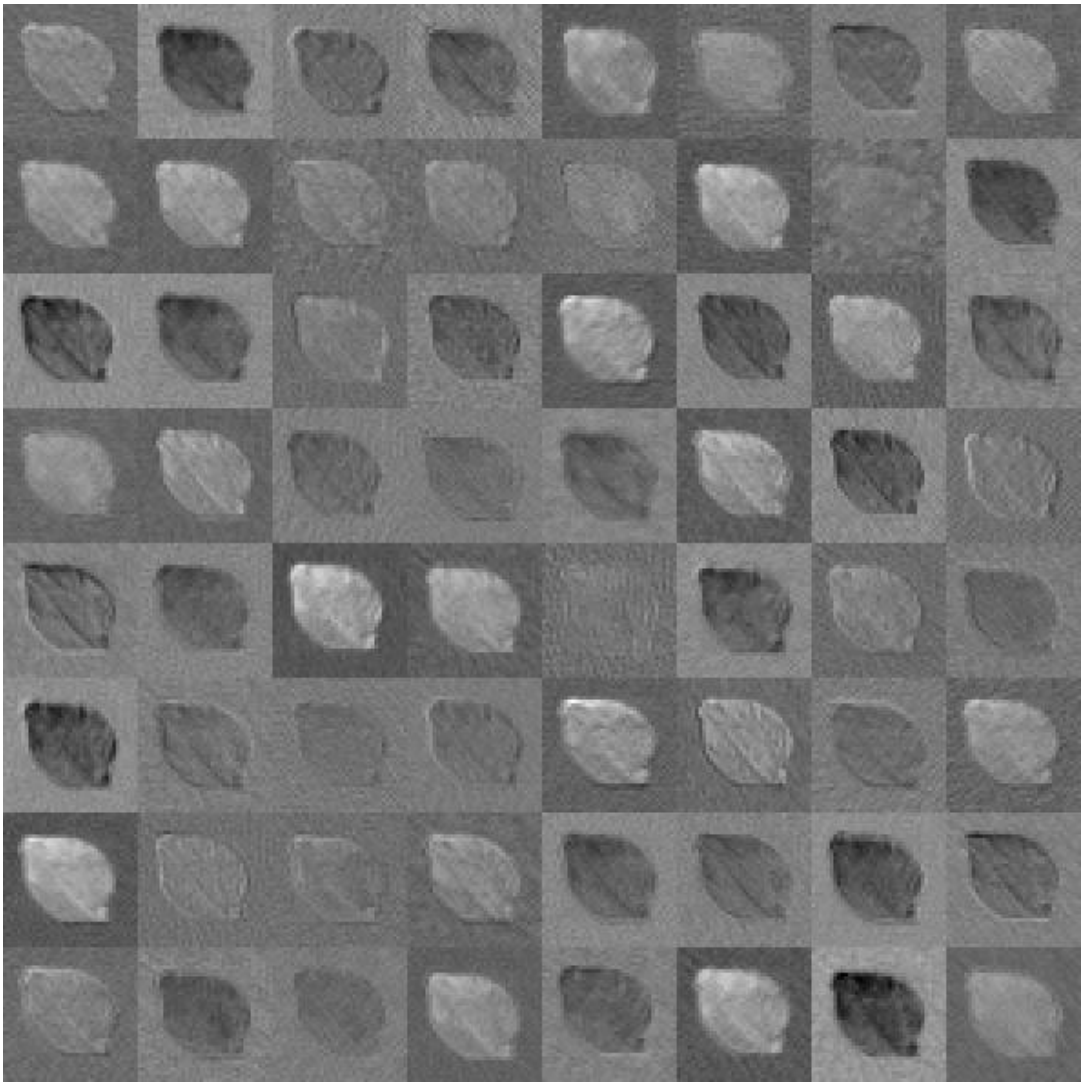
Şekil 4.5. Giriş olarak kullanılan tek tohum görüntüsü



Şekil 4.6. AlexNet yapısındaki ilk 96 filtrenin Şekil 4.5'deki girişe ait çıkış görüntüleri

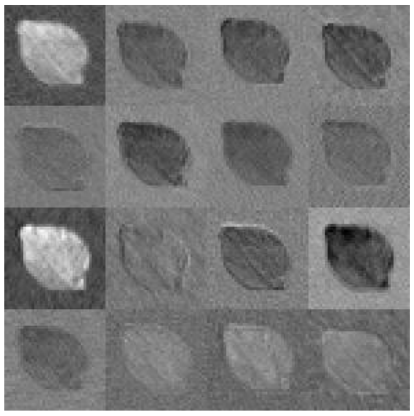
Şekile dikkatlice bakılırsa tüm filtre çıkışlarının tohuma ait bilgileri net bir şekilde taşımadığını görebiliriz. Bu aslında tez boyunca anlatılan literatürdeki derin ağ yapılarının büyük veri setlerini sınıflandırabilecek kapasitede derin ağlar olduğunu ve

bu yüzden fazla katman ve filtreye sahip olduğunu göstermektedir. Şekil 4.6'ya bakılarak tohum ayıklama işlemi için AlexNet ağ yapısı kullanılırsa ilk evrişim katmanında 96 adet filtreye gerek olmadığı net bir şekilde gözükmektedir. Çünkü bazı filtrelerin çıkışında tohuma ait öznitelikler net bir şekilde bulunmamaktadır. Tüm bu sebeplerden dolayı daha küçük ağ yapıları önerilmiş ve devamında ölçekle ve budama yöntemi ile doğruluktan ödün vermeyecek şekilde filtre sayıları sürekli yarıya indirilmiştir. Şekil 3.12'de önerilen özel ağ yapısının ilk evrişim katmanında 64 adet filtre bulunmaktadır. Custom77 ağının Type1.C veri seti ile eğitildikten sonra şekil 4.5'deki tek bir tohum için ilk evrişim katmanının çıkışı Şekil 4.7'deki gibidir.



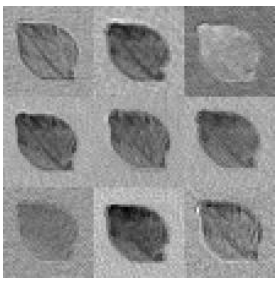
Şekil 4.7. Custom77 (conv1:64) (conv2:96) yapısındaki ilk 64 filtrenin Şekil 4.5'deki girişe ait çıkış görüntüleri

Şekil 4.7'ye bakıldığında bazı filtre çıkışlarının birbirine benzediği gözükmektedir. Önerilen özel ağ yapısı ölçekle ve budama yöntemi ile filtre sayıları sürekli yarıya indirildiğinde en optimum ağ yapısı Custom77 (conv1:16) (conv2:24) olarak hesaplanmıştır. Şekil 4.5'deki görüntü bu ağ yapısına giriş olarak verildiğinde bu ağ yapısının ilk katmanında bulunan 16 adet filtrenin çıkışı Şekil 4.8'deki gibidir.



Şekil 4.8. Custom77 (conv1:16) (conv2:24) yapısındaki ilk 16 filtrenin Şekil 4.5'deki girişe ait çıkış görüntüleri

Son olarak önerilen özel ağ genetik algoritma ile optimize edildiğinde Çizelge 4.9'a göre üç farklı sonuç elde edilmiştir. Bunlardan bir tanesi olan custom77 (conv1: 9, conv2: 46, K1:5, Sc1:2, K2:3, Sc2:3) optimum ağ modeli ilgili veri seti ile eğitildikten sonra Şekil 4.5'deki görüntü bu ağa giriş olarak verilmiştir. Bu girişe ait birinci katmandaki 9 filtrenin çıkış görüntüsü Şekil 4.9'daki gibidir.



Şekil 4.9. Custom77 (conv1: 9, conv2: 46, K1:5, Sc1:2, K2:3, Sc2:3) yapısındaki ilk 9 filtrenin Şekil 4.5'deki girişe ait çıkış görüntüleri

4.6. Optimize Edilen Ağların Hafızada Kapladığı Alanların Hesaplanması

Optimize edilen ağın verimliliğine ölçebilmek için bir diğer önemli kriter ise ağın büyüklüğü yani hafızada kapladığı boyuttur. Bunu ölçebilmek için ağın eğitilmesi gereken parametre sayısının hesaplanması ve 32 bitlik işlem yapıldığı için bu toplam sonucunun 4 ile çarpılıp bayt cinsinden sonucun bulunması gerekmektedir. Tüm bu işlemler Çizelge 4.14'de detaylı bir şekilde açıklanmış olup: önerilen en büyük ağ olan Custom227 ağının toplam parametre sayısı 18.003.988 ve bu ağın hafızada kapladığı alan ise 68,7 MB olarak hesaplanmıştır. Çizelge 4.14'deki diğer sütunlarda ölçekle ve budama metodu ve genetik algoritma ile optimize edilen ağlara ait parametre sayıları gösterilmektedir. Bu dört ağ yapısı bu tezde önerilmiş en optimum dört ağ yapısı olup bunları parametre sayısı açısından karşılaştıracak olursak en az parametre sayısı ölçekle ve budama metodu ile hesaplanan Custom77 ağına ait olduğu gözükmektedir. Optimum ağlar için hesaplanan tüm değerler Custom227 ağ yapısından çok küçük olup en uygun ağın seçilmesine karar verilmesinde test doğruluğu ve FLOPs gibi diğer parametreleri de göz ardı etmemek gerekecektir.

Çizelge 4.14. Farklı ağların parametre sayılarının karşılaştırılması ve hafızada kapladıkları alanların gösterilmesi

Katmanlar	Custom227	Custom77 (N1=16, N2=24)	Genetik 1. Koşturma Sonucu	Genetik 2. Koşturma Sonucu	Genetik 3. Koşturma Sonucu
	Parametre Sayısı	Parametre Sayısı	Parametre Sayısı	Parametre Sayısı	Parametre Sayısı
Conv_1	4.864	1.216	836	684	988
Havuzlama_1	0	0	0	0	0
Conv_2	55.392	3.480	6.800	3.772	6962
Havuzlama_2	0	0	0	0	0
TBK_1	17.915.952	221.232	626.736	423.984	543.792
TBK_2	27.672	27.672	27.672	27.672	27.672
TBK_3	98	98	98	98	98
Softmax	10	10	10	10	10
Toplam Parametre Sayısı	18.003.988	253.698	662.142	456.210	579.512
Gerekli Hafıza	68,7 MB	**0,96MB**	2,51MB	1,73MB	2,19MB
FLOPS	46.072.682	863.530	**715.453**	1.054.321	813.800

Elde edilen FLOPs, parametre sayısı ve gerekli hafıza sonuçları literatürdeki diğer ağ yapıları ile karşılaştırıldığında sonuçlar Çizelge 4.15'deki gibidir. Literatürden en bilindik AlexNet ağ yapısı ve küçük ve hızlı olduğu bilinen MobileNet-V2 ile EfficientNet-B0 ağ yapıları adil bir kıyaslama için seçilmiştir. Literatür ile karşılaştırmak için ölçekle ve budama yöntemi ile elde edilen ağ yapısı seçilmiştir. Tüm bu sonuçlara bakıldığında önerilen ve optimize edilen ağların literatürdeki tüm ağlardan FLOPs değerlerinin, parametre sayılarının ve gerekli olan hafızanın daha az olduğundan söz edilebilir.

Çizelge 4.15. Sonuçların FLOPs, parametre sayısı ve gerekli hafıza açısından literatür ile karşılaştırılması

	FLOPs	Parametre Sayısı	Gerekli Hafıza
Custom77 (conv1:16),(conv2:24)	863K	253K	0,96MB
AlexNet	720M	62M	236MB
MobileNet-V2	1,209G	3,4M	12,9MB
EfficientNet-B0	390M	5,3M	20,1MB

4.7. Ağların Sonuç Çıkarım Sürelerinin Hesaplanması

Çıkarım süresi ağın test aşamasında ne kadar sürdüğünü belirten süredir. Örneğin AlexNet ağı tohum ayıklama veri seti eğitildikten sonra bir tohum görüntüsünün AlexNet ağ yapısı ile test edilmesi işlemi x sürede gerçeklenmiş ise bu süreye çıkarım süresi denir. Ağın eğitim süresini ve test çok farklı terimler olmakla birlikte, test süresi daha çok gerçek zamanlı sistemler için önem arz etmektedir. Tohum ayıklama işlemi de endüstride gerçek zamanlı olarak yapıldığı için test süresinin kısa olması sistem için kritik bir parametredir. Test süresini, kullanılan donanım (CPU, GPU, Mobil GPU), kullanılan ağ yapısı, ağ yapısındaki FLOPs değeri, ağ yapısındaki parametre sayısı, ağ yapısındaki okuma yazma işlemlerinin çokluğu ve kullanılan kütüphaneler (Keras/Pytorch/Tensorflow) gibi birçok parametre etkilemektedir. Bu tez çalışmasında donanım olarak GPU, kütüphane olarak Keras ve ağ yapısı olarak AlexNet ve özel ağ modelleri kullanılmıştır. Bu çalışmada kullanılan ağ yapılarına ait test sonuçları Matlab ve Python/Keras için ayrı ayrı değerlendirilmiş olup; tüm sonuçlar Çizelge 4.16'daki gibidir. Özel ağ modellerinde de ağ yapısını optimize etmek için FLOPs değeri optimizasyon parametresi olarak kullanılmıştır. Yukarıda bahsedilen diğer parametrelerin de çıkarım süresine etkisi olduğu için FLOPs değerinin çıkarım süresini doğru orantıda değiştireceğini söylemek doğru olmayacaktır. Nitekim Custom227 FLOPs değeri 46M iken Custom77 (conv1:64),(conv2:96) FLOPs değeri 4,4M'dur. Arada on kat fark olmasına rağmen Çizelge 4.16'da test sürelerine bakıldığında aradaki farkın 0,32ms/0,18ms = 1,77 kat olduğu görülmektedir.

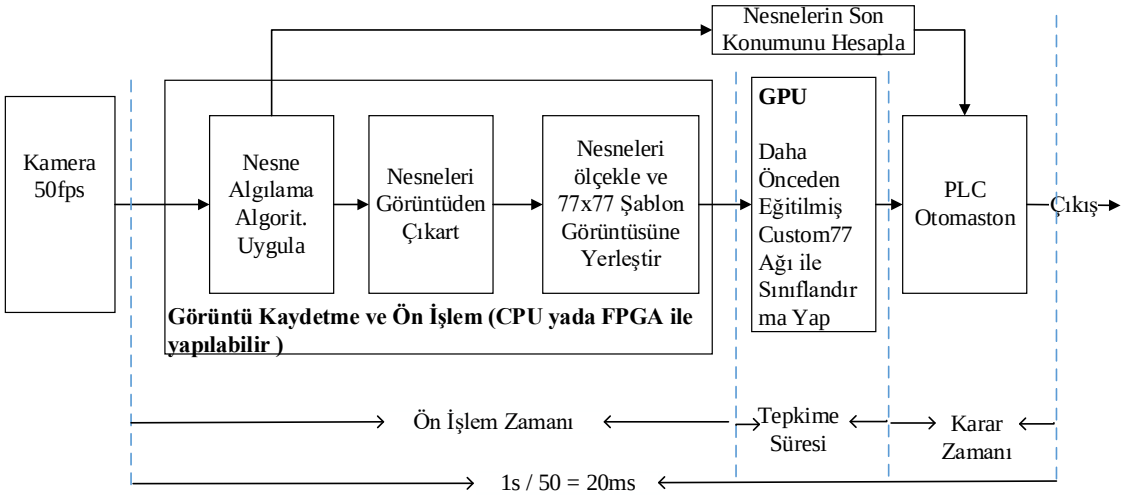
Çizelge 4.16. Farklı ağ yapılarına ait çıkarım süreleri

		Type1 Veri Seti ile Yapılan Sınıflandırma Test Süreleri			
		Matlab Sonuçları		Python/Keras Sonuçları	
CNN Yapısı	Filtre Sayıları	1 nesne için çıkarım süresi(B.S=1)	1 nesne için çıkarım süresi(B.S=32)	1 nesne için çıkarım süresi(B.S =1)	1 nesne için çıkarım süresi(B.S =32)
Alexnet	Eğitilmiş AlexNet	7,57ms	1,38ms	3,96ms	0,41ms
Custom227	(conv1:64) (conv2:96)	5,98ms	0,99ms	2,13ms	0,32ms
Custom157	(conv1:64) (conv2:96)	5,01ms	0,87ms	1,83ms	0,22ms
Custom77	(conv1:64) (conv2:96)	4,39ms	0,77ms	1,77ms	0,18ms
Custom77	(conv1:32) (conv2:48)	4,05ms	0,75ms	1,71ms	0,14ms
Custom77	(conv1:16) (conv2:24)	3,98ms	0,73ms	1,65ms	0,12ms
Custom37	(conv1:64) (conv2:96)	4,27ms	0,71ms	1,70ms	0,13ms

4.8. Sonuçların Gerçek Zamanlı Ayıklama Sistemleri Açısından Tartışılması

Gerçek zamanlı tohum ayıklama sisteminin nasıl çalıştığı Şekil 4.10'da detaylı bir şekilde açıklanmıştır. Bu çalışmada kullanılan kamera 73fps olup; tohumlar düşerken her bir tohumun görüntüsünün minimum bir kere çekilmesi için kameranın hızının 50fps olması gerektiği teorik olarak hesaplanmış ve deneysel olarak test edilmiştir. Böylesi gerçek zamanlı bir sistemde eğer ki kullanılan kamera 50fps ise bu şu demektir: her bir görüntü (frame) en fazla 1sn/50 = 20ms'de işlenmesi gerekmektedir. Derin öğrenme ile tohum ayıklama işlemi

Şekil 4.10'da gösterildiği gibi ön işlemler ve GPU ile sınıflandırma olmak üzere temel iki işlemden oluşmaktadır (Son işlem olan sınıflandırma sonucuna göre otomasyonun çalıştırılması donanımın saat darbesi mertebelerinde olunduğu için yani ns mertebelerinde olduğu ihmal edilebilecek seviyelerdedir.). Bu açıdan bakıldığında sistemin gerçek zamanlı olabilmesi için yani 20ms'lik sürede hem ön işlemlerin hem de sınıflandırma işlemlerinin tamamlanmış olması gerekmektedir. Ön işlemlerde kullanılan nesne tespit algoritması ve GPU ile sınıflandırma işlemleri yüksek işlem yüküne sahip işlemler olup; gerçek zamanlı sistemlerin geliştirilmesi için bu işlemlerin hızlandırılması büyük önem arz etmektedir. Örneğin bu çalışmada kullanılan kamera çözünürlüğü 1440x1080 olup; nesne tespit algoritması CPU ile örnek bir görüntü üzerinde(örnek görüntüde toplam 8 adet tohum bulunmaktadır.) gerçeklendiğinde işlem 100ms sürmüştür. Bu sonuca istinaden nesne tespit algoritmasının CPU üzerinde gerçeklenirse gerçek zamanlı bir sistemden söz edilmesi söz konusu olamaz. Nesne tespit algoritması ve diğer ön işlemleri hızlandırmak için FPGA gibi hızlı donanımların kullanılması daha doğru olacaktır (Johannes vd. 2007). FPGA üzerinde 8k x 8k çözünürlükte 400 fps'de çalışabilen bir nesne tespit algoritmasını gerçeklemiştir. Bu da tüm ön işlemlerin 1ms'den daha kısa bir sürede FPGA üzerinde gerçeklenmesi demektir.



Şekil 4.10. Gerçek zamanlı tohum ayıklama sistemi

Gerçek zamanlı sistem GPU ile sınıflandırma açısından tartışılırsa: bu çalışmada minimum kamera hızı 50fps olarak hesaplanmıştır. Yine kayıtlı 1440x1080 boyutundaki

görüntüler analiz edildiğinde bir çerçevede en fazla 10 adet tohumun olunduğu görülmüştür. Bu şu anlama gelmektedir: gerçek zamanlı bir sınıflandırma yapabilmek için 10 adet tohumun sınıflandırma işlemi maksimum 20ms-1ms(ön işlem süresi) = 19ms'de gerçeklenmesi demektir. Yani batch-size = 1 seçilirse her bir tohumun 19ms/10 = 1,9ms'de sınıflandırılması gerekmektedir.

Çizelge 4.14'de iki farklı batch size değeri kullanılmıştır B.S. = 1 şu anlama gelmektedir: Şekil 4.10'daki gerçek zamanlı sistemde ilk önce ön işlemler yapılmaktadır. Örneğin bir çerçevede 10 adet tohum varsa bu tohumlar ön işlemde tek tek tespit edilip boyutları ağ için uygun hale getirildikten sonra bir dosyada kaydedilmektedir. Eğer ki B.S. = 1 ise sınıflandırma işlemi için kayıtlı oldukları dosyadan birer birer okunmaktadır. Diğer durumda dosyadaki kayıtlı tüm tohumlar tek bir seferde sınıflandırma işlemi için GPU'da işleme sokulmak istenirse bu durumda batch size en az maksimum kayıtlı tohum sayısı kadar seçilmelidir. Sınıflandırma sonucunda tohumun OK ya da NOK olması durumuna göre ayıklama işleminin yapılacağından dolayı B.S.=1 seçerek her bir tohumun sonucuna göre otomasyonu çalıştırmak uygulamada daha kolay olacaktır. Bu yüzden kurulacak gerçek zamanlı sistemi B.S = 1 için yorumlamak daha doğru olacaktır. B.S. = 1 için tablo 11 yorumlanırsa: daha önce minimum zaman koşulu 1,9ms olarak hesaplanmıştı. Böyle bir sistemde AlexNet yapısı kullanılırsa AlexNet yapısının çıkarım süresi 3,96ms olacağı için sistemin gerçek zamanlı olmasından bahsedilemez. Nitekim Custom 227 ağ yapısının çıkarım süresi 2,13ms olup bu ağ yapısının da gerçek zamanlı sistemde kullanılmasından söz edilemez. Ancak Çizelge 4.14'deki Custom157 ve Custom77'ye ait diğer yapıların kullanılması halinde sistemin gerçek zamanlı olabileceğinden söz edilebilir. Bu açıdan bakıldığında derin öğrenme ağlarının gerçek zamanlı sistemlerde kullanılabilmesi için optimizasyon teknikleri ile ağların küçültülmesi ve hızlandırılmasının ne kadar önemli olduğundan bahsedilebilir.

4.9. Sonuçların Donanım Açısından Tartışılması

Sonuçların donanım açısından tartışılması için öncelikle iki tanıma açıklık getirilmesi gerekmektedir. İlk tanım bu tez çalışmasında detaylı bir şekilde ele alınan kayan nokta işlem sayısı (FLOPs) diğer tanım ise bir saniyede donanım tarafından işlenebilecek kayan nokta işlem sayısı (FLOPS). İlk tanım FLOPs daha çok derin öğrenme ağlarının model karmaşıklığını göstermek üzere kullanılırken; ikinci tanım ise daha çok donanım üretici firmalar tarafından donanımın işlem kapasitesini ifade etmek için kullanılır. Bu ifadeleri çok basit örnek ile açıklamak gerekirse, en bilindik AlexNet yapısının FLOPs değeri 720 Milyon'dur. Yani tek bir görüntünün AlexNet ile sınıflandırılması için 720 Milyon adet kayan nokta işleminin yapılması gerekmektedir. AlexNet yapısının bir gömülü bir sistem üzerinde (ARM ya da Mobil GPU) koşturulduğunu düşünürsek; AlexNet yapısının bu gömülü sistem üzerinde maksimum 1 saniyede gerçeklenmesi için (bu da uygulamada 1fps demektir) gerekli olan minimum FLOPS değeri 720 Milyon olacaktır. Yani kullanılan donanım saniyede minimum 720 Milyon kayan nokta işlemini yapabilecek kapasitede olması gerekecektir. Bu örneği genişletecek olursak AlexNet yapısının bir saniyede 10 kere gerçeklenebilmesi için (yani 10 fps lik bir hızda) kullanılan donanımın FLOPS değeri minimum 720 Milyon x 10 = 7,2 Milyar olması gerekmektedir.

Bu çalışmada kullanılan CPU Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz olup; bu donanıma ait FLOPS değeri 32,74 GFLOPS'tur (Anonymous 3). Yine kullanılan GPU modeli Nvidia RTX 2080 Super olup; bu donanıma ait FLOPS değeri 11,15 TFLOPS'tur (Anonymous 4). Mobil GPU'lardan örnek vermek gerekirse Apple Iphone 7'de bulunan PowerVR GT7600 GPU'suna ait FLOPS değeri ise 10 GFLOPS olarak belirtilmektedir (Deng 2019). Bu tez çalışmasında ölçekle ve bu da metodu ile küçültülen ağın FLOPs değeri 863.530 yine genetik algoritma ile optimize ağın en iyi FLOPs değeri ise 715.453 olarak hesaplanmıştır. Bu açıdan bakıldığında optimize edilen ağların tamamıyla donanım dostu olduğundan bahsedilebilir. Toplam işlem sayısının az olması aynı zamanda birim zamanda tüketilen gücünde az olması demektir. Ağların gömülü bir sistem üzerinde gerçeklendiğinde hafızada kaplayacağı alanlara bakıldığında Çizelge 4.15'e göre AlexNet yapısı için 236MB'lık bir hafızaya gerek vardır. Bu değer MobileNet-V2 ağ yapısı için 12,9MB ve optimize edilen ağlardan birisi olan Custom77 ağ yapısı için ise 0,96MB olarak hesaplanmıştır. Bu açıdan bakıldığında optimize edilen ağların gömülü sistemler için daha uygun olduğundan söz edilebilir.

5. SONUÇLAR

Bu tez çalışmasında nihai hedef endüstriyel bir problemin derin öğrenme ile yüksek hızlarda ve yüksek doğrulukta gerçeklenmesidir. Bunun için uygulamaya özgü özel bir ağ yapısının önerilmesi ve bu ağ yapısının farklı parametrelere göre optimize edilmesidir. Endüstriyel problem olarak endüstride kendine çok geniş bir yer bulan ayıklama (sorting) problemi seçilmiştir. Ayıklama problemi çok genel bir problem olup; bu problemin özel bir uygulama alanı olan tohum ayıklama problemi bu tez çalışmasına konu olarak seçilmiştir. Bu problemi derin öğrenme ile gerçekleyebilmek için ilk olarak veri setleri oluşturulmuştur. Literatür taramasında daha önceki tohum sınıflandırma ve ayıklama çalışmalarından detaylı bir şekilde bahsedilmiş olup; bu tez çalışmasında veri setlerinin endüstriyel problemlere uygun bir şekilde oluşturulması için özgün bir otomatik veri seti oluşturma algoritması önerilmiştir. Önerilen bu algoritma sayesinde farklı ölçekleme yöntemleri (Type1 ve Type2 veri setleri) ve farklı boyutlarda (227x227, 157x157, 77x77, 37x37) veri setleri oluşturma işlemi otomatik hale getirilmiştir. Farklı ölçekleme yöntemleri ile oluşturulan veri setleri literatürdeki AlexNet ve MobileNet ağ yapıları ile test edilmiş ve ölçekleme yönteminin test doğruluğunu nasıl etkilediği açık bir şekilde belirtilmiştir. Tablo 6'daki bu test sonuçlarına göre Type1 veri setlerinde en yüksek doğruluk %99 olarak hesaplanırken bu test doğruluğu Type2 veri setinde %98,50 olarak hesaplanmıştır. Bu karşılaştırma sonuçlarına bakılarak; ayıklama işlemlerinde Type1 veri setinin kullanılmasının daha doğru olacağı açık bir şekilde söylenebilmektedir.

Tohum ayıklama işlemi endüstride yüksek hızlarda gerçeklenen bir problem olup; tohum ayıklama problemini yüksek hızlarda ve yüksek doğrulukta gerçekleyebilmek için probleme özgü Şekil 3.12'deki gibi bir özel ağ yapısı önerilmiştir. Tohum ayıklama problemi için maksimum test doğruluğu %99 olup; önerilen özel ağ yapısında test doğruluğunun maksimum %1 sapma ile en az %98 olması hedeflenmektedir. Bu çalışmada dört farklı boyutta veri seti kullanıldığı için (227x227,157x157, 77x77,37x37) bu veri setlerini gerçekleyebilecek şekilde dört farklı boyutta özel ağ yapısı (Custom227, Custom157, Custom77, Custom37) önerilmiştir. Bu dört farklı ağ yapısının tüm parametreleri aynı olup sadece giriş data boyutları farklı olarak seçilmiştir. Önerilen bu ağ yapılarının test doğruluğunu literatürdeki ağ yapıları ile karşılaştırmak için AlexNet ve MobileNet eğitim ve test aşamasında kullanılan Type1.A veri seti ve bu veri setini gerçeklemek için uygun boyuttaki Custom227 ile ilk testler yapılmış olup; test sonuçlarında maksimum test doğruluğu %99 olarak hesaplanmıştır. Önerilen Custom227 ağ yapısı AlexNet ve MobileNet'e göre FLOPs, parametre sayısı ve çıkarım süresi açısından hızlı bir yapı olmasına rağmen bu yapıyı daha da küçültüp hızlandırmak için Şekil 3.14'deki gibi iki aşamalı ölçekle ve budama yöntemi önerilmiştir. Önerilen bu yöntemin test sonuçları Çizelge 4.5 ve 4.6'daki gibi olup bu sonuçlara göre en optimum ağ yapısı Custom77 (conv1:16, conv2:24) olarak bulunmuştur. Bulanan bu ağa ait test doğruluğu ise %98,16 olarak hesaplanmıştır. Ölçekle ve budama yöntemi ile elde edilen sonuçlar FLOPs açısından karşılattırıldığında Custom227 ağ yapısının FLOPs değeri 46.072.682 iken Custom77 (conv1:16, conv2:24) ağ yapısının FLOPs değeri 863.530 olarak hesaplanmıştır. Bu da FLOPs açısından 46.072.682 / 863.530 = 53,35 kat daha küçük bir yapı demektir.

Ölçekle ve budama metodu, önerilen özel modeller üzerinde başarılı sonuçlar vermiş olup; ağın küçültülmesi ve hızlandırılması işlemi bir sonraki aşamada genetik

algoritma ile yapılmıştır. Özel ağlara genetik algoritmanın uygulanması için özel modeller derinlemesine incelenmiş ve katman seviyesinde FLOPs değerleri tek tek hesaplanmıştır. Yapılan bu hesaplamalar sonucu önerilen özel ağ yapılarının genetik algoritma ile optimize edilebilmesi için filtre sayıları (N1 ve N2), atlama sayıları (Sc1 ve Sc2) ve filtre boyutları (Kc1 ve Kc2) parametre olarak seçilmişlerdir. Yapılan testler sonucunda ölçekle ve budama metoduna benzer sonuçlar elde edilmiştir. Genetik algoritma ile en düşük FLOPs değeri N1=11, N2=68, K1=5, K2=3, Sc1=3, Sc2=2 parametreleri için 715.453 olarak hesaplanmıştır. Bu değer ölçekle ve budama yönteminde hesaplanan 863.530 değerinden daha iyi bir değerdir. Test doğruluğu açısından bakıldığında genetik algoritma ile hesaplanan optimum ağların en yüksek test doğruluğu %98,5 olarak hesaplanmıştır. Bu değer ölçekle ve budama yönteminin sonucu ile aynıdır.

Özel ağ yapılarını optimize etmek için ölçekle ve budama yöntemi ile genetik algoritma kullanılmış olup; her iki yöntemden de başarılı sonuçlar elde edilmiştir. Başka bir değişle iki yöntem birbirinin doğrulamasını yapmıştır. İki yöntemden elde edilen sonuçlar gerçek zamanlı sistemler ve gömülü sistemler için de ayrıca incelenmiştir. Elde edilen sonuçlar gerçek zamanlı sistemler için değerlendirildiğinde sınıflandırma işlemi AlexNet yapısı ile gerçeklendiğinde çıkarım süresi B.S.=1 için 3,96ms olarak hesaplanmıştır. Sınıflandırma işlemi ölçekle ve budama yöntemi ile hesaplanan optimum ağ yapısı ile gerçeklendiğinde çıkarım süresi B.S.=1 için 1,65ms olarak hesaplanmıştır. Bu açıdan değerlendirildiğinde optimize edilen ağ yapısı olan Custom77 AlexNet'e göre 3,96ms/1,65ms = 2,4 kat daha hızlı çıkarım süresine sahiptir. Bu da özel ağ yapılarını gerçek zamanlı sistemler için daha tercih edilebilir kılmaktadır. Yine elde edilen sonuçlar parametre sayısı ve hafızada kapladıkları alan bakımında kıyaslandıklarında: AlexNet yapısı için 236MB, MobileNet-V2 ağ yapısı için 12,9MB ve optimize edilen Custom77 ağ yapısı için ise 0,96MB hafızaya ihtiyaç duyulmaktadır. Bu sonuçlara göre optimize edilen ağ yapısı MobileNet-V2 ağ yapısına göre 13,43 kat daha az yer kapladığı söylenebilir. Bu da önerilen ve devamında optimize edilen özel ağ yapılarını gömülü sistem gerçeklemelerinde verimli kılmaktadır.

Bu tez çalışmasındaki elde edilen çıktılar, literatürde ve endüstride kendine geniş bir yer bulmaktadır. Elde edilen mevcut çıktılar birçok farklı problemde efektif bir şekilde kullanılabilir. Tohum ayıklama problemi, endüstrideki ayıklama problemlerinden bir tanesi olup; bu tez çalışmasındaki çıktılar çok rahat bir şekilde diğer ayıklama (meyve ayıklama, yiyecek içecek ayıklama ve maden ayıklama gibi) problemlerinde de uygulanabilir. Öyle ki; veri seti oluşturmak için geliştirilen deney düzeneği ve otomatik veri seti oluşturma algoritması geliştirilerek çok rahat bir şekilde bu problemlere uygulanabilir. Bu tezdeki çalışmaların devamı niteliğinde aşağıdaki çalışmaların gelecekte yapılması ve tez çalışmasının genişletilmesi hedeflenmektedir.

1) Bu tezdeki çalışmalar gelecekte daha genel bir yapıya dönüştürülüp ayıklama problemleri için veri setlerinin oluşturulmasından, bu veri setlerine uygun en optimum ve en hızlı ağ yapısını otomatik bir şekilde hesaplayan bir sisteme dönüştürülmesi hedeflenmektedir. Bu sayede kullanıcı dostu bir ara yüz ve tasarlanarak kullanıcının veri seti oluşturma teknikleri, evrişimli sinir ağ yapıları ve optimizasyon teknikleri hakkında çok bir bilgisi olmasa da geliştirilecek bu yazılım sayesinde yazılımın birçok şeyi kullanıcı için otomatik olarak yapması hedeflenmektedir.

2) Yine gelecekteki çalışmalarda derin öğrenme ile sınıflandırma işlemlerinin gerçek zamanlı olabilmesi için FPGA+GPU hibrit yapıları geliştirilmesi ve bunun genelleştirilmesi hedeflenmektedir. Bu sayede hem görüntü işleme hem de derin öğrenmenin bir arada olduğu yüksek hızlarda işlemlerin yapıldığı uygulamaların geliştirilmesi hedeflenmektedir.

3) Bu tez çalışmasında optimize edilen ağ yapıları gömülü sistemler için olumlu sonuçlar vermiş olup; gelecekte optimize edilen bu ağ yapılarının FPGA üzerinde çalıştırılması hedeflenmektedir. Bu sayede ağların çıkarım süreleri daha da küçültülerek derin öğrenme ağlarının gerçek zamanlı sistemlere daha uygun hale getirilmesi hedeflenmektedir.

6. KAYNAKLAR

Abade, A.S., Ferreira, P.A. and de Barros Vidal, F. 2020. Plant Diseases recognition on images using Convolutional Neural Networks: A Systematic Review.

Alves, T.S., Pinto, M.A., Ventura, P., Neves, C.J., Biron, D.G., Junior, A.C., De Paula Filho, P.L. and Rodrigues, P.J. 2020. Automatic detection and classification of honey bee comb cells using deep learning. *Comput. Electron. Agric.* https://doi.org/10.1016/j.compag.2020.105244

Aktas H. ve Polat O. 2020 "Derin Öğrenme ile Tohumların Sınıflandırılmasında Renk ve Kenar Bilgisinin Etkisi" 6. Ulusal Yüksek Başarımlı Hesaplama Konferansı, 8-9 Ekim 2020, Ankara Yıldırım Beyazıt Üniversitesi, Ankara

Anonymous 1: http://www.image-net.org/ [Son erişim tarihi: 04.12.2020].

Anonymous 2: https://www.programmersought.com/article/1732544989 [Son erişim tarihi: 02.12.2020].

Anonymous 3: https://mindmodeling.org/cpu_list.php [Son erişim tarihi: 01.12.2020].

Anonymous 4: https://www.techpowerup.com/gpu-specs/geforce-rtx-2080-super.c3439 [Son erişim tarihi: 30.11.2020].

Aszemi, N.M. and Dominic, P.D.D. 2019. Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms. IJACSA) *Int. J. Adv. Comput. Sci. Appl*.

Back, T. 1996. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Illustrate. ed. Oxford University Press.

Baker, B., Gupta, O., Naik, N. and Raskar, R. 2016. Designing Neural Network Architectures using Reinforcement Learning. 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.

Banner, R., Hubara, I. and Soudry, D. 2018. Scalable Methods for 8-bit Training of Neural Networks. *Adv. Neural Inf. Process. Syst*.

Banzhaf, W. 1997. Genetic Programming: An Introduction (The Morgan Kaufmann Series in Artificial Intelligence), 1st editio. ed. Morgan Kaufmann.

Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. 2008. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst*. https://doi.org/10.1016/j.cviu.2007.09.014

Beheshti, Z., Shamsuddin, S.M. and Shamsuddin, S.M.H. 2013. A review of population-based meta-heuristic algorithm Web Caching View project A review on handwritten character and numeral recognition for Roman, Arabic, Chinese and Indian scripts View project A Review of Population-based Meta-Heuristic Algorithm. *Int. J. Adv. Soft Comput. Appl*.

Bianchi, L., Dorigo, M., Gambardella, L.M. and Gutjahr, W.J. 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Nat. Comput.* 8, 239–287. https://doi.org/10.1007/s11047-008-9098-4

Bianco, S., Cadene, R., Celona, L. and Napoletano, P. 2018. Benchmark analysis of

representative deep neural network architectures. *IEEE Access* 6, 64270–64277. https://doi.org/10.1109/ACCESS.2018.2877890

Bircanoglu, C., Atay, M., Beser, F., Genc, O. and Kizrak, M.A. 2018. RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks, 2018 IEEE (SMC) International Conference on Innovations in Intelligent Systems and Applications, INISTA 2018. https://doi.org/10.1109/INISTA.2018.8466276

Cai, H., Chen, T., Zhang, W., Yu, Y. and Wang, J. 2017. Efficient Architecture Search by Network Transformation. 32nd AAAI Conf. Artif. Intell. AAAI 2018 2787–2794.

Changpinyo, S., Sandler, M. and Zhmoginov, A. 2017. The Power of Sparsity in Convolutional Neural Networks. *arXiv* :1702.06257

Chin, T.-W., Zhang, C. and Marculescu, D. 2018. Layer-compensated Pruning for Resource-constrained Convolutional Neural Networks. *arXiv* :1810.00518

Chollet, F., 2018. Deep Learning with Phyton, Manning.

Chollet, F., 2017. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* :1610.02357

Cun, Y. Le, Guyon, I., Jackel, L.D., Henderson, D., Boser, B., Howard, R.E., Denker, J.S., Hubbard, W. and Graf, H.P. 1989. Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning. *IEEE Commun. Mag.* https://doi.org/10.1109/35.41400

Dalal, N. and Triggs, B. 2005. Histograms of oriented gradients for human detection, içinde: Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005. https://doi.org/10.1109/CVPR.2005.177

Davis, L. 1991. Handbook of genetic algorithms. Van Nostrand Reinhold, New York.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. 2010. ImageNet: A large-scale hierarchical image database. Institute of Electrical and Electronics Engineers (IEEE), ss. 248–255. https://doi.org/10.1109/cvpr.2009.5206848

Deng, Y. 2019. Deep learning on mobile devices: A Review Mobile Multimedia/Image Processing, Security, and Applications 2019. SPIE, s. 11. https://doi.org/10.1117/12.2518469

Duong, L.T., Nguyen, P.T., Di Sipio, C. and Di Ruscio, D. 2020. Automated fruit recognition using EfficientNet and MixNet. *Comput. Electron. Agric.* https://doi.org/10.1016/j.compag.2020.105326

ELDEM, A. 2020. Buğday Tohumlarının Derin Sinir Ağı Uygulaması ile Sınıflandırılması. *Eur. J. Sci. Technol.* 19, 213–220. https://doi.org/10.31590/ejosat.719048

Elihos, A., Balci, B., Alkan, B. and Artan, Y. 2019. Deep Learning Based Segmentation Free License Plate Recognition Using Roadway Surveillance Camera Images. *arXiv* :1912.02441

Guo, Y., Yao, A. and Chen, Y. 2016. Dynamic Network Surgery for Efficient DNNs. *Adv. Neural Inf. Process. Syst.*

Han, S., Pool, J., Narang, S., Research, B., Mao, H., Tang, S., Elsen, E., Catanzaro, B., Tran, J. and Dally, W.J. 2016. DSD: Regularizing Deep Neural Networks with Dense-Sparse-Dense Training Flow. ICLR 2017

Han, S., Pool, J., Tran, J. and Dally, W.J. 2015. Learning both Weights and Connections for Efficient Neural Networks. Adv. Neural Inf. Process. Syst. NIPS 2015

Heo, Y.J., Kim, S.J., Kim, D., Lee, K. and Chung, W.K. 2018. Super-high-purity seed sorter using low-latency image-recognition based on deep learning. *IEEE Robot. Autom. Lett.* https://doi.org/10.1109/LRA.2018.2849513

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* :1704.04861

Hu, J.F., Huang, T.Z., Deng, L.J., Jiang, T.X., Vivone and G., Chanussot, J. 2020. Hyperspectral Image Super-resolution via Deep Spatio-spectral Convolutional Neural Networks. *arXiv* :2005.14400

Huang, K.Y. and Cheng, J.F. 2017. A Novel Auto-Sorting System for Chinese Cabbage Seeds. *Sensors* 17, 886. https://doi.org/10.3390/s17040886

Huang, S., Fan, X., Sun, L., Shen, Y. and Suo, X. 2019. Research on Classification Method of Maize Seed Defect Based on Machine Vision. J. *Sensors* https://doi.org/10.1155/2019/2716975

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. and Bengio, Y. 2016. Binarized Neural Networks. Advances in Neural Information Processing Systems NIPS *2016*. ss. 4114–4122.

Hussain, L., Ajaz, R.H., Jammu, A. and Muzaffarabad, K. 2015. Seed Classification using Machine Learning Techniques. *Journal of Multidisciplinary Engineering Science and Technology*, 2015. 2(5): p. 1098-1102

Ioffe, S. and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 32nd International Conference on Machine Learning, ICML 2015. ss. 448–456.

Janis, C. 1976. The Evolutionary Strategy of the Equidae and the Origins of Rumen and Cecal Digestion. *Evolution (N. Y).* 30, 757. https://doi.org/10.2307/2407816

Jarrett, K., Kavukcuoglu, K., Ranzato, M. and LeCun, Y. 2009. What is the best multi-stage architecture for object recognition?, Proceedings of the IEEE International Conference on Computer Vision. ss. 2146–2153. https://doi.org/10.1109/ICCV.2009.5459469

Jiang, M., Huang, Z., Qiu, L., Huang, W. and Yen, G.G. 2018. Transfer Learning-Based Dynamic Multiobjective Optimization Algorithms. *IEEE Trans. Evol. Comput. 22*, 501–514. https://doi.org/10.1109/TEVC.2017.2771451

Johannes, T., Th Schwarzbacher, A., Hoppe, B., Noffz, K. and Trenschel, T. 2007. Development of a FPGA Based Real-Time Blob Analysis Circuit. ISSC.

Karlekar, A. and Seal, A., 2020. SoyNet: Soybean leaf diseases classification. *Comput. Electron. Agric.* https://doi.org/10.1016/j.compag.2020.105342

Kamal, K., Yin, Z., Wu, M. and Wu, Z. 2019. Depthwise separable convolution

architectures for plant disease classification. *Comput. Electron. Agric.* https://doi.org/10.1016/j.compag.2019.104948

Kim, P. 2017. MATLAB Deep Learning, MATLAB Deep Learning. https://doi.org/10.1007/978-1-4842-2845-6

Kiratiratanapruk, K. and Sinthupinyo, W. 2011. Color and texture for corn seed classification by machine vision, 2011 International Symposium on Intelligent Signal Processing and Communications Systems: "The Decade of Intelligent and Green Signal Processing and Communications", ISPACS 2011. https://doi.org/10.1109/ISPACS.2011.6146100

Knoll, F.J., Czymmek, V., Harders, L.O. and Hussmann, S. 2019. Real-time classification of weeds in organic carrot production using deep learning algorithms. *Comput. Electron. Agric.* https://doi.org/10.1016/j.compag.2019.105097

Knoll, F.J., Czymmek, V., Poczihoski, S., Holtorf, T. and Hussmann, S. 2018. Improving efficiency of organic farming by using a deep learning classification approach. *Comput. Electron. Agric.* https://doi.org/10.1016/j.compag.2018.08.032

Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012. ImageNet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems. ACM 60, 84–90. https://doi.org/10.1145/3065386

Kurtulmuş, F., Alibaş, İ. and Kavdir, I. 2016. Classification of pepper seeds using machine vision based on neural network. *Int. J. Agric. Biol. Eng.* 9, 51–62. https://doi.org/10.3965/j.ijabe.20160901.1790

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2323. https://doi.org/10.1109/5.726791

LeCun, Y., Kavukcuoglu, K. and Farabet, C. 2010. Convolutional Networks and Applications in Vision, ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems. ss. 253–256. https://doi.org/10.1109/ISCAS.2010.5537907

Li, H., De, S., Xu, Z., Studer, C., Samet and H., Goldstein, T. 2017. Training Quantized Nets: A Deeper Understanding. *Adv. Neural Inf. Process. Syst.*

Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P. 2016. Pruning Filters for Efficient ConvNets. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 4510–4520.

Li, J., Liao, G. and Xiao, F. 2008. Rapeseed seeds colour recognition by machine vision, içinde: Proceedings of the 27th Chinese Control Conference, CCC. IEEE Computer Society, ss. 146–149. https://doi.org/10.1109/CHICC.2008.4604918

Li, Y., Mahjoubfar, A., Chen, C.L., Niazi, K.R., Pei, L. and Jalali, B. 2019. Deep Cytometry: Deep learning with Real-time Inference in Cell Sorting and Flow Cytometry. *Sci. Rep.* https://doi.org/10.1038/s41598-019-47193-6

Liu, H., Simonyan, K., Vinyals, O., Fernando, C. and Kavukcuoglu, K. 2017. Hierarchical Representations for Efficient Architecture Search.

arXiv:1711.00436

Lowe, D.G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* https://doi.org/10.1023/B:VISI.0000029664.99615.94

Luo, J.H., Wu, J. and Lin, W. 2017. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. ICCV 2017, 22-29 October 2017, Venice, Italy

Mallick, S. And Nayak, S. 2018. https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/ [Son erişim tarihi: 29.11.2020].

Mehta, J., Ramnani, E. and Singh, S. 2018. Face Detection and Tagging Using Deep Learning, içinde: 2nd International Conference on Computer, Communication, and Signal Processing: Special Focus on Technology and Innovation for Smart Environment, ICCCSP 2018. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICCCSP.2018.8452853

Mitchell, M. 1998. An Introduction to Genetic Algorithms (Complex Adaptive Systems), *MIT Press.*

Molchanov, P., Tyree, S., Karras, T., Aila, T. and Kautz, J. 2016. Pruning Convolutional Neural Networks for Resource Efficient Inference. 5th Int. Conf. Learn. Represent. ICLR 2017

Mureşan, H. and Oltean, M. 2018. Fruit recognition from images using deep learning. *Acta Univ. Sapientiae, Inform.* https://doi.org/10.2478/ausi-2018-0002

Nakahara, H. and Sasao, T. 2015. A deep convolutional neural network based on nested residue number system, 25th International Conference on Field Programmable Logic and Applications, FPL 2015. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/FPL.2015.7293933

Nassif, A.B., Shahin, I., Attili, I., Azzeh, M. and Shaalan, K. 2019. Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access 7*, 19143–19165. https://doi.org/10.1109/ACCESS.2019.2896880

Parnian, A.R. and Javidan, R. 2014. Autonomous Wheat Seed Type Classifier System General Terms. *Int. J. Comput. Appl.*

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q. and Kurakin, A. 2017. Large-Scale Evolution of Image Classifiers. 34th Int. Conf. Mach. Learn. ICML 2017 6, 4429–4446.

Rosebrock, A. 2017. Deep Learning for Computer Vision with Python, 1st Ed.

Ruvalcaba-Cardenas, A.D., Scoleri, T. and Day, G. 2019. Object Classification using Deep Learning on Extremely Low-Resolution Time-of-Flight Data, 2018 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2018. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/DICTA.2018.8615877

Shahriari, B., Swersky, K., Wang, Z., Adams, R.P. and Freitas, N. 2016. Taking the human out of the loop: A review of Bayesian optimization. *in Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, doi:10.1109/JPROC.2015.2494218.

Shakeel, M.F., Bajwa, N.A., Anwaar, A.M., Sohail, A. and Khan, A. 2019. Detecting

Driver Drowsiness in Real Time Through Deep Learning Based Object Detection, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, ss. 283–296. https://doi.org/10.1007/978-3-030-20521-8_24

Sharma, S. and Mehra, R. 2019. Implications of Pooling Strategies in convolutional neural networks: A Deep Insight. Found. *Comput. Decis. Sci.* https://doi.org/10.2478/fcds-2019-0016

Sifre, L. 2014. Rigid-Motion Scattering For Image Classification. PhD Thesis, Ecole Polytechnique, Palaiseau, https://doi.org/10.1.1.672.7091

Steinbrener, J., Posch, K. and Leitner, R. 2019. Hyperspectral fruit and vegetable classification using convolutional neural networks. *Comput. Electron. Agric.* 162, 364–372. https://doi.org/10.1016/j.compag.2019.04.019

Suganuma, M., Shirakawa, S. and Nagao, T. 2017. A genetic programming approach to designing convolutional neural network architectures, GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference. Association for Computing Machinery, Inc, ss. 497–504. https://doi.org/10.1145/3071178.3071229

Sun, Y., Xue, B., Zhang, M., Yen, G.G. and Lv, J. 2020. Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Trans. Cybern.* 50, 3840–3854. https://doi.org/10.1109/TCYB.2020.2983860

Sun, Y., Yen, G.G. and Yi, Z. 2017. Reference line-based Estimation of Distribution Algorithm for many-objective optimization. *Knowledge-Based Syst.* 132, 129–143. https://doi.org/10.1016/j.knosys.2017.06.021

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. 2015. Going deeper with convolutions, içinde: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy, C., Vanhoucke, V., Ioffe, S. and Shlens, J. 2016. Rethinking the Inception Architecture for Computer Vision. *arXiv:*1512.00567

Tan, M. and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 36th Int. Conf. Mach. Learn. ICML 2019 2019-June, 10691–10700.

Veeramani, B., Raymond, J.W. and Chanda, P. 2018. DeepSort: Deep convolutional networks for sorting haploid maize seeds. *BMC Bioinformatics*. https://doi.org/10.1186/s12859-018-2267-2

Wen, W., Wu, C., Wang, Y., Chen, Y. and Li, H. 2016. Learning structured sparsity in deep neural networks, Advances in Neural Information Processing Systems. Neural information processing systems foundation, ss. 2082–2090.

Xie, L., Yuille, A., 2017. Genetic CNN. *arXiv:*1703.01513

Zeiler, M.D. and Fergus, R. 2014. Visualizing and understanding convolutional networks, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag, ss. 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

Zhang, X., Liu, F., He, Y. and Li, X. 2012. Application of Hyperspectral Imaging and Chemometric Calibrations for Variety Discrimination of Maize Seeds. *Sensors* 12, 17234–17246. https://doi.org/10.3390/s121217234

Zhang, X., Zhou, X. and Lin, M. 2018. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *arXiv*:1707.01083

Zhong, Z., Yan, J., Wu, W., Shao, J. and Liu, C.L. 2017. Practical Block-wise Neural Network Architecture Generation. Proc. *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* 2423–2432.

Simonyan K. and Zisserman A. 2018. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, ICLR 2015, San Diego, CA, USA,

Zoph, B. and Le, Q. V, 2016. Neural Architecture Search with Reinforcement Learning. 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.

Zoph, B., Vasudevan, V., Shlens, J. and Le, Q. V. 2017. Learning Transferable Architectures for Scalable Image Recognition. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 8697–8710.

7. EKLER

```
% FLOPs VE TOPLAM PARAMETRENİN MATLAB İLE HESAPLANMASI


%KULLANICI AŞAĞIDAKİ PARAMETRELERİ TEK TEK TANIMLAR:
%varSize (giriş data boyutu)
%num_of_filt_conv1 (birinci evrişim katmanındaki filtre sayısı)
%num_of_filt_conv2 (ikinci evrişim katmanındaki filtre sayısı)
%K_c1 (birinci evrişim katmanındaki filtrelerin boyutu)
%S_c1 (birinci evrişim katmanındaki atlama miktarı)
%K_c2 (ikinci evrişim katmanındaki filtrelerin boyutu)
%S_c2 (ikinci evrişim katmanındaki atlama miktarı)
%Cin  (giriş data derinliği RGB için 3)
%K_p1 (birinci havuzlama katmanındaki filtrelerin boyutu)
%S_p1 (birinci havuzlama katmanındaki atlama miktarı)
%K_p2 (ikinci havuzlama katmanındaki filtrelerin boyutu)
%S_p2 (ikinci havuzlama katmanındaki atlama miktarı)


%%% Conv_1 Outputs %%%%%%%%%%%%%%%%
Cout_c1 = num_of_filt_conv1;
H_c1 = round(((varSize-K_c1)/S_c1)+1);
W_c1 = round(((varSize-K_c1)/S_c1)+1);
NFLOPs_c1 = (((K_c1*K_c1)*Cin)*Cout_c1 + Cout_c1)*(H_c1*W_c1);
NoP_c1 = ((K_c1*K_c1)*Cin +1)*num_of_filt_conv1

%%% Pool_1 Outputs %%%%%%%%%%%%%%%%
H_p1 = round((H_c1-K_p1)/S_p1);
W_p1 = H_p1;
Cout_p1 = Cout_c1;
NFLOPs_p1 = H_c1*W_c1*Cout_c1;

%%% Conv_2 Outputs %%%%%%%%%%%%%%%%
Cout_c2 = num_of_filt_conv2;
H_c2 = round(((H_p1-K_c2+1)/S_c2));
W_c2 = round(((W_p1-K_c2+1)/S_c2));
NFLOPs_c2 = (((K_c2*K_c2)*Cout_p1)*Cout_c2 + Cout_c2)*(H_c2*W_c2);
NoP_c2 = ((K_c2*K_c2)*Cout_p1 +1)*num_of_filt_conv2

%%% Pool_2 Outputs %%%%%%%%%%%%%%%%
H_p2 = round((H_c2-K_p2+1)/S_p2);
W_p2 = H_p2;
Cout_p2 = Cout_c2;
NFLOPs_p2 = H_c2*W_c2*Cout_c2;
activation_p2 = W_p2*W_p2*Cout_p2

%%% FC_1 paramters %%%%%%%%%%%%%%%%
```

```
NFOPs_fc1 = activation_p2*48 + 48
param_fc1 = (48*activation_p2+1)*48

%%% FC_2 paramters %%%%%%%%%%%%%%%%
NFLOPs_fc2 = 48*24 + 24
param_fc2 = (24*48+1)*24

%%% FC_3 paramters %%%%%%%%%%%%%%%%
NFLOPs_fc3 = 2*48 + 2
param_fc3 = (2*24+1)*2

%%% Softmax paramters %%%%%%%%%%%%%%%%
param_soft = (2*2+1)*2

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Total_param = NoP_c1 + NoP_c2 + param_fc1 + param_fc2 + param_fc3 + param_soft

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Total_NFLOPs= NFLOPs_c1 + NFLOPs_p1 + NFLOPs_c2 + NFLOPs_p2 +
NFLOPs_fc1 + NFLOPs_fc2 + NFLOPs_fc3
```

ÖZGEÇMİŞ

Hakan AKTAŞ

hakanaktas5151@gmail.com

ÖĞRENİM BİLGİLER

Doktora 2015-2020	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı, Antalya
Yüksek Lisans 2012-2015	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı, Antalya
Lisans 2004-2008	İstanbul Üniversitesi Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği

MESLEKİ VE İDARİ GÖREVLER

Araştırma Görevlisi 2012-2020	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı
Kurucu/Yönetici 2013- Devam Ediyor	Gitek Elektrik-Elektronik Mühendislik Ar. Ge. San. ve Tic. Ltd. Şti.
Araştırmacı 2012 Eylül – 2013 Ocak	Fraunhofer IIS Erkangen/Almanya

ESERLER

Uluslararası hakemli dergilerde yayımlanan makaleler

1- Aktaş H. and San B.T. 2019 "Landslide Susceptibility Mapping Using an Automatic Sampling Algorithm Based on Two Level Random Sampling", *COMPUTERS & GEOSCIENCES*, vol.1, no.1, pp.1-62

2- Bozkurt, Y., Mikail, N., Uluşar, Ü. D., Aktas, H. and Doğan, C. 2017 "Prediction Of Bodyweight Of Holstein And Brown-Swiss Male Cattle By Using Digital Images." *Scientific Papers. Series D. Animal Science(LX),* 196-201.

Ulusal bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler

1- Aktas H. ve Polat O. 2020 "Derin Öğrenme ile Tohumların Sınıflandırılmasında Renk ve Kenar Bilgisinin Etkisi" *6. Ulusal Yüksek Başarımlı Hesaplama Konferansı*, 8-9 Ekim 2020, Ankara Yıldırım Beyazıt Üniversitesi, Ankara

2- Aktas, H. 2018 "Implementation of GOST 28147-89 Encryption and Decryption Algorithm on FPGA." International Conference on Cyber Security and Computer Science(ICONCS 2018)

3- Bozkurt, Y., Mikail, N., Uluşar, Ü. D., Aktas, H. and Doğan C. 2017 "Prediction of Bodyweight of Holstein and Brown-Swiss Male Cattle By Using Digital Images." International Conference Agriculture for Life, Life for Agriculture, 59

4- Aktas, H., Sever, R. and Töreyin, B.U. 2016 "A two stage template matching algorithm and its Implementation on FPGA," IEEE Signal Processing and Communications Applications Conference, vol.34, pp. 2214–2217,

5- Sever, R. and Aktas, H. 2015 "A Fast Implementation of the Extended Kalman Filter on FPGA for UAV Applications" 4th International Conference On Materials Engineering For Advanced Technologies (ICMEAT 2015*)* Pages: 602-605

PATENTLER

1- Aktaş H, Akdeniz Üniversitesi, "OKUL SERVİSİ GÖRÜNTÜ İŞLEME İLE GÜVENLİK SİSTEMİ", TÜRKIYE, Patent, 2018/19777, Temmuz 2020

ÖDÜLLER VE BURSLAR

1- Aktaş H,Akdeniz Üniversitesi, "ISIF'20 Bronz Madalya , Türk Patent, Eylül 2020

2- 2211-C Yurt İçi Öncelikli Alanlar Doktora Bursu, TÜBİTAK, Mart 2019

3- Aktaş H, "Silikon Vadisi Seyahat Ödülü, TİM- TEB Ortaklığı, Nisan 2018

4- Aktaş H, "ATSO 2017 Ürün İnovasyon Ödülü, ATSO, Eylül 2017