EVRİŞİMLİ SİNİR AĞLARINDA HİPER PARAMETRELERİN ETKİSİNİN İNCELENMESİ

ANALYSIS OF THE EFFECTS OF HYPERPARAMETERS IN CONVOLUTIONAL NEURAL NETWORKS

FERHAT KURT

PROF. DR. MEHMET ÖNDER EFE

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Mühendisliği Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

2018

FERHAT KURT' un hazırladığı "**Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi**" adlı bu çalışma aşağıdaki jüri tarafından **BİLGİSAYAR MÜHENSİLİĞİ ANABİLİM DALI'** nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Dr. Öğr. Üyesi İ. Aykut ERDEM
Başkan .....

Prof. Dr. Mehmet Önder EFE
Danışman .....

Dr. Öğr. Üyesi Adnan ÖZSOY
Üye .....

Dr. Öğr. Üyesi Tayfun KÜÇÜKYILMAZ
Üye .....

Dr. Öğr. Üyesi A. Murat ÖZBAYOĞLU
Üye .....

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak onaylanmıştır.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Fen Bilimleri Enstitüsü Müdürü

YAYINLAMA VE FİKRİ MÜLKİYET HAKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

☒ **Tezimin/Raporumun tamamı dünya çapında erişime açılabilir ve bir kısmı veya tamamının fotokopisi alınabilir.**
(Bu seçenekle teziniz arama motorlarında indekslenebilecek, daha sonra tezinizin erişim statüsünün değiştirilmesini talep etseniz ve kütüphane bu talebinizi yerine getirse bile, tezinin arama motorlarının önbelleklerinde kalmaya devam edebilecektir.)

☐ **Tezimin/Raporumun ..... tarihine kadar erişime açılmasını ve fotokopi alınmasını (İç Kapak, Özet, İçindekiler ve Kaynakça hariç) istemiyorum.**
(Bu sürenin sonunda uzatma için başvuruda bulunmadığım taktirde, tezimin/raporumun tamamı her yerden erişime açılabilir, kaynak gösterilmek şartıyla bir kısmı ve ya tamamının fotokopisi alınabilir)

☐ **Tezimin/Raporumun ..... tarihine kadar erişime açılmasını istemiyorum, ancak kaynak gösterilmek şartıyla bir kısmı veya tamamının fotokopisinin alınmasını onaylıyorum.**

☐ **Serbest Seçenek/Yazarın Seçimi**

29 / 05 / 2018

Ferhat Kurt

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

29/05/2018

FERHAT KURT

ÖZET

EVRİŞİMLİ SİNİR AĞLARINDA HİPERPARAMETRELERİN ETKİSİNİN İNCELENMESİ

FERHAT KURT
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Prof. Dr. MEHMET ÖNDER EFE
Mayıs 2018, 114 sayfa

Bu çalışmada, IMAGE-NET yarışması kapsamında 2012 yılından günümüze resim tanımada popülerlik kazanan evrişimli sinir ağlarının yapısı, çalışma sistemi ve ağı oluşturan hiper parametreler konusunda literatür incelemesi ve deneysel çalışma yapılmıştır. Çalışmada, ILSVRC2012 eğitim verisinden 50 sınıf ve her sınıfa ait 600 örneklemden oluşan bir veri seti ile evrişimli sinir ağı hiper parametreleri için farklı seçenek değerleri belirlenmiş ve süper bilgisayarlarında dahil edildiği derin öğrenme istemcisi, parametre ve değerlendirme sunucusu mimari yapısı üzerinde eğitimler yapılmıştır. Yapılan eğitimler sonucunda şekil ve çizelgeler üzerinden model başarımları değerlendirilerek yeni hiper parametre değerleri oluşturulmuş ve ilave eğitimler yapılmıştır. Toplamda yapılan 410 farklı eğitim sonucunda veri setine ön işleme yapılması, aktivasyon fonksiyonuna göre öğrenme katsayısı seçimi, paket normalizasyonu ve seyreltme işleminin kullanılmasının model başarımını arttırdığı tespit edilmiştir.

Anahtar Kelimeler: Yapay Sinir Ağları, Evrişimli Sinir Ağları, Derin Öğrenme, Hiper Parametre, Eniyileme

ABSTRACT

INVESTIGATION OF THE EFFECTS OF HYPERPARAMETERS IN CONVOLUTIONAL NEURAL NETWORKS

Ferhat KURT
Master of Science, Department of Computer Engineering
Supervisor: Prof. Dr. Mehmet Önder EFE
May 2018, 114 pages

In this study, literature review and experimental study were carried out on the hyperparameters constituting the structure, working system and network of the irregular neural networks that gained popularity in the definition of the day-to-day picture in 2012 within the scope of IMAGE-NET contest. In the study, ILSVRC2012 training dataset consisting of 50 classes and 600 samples and different option values for convolutional neural network hyperparameters were determined and trainings were conducted on the learning structure of the deep learning client, parameter and evaluation server included in the supercomputers. Consequently, these trainings, the model performances were evaluated through diagrams and charts and new hyper parameter values were created and additional trainings were made. As a result of 410 separate trainings in total, it has been determined that preprocessing of data sets, learning rate selection in accordance with optimizer, packet normalization and use of dropout process, increases model performance.

Keywords: Artificial Neural Networks, Convolutional Neural Networks, Deep Learning, Hyperparameters, Optimization

TEŞEKKÜR

Tez çalışması süresince kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam Prof. Dr. Mehmet Önder EFE'ye teşekkürü bir borç bilirim.

Çalışmalarım boyunca manevi desteğiyle beni hiçbir zaman yalnız bırakmayan eşim Neslihan KURT'a sonsuz teşekkür ederim.

Çalışmanın gerçekleşmesinde desteklerini esirgemeyen iş arkadaşlarıma ve eğitim altyapısının oluşturulmasına katkı sağlayan NVIDIA, PNY ve Microsoft firmalarına teşekkürler ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER

Sayfa

ÇİZELGELER

Sayfa

KISALTMALAR

Adagrad	Adaptive Gradient (Adaptif Eğim)
Adam	Adaptive Moment Estimation (Adaptif Moment Tahmin)
AF	Aktivasyon Fonksiyonu
CPU	Merkezi İşlem Birimi
Eİ	Eniyileyici
ESA	Evrişimli Sinir Ağı
FB	Filtre Boyutu
FS	Filtre Sayısı
GPU	Grafik İşleme Birimi
ILSVRC	Large Scale Visual Recognition Challenge Büyük Ölçekli Görsel Tanıma Yarışması
LRN	Local Response Normalization Lokal Cevap Normalizasyonu
Nadam	Nesterov Accelerated Adaptive Moment Estimation (Nesterov Hızlandırılmış Adaptif Moment Tahmin)
ÖK	Öğrenme Katsayısı
PN	Paket Normalizasyonu
R-CNN	Region-Based Convolutional Neural Networks (Bölge-Bazlı Evrişimli Sinir Ağları)
RGB	Red, Green, Blue (Kırmızı, Yeşil, Mavi renk uzayı)
RMSprop	Root Mean Square Propagation (Kare Ortalamalarının Karekökü Yayılımı)
SGD	Stochastic Gradient Descent (Olasalıksal Eğim Düşümü)
Sİ	Seyreltme İşlemi
SO	Seyreltme Oranı
TB	Tam Bağlaşımlı
TOP-1	En yüksek olasılığa sahip ilk çıkarım sonucunun veri etiketiyle aynı olma durumu
TOP-5	En yüksek olasılığa sahip ilk 5 çıkarım sonucu arasında veri etiketinin olması durumu
WRN	Wide Resiudal Network (Geniş Kalıntı Ağ)

1. GİRİŞ

Evrişimli sinir ağları, 2012 yılında ImageNet Büyük Ölçekli Görsel Tanıma Yarışmasında (ImageNet Large Scale Visual Recognition Challenge – ILSVRC) birinci olan AlexNet algoritmasında kullanıldıktan sonra, popülerlik kazanmaya başlamıştır. 2012 yılından günümüze evrişimli sinir ağları resim sınıflandırmada Çizelge 1.1'de görüldüğü gibi her sene başarımı daha üst seviyeye taşımıştır. Resim tanıma başarımının, sınırlandırılmış alanlarda insan algısına yakın doğruluğa ulaşması ile birlikte evrişimli sinir ağları ile eğitilen modeller günlük hayatımızda birçok alanda yer almaya başlamıştır.

Çizelge 1.1. 2012-2017 yılları arasında dereceye giren ağların hata-doğruluk tablosu

Yıl	**Algoritma**	**Başarım (Top-5 Doğruluk)**
2012	AlexNet	%84,7
2013	ZFNet	%85,2
2014	**VGGNet**	**%92,7**
2014	GoogLeNet	%93,33
2015	ResNet	%96,43
2016	Trimp	%97.01
2017	SENet	%97,75

Evrişimli sinir ağlarında, standart model eğitim sürecinde veriden eğitim ile doğrudan öğrenilen model parametreleri olduğu gibi, veriden doğrudan öğrenilemeyen ve önceden tasarımcı tarafından tanımlanması gereken parametreler bulunmaktadır. Tasarımcı tarafından tanımlanan sınıf ve örneklem sayısı, filtre sayısı, filtre boyutu, aktivasyon fonksiyonu, eniyileme, öğrenme katsayısı, paket boyutu ve görüntü ön işleme gibi tercihlere hiper parametre denilmektedir [1, 2].

Uygun hiper parametre değerlerine ulaşmak için kullanılabilecek dört temel yöntem bulunmaktadır [3].

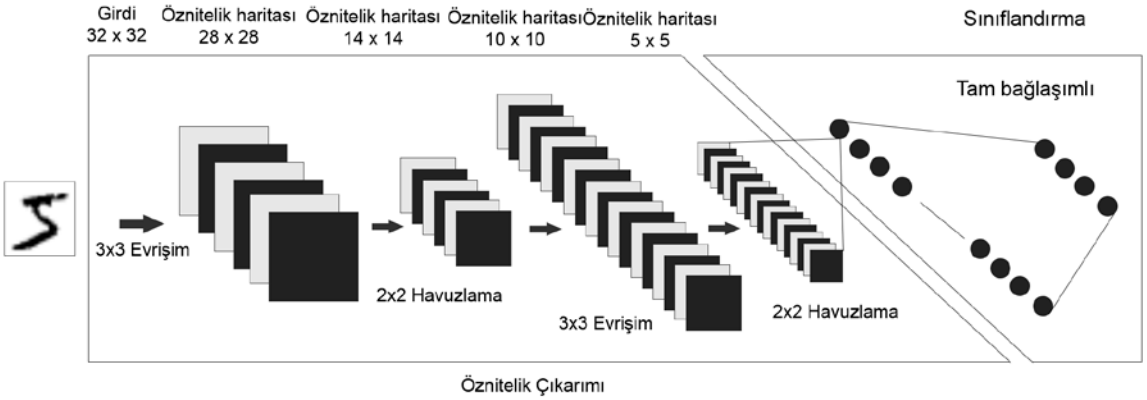
a. **Elle Arama**: Bu yöntem, daha önceki bilgiler de kullanılarak en iyi hiper parametrelerin tahmin edilmesi ve denenmesi ile sorunu çözmeyi hedefler. Sonuca bağlı olarak hiper parametreler değiştirilir ve sonuç gözlenir. En iyi sonuç elde edilene kadar bu işlem devam eder.
b. **Izgara Arama**: Bu arama türünde, her bir konfigürasyon parametresi için bir değerler kümesi seçilir ve bu değerlerin tüm kombinasyonları değerlendirilir.

Nihai sonuç olarak bu kombinasyonların en iyisini döndürür. Bu işlem iç içe döngüler şeklinde basitçe uygulanabilir [4].

c. **Rastgele Arama**: Bu arama yöntemi ise rastgele parametreler oluşturur ve bu parametlerin sonucunu, elde ettiği en iyi sonuç ile karşılaştırılır. Eğer en iyi sonuçtan daha iyi bir sonuç elde edildiyse, en iyi parametreler, en iyi sonucu veren parametreler ile değiştirilir. Bu işlem, durma koşulu sağlanana kadar devam eder [4].
d. **Bayesian Eniyileme**: Bu yöntem en iyi kombinasyonu bulmak için makine öğrenmesi algoritması kullanmaktadır. Gaussian işlem tekniği ile daha önceki sonuçlara bakılarak, hiper parametreler için yeni değerler tahmin edilir. Durma koşulu sağlanana kadar bu işlem devam eder. En iyi sonuç bulunduğu zaman kaydedilir ve daha iyi bir sonuç elde edilene kadar bu sonuç tutulur [5].

Evrişimli sinir ağları yapısını oluşturan filtre sayısı, filtre boyutu, aktivasyon fonksiyonu, en iyileme, görüntü ön işleme gibi birçok hiper parametrenin eğitim başarımına etkisini gözlemleyebilmek için 144 farklı eğitim işi planlanmıştır. Bu 144 farklı eğitim işi paket boyutu 128 ve 256 olacak şekilde toplamda 288 iş olarak eğitilmiş ve çıkan sonuçların analizleri yapılarak 120 farklı eğitim işi daha yapılmıştır. Planlanan eğitimlerin çok olması nedeniyle süper bilgisayarların da dahil edildiği 2 adet DGX-1 ve 40 adet K80 GPU'lardan oluşturulan derin öğrenme eğitim istemcileri, parametre sunucusu ve değerlendirme sunucusundan oluşan bir mimari yapı tesis edilmiş ve uygulama altyapısı kurulmuştur. Çalışmada kullanılan veri seti, ILSVRC2012 veri setinden oluşturulmuştur. ILSVRC2012, 1000 sınıf ve her sınıfta 1300 resimden oluşmaktadır. Bu çalışmada ILSVRC2012 veri setinden 50 sınıf ve bu sınıflara ait 600 resim rastgele seçilmiştir.

Evrişimli sinir ağı mimari modeli, Şekil 1.1’de görüldüğü gibi öznitelik çıkarma (feature extraction) ve sınıflandırma (classification) olmak üzere iki bölümden oluşmaktadır [6].



Şekil 1.1. Evrişimli sinir ağı mimari modeli

2. bölümde literatür taraması kapsamında evrişimli sinir ağının nasıl çalıştığının anlaşılabilmesi için sınıflandırma ve evrişim kısmına yer verilmiştir. Sınıflandırma kısmında sinir ağının yapısı, ileri ve geriye yayılım; evrişimli sinir ağları kısmında ise evrişim katmanı, havuzlama katmanı, hiper parametreler, IMAGE-NET sınıflandırma kategorisi kapsamında 2012-2017 yılları arasında dereceye giren ağ yapıları ve yaygın olarak kullanılan veri setleri ile ilgili bilgi verilmiştir.

3. bölümde evrişimli sinir ağı eniyilemeye yönelik farklı parametre değerleri ile oluşturulan 144 farklı işin eğitim sonuçları şekil ve çizelgeler üzerinden anlatılmış ve yorumlanmıştır.

4. bölümde eğitim sonrasında en başarılı olduğu değerlendirilen bir ağın TOP-1 ve TOP-5 değerleri hata matrisi üzerinden gösterilmiştir. Modelin çıkarım hızı eniyilemesine yönelik olarak tasarlanan NVIDIA TensorRT yapısı hakkında bilgi verilmiştir.

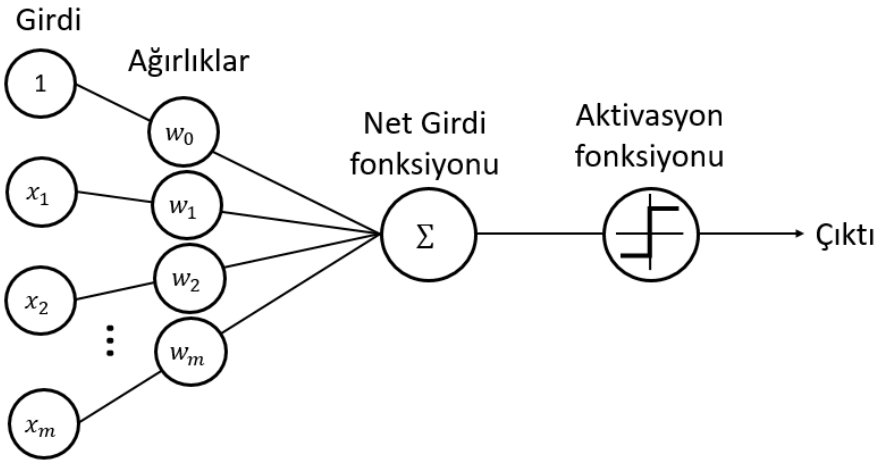
5. bölümde yapılan çalışmanın sonuç değerlendirmesi yapılmıştır. Bu çalışmada olabildiğince Türkçe terminoloji kullanılmıştır. Çevrilen İngilizce kelimelerin Türkçe karşılıkları EK-5'te yer almaktadır.

2. LİTERATÜR TARAMASI

2.1. Sinir Ağı / Çok Katmanlı Perseptron

2.1.1. Perseptron

Perseptron, ilk olarak 1958 yılında F. Rosenblatt tarafından tanıtılmıştır. Perseptron, insanın veriyi duyuları (görme, duyma, işitme gibi) aracılığıyla nasıl algıladığı, veriyi nasıl öğrendiği, bilginin nasıl depolanıp hatırlandığı ve bu sürecin matematiksel olarak nasıl ifade edilebileceği gibi kuramsal sorular sayesinde ortaya çıkmıştır. Perseptron Şekil 2.1'de gösterilen yapıya sahiptir [7].



Şekil 2.1. Rosenblatt'a göre perseptron modeli

Tek perseptron giriş, hesaplama ve çıkış katmanı olmak üzere üç parçadan oluşmaktadır. Perseptron bir veya daha fazla girdi alabilir ve hesaplama katmanındaki sonuca göre bir çıktı üretebilir. Giriş katmanında, Rosenblatt her katmanı $x1, x2, \ldots xn$ olarak ve her ağırlığı $w1, w2, \ldots, wn$ olarak tanımlamıştır. Bu değerler birer reel sayı olarak tanımlanmıştır. Ağırlık, perseptron modeline belirli bir hedefe yönelik çıktı üretme yeteneği sağladığı için çok önemlidir.

Perseptron, sıfır ve birlerden oluşan bir değer üretmektedir. Girdi katmanındaki değerler ile ilgili ağırlık değerlerinin çarpımının toplamına kutuplama değeri eklendiğinde, sonuç eşik değerinden daha büyük ise perseptron 1, değilse 0 olarak çıktı üretecektir. Örneğin, perseptron girdisinin toplam değerinin 0.42 ve eşik değerinin 0 olduğunu varsayılırsa, toplam değer eşik değerinden büyük olduğu için

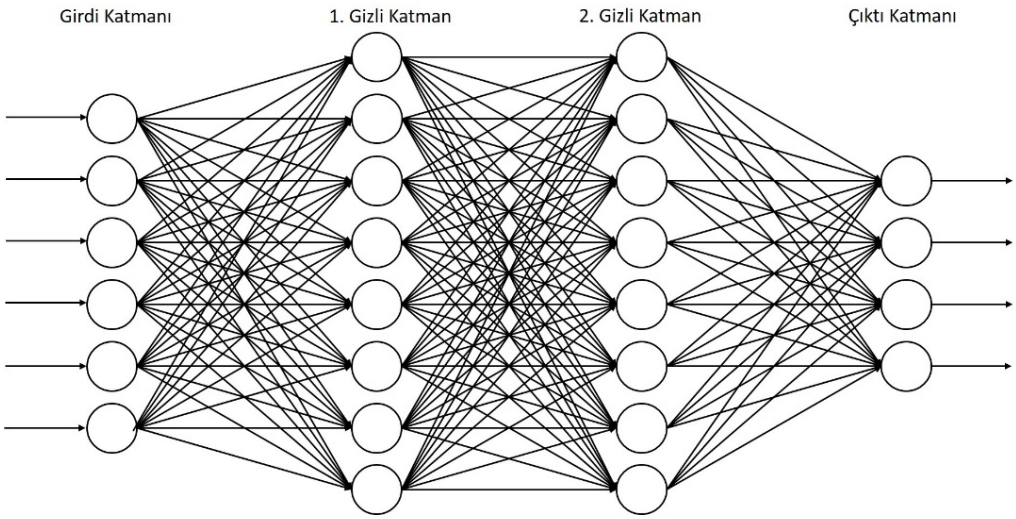
bu nöron ateşlenecek ve çok katmanlı bir ağ yapısında kendinden sonra gelen diğer perseptronlara bilgiyi aktaracaktır. Perseptronun çıkış üreten fonksiyonu Denklem (2.1)'de gösterilmiştir.

$$x = \sum_j w_j x_j + b$$
$$y = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (2.1)$$

Denklem (2.1)'de çıktı (y), girdi katmanı (X_j), ağırlıklar (W_j), kutuplama değeri (b) olarak gösterilmiştir.

2.1.2. İleriye Yayılım Modeli

Tek perseptron kompleks problemlerin çözümünde yeterli olmamaktadır. Bu nedenle Şekil 2.2'de gösterildiği gibi çok katmanlı perseptron yapısı ya da daha genel adıyla sinir ağları oluşturulmuştur [8].



Şekil 2.2. Gizli katmanlara sahip bir yapay sinir ağı

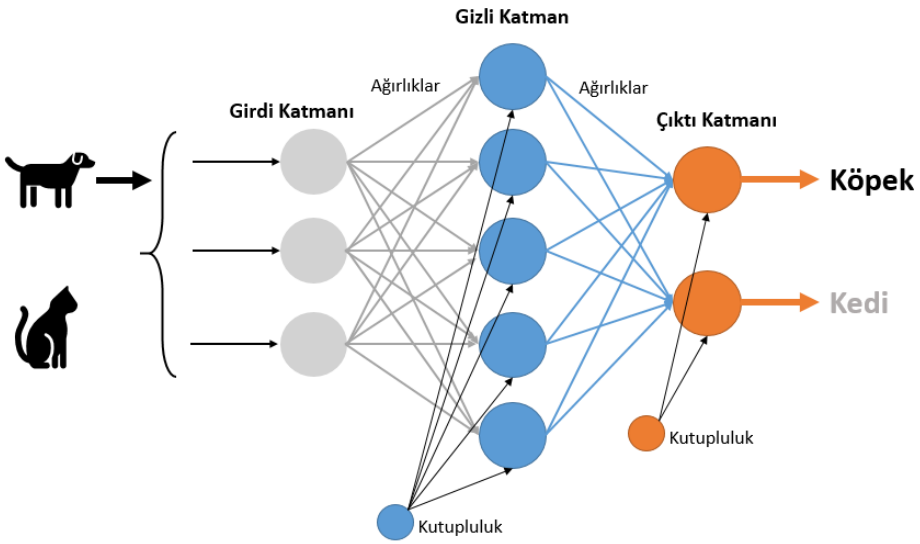
Yapay sinir ağı giriş katmanı, gizli katmanlar ve çıkış katmanı olmak üzere üç bölümden oluşmaktadır. Giriş katmanı ağa verinin Gizli katman terimi, sadece giriş katmanından sonra gelen katmanları ifade eder. Gizli katmanlarla ilgili olarak, Hornik [8], en az bir gizli katmana sahip sinir ağlarının evrensel yaklaşıklayıcı (universal approximators) olarak sınıflandırılabileceğini ve kabul edilebilir bir doğruluk ile

herhangi dereceden bir fonksiyonu tahmin edebileceğini; başarısız bir uygulamanın yetersiz öğrenmeden, yetersiz gizli katmanlardan veya girdi ile çıktı arasında belirleyici bir ilişkinin bulunmamasından kaynaklandığını belirtmiştir.

Girdi katmanındaki her nöron girdiye ait vektördeki değerlerden birini değer olarak alır. Bu girdi değeri siyah-beyaz el yazısı görüntüsü, bir sayı veya RGB formatındaki bir resmin piksel değerleri olabilir. Daha sonra girdi katmanındaki her nöronun kendinden sonra gelen gizli katmandaki her nörona olan bağlantısı için bir ağırlık değeri atanır. Atanan ağırlık değerleri, ağın performans ve doğruluğunu etkilediği için dikkatli seçilmelidir. Ağırlık atama işleminden sonra yapılan işlem, ağ içerisinde bulunan her bir girdi ile ilgili ağırlık değerinin çarpılması ve kutuplama değerinin eklenerek aktivasyon fonksiyonundan geçirilmesidir. Yapılan işlem Denklem (2.2)'de gösterilmektedir. Bir katmandaki değerin bir sonraki katmana aktarılmasından dolayı bu ağa ileri yayılımlı ağ denir.

$$v_j = \sum_i w_{ij} x_i + b_l$$
$$y_j = \sigma(v_j) \tag{2.2}$$

İleri yayılımlı bir ağın son katmanındaki çıktı, ağın sonucudur. Çıktı katmanında bulunan nöron sayısı Şekil 2.3'te gösterildiği gibi ağın amacına yönelik değişkenlik gösterebilmektedir.



Şekil 2.3. Sınıflandırma örneği

Çıktı değeri hesaplandıktan sonra bu sonucun, atanan ağırlık değerleri ve kutuplama değeri için ne kadar doğrulukta olduğu daha önceden etiketlenmiş eğitim seti ile belirlemesi gerekmektedir. Bu süreç hata sinyali bulunarak yapılabilir. Hata sinyali; çıkış nöronundan çıkan ve ağ boyunca geriye doğru (katmandan katmana) yayılan sinyal olarak tanımlanabilir [9]. Hata sinyali olarak adlandırılmasının sebebi, ağdaki her nörona bağlı bir hesaplama, hata bağımlı bir fonksiyon içerir. Hata sinyali ile birlikte maliyet fonksiyonu hesaplanmasının önemi ve görevi ilerleyen kısımlarda tartışılacaktır. İleri yönde yayılım Denklem (2.3) ile açıklanabilir:

$$y_j^{(l)}(n) = \sigma\left(\sum_i w_{ji}^{(l)}(n) Y_i^{(l-1)}(n)\right) \tag{2.3}$$

l-1'inci katman tarafından beslenen *l*'inci katmandaki *j* nöronuna ait sinaptik ağırlık değeri $w_{ji}^{(l)}(n)$, bir önceki katmanın çıktısı $Y_i^{(l-1)}(n)$, çıkış katmanındaki nöronun değeri ise $y_j^{(l)}(n)$ ile belirtilmiştir. Eğer nöron *j* çıktı katmanında ve L ağın derinliğini belirtiyor ise ifade Denklem (2.4)'de olduğu gibi tanımlanabilir:

$$y_j^{(L)}(n) = o_j(n) \tag{2.4}$$

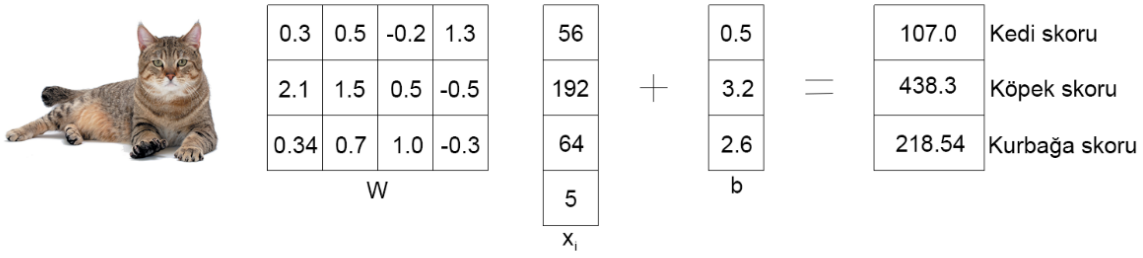
$o_j(n)$ j'inci nöronun çıktısını belirtmektedir.

Hata sinyali ise, $d_j(n)$ *j*'inci nöronun beklenen çıktısı olmak üzere Denklem (2.5)'de olduğu gibi hesaplanabilir:

$$e_j(n) = d_j(n) - o_j(n) \tag{2.5}$$

2.1.3. Maliyet Fonksiyonu

Bu fonksiyon, ağın ürettiği sonuç ile gerçek değer arasındaki farkın değerlendirmesi olarak tanımlanabilir [10]. Şekil 2.4'te girdisi resim, çıktısı 3 sınıf olan bir ağ yapısı gösterilmektedir. Ağa, x_i ile gösterilen, RGB piksel değerleri tek sütun haline getirilmiş kedi resmi verilmektedir. W ve b ile gösterilen matris değerleri başlangıçta rastgele oluşturulmaktadır. Girdi (x_i) ile W matrisinin çarpılması ve kutuplama değerinin (b) eklenmesiyle skor üretilmektedir. Şekil 2.4'te görülebileceği gibi 'kedi' için üretilen skor −96.8, aynı zamanda 'köpek' için üretilen skor ise 437.9'dur. Açıkça anlaşılabileceği gibi yanlış olmasına rağmen ağı girdi olarak verilen resmin köpek olduğu tahmin etmektedir.



Şekil 2.4. Bir ağın resme bağlı olarak bir sonuç üretmesi

Modeli geliştirmek için veriyi çoğaltmak doğruluğu beklediğimizden çok daha az arttıracaktır. Çünkü bu ağırlık parametreleriyle model belirli bir çıkış sınıfına doğru eğimli hale gelmiştir. Sonuç olarak istenen sonuç elde edilene kadar ve hata bu anlamda düşene kadar ağırlıkların her döngüde tekrar hesaplanması gerekmektedir. Maliyet fonksiyonu Denklem (2.6) ile hesaplanabilir [9]:

$$\mathfrak{E}(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{2.6}$$

Tüm ağın kaybı $\mathfrak{E}(n)$ ile gösterilmiştir. C, çıkış katmanındaki tüm nöronları temsil etmektedir. e_j^2 ise olasılıksal eğim düşümü algoritmasına göre hesaplanmış hata sinyalini ifade etmektedir. Eğitim için N adet veri var ise, ortalama maliyet Denklem (2.7) ile hesaplanabilir:

$$\begin{aligned}\mathfrak{E}_{av}(N) &= \frac{1}{N}\sum_{n=1}^{N}\mathfrak{E}(n) \\ &= \frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C} e_j^2(n)\end{aligned} \tag{2.7}$$

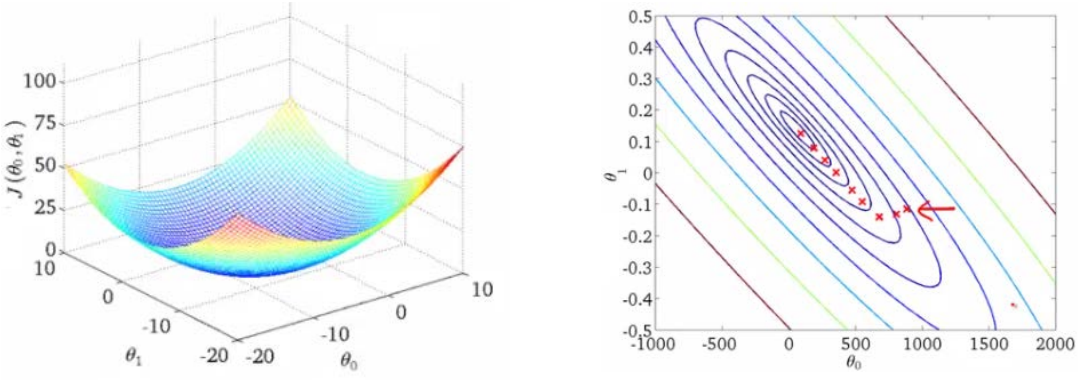
Maliyet fonksiyonunu hesaplamak için kullanılan matematiksel fonksiyon her zaman Denklem (2.7)'de belirtilen ifade olmak zorunda değildir.

2.1.4. Geri Yönde Yayılım

Maliyet fonksiyonunu kullanılarak her çıktı katmanındaki nöronun hatası hesaplanabilir. Ayrı ayrı hesaplanan hata değerleri toplanarak toplam hata elde edilir. Geri yönde yayılım algoritmasının amacı maliyet fonksiyonunu kullanarak bütün katmanlardaki ağırlık değerlerini yeniden hesaplamak ve bu sayede hatayı enküçüklemektedir. Bu süreç (ileri ve geri yönde yayılım), maliyet fonksiyonundaki düşüş artık önemsenmeyecek noktaya gelinceye kadar devam eder.

Geri yönde yayılımda eğim düşüm fonksiyonu, maliyet fonksiyonunun hangi yönde değişmesi gerektiğini gösterir.

Konveks fonksiyonunun yüzey grafiği Şekil 2.5 (a)'da 3 boyutlu, Şekil 2.5 (b)'de ise 2 boyutlu olarak gösterilmektedir [11]. Hedef global minimum olarak adlandırılan en düşük noktaya ulaşabilmektir. Bunu gerçekleştirebilmek için eğim düşümü algoritması ile global minimum noktasına ulaşana kadar her yerel eğimin türevinin hesaplanması gerekmektedir.

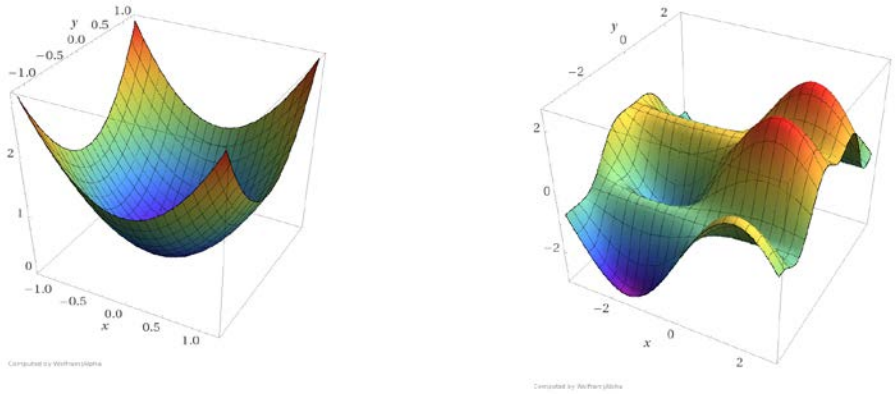


(a) (b)

Şekil 2.5. Konveks fonksiyonun yüzey grafiği [11]

Gerçekte yapay sinir ağı konveks olmayan bir fonksiyondur. Konveks olmayan fonksiyonun birden fazla yerel minimum noktası bulunmaktadır. Şekil 2.6 (a) konveks ve (b) konveks olmayan fonksiyonların farkını göstermektedir [12]. Konveks bir fonksiyonda herhangi bir başlangıç değeri ile global minimum noktasına erişmek kolay iken konveks olmayan bir fonksiyonda bu işlem daha zordur. Global minimum noktasına erişmek için ağda kullanılacak hiper parametrelerin uygun şekilde ayarlanması gerekmektedir. Bunlar başlangıç ağırlıkları, öğrenme katsayısı, döngü sayısı ve eğim düşümü türleridir.

Bu bölümün ilerleyen kısımlarında eğim düşümü türleri tartışılacaktır. Geri kalan hiper parametreler daha sonra ele alınacaktır.



(a) (b)

Şekil 2.6. Konveks ve konveks olmayan fonksiyonlar [12]

Basit bir sinir ağının kaybı Denklem (2.8) ile ifade edilebilir [13]:

$$E = (O_{çıkış(i)} - y_{(i)})^2 \tag{2.8}$$

$O_{çıkış(i)}$ değeri, bir önceki nöronun değeri ile ağırlık parametresinin çarpımı ve kutuplama değerinin toplamının bir aktivasyon fonksiyonundan geçirilmiş halidir. Bu örnek için aktivasyon fonksiyonu Denklem (2.9)'da belirtildiği gibi sigmoid kullanılmıştır [13]:

$$\begin{aligned} O_{giriş(i)} &= w^{(L)} h_{çıkış(i)}^{(L-1)} + b^{(L)} \\ O_{çıkış(i)} &= \sigma(O_{giriş(i)}) \end{aligned} \tag{2.9}$$

$a^{(L-1)}$ değeri, kendinden önceki nörona yani $a^{(L-2)}$'ye bağlıdır.

Hesaplanmak istenen değer maliyet fonksiyonunun ağırlık değişiminden ne kadar etkilendiğini bulmak için maliyet fonksiyonunun ağırlık değerine göre türevini ($\frac{\partial E}{\partial w^{(L)}}$) hesaplanmalıdır.

Türev kurallarından Denklem (2.10)'deki zincir kuralını uygulanacaktır:

$$\frac{\partial E}{\partial w^{(L)}} = \frac{\partial E}{\partial O_{çıkış(i)}} \frac{\partial O_{çıkış(i)}}{\partial O_{giriş(i)}} \frac{\partial O_{giriş(i)}}{\partial w^{(L)}} \tag{2.10}$$

Yukarıda belirtilen eşitliklerin Denklem (2.11) ayrı ayrı türevi alınarak eşitlik elde edilir [13].

$$\frac{\partial E}{\partial O_{çıkış(i)}} = 2\left(O_{çıkış(i)} - y_{(i)}\right) \tag{2.11}$$

$$\frac{\partial O_{\text{çıkış}(i)}}{\partial O_{\text{giriş}(i)}} = \sigma'\left(O_{\text{giriş}(i)}\right)$$

$$\frac{\partial O_{\text{giriş}(i)}}{\partial w^{(L)}} = h_{\text{çıkış}(i)}^{(L-1)}$$

$$\frac{\partial E}{\partial w^{(L)}} = 2\left(O_{\text{çıkış}(i)} - y_{(i)}\right)\sigma'\left(O_{\text{giriş}(i)}\right)h_{\text{çıkış}(i)}^{(L-1)}$$

Bu ifade maliyet fonksiyonunun ağırlık değişimlerinden nasıl etkileneceğini göstermektedir. Hedef, maliyet fonksiyonunu minimumuna ulaşmak olduğu için denklemde değerlerin değişiminin maliyet fonksiyonunu nasıl etkileyeceği anlaşılabilmektedir. Aynı ifade kutuplama değerleri için de kullanılmaktadır. Olabildiğince basitleştirilmiş olan bu yöntem, yani zincir kuralı bütün ağa uygulanabilir. Daha karmaşık bir yapay sinir ağına uygulandığında bu değişim, Denklem (2.12) ile ifade edilebilir:

$$\frac{\partial E}{\partial w^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial E_k}{\partial w^{(L)}} \tag{2.12}$$

Ağırlık değişimleri Denklem (2.13) ile hesaplanabilir:

$$\Delta w_{ij}^{k} = -\eta\frac{\partial E}{\partial w_{ij}^{k}} \tag{2.13}$$

Burada η öğrenme katsayısıdır.

2.1.5. Hiper Parametreler

Uygun hiper parametre seçimi yapay sinir ağı eğitiminin daha az maliyetle ve yüksek başarımla gerçekleşmesini sağlar. Yapay sinir ağlarında kullanılan hiper parametreler aşağıda detaylandırıldığı gibi altı grupta toplanmaktadır.

2.1.5.1. Öğrenme Katsayısı

Öğrenme katsayısı, eğim düşümü algoritmalarında kullanılan bir katsayıdır. Hata düzeltme katsayısı olarak da bilinir. Öğrenme katsayısı, eğim düşümü algoritmalarının yakınsamasını sağlamaktadır. Bu katsayı, fazla büyük olduğunda hedefe ulaşamayabilir; global minimum noktasının etrafında dolaşabilir ya da ıraksayabilir. Aynı zamanda katsayı çok küçük seçildiğinde, algoritma her döngüde çok küçük adımlarla ilerleyeceği için yakınsama çok uzun sürecektir [9].

2.1.5.2. Aktivasyon Fonksiyonu

Perseptronda, toplam değer birim basamak fonksiyonundan geçirilerek çıktı üretilmekteydi. Zaman içerisinde yapay sinir ağlarının kompleks problemleri çözmesi için Çizelge 2.1'de belirtilen aktivasyon fonksiyonları geliştirilmiştir.

Çizelge 2.1. Aktivasyon fonksiyonları

İsim	Fonksiyon
Sigmoid [10]	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh [10]	$tanh(x) = 2\sigma(2x) - 1$
ReLU [10]	$f(x) = max(0, x)$
Leaky ReLU [10]	$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$
ELU [14]	$f(x) = \begin{cases} x, & if\ x > 0 \\ \alpha(e^x - 1), & if\ x \leq 0 \end{cases}$
SELU [15]	$selu(x) = \lambda \begin{cases} x, & x > 0 \\ ae^x - a, & x \leq 0 \end{cases}$
PReLU [16]	$f(x) = \begin{cases} x, & if\ x > 0 \\ \alpha x, & if\ x \leq 0 \end{cases}$
Softmax [10]	$f(x) = \frac{e^{z_j}}{\sum_k e^{z_k}}$

2.1.5.3. Eniyileme Türleri

Eniyileme, ağın ürettiği çıkış değeri ile gerçek değer arasında farkı enküçüklemek için kullanılan yöntemdir. Sinir ağlarını eniyilemek için en popüler algoritmalardan biri eğim düşümüdür. Verinin büyüklüğüne bağlı olarak kullanabilecek üç adet eğim düşümü türü bulunmaktadır [17]. Bu eğim düşümü türleri ile birlikte çeşitli eniyileme algoritmaları kullanılabilir.

2.1.5.3.1. Eğim İnişi Türleri

2.1.5.3.1.1. Paket Eğim Düşümü (Batch Gradient Descent)

Standart eğim düşümü olarak da bilinen paket eğim düşümü tüm veri seti üzerinde eğim düşümünü hesaplamaktadır. Tüm veri setini tek seferde kullanmasından dolayı bellek yetersizliği sorunları ortaya çıkmaktadır. Hesaplama maliyeti yönünden yüksek de olsa diğer eğim düşüm türlerine göre daha kararlıdır. Maliyet fonksiyonunun $\boldsymbol{\theta}$ parametresine bağlı olarak eğiminin hesaplaması Denklem (2.14)'de gösterilmiştir.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t) \quad (2.14)$$

$\theta \in \mathbb{R}^d$ model parametreleri, η öğrenme katsayısı ve $\nabla_\theta J(\theta)$ parametrelere bağlı hedef fonksiyon eğimini belirtmektedir.

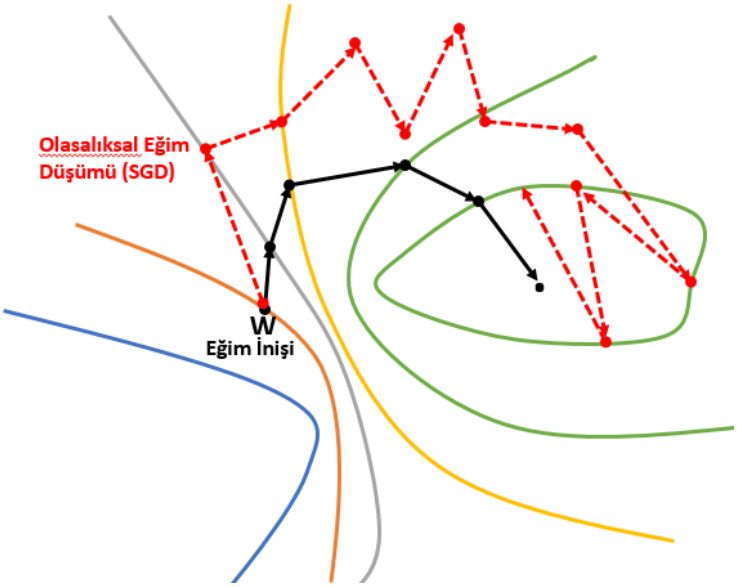
Tek seferde tüm veri seti ele alınarak eğimler hesaplandığı için bu yöntem oldukça yavaştır ve hafızaya sığmayacak büyüklükteki veri setleri için uyarlanması güçtür. Ayrıca bu yöntem modelimizi online olarak uygulanmaya elverişli değildir. Tüm bu olumsuzluklarına rağmen bu yöntem içbükey fonksiyonlar için global minimum noktasını, içbükey olmayan fonksiyonlar için yerel minimum noktasını bulmayı garanti etmektedir [17].

2.1.5.3.1.2. Olasılıksal Eğim Düşümü (Stochastic Gradient Descent - SGD)

Bir önceki yöntemin aksine olasılıksal eğim düşümü, her eğitim örneği $x^{(i)}$ ve etiket $y^{(i)}$ için bir parametre güncellemesi yapar [17]. Alınan her eğitim verisi rastgele seçilmektedir. Matematiksel olarak Denklem (2.15)'de olduğu gibi ifade edilebilir.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J\left(\theta_t; x^{(i)}; y^{(i)}\right) \quad (2.15)$$

Bu yöntemde, global minimum noktasına giden yol, Şekil 2.7'de iki boyutlu çizilen paket eğim düşümünde olduğu gibi doğrudan giden bir yol değil, zikzak çizerek ilerleyen bir yoldur. Zikzak çizmesine rağmen yine de global minimum noktasına ulaşmaktadır. Bu yöntem bir önceki yöntemin aksine, tüm eğitim verisi yerine sadece eğitim örneğini alarak ağırlık güncellemesi yaptığı için daha verimlidir [18].



Şekil 2.7. Olasılıksal ve paket eğim düşümü

2.1.5.3.1.3. Mini Paket Eğim Düşümü (Mini-Batch Gradient Descent)

Mini paket eğim düşümü, eğitim setindeki her bir mini paket için bir güncelleme gerçekleştirir. Modelin güncellenme sıklığı paket eğim düşümü yönteminden daha fazladır. Bu yöntem daha fazla yerel minimum noktasından kaçınarak global minimum noktasına hızlı bir yakınsama sağlamaktadır. Hesaplama işlemi Denklem (2.16)'de olduğu gibi ifade edilebilir [17]:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J\left(\theta_t; x^{(i,i+n)}; y^{(i,i+n)}\right) \quad (2.16)$$

2.1.5.3.2. Eğim Düşümü Eniyileme Algoritmaları

Eğim düşümü eniyileme algoritmalarında kullanılan fonksiyonlar Çizelge 2.2'de gösterilmektedir.

Çizelge 2.2. Eğim düşümü eniyileme algoritmaları

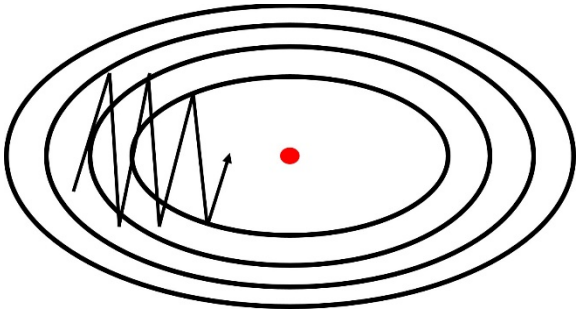
Adaptif Öğrenme Katsayısı	Fonksiyon	Açıklama
Olasılıksal eğim düşümü [17]	$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J\left(\theta_t; x^{(i)}; y^{(i)}\right)$	
Momentum [17]	$v_{t+1} = \gamma v_t + \eta \nabla_\theta J(\theta)$ $\theta_{t+1} = \theta_t - v_{t+1}$	Önceden hesaplanmış diğer eğim değerleri de parametre güncelleme ifadesine katılarak, eğim düşümü algoritmasının global minimum noktasına ulaşması hızlandırılmıştır.
Nesterov Hızlandırılmış Eğim [17]	$v_{t+1} = \gamma v_t + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$ $\theta_{t+1} = \theta_t - v_t$	Parametrelere bağlı hedef fonksiyon eğimi, bir önceki momentum değeri, parametrelerden çıkarılarak hesaplanır.
RMSprop [17]	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$	Eğimin kaybolmasını veya aşırı artması durumlarını engellemek için, öğrenme katsayısı eğimin değerine göre güncellenir.
Adagrad [17]	$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i}$	RMSprop'tan tek farkı eğimler üssel olarak azalan ortalama ile değil, toplamı ile hesaplanmasıdır.
Adadelta [17]	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$	Matematiksel olarak RMSprop ile aynı görünse de bu algoritma her adımda öğrenme katsayısını güncellemez. Tüm öncelik gradyan değerlerini almak yerine bunu belirli bir sayıda kısıtlar.

Adam [17]	$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\frac{v_t}{1-\beta_2^t}} + \epsilon} \frac{m_t}{1-\beta_1^t}$	Adagrad ve RMSprop algoritmalarının faydalarını kullanan bu algoritma, ek olarak momentum değişikliklerini de hesaplar. RMSprop ve momentumun birleşimi olarak da görülebilir.
Adamax [17]	$\theta_{t+1} = \theta_t - \frac{\eta}{(1-\beta_1^t)} \frac{m_t}{u_t}$	Normun sonsuza götürüldüğü Adam algoritması, ilginç bir şekilde daha karalı olduğu gözlemlenmiştir.
Nadam [17]	$\theta_{t+1} = \theta_t - \frac{\alpha_t}{\sqrt{\hat{n}_t + \epsilon}} \hat{m}_t$	Bu algoritma Adam ile Nesterov algoritmasının bir kombinasyonudur.

2.1.5.3.2.1. Momentum

Momentum, maliyet fonksiyonunun küçültülmesi sırasında olasılıksal eğim düşümünün en küçük değere ulaşmasını kolaylaştırmak için kullanılır. Maliyet fonksiyonunun geri yayılım yapılarak ağırlık ve kutuplama değerlerine göre küçültülmesi sürecinde her bir t adımında hesaplanan maliyet değerleri farklıdır. Bu da her bir adımda farklı türev değerlerinin oluşması demektir. Momentum yöntemiyle, -belirlenen sayıda- daha önceden hesaplanmış türev değerlerinin üssel olarak ağırlıklandırılmış ortalaması alınmaktadır. Bu değer, daha sonra ağırlık ve kutuplama değeri güncelleme eşitliğinde çarpan olarak kullanılmaktadır.

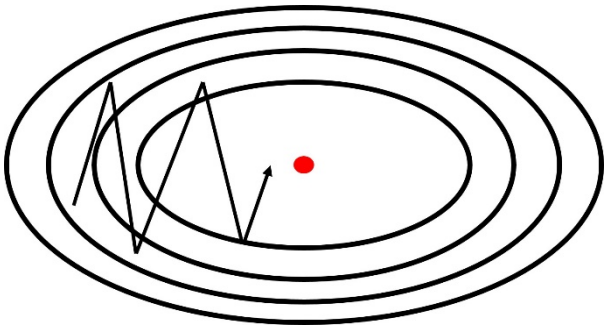
Olasılıksal eğim düşümü tek başına uygulandığında, başlangıçta belli bir büyüklüğe sahip olan maliyet değeri, her adımda hesaplanan farklı türev değerlerine göre en küçük noktaya Şekil 2.8'de görüldüğü gibi çok fazla salınım yaparak ilerlemektedir. Bu şekilde bir kullanımda en küçük değere ulaşmak uzun sürmektedir [19].



Şekil 2.8. Momentum olmaksızın SGD

Bu salınımların dikeyde küçük değerler; yatayda büyük değerler alması, salınımın daha az olacağını dolayısıyla en küçük değere daha kısa sürede ulaşılacağını

gösterir. Şekil 2.9'da eğim düşümü momentum ile birlikte kullanıldığında salınımın değişimi gösterilmiştir [19].



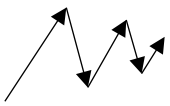
Şekil 2.9. Momentum ile SGD

Eğim düşümü, Momentum ile yapıldığında Denklem (2.17)'deki akış gerçekleşecektir [17].

$$v_{t+1} = \gamma v_t + \eta \nabla_\theta J(\theta) \qquad (2.17)$$
$$\theta_{t+1} = \theta_t - v_{t+1}$$

v_t momentum değerini, γ önceki momentum değerinin katsayısını (1'den küçük sayı) ve η öğrenme katsayısını belirtmektedir.

Şekil 2.10'da görüldüğü üzere momentum olmadan olasılıksal eğim düşümü işlemi gerçekleştirildiğinde dikey yönde her bir adımda bir önceki adıma ters yönde hareketlenme olduğundan bu adımlar birbirlerini dengelerler ve dikey yönlü hareket değerlerinin ortalaması sıfıra yakın olur. Aynı durum yatay yönlü hareketler için geçerli değildir. Çünkü bütün türevlerin yönü yatayda aynıdır. Bu yüzden ortalama alındığında büyük bir değer ortaya çıkar.



Şekil 2.10. Momentum olmadan olasılıksal eğim düşümü

Şekil 2.11'de eğim düşümü momentum kullanılarak yapıldığında hareketlenme değerleri dikey eksende azalmış; yatay eksende artmıştır. Bu da maliyet fonksiyonunun en küçük değerine ulaşılması için gerekli zamanı kısaltmıştır.

Şekil 2.11. Momentum kullanıldığında olasılıksal eğim düşümü

2.1.5.3.2.2. Nesterov Hızlandırılmış Eğim (Nesterov Accelerated Gradient)

Momentum yöntemi, maliyet değerinin en küçük değere ulaşmasını kolaylaştırıyor olsa da eğim düşümünün canlandırılması için sıkça kullanılan vadinin bir tepeciğinden bırakılan top örneğindeki topun, kontrolsüzce tepenin eğimini izlemesi yetersiz bir çözümdür. Ters yönde bir eğimle karşılaştığında ilerlemesini kendince yavaşlatacak akıllı bir topa ihtiyaç vardır.

Nesterov hızlandırılmış eğim, momentum ifadesine bu türden öngörü kazandıran bir yöntemdir. θ parametre değerlerinin, momentum ifadesi γv_t ile güncelleneceği bilinmektedir. Buna göre $\theta - \gamma v_t$'i hesaplamak, parametrelerin bir sonraki yaklaşık değerlerini verir. Hesaplama işlemi Denklem (2.18)'da gösterilmiştir.

$$v_{t+1} = \gamma v_t + \eta \nabla_\theta J(\theta - \gamma v_t) \tag{2.18}$$
$$\theta_{t+1} = \theta_t - v_{t+1}$$

Nesterov hızlandırılmış eğimin başında belirtildiği gibi hata fonksiyonunun küçültme işlemi sırasında kontrollü bir şekilde ilerlemek, aşırı hızlı küçülmeyi önler ve daha hızlı bir çözüm üretilmesinde katkı sağlar [20].

2.1.5.3.2.3. Adagrad

Adagrad algoritmasının görevi, öğrenme katsayısını parametrelere uyarlamaktır. Sıkça değişmeyen parametreler için büyük güncellemeler; sıkça değişen parametreler içinse küçük güncellemeler yapılmasını sağlar [17].

Daha önceki bölümlerde her bir θ_i parametresinin aynı η öğrenme katsayısı kullanılarak güncellendiğinden bahsedilmişti. Adagrad'da her bir parametre θ_i için, her bir t işlem adımında farklı öğrenme katsayısı kullanılır [17].

Adagrad algoritmasının parametre güncelleme yöntemi, Denklem (2.19)'de gösterilmiştir [17].

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{2.19}$$

$G_{t,ii}$: Her bir i, i konumundaki köşegen elemanı; θ_i parametresine göre, t. iterasyona kadar hesaplanmış eğim değerlerinin kareleri toplamıdır. G_t, bir köşegen matristir.

$g_{t,i}$: t anında, θ_i parametresine göre hesaplanmış, maliyet fonksiyonun eğim değeridir.

ϵ: Öğrenme katsayısının 0'a bölünmediğinden emin olmak için atanan bir sabit değerdir. Değeri, genel olarak 10^{-8}'dir.

Öğrenme katsayısını elle ayarlamak zorunda kalmamak, bu yöntemin en önemli

avantajlarından biridir. Çoğu yöntemde öğrenme katsayısı için 0.01 değeri varsayılan olarak kullanılır ve o şekilde bırakılır [17].

Adagrad'ın en büyük dezavantajı, güncelleme işleminde öğrenme katsayısı değerinin bölündüğü ifadenin eğitim süresince büyümeye devam etmesidir.

Öğrenme katsayısının her adımda daha büyük bir değere bölünmesi, değerin aşırı küçülmesine sebep olacaktır. Bir sonraki algoritma buna çözüm aramaktadır [17].

2.1.5.3.2.4. Adadelta

Adadelta, Adagrad'ın güncelleme ifadesinde öğrenme katsayısının agresifçe küçülmesi dezavantajını hafifletmek için çıkarılmıştır. Adagrad'daki geçmiş bütün eğimlerin karesi değerlerinin tamamını kullanmak yerine, değer miktarını belli bir çerçeve boyutu w ile kısıtlamıştır [17].

Bu yöntemde bahsedilen w çerçevesinde bulunan bütün geçmiş eğim değerleri bağımsız olarak depolanmazlar; geçmiş eğimlerin kareleri, üssel olarak ağırlıklandırılmış ortalamaları alınarak tek değer halinde hesaba katılır. Yöntem Denklem (2.20)'de gösterilmiştir [17].

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.20}$$

$E[g^2]_t$ geçmiş eğim karesi değerlerinin ortalamasıdır.

g_t^2: t anında, θ parametresine göre hesaplanmış, maliyet fonksiyonunun eğitim karesidir.

Adadelta yönteminin parametre güncelleme ifadesinde öğrenme katsayısı, eşitlikten çıkarılmaktadır. Bu sayede başlangıçta bir varsayılan öğrenme katsayısı belirlenmesine gerek kalmamaktadır [17].

2.1.5.3.2.5. RMSprop

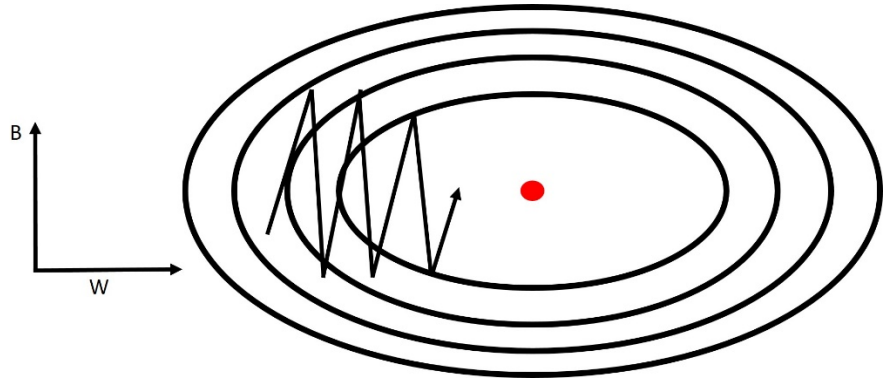
RMSprop, Adadelta yöntemi ile hemen hemen aynı süreçte geliştirilmiş ve yine Adagrad'ın agresif küçültme politikasına bir çözüm sunmayı amaçlamıştır [17]. Daha önce Momentum konusunda bahsedildiği üzere maliyet değerinin, Şekil 2.9'daki küçültme işleminin yapıldığı düzlemde, en küçük değere daha çabuk ulaşabilmesi için yaptığı salınım hareketlerinin dikeyde yavaş; yatayda hızlı olması gerekir.

RMSprop yönteminde bu hızın kazandırılması için Denklem (2.21)'deki akış takip edilmiştir [17].

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1 g_t^2 \tag{2.21}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Şekil 2.12'de düzlemde dikey hareketlenmelerin olduğu eksene B (Kutuplama) ekseni, düşey hareketlenmelerin olduğu eksene ise W (Ağırlık) ismi verilsin. Maliyet değerinin izlediği yol çizgilerinin her biri o andaki türev vektörünü ifade etmektedir. Görüldüğü üzere türev değerleri B ekseninde büyük; W ekseninde küçüktür. RMSprop yönteminde amaç Momentum'da da olduğu gibi dikeydeki hareketlenme değerini küçültüp, yataydaki hareketlenme değerini büyütmektir. Bu yüzden RMSprop'ta ağırlık güncellemesi yapılırken dW küçük değere; dB büyük değere bölünür. Paydadaki ϵ değeri paydanın sıfır olmasını engellemek için genellikle küçük değerli pozitif bir sayıdır.



Şekil 2.12. RMSprop salınımı

2.1.5.3.2.6. Adaptif Moment Tahmin (Adaptive Moment Estimation - Adam)

Adaptif moment tahmin yönteminde, RMSprop'ta yapıldığı gibi geçmiş eğimlerin karelerinin üssel olarak ağırlıklandırılmış ortalamalarının (v_t) depolanmasının yanında, momentumdaki geçmiş eğimlerin üssel olarak ağırlıklandırılmış ortalamalarını (m_t) da tutar:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \qquad (2.22)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Adam algoritmasının yaratıcıları, başlangıçta sıfır vektörü olarak tanımlanan m_t ve v_t değerlerinin, özellikle ilk iterasyonlarda ve β_1 ile β_2 1'e yakınken sıfıra eğilimli olduklarını gözlemlemişlerdir [17].

Bu eğilime, kutuplama-düzeltmeli birinci ve ikinci moment hesaplamalarını yaparak karşı koymuşlardır:

$$m'_t = \frac{m_t}{1-\beta_1^t} \qquad (2.23)$$

$$v'_t = \frac{v_t}{1-\beta_2^t}$$

Denklem (2.22) ve (2.23)'de de yer alan β_i^t ($i = 1$ ve $i = 2$) ifadesinde kullanılan t değeri, kuvvet olarak hesaba katılmıştır. Daha sonra bu iki ifade kullanılarak daha önce Adadelta ve RMSprop'ta yapıldığı gibi parametreler güncellenir. Buna göre Adam güncelleme Denklem (2.24)'de belirtilmiştir [17].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t + \epsilon}} m'_t \qquad (2.24)$$

Bu kuralın varsayılan değerleri β_1 için 0.9; β_2 için 0.999 ve ϵ için 10^{-8} olarak belirtilmiştir.

2.1.5.3.2.7. Adamax

Adam güncelleme kuralındaki v_t faktörü, eğimi, önceki eğimlerin l_2 normuna (v_{t-1} terimi aracılığı ile) ve geçerli eğim $|gt|^2$'ye ters orantılı olarak ölçeklendirir [17].
Algoritma için en iyi varsayılan değerler; öğrenme katsayısı η için 0.002, β_1 için 0.9 ve β_2 için 0.999 olarak algoritmanın geliştiricileri tarafından belirlenmiştir [17].

2.1.5.3.2.8. Nadam

Nadam (Nesterov Hızlandırılmış Adaptif Moment Tahmin - Nesterov Accelerated Adaptive Moment Estimation), Adam ve NAG'ı kombine eder. NAG'ı Adam ile bağdaştırabilmek adına Momentum ifadesi üzerinde değişiklik yapılmalıdır. NAG sayesinde eğim değeri hesaplanmadan önce parametreler Momentum ifadesiyle güncellenerek, eğim yönünde daha tutarlı bir adım atılır [17].

2.1.5.4. Ağın Genişliği ve Derinliği

Ağın genişliği, gizli bir katmandaki gizli düğümlerin miktarını ifade eder. Ağın derinliği ise ağda bulunan gizli katmanların sayısını ifade eder. Öğreticili öğrenme tekniğinde ağın genişliği ve derinliği önemli bir rol oynamaktadır. Lu ve arkadaşları [21], genişliğin sinir ağını nasıl anlamlandırdığı hakkında bir araştırma yapmışlardır. Bu çalışma ReLU ağları için derinliğin genişlikten daha anlamlı olabileceğini göstermiştir.

Birçok araştırmacı tarafından daha derin bir yapıya sahip olan ağların yapabilecekleri konusu araştırılmaktadır. Eldan ve Shamir [22], 3 katmanlı ileri yönelimli sinir ağlarının, 2 katmanlı olan ağlardan daha yüksek doğruluk ile basit

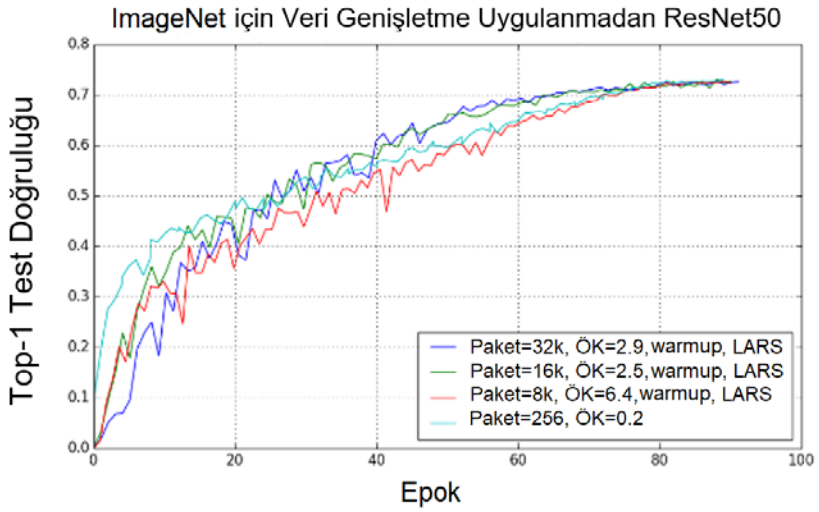
fonksiyonları ifade edebildiklerini belirtmişlerdir. Aynı zamanda Bengio [23], ağın derinliği arttıkça potansiyelinin de logaritmik olarak arttığını ortaya koymuştur.

2.1.5.5. Epok, Döngü Sayısı ve Paket Boyutu

Epok, tüm veri setinin bir ileri yönde bir de geri yönde tüm ağdan geçmesidir. Iterasyon, belirlenen paket boyutu kadarlık verinin ileri ve geri yönde ağdan geçmesidir. Paket boyutu, ileri ve geri yönde yayılım için veri setinden alınan veri miktarıdır. Paket boyutu yükseldikçe, daha fazla hafıza gerekir. Yüksek paket boyutu seçilmesi her iterasyon için kullanılacak eğitim örneğinin de büyümesi demektir. 1000 adet veriden oluşan bir veri seti için paket boyutu 500 seçilirse, bir epoğu tamamlamak için iki iterasyon gereklidir.

Epok sayısının yüksek olması, ağın daha yüksek doğrulukta sonuçlar üreteceği anlamına gelmez. Genelde bu iki parametre, değerin her adımda ne kadar değiştiğini ölçmek için kullanılır (Top-1 doğruluk, hata payı, ağırlık değişimi gibi).

You ve arkadaşları [24], epok ve Top-1 test doğruluğu korelasyonunu gösteren Şekil 2.13'ü hazırlamışlardır. Bu şekilde epok sayısı ve diğer hiperparametre değerleri ne olursa olsun, bir noktadan sonra doğruluktaki kazanımın daha yavaş olduğu ve doğruluğun daha fazla artmadığı gözlemlenebilir. Aynı durum Hoffer'in [25] yaptığı araştırmada da gözlemlenmiştir. Doğrulama hatasında düşüş 5000'inci iterasyondan sonra başlamış ve 20000'inci iterasyondan sonra yavaşlamıştır.



Şekil 2.13. Top-1 doğruluğu ve epok sayısının arasındaki ilişki [26]

Aynı zamanda, paket boyutu da ağın doğruluğunu etkileyebilir. Goyal ve arkadaşları [27] tarafından yapılan bir araştırmada paket boyutunun 8192 olması veya 256 olması çok yakın doğruluk değerleri vermiştir. Paket boyutunun çok fazla artması Top-1 doğrulama hatasının da çok hızlı artmasına neden olmuştur. Bu sonuca benzer olarak You ve arkadaşları [24], ResNet50 doğruluğunun, paket boyutu olarak 10000'ni aştığında hatanın arttığı sonucuna varmıştır.

2.1.5.6. Düzenlileştirme (Regularization)

Düzenlileştirme tekniği, ağın girdideki gürültüleri de öğrenerek daha fazla parametreye sahip olmasını engeller. Ağın ezberlemesi durumu, girdinin gürültüsü ile birlikte öğrenilmesiyle meydana gelir. Bu yöntem, ağın ezberlemesini önler. Düzenlileştirme, aynı zamanda algoritmanın genelleştirilmesini de sağlar ve ağın karmaşıklığı arttıkça katsayıların girdi ile mükemmel uyuşmasını önler [28].

Ezberlemeyi önlemek için veri setini artırmak bir çözümdür. Aynı zamanda, L1 ve L2 Düzenlileştirmesi (Regularization), Seyreltme (Dropout), Ağırlık Sıfırlama (Dropconnect) ve veri genişletme yöntemleri de kullanılabilir. L1 ve L2 en sık kullanılan düzenlileştirme yöntemleridir. L1 düzenlileştirme yönteminde, parametrelerin mutlak değerlerinin toplamını azaltmak için fonksiyona düzenlileştirme terimi eklenir. L2 tekniğinde ise düzenlileştirme terimi, parametrelerin karelerinin toplamını azaltmak için eklenir. L1 tekniği kullanıldığında parametre vektörü seyrek bir hal alacaktır. Bunun sebebi, düzenlileştirme tekniğinin bazı parametreleri sıfıra eşitlemesidir. Düzenlileştirme algoritmaları Çizelge 2.3'te gösterilmektedir.

Çizelge 2.3. Düzenlileştirme algoritmaları

Düzenlileştirme Tekniği	Fonksiyon	Açıklama
L1 Düzenlileştirme [28]	$R_{L_1}(w) = \|W\| = \sum_{k=1}^{Q} w_k$	Algoritma sadece en iyi birkaç özniteliğe yönelip diğerlerini ihmal eder. Ayrıca her özniteliğin sonuca etkisini azaltır [28].
L2 Düzenlileştirme [28]	$R_{L_2}(w) = \|W\|^2 = \sum_{k=1}^{Q} w_k^2$	Modele karmaşıklık cezası ekler. Daha az etkili olan özniteliklerin daha fazla, baskın özniteliklerin daha az etkili olması için ceza değeri ekler [28].

Seyreltme (Dropout) [29]	$\tilde{y}^{(l)} = r^{(l)} * y^{(l)}$	Nöron değerlerini sıfırlar. Herhangi bir l katmanı için, $r^{(l)}$ nöronların aktif olma olasılıklarını içeren vektörü, $y^{(l)}$ bir önceki katmanın çıktısını, $*$ işlemi element-wise çarpım işlemidir [29].
Ağırlık Sıfırlama (Dropconnect) [30]	$r = a(u) = a\big((M * W)v\big)$	Seyreltme ile benzerdir. Nöronun değil ağırlıkların bir kısmını sıfırlar. v girdi vektörü, W ağırlık vektörü, a aktivasyon fonksiyonu, M bağlantı bilgilerini kodlayan ikili bir matristir. $*$ işlemi element-wise çarpım işlemidir [30].

2.1.5.7. Ağırlık Başlangıç Değerleri

Ağırlık başlangıç değerleri, ağın başarımını ve eğitim süresini doğrudan etkilemektedir. Ağırlık değerlerinin 0 olarak atanması, bütün nöronların aynı çıktı değerini hesaplamasını sağlayacağından geri yayılım işlemi sırasında eğim değerlerinin aynı çıkmasına, dolayısıyla tüm ağırlıkların aynı parametre değeriyle güncellenmelerine neden olacaktır [10].

Başlangıçta rastgele değerler olarak atanan ağırlıklar, tam olarak 0 değerine sahip olmamakla birlikte küçük rastgele değerlere sahip olmalıdır. Buna simetri kırılması (symmetry breaking) denir [31]. Bu şekilde rastgele ve benzersiz ağırlıkların olması, farklı güncelleme değerlerinin üretilmesini ve ağırlıkların birbirlerinden farklı şekillerde ağa adapte olmalarını sağlayacaktır [10].

Çok küçük değerlerin daha iyi sonuçlar vereceği kesin değildir. Çünkü çok küçük değerler almış herhangi bir ağırlık katmanı, geri yayılım algoritması sırasında çok küçük eğim değerlerinin hesaplanmasına yani kaybolan eğim (vanishing gradient) adı verilen bir soruna neden olacaktır [32].

Bu tez çalışması kapsamında model katmanlarında varsayılan olarak “Glorot Düzgün” yöntemi kullanılmıştır. Aşağıda ağırlık başlangıç değeri belirleme yöntemleri listelenmiştir [33]:

a. **Bir**: Başlangıç ağırlıklarına 1 değeri atanır.
b. **Sabit**: Başlangıç ağırlıkları olarak elle girilmiş sabit bir değer atanır.
c. **Rastgele Normal**: Ağırlık birimlerinin değerleri normal dağılım ile belirlenir.
d. **Rastgele Düzgün**: Ağırlık birimlerinin değerleri düzgün dağılım ile belirlenir.

e. **Kesikli Normal**: Ağırlık birimlerinin değerleri kesikli normal dağılım ile belirlenir. Rastgele normal dağılıma benzer sonuçlar üretilir ama standart sapması 2'den büyük olan değerler hesaba katılmaz ve bu değerler için yeniden hesaplama yapılır. Bu yöntem, sinir ağları ve filtreler için önerilen yöntemdir.

f. **Varyans Ölçeklendirme**: Bu yöntemde ağırlık başlangıç değerlerinin belirlenmesi, ağırlık matrisinin boyutlarına göre ölçeklendirmeyle yapılır. Eğer değer belirleme işlemi normal dağılım seçeneğiyle yapılıyorsa, ağırlıklar sıfır merkezli kesikli normal atamaya göre değer alırlar. Bu durumda, kesikli normal dağılımın standart sapma değeri (σ) işlemi Denklem (2.25)'e göre değer alır:

$$\sigma = \sqrt{\frac{s}{n}} \tag{2.25}$$

İfadede belirtilen s ölçeklendirme değeridir. n seçilen moda göre aşağıdaki 3 değerden birini alır:

Eğer fan_in modu seçilmişse n, ağırlıklar matrisindeki girdi birimlerinin sayısıdır.

Eğer fan_out modu seçilmişse n, ağırlıklar matrisindeki çıktı birimlerinin sayısıdır.

Eğer fan_avg modu seçilmişse n, ağırlıklar matrisindeki girdi ve çıktı birim sayısının ortalamasıdır.

Eğer değer belirleme işlemi düzgün dağılım seçeneğiyle yapılıyorsa ağırlıklar [-limit, limit] aralığında düzgün atamaya göre değer alırlar. Limitin değeri, Denklem (2.26)'ya göre belirlenir.

$$limit = \sqrt{\frac{3s}{n}} \tag{2.26}$$

g. **Dikey (Orthogonal)**: Ağırlıklar matrisi, elemanları rastgele değerlerden oluşan bir kare matristir [34].

h. **Birim (Identity)**: Ağırlıklar matrisi, bir birim matristir.

i. **Lecun Düzgün**: LeCun düzgün başlangıç değer tanımlayıcısında ağırlıklar, [-limit, limit] aralığında düzgün atamaya göre değer alır. Burada limit değeri Denklem (2.27)'ye göre belirlenir [35].

$$limit = \sqrt{\frac{3}{fan\_in}} \tag{2.27}$$

Yukarıdaki eşitlikte ifade edilen fan_in, ağırlıklar matrisindeki girdi birimlerinin sayısıdır.

j. **Glorot Normal**: Glorot normal başlangıç değer tanımlayıcısı aynı zamanda Xavier normal başlangıç değer tanımlayıcısı olarak da anılmaktadır. Ağırlıklar, 0 merkezli kesikli dağılıma göre değer alır. Atanacak değerlerin standart sapma (σ) parametresi Denklem (2.28)'e göre hesaplanır [36]:

$$\sigma = \sqrt{\frac{2}{fan\_in + fan\_out}} \tag{2.28}$$

Yine bu eşitlikte kullanılan fan_in ve fan_out ifadelerinin karışıkları varyans ölçeklendirme başlığı altında açıklanmıştır.

k. **Glorot Düzgün**: Glorot düzgün başlangıç değer tanımlayıcısı aynı zamanda Xavier düzgün başlangıç değer tanımlayıcısı olarak da anılmaktadır. Ağırlıklar, [-limit, limit] aralığında düzgün atamaya göre değer alır. Limit değeri Denklem (2.29)'a göre hesaplanır [36].

$$limit = \sqrt{\frac{6}{fan\_in + fan\_out}} \tag{2.29}$$

l. **He Normal**: He normal başlangıç değer tanımlayıcısında ağırlıklar, 0 merkezli kesikli dağılıma göre değer alır. Atanacak değerlerin standart sapma (σ) parametresi Denklem (2.30)'a göre hesaplanır [16].

$$\sigma = \sqrt{\frac{2}{fan\_in}} \tag{2.30}$$

m. **LeCun Normal**: LeCun normal başlangıç değer tanımlayıcısında ağırlıklar, 0 merkezli kesikli dağılıma göre değer alır. Atanacak değerlerin standart sapma (σ) parametresi Denklem (2.31)'e göre hesaplanır [15, 35].

$$\sigma = \sqrt{\frac{1}{fan\_in}} \tag{2.31}$$

n. **He Düzgün**: He düzgün varyans ölçeklendirme başlangıç değer tanımlayıcısında ağırlıklar, [-limit,limit] aralığında düzgün atamaya göre değer alır. Burada limit değeri Denklem (2.32)'ye göre belirlenir [16].

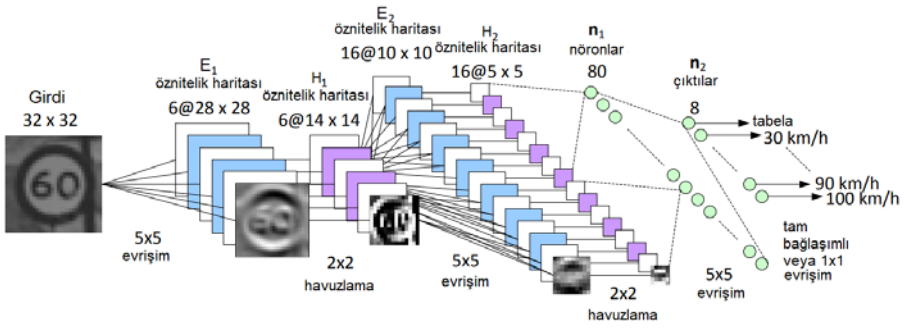
$$limit = \sqrt{\frac{6}{fan\_in}} \tag{2.32}$$

2.2. Evrişimli Sinir Ağları

Son 7 yılda araştırma ve uygulama alanında ESA (Evrişimli Sinir Ağları) önemli bir gelişme göstermiştir. Bunun sebebi, evrişimli sinir ağlarının günümüzde üretilen çok büyük miktarda veri ile çalışabilmesidir. Geniş açıdan baktığımızda, uydulardan üretilen resimler, videolar ve benzeri veriler ile sınıflandırma işlemi, yüksek bir doğruluk ile evrişimli sinir ağları kullanılarak gerçekleştirilmektedir. ESA'nın başarısı Tesla veya Google gibi büyük firmaların kullanıma sunduğu kendi kendine süren araç ve çevrimiçi servislerde de görülmektedir. Yüksek miktarda veri, yeni nesil ESA algoritmaları ve yüksek performanslı GPU'lar sayesinde hayatımıza giren kendi kendine süren araçlar ve robotlar endüstri devriminde yeni bir sayfanın açılmasına katkı sağlamıştır.

2.2.1. ESA Mimarisi

Geleneksel ESA mimarisi genelde girdi katmanı, evrişim katmanı, havuzlama katmanı, tam bağlaşımlı katman (fully-connected layer) ve çıktı katmanı olmak üzere beş ana katman içerir. Araştırmacılar, bu beş katmanın farklı dizilişleriyle oluşturulmuş AlexNet, ResNet, Inception, VGG gibi ESA mimarileri ile geliştirme çalışmaları yapmaktadırlar. Bu bölümde ESA mimarisinde bulunan üç ana katman daha detaylı olarak incelenecektir. Örnek ESA yapısı Şekil 2.14'te görülmektedir [37].



Şekil 2.14. Örnek CNN yapısı [37]

2.2.1.1. Evrişim Katmanı

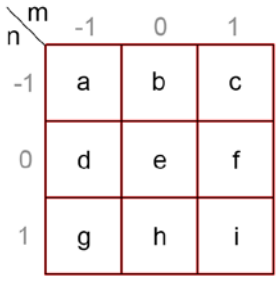
Evrişim konsepti ilk olarak LeCun [38] tarafından tanıtılmıştır. Evrişim özelleştirilmiş bir doğrusal işlemdir. Bu ağlar basitçe, en az bir katmanında matris çarpımı yerine evrişim işlemi gerçekleştiren ağlardır [39]. Evrişimi farklı alanlarda kullanma çalışmaları da yapılmıştır. Jackson [40], eşit olmayan frekans örneklerinden resmi yeniden oluşturmak için bu fonksiyondan faydalanmıştır. Santos [41], tren raylarının davranışını incelemek için evrişimi kullanmıştır. Ayrık zamanlı evrişim işlemi Denklem (2.33) ile ifade edilir [39]:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.33}$$

Denklem (2.33)'de filtre (kernel) w, girdi x, zaman t, ve s sonuç olarak ifade edilmiştir. Girdi olarak resim gibi iki boyutlu bir girdi kullanıldığında evrişim işlemi, Denklem (2.34) ile ifade edilir [39]:

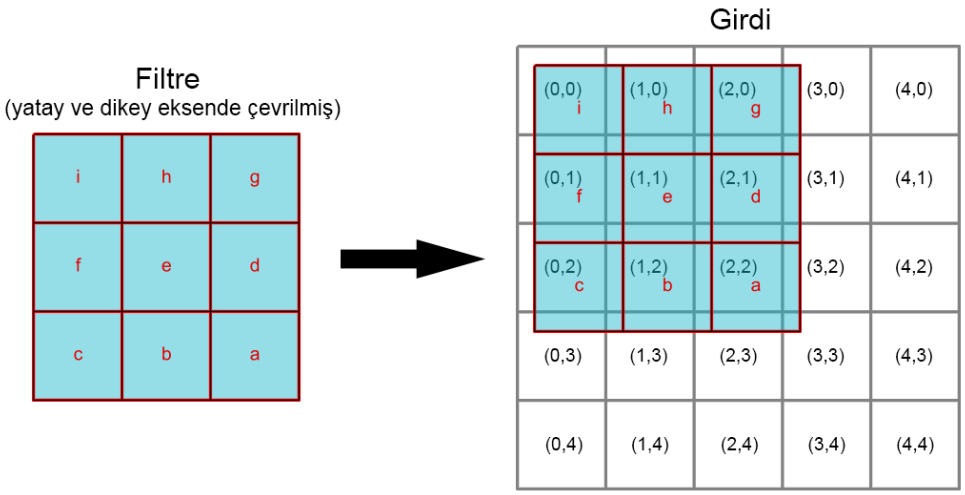
$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i,j)K(i-m, j-n) \tag{2.34}$$

Denklem (2.34)'de i ve j terimleri, evrişim işlemi sonucunda elde edilecek yeni matrisin konumlarını ifade etmektedir. Birçok durumda filtrenin merkezi, orijinde olacak şekilde konumlandırılır [39]. Şekil 2.15'te gösterilen filtre örneğinde, e noktası orijinde konumlandırılmıştır [42]. Bu sebeple bu nokta (0,0) noktasına denk gelmektedir. Denklem (2.34)'de yer alan m ve n terimleri, Şekil 2.15'teki resimde görülebileceği gibi filtrenin her bir konumunu ifade etmektedir.



Şekil 2.15. Örnek filtre

Şekil 2.16'da gösterilen girdiye, Şekil 2.15'teki filtre ile evrişim işlemi uygulandığında, çıktının (1,1) konumundaki değeri Denklem (2.35) ile hesaplanır.



Şekil 2.16 Örnek evrişim işlemi

$$\begin{aligned}(I * K)(1,1) = {} & I(0,0)K(1-0,1-0) + I(1,0)K(1-1,1-0) \\ & + I(2,0)K(1-2,1-0) + I(0,1)K(1-0,1-1) \\ & + I(1,1)K(1-1,1-1) + I(1,2)K(1-1,1-2) \\ & + I(0,2)K(1-0,1-2) + I(1,2)K(1-1,1-2) \\ & + I(2,2)K(1-2,1-2)\end{aligned} \tag{2.35}$$

Şekil 2.16'da, filtre çevrilmiş bir şekilde girdi ile işleme girmektedir. Bu çevirme işlemi hem yatay hem de dikeyde gerçekleşmektedir. Birçok yapay sinir ağı kütüphanesinde, filtrenin çevrilmeden girdi ile işleme girmesi için çapraz korelasyon fonksiyonu uygulanmaktadır. Bu fonksiyon filtrenin herhangi bir şekilde çevrilmeden girdi ile işleme girmesine olanak sağlamaktadır. Çapraz korelasyon, Denklem (2.36) ile ifade edilir [39]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.36}$$

Evrişim işlemi, diğer makine öğrenmesi algoritmalarına göre parametre paylaşımı, seyrek etkileşim ve eş değişken kullanması nedeniyle avantajlar sunmaktadır [39].

Seyrek etkileşimler, ağın, her biri sadece seyrek etkileşimleri açıklayan basit yapı bloklarından inşa ederek, birçok değişken arasındaki (örneğin kenarlar gibi) karmaşık etkileşimleri verimli bir şekilde tanımlamasına olanak tanır. Bu da daha az

parametrenin depolanması anlamına gelir ve istatistiksel verimliliği arttırır. Parametre paylaşımı, aynı parametrelerin fonksiyon içinde birden fazla kullanılması anlamına gelmektedir. Bu sayede yine depolama alanında daha az parametre bulunacağından verimlilik sağlanmış olur.

Bir fonksiyonun eşdeğerli olması, bu fonksiyonun girdisi değiştikçe çıktısının da aynı doğrultuda değişeceği anlamına gelmektedir. Eğer $f(g(x)) = g(f(x))$ sağlanıyorsa, f fonksiyonu g fonksiyonu ile eşdeğerlidir. Evrişim işleminde eğer g fonksiyonunu, girdinin her pikselini bir adım sağa kaydırmak gibi girdiyi değiştiren herhangi bir fonksiyon ise, bu durumda evrişim, g fonksiyonu ile eşdeğerli olmaktadır.

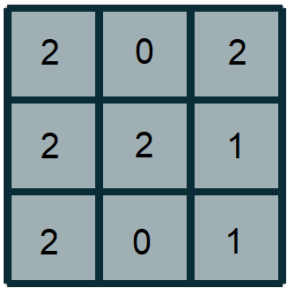
I, resmin herhangi bir koordinatındaki parlaklık değerini veren bir fonksiyon, g, girdiyi bir piksel sağa kaydıran fonksiyon olmak üzere:

$$I' = g(I)$$
$$I'(x, y) = I(x - 1, y) \quad (2.37)$$

g fonksiyonu, parametre olarak f fonksiyonunu aldığında, f fonksiyonundaki her piksel bir birim sağa kayacaktır. Eğer bu dönüşüm I fonksiyonuna uygulandıktan sonra evrişim işlemi uygulanırsa elde edilecek sonuç, I I' fonksiyonunun evrişim işlemine tabi tutulup çıkan sonucun g fonksiyonuna parametre olarak verilmesiyle elde edilen sonuç ile aynı olacaktır. Bu durum evrişim işleminin eşdeğerli olmasından kaynaklanmaktadır. I fonksiyonunun önce bir piksel sağa kaydırıp daha sonra evrişim işlemi yapıldığında elde edilen sonuç, I' fonksiyonunun önce evrişim işlemi yapılıp daha sonra bir piksel kaydırılmasıyla elde edilen sonuç ile eşit olmaktadır [39].

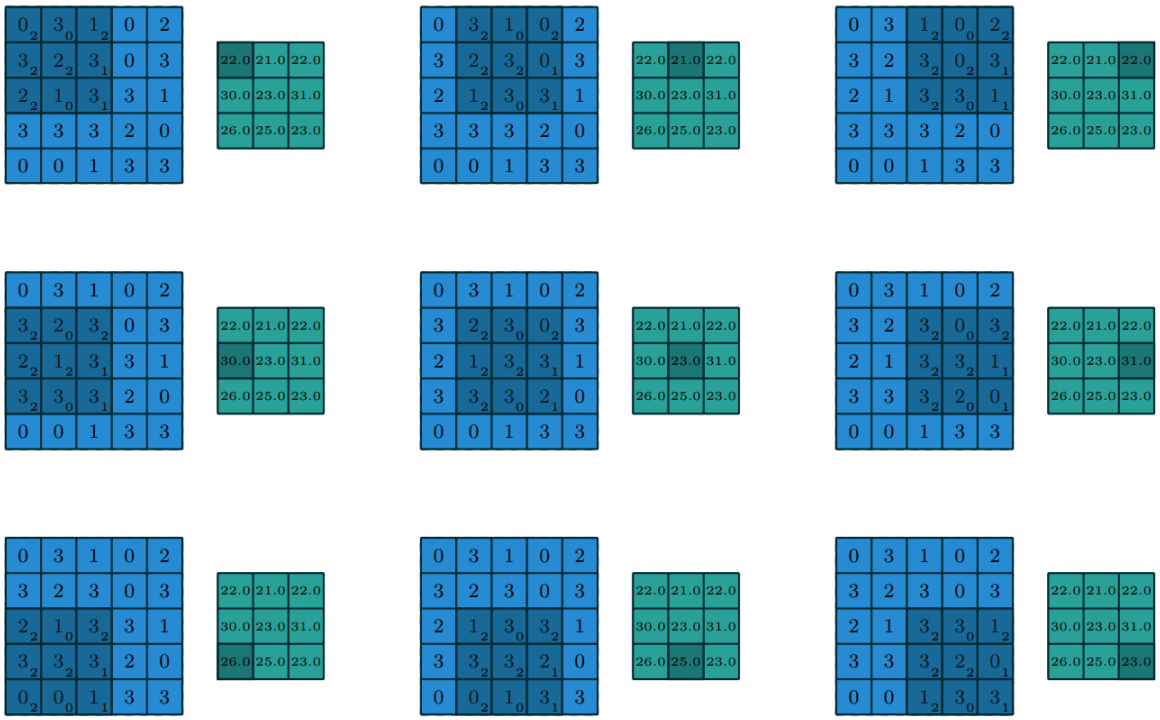
Resmi döndürme gibi bazı dönüşüm tekniklerinde bu özellik kullanılamamaktadır. Bu tarz dönüşümler için başka mekanizmalar kullanılmalıdır [39].

Evrişim katmanında kullanılan 3x3'lük filtre örneği Şekil 2.17'de yer almaktadır [43].



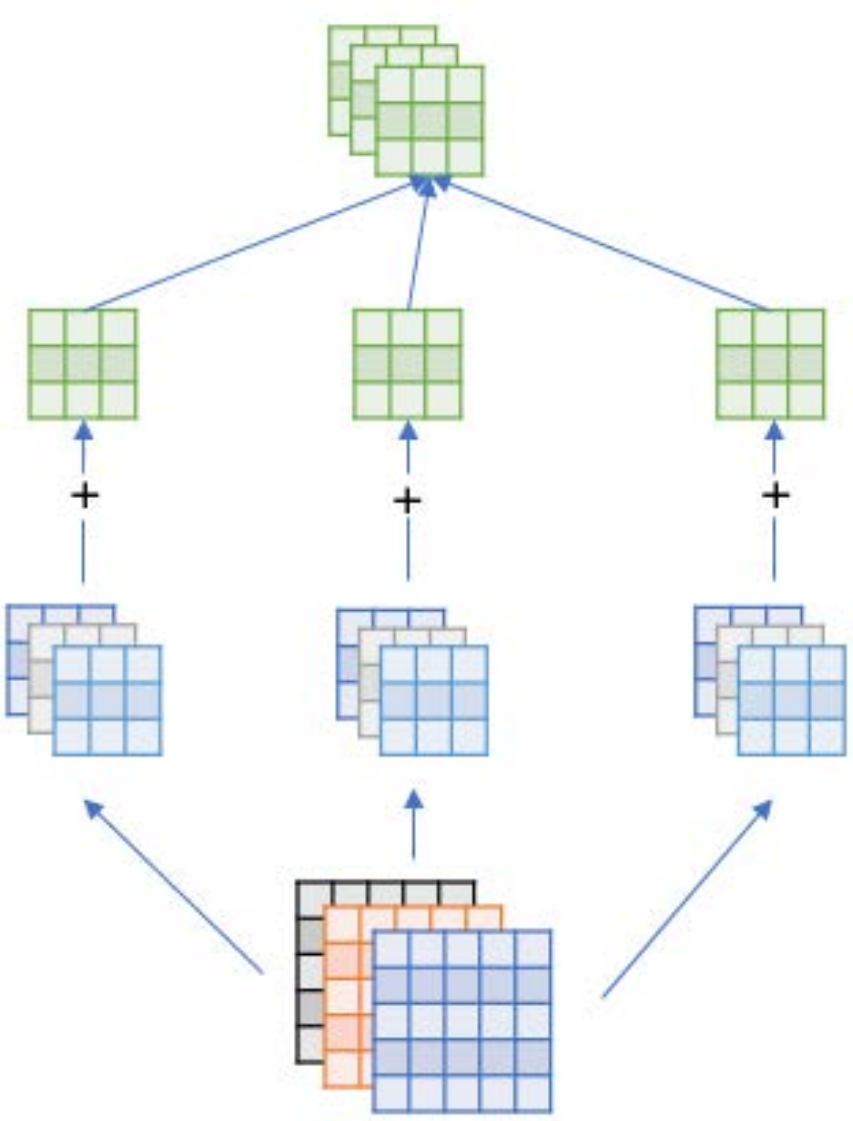
Şekil 2.17. Örnek filtre

Filtre, girdi üzerinde kayarak ilerler. Her konumda filtre ile girdinin üst üste geldiği yerler çarpılıp toplanarak bir çıktı üretilir ve yeni bir matrise yazılır. Bu çıktıları oluşturan matrise çıktı yani öznitelik haritası denir. Şekil 2.18'de 5x5 boyutlarında bir veriye 3x3 boyutunda filtre adım sayısı 1 olacak şekilde uygulanmaktadır [43].



Şekil 2.18. 5x5 boyutundaki veriye 3x3 filtre uygulanması

Filtre gezdirme işlemi Şekil 2.19'da gösterildiği farklı filtreler kullanılarak yapılabilir [43]. Birinci yol takip edildiğinde, girdi 1 ve girdi 2'nin filtre 1 ile evrişim işlemine girdiği ve bir adet çıktı ürettiği görülmektedir. Girdiler aynı şekilde ikinci ve üçüncü yolda bulunan farklı filtreler ile işleme sokulduğunda her işlem sonucunda bir çıktı üretildiği görülmektedir [37].



Şekil 2.19. İki adet girdiye üç farklı filtre uygulanması örneği

Şekil 2.18 2 boyutlu evrişime bir örnektir. Bu işlem N katmanlı veriler için de genelleştirilebilir. Örnek olarak 3 katmanlı bir veride, filtre de 3 katmanlı olmalıdır. Filtrenin her katmanı girdinin ilgili katmanında kayarak ilerleyecektir [43].

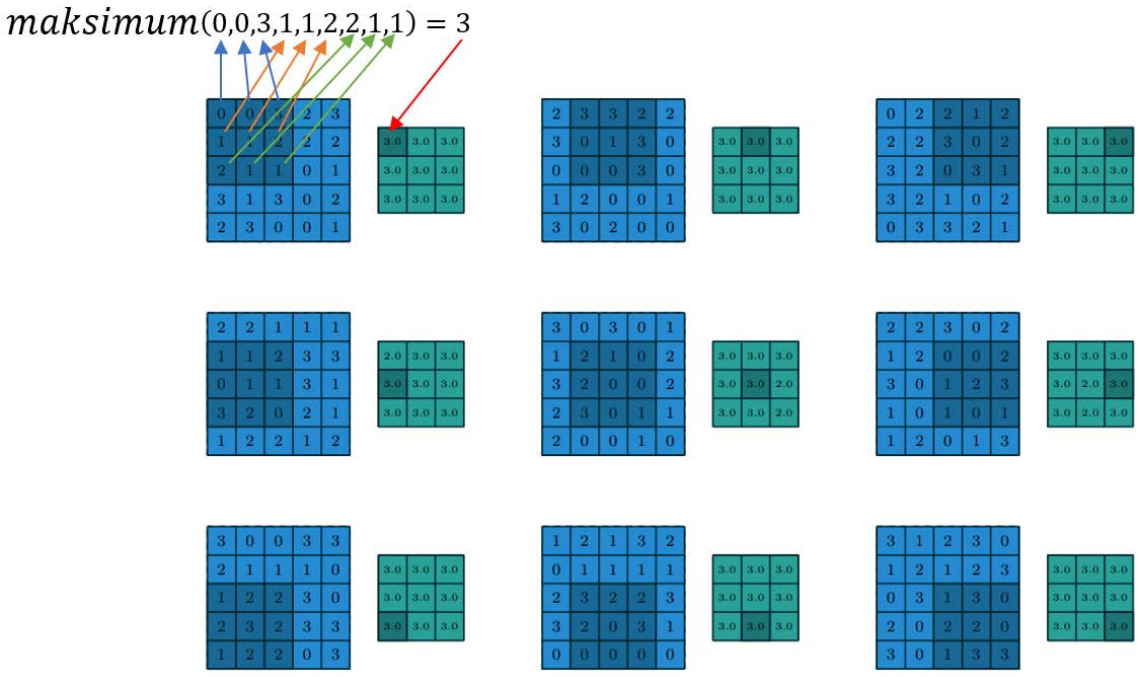
En iyi filtreyi tasarlamak aynı zamanda en verimli filtre parametrelerini seçmeyi gerektirir. Bu parametreler filtre ağırlık değerleri, filtre boyutudur. Bu parametrelerin düzenlenmesi ağ çıktısının verimini artırmayı ve hafıza kullanımını azaltmayı sağlasa da eniyileme için sadece bu alanlara odaklanmak yeterli olmamaktadır. Bazı araştırmacılar filtre boyutunun ağın çıktısına etkisini de araştırmıştır. Szegedy ve arkadaşları [44] 5x5 n tane filtre ile 3x3 aynı sayıda filtreyi karşılaştırmış ve 5x5 filtre ile yapılan hesaplamanın 2.78 kat daha maliyetli olduğu sonucuna varmıştır.

2.2.1.2. Havuzlama Katmanı (Pooling)

Havuzlama işlemlerinde, alt bölgeleri özetlemek için ortalama veya maksimum değer alma gibi bazı fonksiyonlar kullanılarak çıktının boyutu düşürülmektedir.

Havuzlama işlemi belirlenen bir alan içindeki değerlerin ortalamasını alarak veya maksimumunu hesaplayarak bir değer çıkarır. Girdide havuzlama işlemi kayan pencere yöntemiyle uygulanmaktadır. Kayan pencere her seferinde belirlenen havuzlama yöntemine göre denk geldiği girdi alanından bir değer oluşturur ve bunu çıktı katmanına ekler [43].

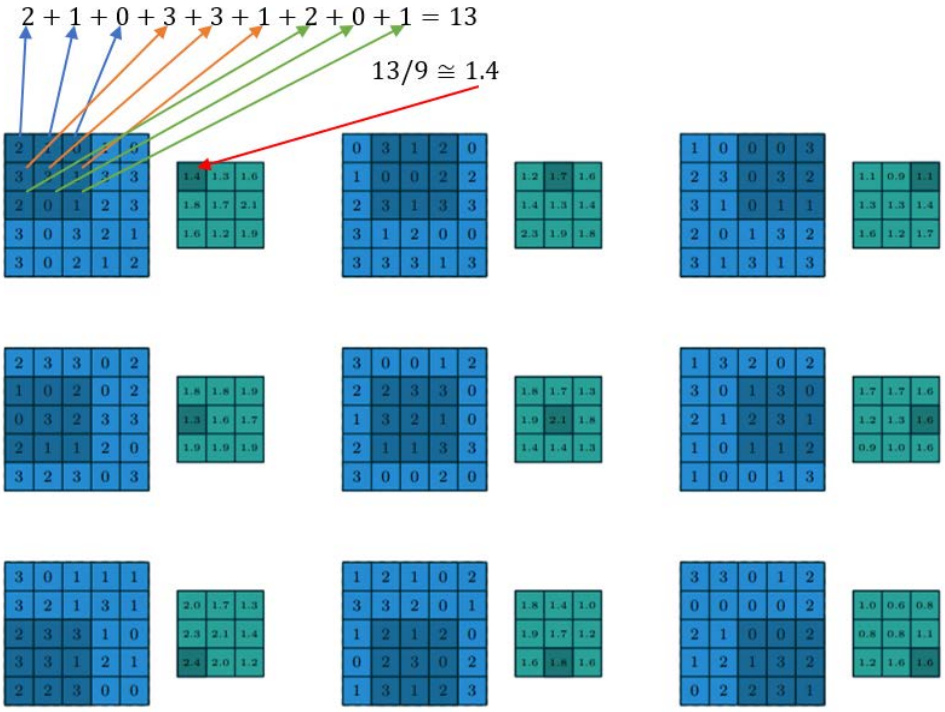
Bir evrişimli sinir ağında havuzlama katmanları sayesinde, daha önce bahsedilen girdinin küçük parçaları, tercih edilen yönteme göre tek bir sabit değere indirgenmiş olur. Buna göre havuzlama katmanı hesaplamaları, evrişim katmanı hesaplamalarına göre daha az maliyetlidir. Şekil 2.20'de listelenen havuzlama türlerinden en yaygın olarak kullanılan türü, girdinin parçalara ayrılıp her bir parçadaki en yüksek değerin alındığı maksimum havuzlamadır. 3x3'lük maksimum havuzlama işleminin 5x5'lik girdi üzerinde 1x1'lik adım sayısı ile hesaplanması Şekil 2.20'de gösterilmiştir [43].



Şekil 2.20. Maksimum havuzlama örneği

Havuzlama türlerinden sıklıkla kullanılan ortalama havuzlamada, girdi yine maksimum havuzlamadaki gibi parçalara ayrılarak her bir parçadaki değerlerin

ortalaması alınır. 3x3'lük ortalama havuzlama işleminin 5x5'lik girdi üzerinde 1x1'lik adım sayısı ile hesaplanması Şekil 2.21'de gösterilmiştir [43].



Şekil 2.21. Ortalama havuzlama örneği

Havuzlama katmanının çıktı boyutunun (o); girdi boyutu (i), işlemin gerçekleştirileceği pencere (parça) boyutu (k) ve adım sayısı (s) ile ilişkisi Denklem (2.38)'de gösterilmektedir [43].

$$o = \left[\frac{i-k}{s}\right] + 1 \tag{2.38}$$

Denklem (2.38)'de belirtilen ilişki, bütün havuzlama türleri için geçerlidir [43].

2.2.1.3. Tam Bağlaşımlı Katman

Bu katman yapay sinir ağı gibi çalışmaktadır. Evrişim ve havuzlama işlemleri sonucunda üretilen değerler bu katman tarafından girdi olarak alınarak işleme sokulmakta ve çıkış katmanında sınıf sayısı kadar sonuç üretilmektedir

2.2.2. Hiper Parametreler

Bir önceki bölümde evrişimli sinir ağlarına ait evrişim katmanı, havuzlama katmanı ve tam bağlaşımlı katman özellikleri hakkında bilgi verildi. Temel sinir ağlarında olduğu gibi evrişimli ağlarının da en iyi sonucu verebilmesi için hiper parametrelerin

ideal bir biçimde ayarlanması gerekmektedir. Her bir hiper parametrenin detayları bir sonraki bölümde tartışılacaktır.

2.2.2.1. Filtre Derinliği, Filtre Boyutu, Adım Sayısı ve Dolgu (Padding)

Derinlik, kullanmak istediğimiz filtre sayısıdır ve her filtre girdideki farklı bir özeliği aramaktadır. Örneğin evrişim katmanı girdi olarak bir resim aldığında, her bir filtrede bulunan farklı nöronlar, bu girdide bulunan köşe veya farklı renklerden dolayı aktif olacaktır. Girdinin belli bir yerine bakan fakat farklı özellikler arayan nöronların hepsine derinlik kolonu denilmektedir [10].

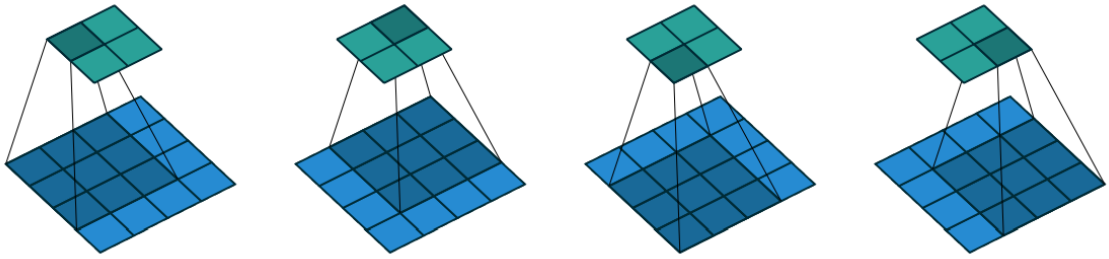
Karen ve Andrew [45], filtre boyutlarını üzerine yaptığı çalışmada, bir adet 5x5 boyutunda filtre yerine 2 adet 3x3 boyutlarında filtre; 7x7 boyutlarında bir filtre yerine üç adet 3x3 boyutlarında filtre ve adım değeri 1 olarak kullanılmıştır. 7x7 filtre yerine 3x3 boyutunda filtre kullanılması, ilk olarak bir tane doğrusal olmayan katman yerine üç adet kullanılmasına yol açmış bu da fonksiyonun daha ayırt edici olmasını sağlamıştır. İkinci olarak parametre sayısı düşürülmüştür. Örneğin üç adet 3x3 boyutunda filtre kullanarak elde edilen parametre sayısı, C_1 kanal sayısı, C_2 filtre sayısı olmak üzere $3(3^2C_1C_2) = 27C_1C_2$ olurken, bir adet 7x7 boyutunda bir filtre kullanıldığında $7^2C_1C_2 = 49C_1C_2$ olmaktadır. Bu da %81 daha fazla parametre demektir.

Adım sayısı, filtrenin girdi üzerinde ne kadar piksel kaydırılacağını belirten hiper parametredir. Örneğin adım sayısı 1 seçildiğinde, filtre evrişim işlemi sonrasında 1 piksel kayacaktır. Eğer adım sayısı 2 olursa filtre 2 piksel kayacaktır. Adım sayısının 2 veya daha fazla bir değere sahip olması, çıktı boyutunun girdiden daha düşük olmasına neden olacaktır [10].

Filtre, girdinin en solundan başlayıp en sağına ulaşana kadar birer adım ilerlediğinde çıktının genişliği, ilerleme sayısı artı bir olmaktadır. Aynı yöntem yükseklik için de uygulanabilir. Matematiksel olarak ifade edersek herhangi bir girdi boyutu (i) ve filtre boyutu (k) değeri için adım sayısı 1 ve dolgu değeri sıfır olmak üzere sonuç (o) Denklem (2.39) ile hesaplanır [43]:

$$o = (i - k) + 1 \tag{2.39}$$

Şekil 2.22'de girdi genişliği (i) 4, filtre boyutu (k) 3 için, adım sayısı (s) 1 ve dolgu 0 olduğunda sonuç $o = (i - k) + 1 = (4 - 3) + 1 = 2$ olur [43].

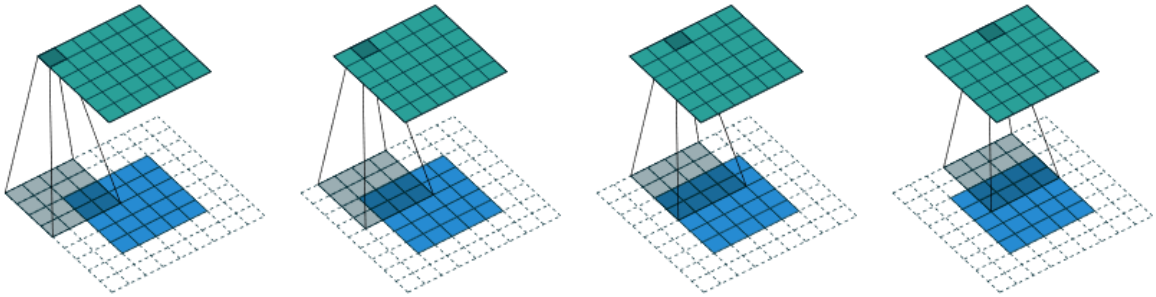


Şekil 2.22. Dolgu kullanılmayan evrişim örneği [43]

Dolgu, girdinin etrafının sıfırlar ile doldurması işlemidir. Dolgu kullanılmasının sebebi genellikle evrişim işlemi sonrasında çıktının boyutunun kontrol edilmek istenmesidir. Dolgunun çıktı boyutuna etkisi, girdi boyutu (i), filtre boyutu (k), adım sayısı 1 ve herhangi bir dolgu (p) değeri için sonuç Denklem (2.40) ile hesaplanır [43]:

$$o = (i - k) + 2p + 1 \tag{2.40}$$

Şekil 2.23'te girdi genişliği (i) 5 ve filtre genişliği (k) 4 için, adım sayısı (s) 1 ve dolgu (p) 2 olduğunda çıktı $o = (i - k) + 2p + 1 = (5 - 4) + 2x2 + 1 = 6$ olur [43].

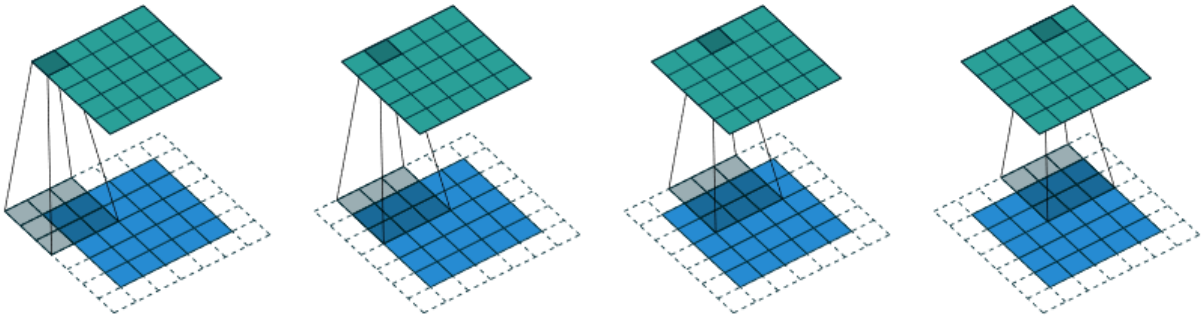


Şekil 2.23. İsteğe bağlı dolgulu evrişim örneği [43]

Evrişim sonucu girdi boyutu ile çıktı boyutunun aynı olması istendiğinde dolgu değeri 1 olmalıdır. Bu duruma Yarım (Aynı) Dolgu (Half-Padding) denir ve matematiksel olarak aşağıdaki gibi ifade edilebilir. Herhangi bir i ve k değeri için $(k\ değeri\ tek\ ise;\ k = 2n + 1, n \in N)$ adım sayısı (s) = 1 ve dolgu $p = [k/2] = n$ olmak üzere sonuç Denklem (2.41) ile hesaplanır [43]:

$$o = i + 2\left[\frac{k}{2}\right] - (k-1)$$
$$o = i + 2n - 2n \tag{2.41}$$
$$o = i$$

Şekil 2.24'te girdi genişliği (i) 5 ve filtre genişliği (k) 3 için, adım sayısı (s) 1 ve dolgu (p) 1 olduğunda sonuç $o = i = 5$ olur [43].

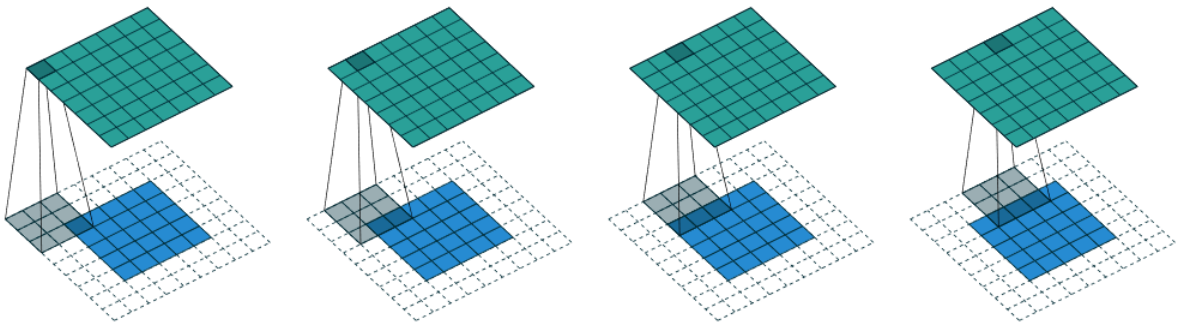


Şekil 2.24. Yarım (Aynı) dolgulu evrişim örneği [43]

Evrişim genel olarak çıktının boyutunu girdiye göre düşürse de bazen bu durumun tersine de ihtiyaç duyulabilir. Çıktı boyutunu artırmak için uygun dolgu değeri tercih edilmelidir. Herhangi i ve k değeri için adım sayısı 1 ve dolgu $p = k - 1$ olmak üzere sonuç Denklem (2.42) ile hesaplanır [43]:

$$o = i + 2(k-1) - (k-1)$$
$$o = i + (k-1) \tag{2.42}$$

Şekil 2.25'te girdi genişliği (i) 5 ve filtre genişliği (k) 3 için, adım sayısı (s) 1 ve dolgu (p) 2 olduğunda sonuç $o = i + (k-1) = 5 + (3-1) = 7$ olur [43].



Şekil 2.25. Tam dolgulu evrişim örneği [43]

Yukarıdaki açıklamalar adım sayısı 1 olan evrişimler için geçerlidir. Adım sayısı birden farklı olduğunda Denklem (2.43)'daki genelleştirilmiş ifade kullanımalıdır.

Herhangi girdi boyutu (i), filtre boyutu (k), dolgu değeri (p) ve adım sayısı (s) için sonuç Denklem (2.43) ile hesaplanır [43] :

$$o = \left[\frac{i + 2p - k}{s}\right] + 1 \tag{2.43}$$

2.2.2.2. Havuzlama Boyutu ve Adım Sayısı

Yapay sinir ağlarında havuzlama katmanı, girdinin küçük yön değiştirmelerine karşı çıktının değişmemesini sağlar. En çok kullanılan havuzlama türü, girdiyi parçalara ayırıp her bir parçanın sadece en büyük değerini alan maksimum havuzlamadır. Havuzlama işlemi de evrişim işlemi gibi girdinin farklı parçalarına sürekli uygulanması olduğundan Denklem (2.43)'da tanımlanan ifade burada da kullanabilir. Havuzlama işleminde dolgu değeri sıfır olduğu için ($p = 0$) sonuç Denklem (2.44) ile hesaplanır [43]:

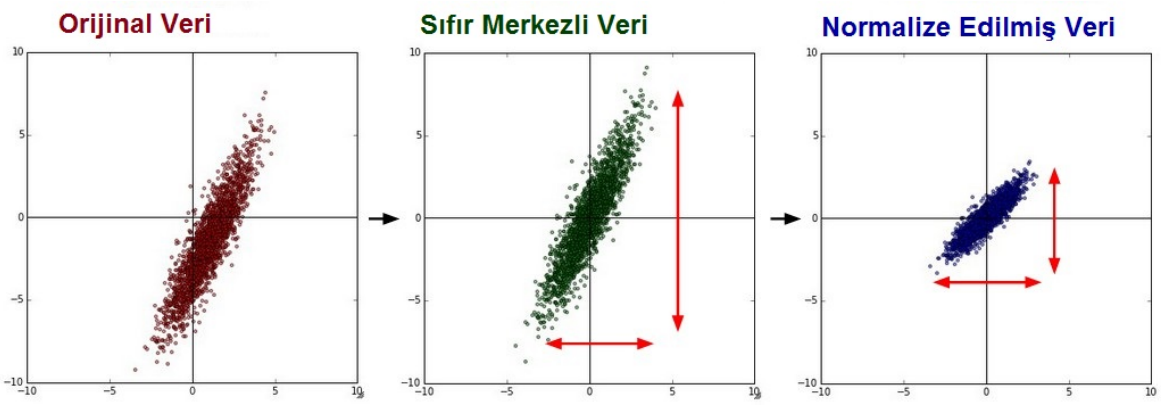
$$o = \left[\frac{i - k}{s}\right] + 1 \tag{2.44}$$

Denklem (2.44) herhangi i, k ve s değeri için geçerli olup herhangi türden bir havuzlama işleminde kullanılabilir [43].

2.2.2.3. Paket Normalizasyonu

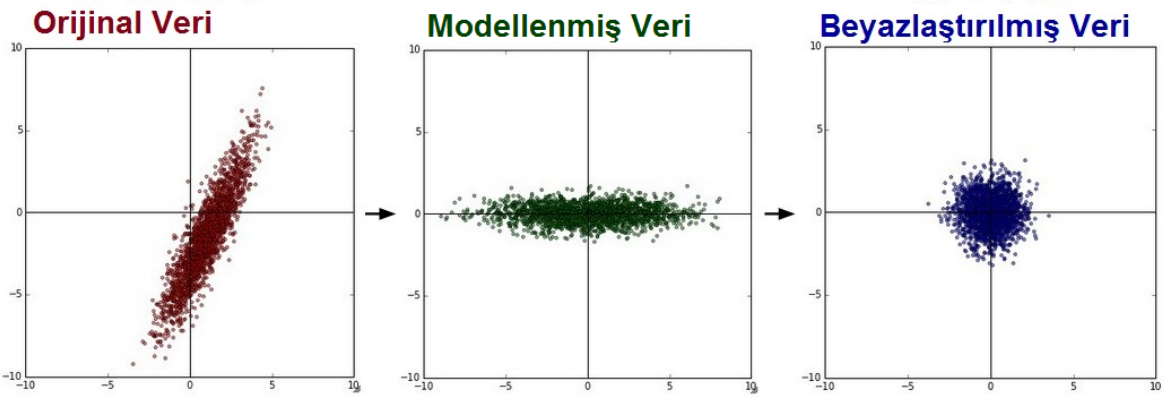
Düzenlileştirme bölümünde bahsedilen düzenlileştirme yöntemlerinden her biri eğitim veri setindeki tüm örneklerin aynı dağılımdan üretildiğini varsaymaktadır. Böyle bir varsayımla veri setini eğitmek ve değişken hiper parametrelere dayalı yorum yapmak net sonuçlar elde etmeye olanak sağlamaz. Derin sinir ağlarının eğitim evresinde girdilerinin beyazlaşması durumunda daha hızlı birleştiği bilinmektedir [46].

Şekil 2.26'da solda orijinal 2 boyutlu giriş verileri, ortada her bir boyuttaki ortalamanın çıkarılmasıyla sıfır merkezli hale getirilmekte ve sağda her boyut standart sapması ile ek olarak ölçeklendirilmektedir. Verilerin kapsamını gösteren çizgiler, ortada eşit olmayan uzunlukta, ancak sağda eşit uzunluktadırlar [10].



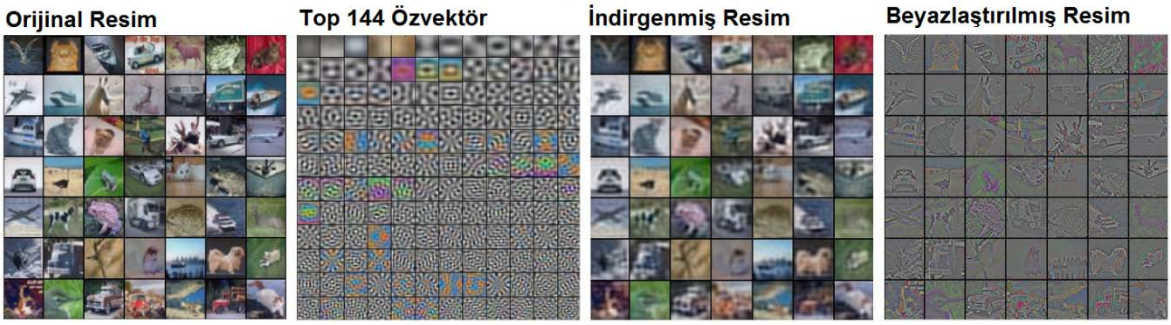
Şekil 2.26. Genel veri ön işleme hattı [10]

Verilerin beyazlaşması, ön işlemenin başka bir şeklidir. Bu işlemde, veriler ilk önce Şekil 2.26'da açıklandığı gibi ortalanır. Daha sonra, verilerdeki korelasyon yapısı hakkında bilgi veren kovaryans matrisi hesaplanır. Bu ön işleme ile veriler arasındaki temel korelasyonu ortadan kaldırılır [10]. Yani beyazlatılmış verinin önceki veriden farkı, sıfır ortalama ve özdeş kovaryans matrisine sahip olmasıdır. Şekil 2.27'de solda orijinal 2 boyutlu giriş verileri ortada PCA'yı gerçekleştirdikten sonra, veriler sıfırda ortalanır ve daha sonra veri kovaryans matrisinin özvektör tabanına döndürülür. Bu görünüm, verileri kovaryans matrisi köşegen olacak şekilde ilişkilendirir. Sağda her bir boyut ek olarak, özdeğerler tarafından veri kovaryans matrisinin kimlik matrisine dönüştürülmesiyle ölçeklendirilir. Bu işlem geometrik olarak, verilerin bir İzotropik Gaussian blobuna (Isotropic Gaussian Blob) gerilmesine ve sıkıştırılmasına karşılık gelir [10].



Şekil 2.27 Beyazlaştırma basamakları [10]

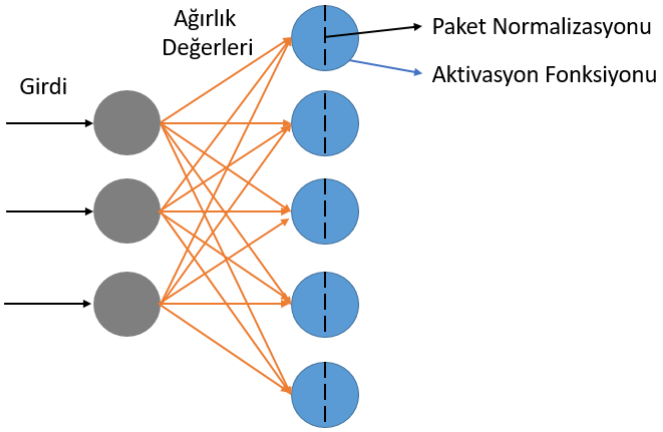
Şekil 2.28’de 49 görüntüden oluşan bir örnek gösterilmektedir. Soldan 2. resimde 3072 öz vektörün en üstteki 144'ü gösterilmektedir. Üst öz vektörler, verilerdeki varyansın çoğunu oluşturur ve görüntülerde daha düşük frekanslara karşılık gelmektedir. Sağdan 2. resimde 49 görüntüden elde edilen 144 öz vektörü, bir görüntünün, belirli bir pikselin bazı konum ve kanaldaki parlaklığı olduğu her bir görüntüyü 3072 boyutlu bir vektör olarak ifade etmek yerine, her bir görüntü yalnızca 144 boyutlu bir vektörle temsil edilmesi gösterilmektedir. Sağ resimde 144 boyutun her biri boyunca varyansın eşit uzunluğa indirgenmesi işlemi gösterilmektedir [10].



Şekil 2.28 Örnek beyazlaştırılmış resimler [10]

Derin sinir ağlarında girdilerin beyazlaştırılması belirli bir aşamaya kadar giriş katmanı için varyasyonlar elde ederek gerçekleştirilir. Her katmanın girdilerinin tam olarak beyazlaşması maliyetli olması ve her yerde farklılaşmanın gerçekleşmemesinden, iki basitleştirme işlemi yapılır. Birincisi, katman girişleri ve çıkışlarındaki özellikleri ortaklaşa beyazlaştırmak yerine, her bir skaler özelliği, sıfır değerine ve 1'in varyansına sahip olacak şekilde, birbirinden bağımsız olarak normalleştirecektir [46]. Sinir ağın gizli katmanları, bu katmanların dağılımındaki değişime bağlı olarak yeni dağılımın uyarlanması için sürekli bir çaba harcamak zorunda kalır. Gizli katmanların girdilerinin dağılımının daha istikrarlı olması, eniyileyicinin doygun rejimde sıkışmasına yani optimize edebileceğinin ulaşabileceği maksimum seviyeye ulaşmasına yardımcı olur [46]. İç katmanların girişlerinin dağılımını düzeltmek için normalleştirme uygulanır. Her aktivasyon bağımsız olarak paket normalizasyonu kullanarak normalleştirme işlemi yapar [47]. Paket normalizasyonu işleminde girdi olarak aldığımız parametrelerin eğitilebilir veya eğitilemez olması paket normalizasyon işleminde kullanılan ortalama ve varyans vektörüne bağlıdır.

Bir yapay sinir ağında paket normalizasyonu işlemi Şekil 2.29’da gösterilmiştir.



Şekil 2.29. Bir yapay sinir ağında paket normalizasyonu işlemi

İç katmanların girdilerinin dağılımını düzeltmek için normalizasyon uygulanır. Normalizasyon işleminde ağ modellerinin eğitimi, paket halindeki verileri daha küçük paketlere ayırarak uygulanmaktadır. Her bir aktivasyon fonksiyonu öncesinde bağımsız bir şekilde gerçekleşen toplu normalizasyon uygulanmasında, normalleştirilmek istenen katman m boyutlu $x = (x_1, x_2 \ldots\ldots x_m)$ olduğunda k boyutunda normalleştirme işlemi için Çizelge 2.4'teki denklemler kullanılır.

Çizelge 2.4. Paket normalizasyonda kullanılan denklemler

Denklem	Açıklama
$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$	μ ortalama değeri, x_i girdi verilerini, m boyutu ifade etmektedir. Küçük paket değerlerinin ortalama değerini hesaplamaktadır.
$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\beta)^2$	σ varyans değerini ifade etmektedir. Mini paket değerlerinin varyans değerini hesaplamaktadır.
$\hat{x}_i \leftarrow \frac{x_i - \mu}{\sqrt{\sigma_\beta^2 + \varepsilon}}$	$\hat{x}_i$ normalizasyon işlemini ifade etmektedir. ε çok küçük bir değer almaktadır.
$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$	γ, β öğrenilebilir parametreler olmak üzere paket normalizasyon işleminin çıktısını (y_i) vermektedir.

Ayrıca normalize edilmiş değerlerin ölçeklendirilmesi ve değiştirilmesi gerekmektedir. Aksi takdirde, bir katmanın normalleştirilmesi, o katmanın temsil edebileceği parametreler açısından sınırlayacaktır. Örneğin, girişleri bir sigmoid fonksiyonuna normalleştirilirse, çıkış sadece doğrusal bölgeye bağlanacaktır. Yani kullanılan normalleştirme yöntemi, çıkışta elde edilecek veriyi doğrudan

etkilemektedir. Böylece normalleştirilmiş giriş yani xk, çıktı yani yk ifadesine dönüştürülür:

$$y^k = \gamma^k \hat{x} + \beta^k \quad (2.45)$$

Denklem (2.45)'da γ ve β öğrenilecek parametrelerdir. Yukarıda açıklandığı gibi x'den y'ye dönüşüm, paket normalizasyon dönüşümü olarak adlandırılır. Paket normalizasyon, bir ağın her katmanının kendi başına diğer katmanlardan daha bağımsız olarak öğrenmesini sağlar. Paket normalizasyon dönüşüm işlemi, ağ eğitiminde kullanıldığı için farklılaşabilir ve katmanların daha az değişim gösteren girdi dağılımlarını öğrenebilir. Böylece eğitimi hızlandırıcı bir etki yaratabilir [46].

2.2.2.3.1. Evrişimli Ağlarda Paket Normalizasyonu

$x = g(Wu + b)$ ifadesi için W ve b öğrenilecek parametreleri, u ise katman girdileridir. g doğrusal olmayan ve bir önceki katmanın çıktısı tarafından gerçekleştirilen işlemdir. $x = (Wu + b)$ işleminin normalleştirilmesi ve doğrusallaştırılması ile paket normalizasyonu elde edilir. $x = (Wu + b)$'yi normalleştirirken, normalleşme adımı sırasında ihmal edileceği için b terimi göz ardı edilmektedir (b'nin etkisi kullanılan parametreye dahil edilmiş olarak ifade edilir).

$z = g(BN(W_u))$ ifadesi normalizasyon işlemi sırasında evrişim özelliğiyle birlikte hareket eder. Yani bir özellik haritasının farklı elemanlarını, farklı konumlarda aynı normalizasyon işlemi ile normalize eder. Bu nedenle bir mini-paket, bir özellik haritasında, tüm konumlar üzerinde ortaklaşa normalleştirilir ve her bir aktivasyon başına parametreler (γ, β) öğrenilir.

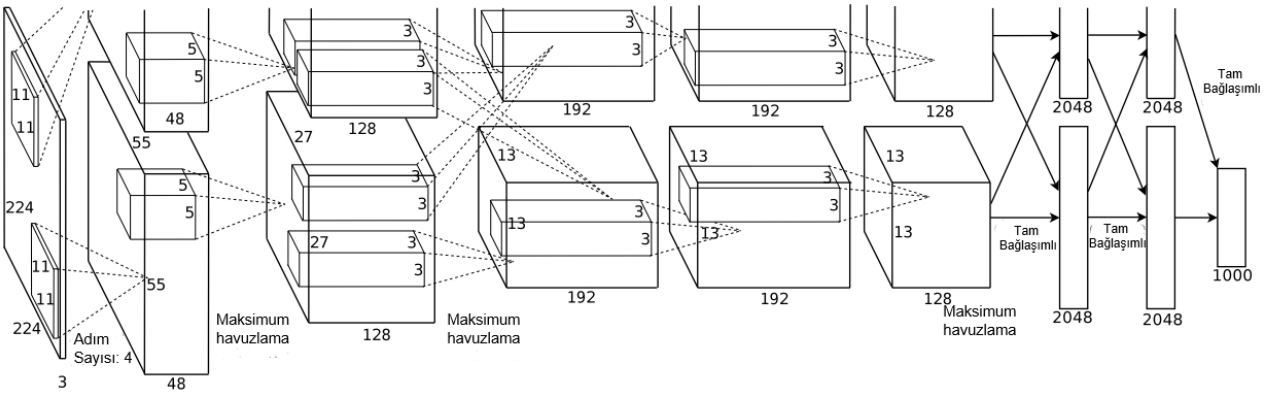
2.2.2.3.2. Paket Normalizasyon İşleminin Avantajları

Paket normalizasyon işlemi, gizli katmanlara gelen verilerin kovaryans kaymasını azaltır. Kovaryans değişimi birden fazla boyutun birbiri ile ilişkisi olup olmadığını anlamak için kullanılabilir. Eğimlerin, parametrelerin ölçeğine veya başlangıç değerlerine bağımlılığını azaltır. Modeli düzenli hale getirir ve seyreltme, fotometrik bozunmalar (photometric distortions), lokal cevap normalizasyonu (Local Response Normalization) ve diğer düzenlileştirme tekniklerine olan ihtiyacı azaltır. Yoğunlaştırılmış doğrusal olmayan ve yüksek öğrenme oranlarının kullanılmasına izin verir.

2.2.3. Hazır Evrişimli Sinir Ağları

2.2.3.1. AlexNet Ağı

AlexNet, Şekil 2.30'da görülebileceği gibi toplamda sekiz katmandan oluşmaktadır. İlk beş katman evrişim katmanı, son üç katmanı ise tam bağlaşımlı katmandan oluşturmaktadır. Son katman, 1000 sınıf arasında bir dağılım gerçekleştiren softmax fonksiyonuna bağlıdır [48].



Şekil 2.30. AlexNet ağı [48]

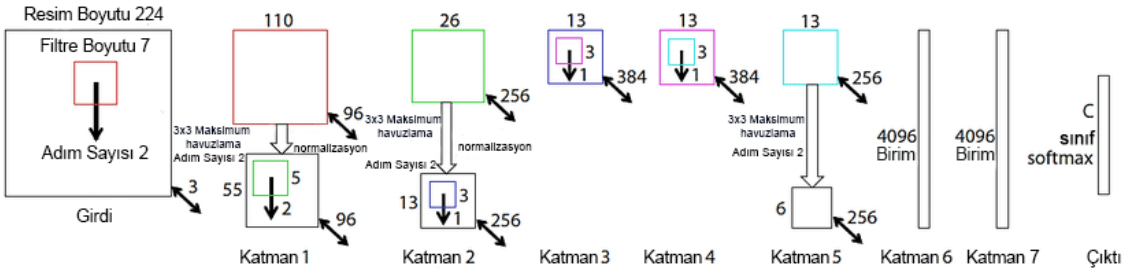
İkinci, dördüncü ve beşinci katmanlara ait filtre, sadece aynı GPU'da bulunan ve kendinden önce gelen katmandaki filtrelere bağlıdır. Üçüncü katman ise ikinci katmandaki tüm filtrelere bağlıdır. ReLU, her katmandan sonra uygulanmaktadır [48].

İlk evrişim katmanının parametreleri; girdi boyutu 224x224x3, filtre boyutu 11x11x3 olan 96 filtre ve adım sayısı 4 pikselden oluşmaktadır. İkinci katman, ilk katmandaki normalize edilmiş ve havuzlama uygulanmış çıktılara 256 adet 5x5x48 filtre uygular. Üçüncü katman 3x3x256 boyutlarında 384 adet filtreyi ikinci katmanın normalize edilmiş ve havuzlama uygulanmış çıktılarına uygulanır. Dördüncü katman 384 adet 3x3x192 boyutlarında, beşinci katman 256 adet 3x3x192 boyutlarında filtre uygular. Tam bağlaşımlı katmanda bulunan her bir katman 4096 nöron içerir [48].

2.2.3.2. ZFNet Ağı

Bu ağ modelinde AlexNet'te tanımlanan evrişim ağ yapısı kullanılmıştır. Ağın çıktısı ve gerçek değeri arasında karşılaştırma yapmak için sınıflandırma problemine uygun olan çapraz düzensizlik maliyet fonksiyonu seçilmiştir. Şekil 2.31'de gösterilen ağın yapısı AlexNet ağına çok benzemektedir. AlexNet'ten farkı, orta katmanlardaki filtrelerin boyutu artırılırken ilk katmandaki filtre boyutu ve adım sayısı

düşürülmüştür [38].



Şekil 2.31. ZFNet ağı

Şekil 2.31'de görüldüğü üzere ZFNet, 8 evrişim katmanlı bir ağdır. Girdi olarak yukarıda verilen AlexNet ağındaki gibi 3 renk kanallı, 224x224 boyutlarındaki resimleri kabul eder. Bu ağda filtre boyutu 7x7x3 olan 96 adet filtre, adım sayısı 4 kullanılarak evrişim işlemine sokulmaktadır. Evrişim işlemi sonucunda oluşan çıktı öncelikle ReLU aktivasyon fonksiyonundan geçirilmekte ve daha sonra 3x3 boyutunda adım sayısı 2 olacak şekilde havuzlama işlemi uygulanmaktadır. Havuzlama işlemi sonucunda 96 tane farklı 55x55'lik çıktı elde edilmesi için bütün çıktıda tonlama normalizasyonu yapılmaktadır [49].

Benzer işlemler 2, 3, 4 ve 5 numaralı katmanlarda da uygulanmaktadır. Sondaki iki katman, en üst evrişim katmanından çıkan öznitelikleri vektör formatında girdi olarak alan tam bağlaşımlı katmanlardır. En son katmanda ise C-yollu softmax fonksiyonu (C: sınıf sayısı) bulunmaktadır [49].

2.2.3.3. VGGNet Ağı

VGGNet, AlexNet ağı ile aynı prensiplere dayanılarak üretilmiştir. Ağın girdisi 224x224 boyutlarından oluşan RGB formatında bir resimdir. Resim ağa verilmeden önce resimdeki her pikselden, eğitim setindeki resimler ile hesaplanan ortalama resme ait piksel değerleri çıkartılır. Ağda kullanılan evrişimlerin boyutları 3x3 şeklindedir. Derinlikleri ise çeşitli değerler verilerek uygulanmıştır. Bir konfigürasyonda ek olarak 1x1 boyutlarında evrişim de uygulanmıştır. Filtre adım sayısı 1 piksel ile sabitlenmiştir. Dolgu değeri ise çıktının boyutları değişmeyecek şekilde ayarlanmıştır. Bazı evrişim katmanlarından sonra toplamda beş kez maksimum havuzlama uygulanmıştır. Maksimum havuzlama, 2x2 boyutlarında ve adım sayısı 2 olarak uygulanmıştır. Evrişim katmanlarını tam bağlaşımlı katmanlar izlemiş ve ilk iki tam bağlaşımlı katmanda her birinde 4096 adet, son katmanında ise 1000 adet nöron kullanılmıştır. Son katman softmax katmanıdır [45].

ILSVRC veri setinde performansı arttırmaması ve gereksiz hafıza tüketimine yol açması nedeniyle bir konfigürasyon dışında, bu ağ yapısında lokal cevap normalizasyonu (LRN) kullanılmamıştır [45]. Bu ağa ait tüm konfigürasyonlar Çizelge 2.5'te verilmiştir.

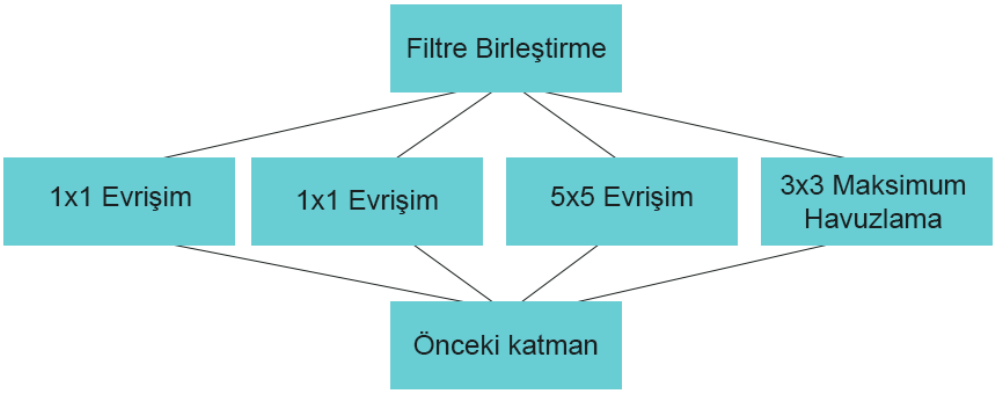
Çizelge 2.5. VGG ağ konfigürasyonları

VGG Konfigürasyonları					
A	A-LRN	B	C	D	E
11 Ağırlık Katmanı	11 Ağırlık Katmanı	13 Ağırlık Katmanı	16 Ağırlık Katmanı	16 Ağırlık Katmanı	19 Ağırlık Katmanı
Girdi (224 x 224 RGB Resim)					
evriş3-64	evriş3-64 **LRN**	evriş3-64 **evriş3-64**	evriş3-64 evriş3-64	evriş3-64 evriş3-64	evriş3-64 evriş3-64
Havuzlama					
evriş3-128	evriş3-128	evriş3-128 **evriş3-128**	evriş3-128 evriş3-128	evriş3-128 evriş3-128	evriş3-128 evriş3-128
Havuzlama					
evriş3-256 evriş3-256	evriş3-256 evriş3-256	evriş3-256 evriş3-256	evriş3-256 evriş3-256 **evriş1-256**	evriş3-256 evriş3-256 **evriş3-256**	evriş3-256 evriş3-256 evriş3-256 **evriş3-256**
Havuzlama					
evriş3-512 evriş3-512	evriş3-512 evriş3-512	evriş3-512 evriş3-512	evriş3-512 evriş3-512 **evriş1-512**	evriş3-512 evriş3-512 **evriş3-512**	evriş3-512 evriş3-512 evriş3-512 **evriş3-512**
Havuzlama					
evriş3-512 evriş3-512	evriş3-512 evriş3-512	evriş3-512 evriş3-512	evriş3-512 evriş3-512 **evriş1-512**	evriş3-512 evriş3-512 **evriş3-512**	evriş3-512 evriş3-512 evriş3-512 **evriş3-512**
Havuzlama					
TB-4096					
TB-4096					
TB-1000					
softmax					

2.2.3.4. GoogleNet Ağı

2014 yılında yapılan ILSVRC14 yarışmasında birinci gelen bu ağ, hesaplama için gereken maliyeti arttırmadan daha derin ve daha geniş bir yapı sunmaktadır. 22 adet katmandan oluşan ağa Inception adı verilmiştir. Bu isim Lin ve arkadaşlarının yaptığı bir çalışmada kullandıkları "Daha derine gitmeliyiz" internet capsinden türetilmiştir [50].

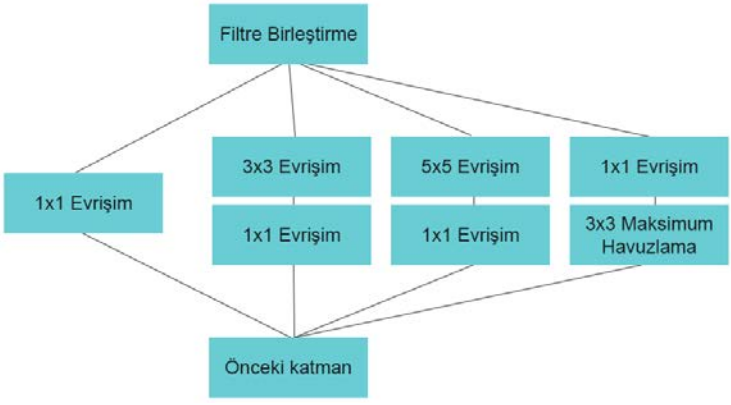
Bu ağ temelde, aynı girdiye uygulanan havuzlama işlemlerini ve birden fazla evrişim işleminin uygulanmasını esas alır. Girdiye aynı anda maksimum havuzlama ve 1x1, 3x3 ve 5x5 boyutlarında filtre ile evrişim işlemi uygulanır ve bu işlemlerin sonucu birleştirilerek tek bir sonuç elde edilir. Bu özelliği sayesinde ağ, aynı anda hem genel hem de spesifik özellikleri çıkarabilmektedir [50].



Şekil 2.32. Basit Inception yapısı

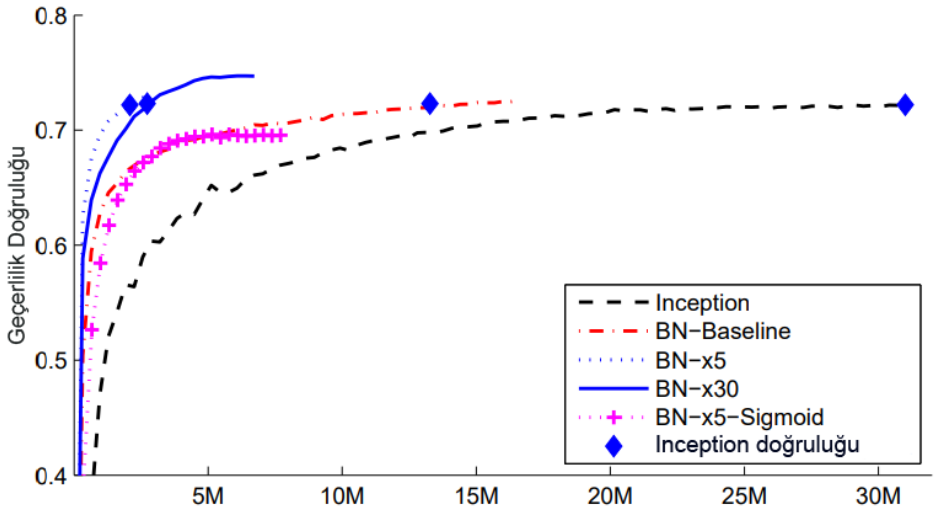
Şekil 2.32'de bahsedilen bu ağ ile ilgili en büyük problem 5x5 evrişimde sayısı az olsa bile çok sayıda filtreli bir evrişim tabakasının üzerinde bulunmasından dolayı hesaplama açısından maliyeti artmaktadır. Bu problem havuzlama katmanları da eklendiğinde daha da belirgin bir hale gelmektedir. Her bir aşamada bu evrişim katmanının çıktısı ve havuzlama katmanının çıktısını birleştirmek çıktı boyutunda kaçınılmaz bir büyümeye sebep olmaktadır [50].

Ağın hesaplama maliyetini düşürmek için her işlemden önce boyut azaltma işlemi yapılmaktadır. Şekil 2.33'te gösterildiği gibi 5x5 ve 3x3 filtre uygulamadan önce 1x1 boyutlarında filtre uygulayarak boyut düşürülmüştür [50].



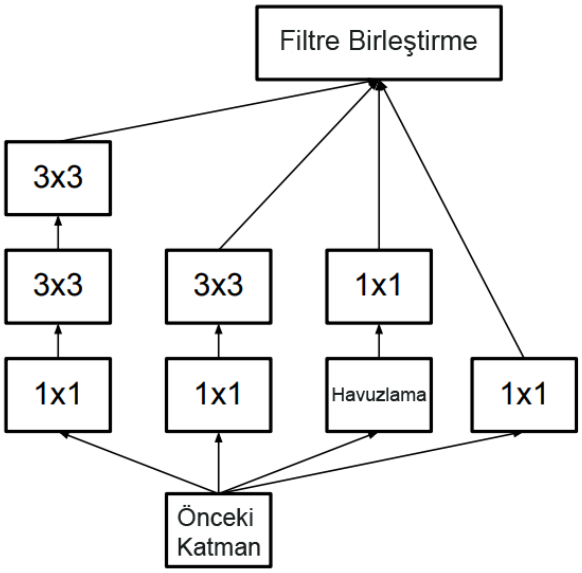
Şekil 2.33. Boyut düşürme işleminden sonra Inception yapısı

Şubat 2015'te bu Inception V2 ağı duyurulmuştur. Bu ağ, ilk ağdan farklı olarak paket normalizasyonu içermektedir. Paket normalizasyonu kısaca her katmanın çıktısında ortalama ve standart sapmanın hesaplanması ve bu değerlerle çıktıların normalleştirilmesidir. Bu işlem tüm sinir ağının aynı aralıkta ve sıfır ortalama ile bir sonuç vermesini sağlar. Şekil 2.34'te Inception ağının farklı paket normalizasyon değerleri ile karşılaştırılması görülmektedir [51].



Şekil 2.34. Inception ağının farklı paket normalizasyon değerleri ile karşılaştırılması [51]

Yine 2015 yılında Inception V3 ağı yayınlanmıştır. Inception V3'te, ağın genişlik ve derinliği eniyilenerek ağdan maksimum bilgi akışı hedeflenmiştir. Derinlik arttıkça ağın genişliği de sistematik olarak artmaktadır. 5x5 ve 7x7 boyutlarında filtre yerine onları karşılayacak iki ya da üç tane 3x3 boyutlu filtreler kullanılmaktadır. Inception V3 ağı Şekil 2.35'te gösterildiği gibidir [44].

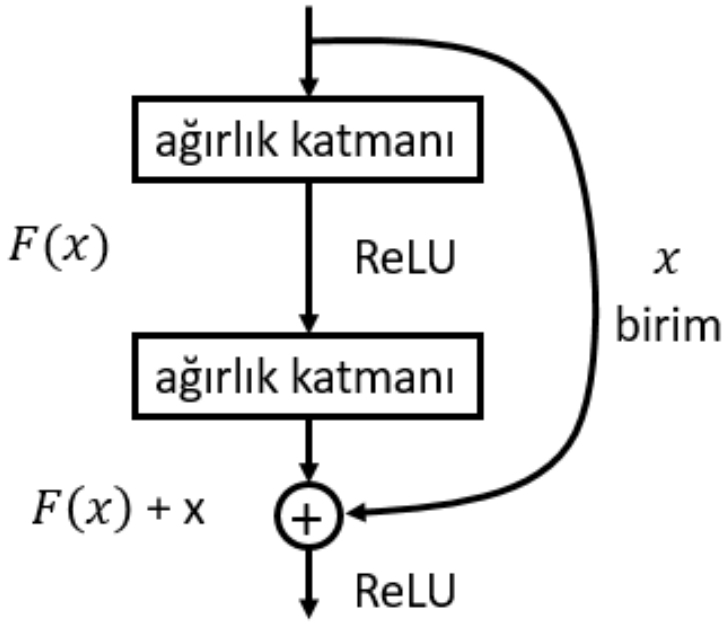


Şekil 2.35. Inception V3 ağı

Geleneksel ağlardan farklı yapıya sahip Resnet ağının, 2015 yılında birinci olmasından sonra daha hızlı eğitim yapılmasını sağlamak maksadıyla Inception yapısıyla kalıntı (residual) ağ bağlantıları birleştirerek Inception V4 ağı geliştirilmiştir [52].

2.2.3.5. Microsoft Resnet Ağı

Şekil 2.36'da görülen klasik ağa bazı kısa yollar eklenilmesiyle elde edilen bu ağ kalıntı bloklardan oluşmaktadır. Kalıntı blokta girdi olarak x değeri alınıp evrişim-aktivasyon-evrişim serisinden geçirilerek bir $f(x)$ fonksiyonu elde edilmektedir. Daha sonra $f(x)$ fonksiyonuna orijinal girdi olan x değeri eklenerek $h(x) = f(x) + x$ üretilmektedir. Klasik evrişim işleminde $h(x)$ fonksiyonu $f(x)$ fonksiyonuna eşitken bu ağda, girdiye evrişim işlemi uygulandıktan sonra orijinal veri de eklenmektedir [53]. Ağı oluşturan blok diyagramı Şekil 2.36'da gösterilmiştir.



Şekil 2.36. Kalıntı (Residual) öğrenme bloğu

2.2.3.6. Trimps-Soushen

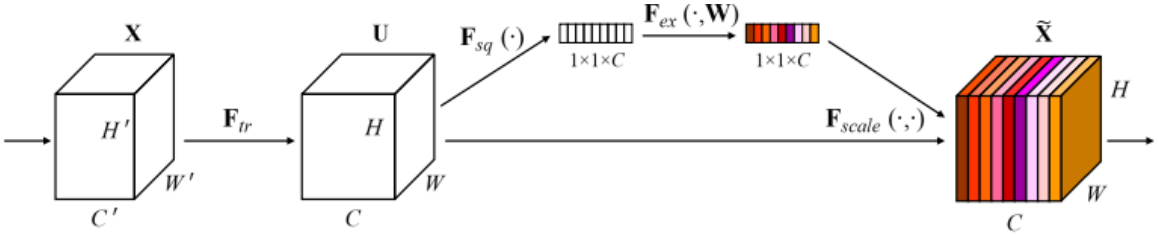
Trimps-Soushen, ILSVRC2016 yarışmasını kazanan ekibin adıdır. Trimps-Soushen, resimlere ait sınıf etiketlerinin tahmin edilmesi görevi için resim sınıflandırma tabanlı Inception, Inception-Resnet, ResNet ve Geniş Kalıntı Ağ (Wide Resiudal Network - WRN) ağlarını kullanarak resim sınıflandırması yapmakta ve nesnelerin konumunu “Faster R-CNN” algoritması kullanılarak tespit etmektedir. Birçok modelden elde edilen sonuçlar farklı şekillerde birleştirilerek model tutarlılığı, ağırlıklar olarak kullanılmaktadır [54].

2.2.3.7. SENet (Squeeze-and-Excitation Networks)

Evrişimli sinir ağları, evrişim işlemine dayanmaktadır. Bu işlemde bölgesel ve renk kanalı tabanlı bilgiler birleştirilerek bilgi niteliği taşıyan öznitelikler ortaya çıkarılmaktadır. Bu ağlar, çıktıları olarak öznitelik haritalarını oluştururken, bütün kanalları eşit olarak ağırlıklandırmaktadır. Bundan farklı olarak SENet ağları, her bir kanalın önemini adaptif olarak belirtmek için içerik farkındalığı olan bir mekanizmaya sahiptir. Bunun için yapılan şey ise her bir kanala tek bir parametre eklenmesi ve o kanalın ne kadar alakalı olduğunu gösteren bir doğrusal ölçütün sunulmasıdır [55].
SENet, ESA’lar için kanallar arası bağımlılıkları neredeyse hiç hesaplama maliyeti yaratmadan geliştiren bir bloktur. Buradaki mantık, her evrişim bloğunun her bir kanalına parametrelerin eklenmesi ve böylece ağdaki her bir öznitelik haritasının

ağırlıklandırmasının adaptif olarak ayarlanmasıdır [55].

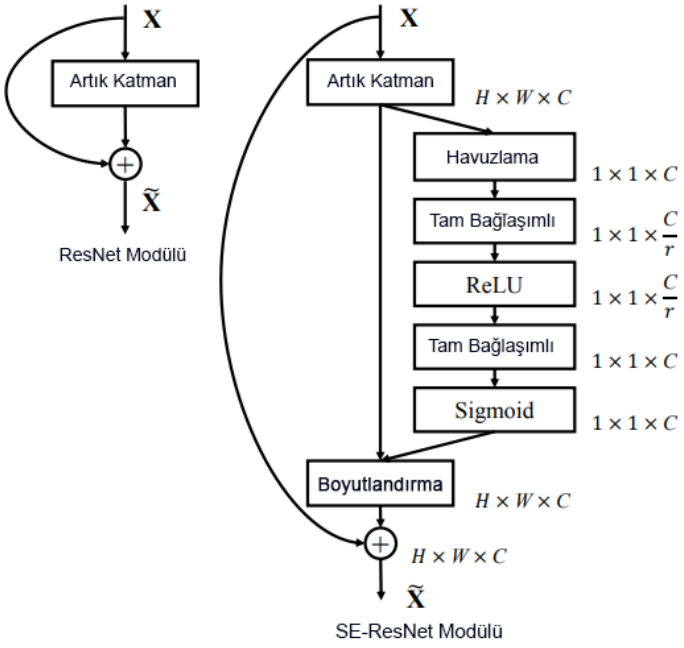
Şekil 2.37'de gösterilen Squeeze-and-Excitation bloğu, bir hesaplama birimidir. SENet'teki evrişim katmanlarında bu bloğun kullanılmasının amacı, ağ içerisinde daha sonraki dönüşümlerde kullanılacak işe yarar özniteliklere karşı ağın hassasiyetini artırmak ve daha az işe yarar olanları bastırmaktır. Bahsedilen blok bu işlemi, sonraki dönüşümlere verilmeden önce filtrelendirme sonuçlarının yeniden ayarlanması için kanallar arası bağımlılıkları belirgin bir şekilde biçimlendirerek yapar [56].



Şekil 2.37. Squeeze-and-Excitation blokları [56]

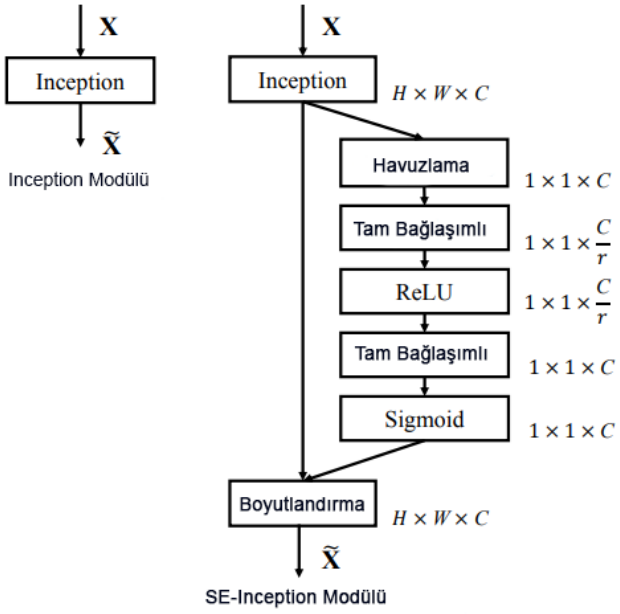
SENet ağlarında yapılan işlemler sırasıyla:

1. Şekil 2.38'de gösterilen bir evrişim bloğu ve o bloğun sahip olduğu kanal sayısı alınır.
2. Ortalama havuzlama yapılarak her bir kanalın değeri sıkıştırılarak (squeeze) tek bir sayısal değer elde edilir.
3. ReLU fonksiyonunu takip eden bir tam bağlaşımlı katman gerekli olan doğrusallığı sağlar.
4. Sigmoid fonksiyonunu takip eden ikinci bir tam bağlaşımlı katman, her bir kanala düzgün bir geçit işlevi verir.
5. En sonunda, evrişim bloğunun her bir öznitelik eşleştirmesi, yan ağın sonuçlarına dayalı olarak ağırlıklandırılır.



Şekil 2.38. Orjinal kalıntı modülü (ResNet) ve SE-ResNet modülü [56]

Bütün bu işlemlerin toplam hesaplama maliyetine katkısı %1'den daha düşüktür ve bu işlemler bütün modellere eklenebilir [55]. Şekil 2.39'da görülen bütün r değerleri 16 olarak belirlenmiştir. Ağ, girdi olarak 224x224 boyutlarında resim almaktadır [56].



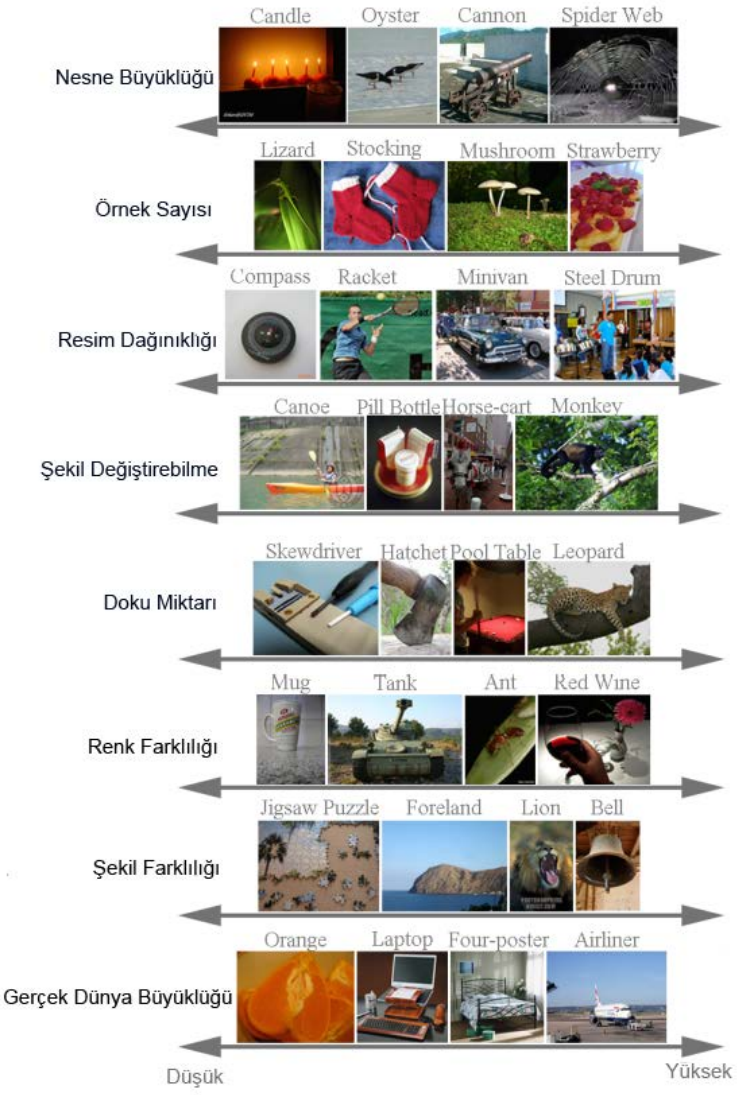
Şekil 2.39. Orijinal Inception modülü ve SE-Inception modülü [56]

2.3. Veri Setleri

2.3.1. ImageNet Veri Seti

ImageNet veri seti şu anda her yıl yapılan ILSVRC'de (ImageNet Large Scale Visual Recognition Challenge) kullanılmaktadır. Bu yarışmada, yarışmacılara resim sınıflandırma, tek objenin yerini belirleme ve nesne tanıma gibi görevler ve her bir görev için farklı veri seti ve etiketler verilmektedir. 2012-2014 yılları arasında kullanılan veri seti toplamda 1000 sınıf, yaklaşık 1,28 milyon eğitim verisi, 50 bin doğrulama verisi, 100 bin test verisinden oluşmaktadır.

Her bir resimdeki nesne sayısı bir veya birden fazla olabilmektedir. Her bir nesnenin resim içerisinde bulunma sıklığı da değişebilir. Şekil 2.40'da veri setinin çeşitliliği göstermektedir.



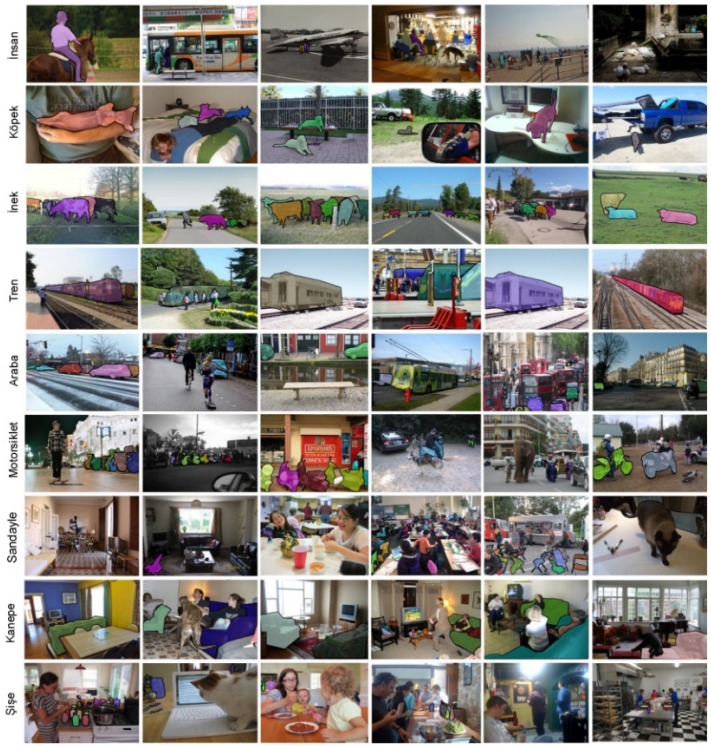
Şekil 2.40. ImageNet veri setinin çeşitliliği [57]

PASCAL veri setine kıyasla ImageNet veri seti daha iyi sınıflandırılmış veri içerir. Örneğin “kedi” sınıfa kedi türleri ve kedi renkleri gibi sınıflar da eklenmiştir. Bu detaylı sınıflandırmadan dolayı bu veri seti araştırmacılar tarafından sıklıkla kullanılmaktadır.

Bilinen bir çok derin öğrenme ağı, AlexNet [48], VGG [45], ResNet [53] gibi, kıyaslama yapmak için bu veri setini kullanmaktadır. Ayrıca bazı araştırmacılar bu veri setindeki resimleri ağa vermeden önce ön işleme yapmışlardır. Zhai [58], standart ImageNet veri setindeki resimleri 64x64 boyutlarına kırpmış ve resimlerin daha az bilgi içermesine rağmen ağın hala yüksek doğrulukta çalıştığını belirtmiştir.

2.3.2. MS COCO Veri Seti

Microsoft Common Objects in Context (COCO) veri seti 91 sınıf içermekte ve bunların 85 tanesi 5000’den fazla etiketlenmiş veri içermektedir. Toplamda, veri setinde 328 bin resimde 2 milyondan fazla etiketlenmiş nesne bulunmaktadır. ImageNet veri setinden daha az sınıf bulundursa da COCO veri seti, her kategori başına daha fazla resim bulundurmaktadır. Ayrıca kategori başına örnek sayısında SUN ve PASCAL veri setinden önemli derecede fazla veri barındırmaktadır. Şekil 2.41’de MS COCO veri setinden örnek resimler yer almaktadır.

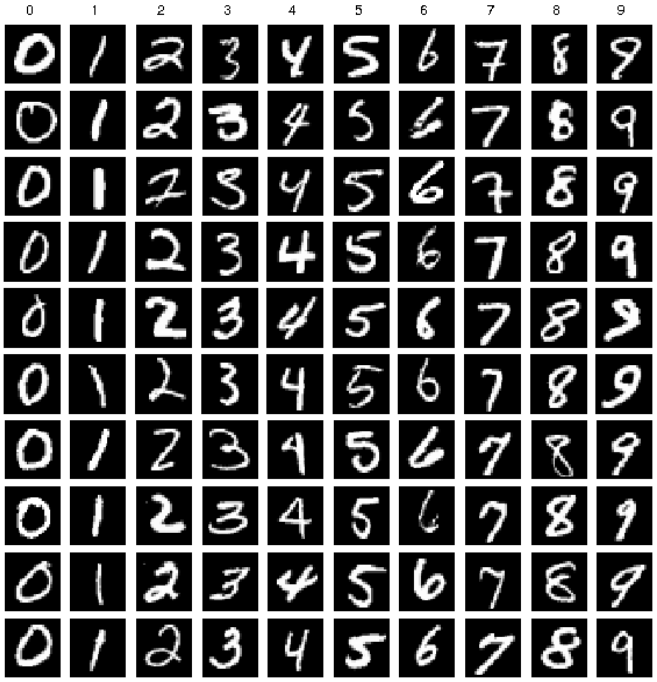


Şekil 2.41. MS COCO veri setine ait bir örnek resimler [59]

2.3.3. MNIST Veri Seti

MNIST, resim işleme ve bilgisayarla görme alanlarında araştırmacıların en eski ve yaygın olarak kullandıkları veri setlerinden biridir. El yazısı rakamları içeren NIST's Special Database 3 ve Special Database 1 veri setlerinden üretilmiştir. Special Database 1 ve Special Database 3 lise öğrencileri ve US Sayım Bürosu çalışanları tarafından yazılmış rakamlardan oluşmaktadır [60]. Şekil 2.42'de örneklerin yer aldığı MNIST veri setinde 60000 adet eğitim verisi ve 10000 adet test verisi bulundurmaktadır [61]. Bu veri setinin karakteristiği, siyah arka plan üzerine beyaz renkte 0'dan 9'a kadar olan sayılardan oluşmaktadır. Her resmin boyutu 28x28 boyutlarındadır. MNIST veri seti, herhangi bir ön işleme veya formatlama gerektirmediğinden, makine öğrenmesi ile bilgisayarla görme ve görüntü işleme alanlarına yeni başlayan kullanıcılar için uygundur.

Bazı araştırmacılar MNIST veri setini ön işlemden geçirmektedir. LeCun [60], MNIST veri setini önceden işleyerek üç farklı versiyon ortaya koymuştur. Bu versiyonlardan birisi resimler 20x20 boyutlarına indirgemektedir. Ciregan [62], resimleri 10,12,14 ve 20 piksele normalize eden bir çalışma yapmıştır. Bir başka çalışmada Tabik [63], MNIST veri setinin evrişimsel sinir ağları için ön işleme tekniklerini ele almıştır.



Şekil 2.42. MNIST veri setinden örnek resimler

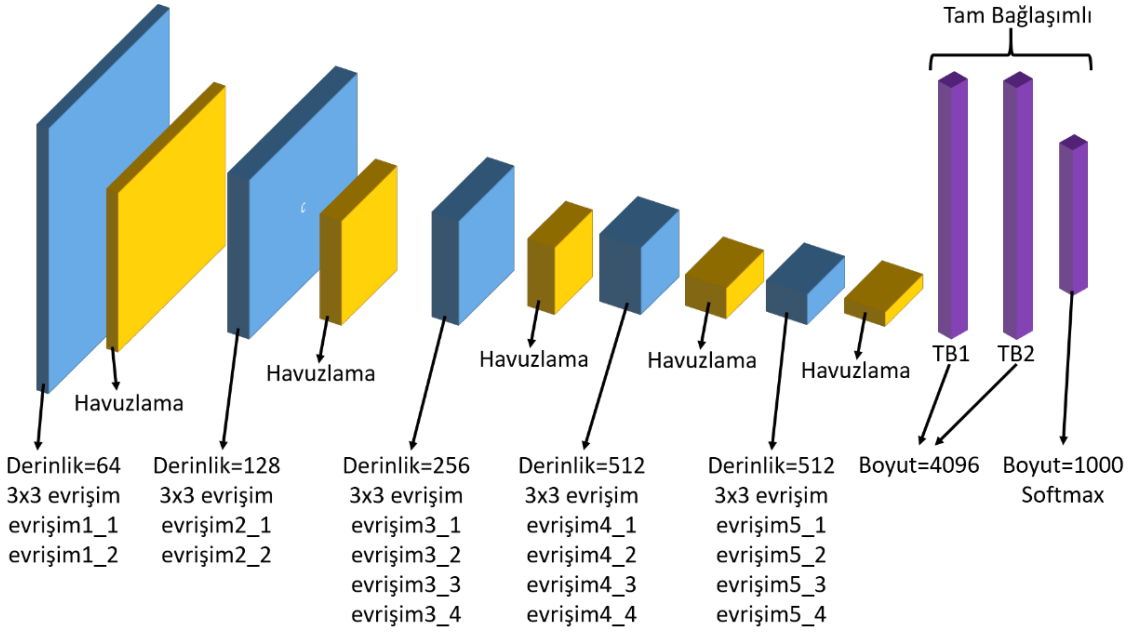
2.3.4. CIFAR-10 / CIFAR-100

CIFAR-10 toplamda 10 sınıf ve 32x32 boyutlarında 60000 adet resimden oluşan, bir veri setidir. 50000 adet eğitim verisi ve 10000 adet test verisi bulunmaktadır. CIFAR-100 veri seti ise her bir sınıf için 600 resim bulundurmakta ve toplamda 100 sınıftan oluşmaktadır. CIFAR-100 içinde bulunan 100 sınıf, 20 adet üst sınıfta gruplandırılmıştır [64]. Şekil 2.43'te CIFAR-10 veri setinden örnek resimler yer almaktadır [54].

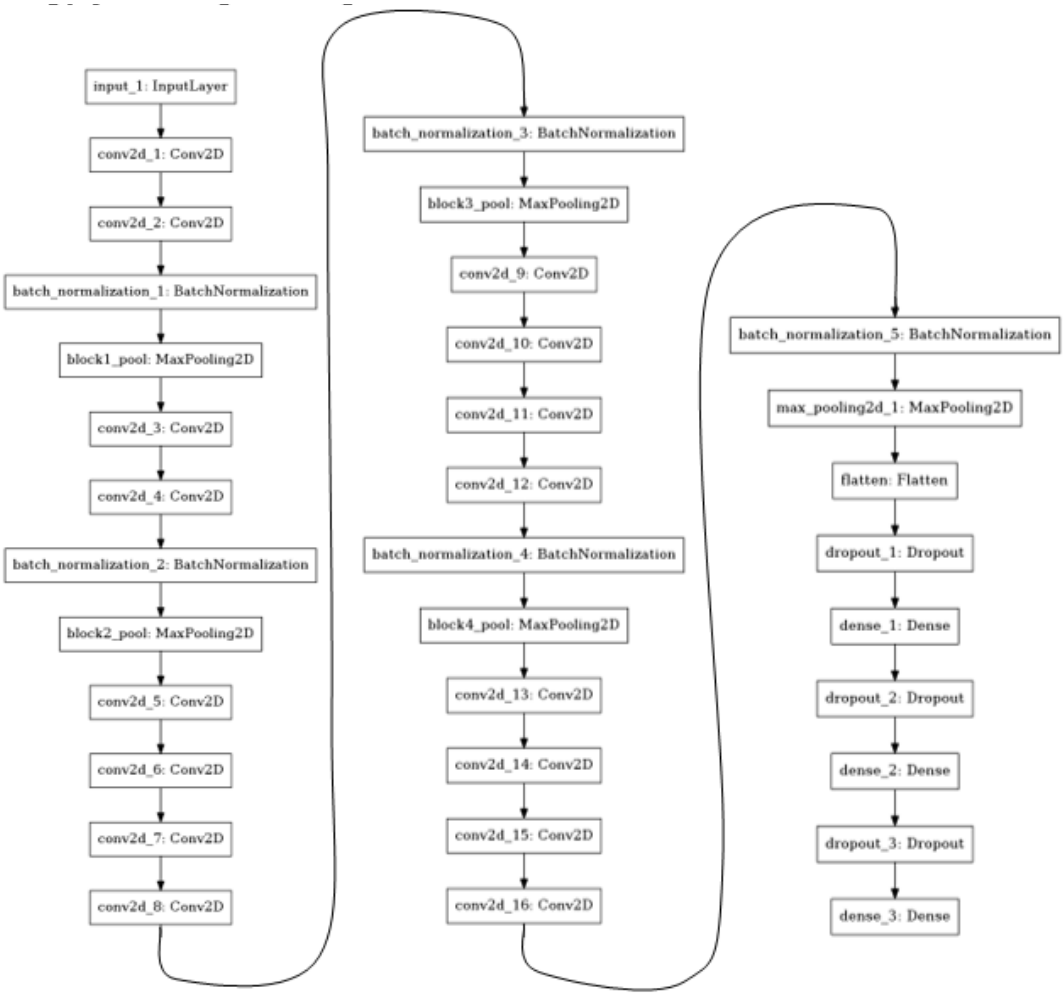
Şekil 2.43. CIFAR-10 veri setinden örnek resimler

3. EVRİŞİMSEL SİNİR AĞLARININ OPTİMİZASYONU

Derin öğrenmede kullanılan hiper parametrelerin eğitime katkısının test edilmesi maksadıyla IMAGE-NET yarışmasında dereceye girmiş ağların incelemesi yapılmıştır. Özellikle hiper parametrelerin kolay düzenlenebildiği VGG-19 modeli seçilmiştir. Bu çalışma kapsamında paket normalizasyonun da incelenebilmesi maksadıyla standart VGG-19 ağ yapısında her havuzlama katmanından önce paket normalizasyon katmanı eklenmiştir. Ayrıca IMAGE-NET verisinde bulunan 1000 sınıf sayısı dikkate alınarak tasarlanan FC katmanında 4096-4096-1000 şeklinde olan sınıflandırma bloğu bu çalışma da 50 sınıf olması nedeniyle 512-256-50 olarak değiştirilmiştir. Standart VGG-19 ağ yapısı Şekil 3.1'de [65], tez çalışması için düzenlenmiş VGG-19 ağ yapısı Şekil 3.2'de görülmektedir.



Şekil 3.1. Standart VGG-19 ağ yapısı



Şekil 3.2. Düzenlenmiş VGG-19 ağ yapısı

3.1. Veri Seti Oluşturulması

ILSVRC2012 eğitim veri seti 138 GB boyutunda 1000 sınıftan oluşmaktadır. Her sınıfta ortalama 1300 adet resim bulunmaktadır.

Tez çalışmasında olabildiğince farklı parametre ile eğitim yapılması hedeflenmiştir. Veri setinin büyüklüğü ve planlanan eğitim sayısı dikkate alındığında eğitimin çok uzun zaman alacağı değerlendirilmiş ve veri setindeki sınıf sayısı ve sınıflardaki resim sayısı azaltılmıştır. Sonuç olarak eğitimlerde Çizelge 3.1'de belirtilen 50 sınıfta 600'er adet resimden oluşan 30000 adetlik bir eğitim veri seti ve yine 50 sınıfta 50'şer adet resimden oluşan 2500 adetlik bir geçerlilik veri seti kullanılmıştır. Her eğitim 100 epok sürecek şekilde planlanmıştır.

Çizelge 3.1. Eğitim veri seti

No.	Etiket	No.	Etiket	No.	Etiket	No.	Etiket	No.	Etiket
1	alp	11	oberman	21	microphone	31	pug	41	tractor
2	ambulance	12	drum	22	missile	32	rule	42	trombone
3	baboon	13	ear	23	mouse	33	schooner	43	umbrella
4	banana	14	fireboat	24	orange	34	screen	44	valley
5	binoculars	15	flamingo	25	oxygen_mask	35	seashore	45	violin
6	bookcase	16	forklift	26	paintbrush	36	soccer_ball	46	volcano
7	cab	17	hammer	27	pelican	37	space_shuttle	47	wall_clock
8	canoe	18	jeep	28	pineapple	38	strawberry	48	wardrobe
9	cello	19	laptop	29	plane	39	tank	49	warplane
10	desk	20	meerkat	30	projector	40	throne	50	white_wolf

Geçerlilik veri setinden örnekler Şekil 3.3'te gösterilmiştir.

Şekil 3.3. Geçerlilik veri setinden resimler

3.2. Derin Öğrenme Kütüphane Seçimi

Derin öğrenme kütüphaneleri, derin sinir ağı tasarlama, eğitme ve geçerlilik işlemleri için kullanışlı yapı taşları sunmaktadır. Derin öğrenme kütüphanelerinin çoğu GPU destekli hızlı eğitim sunmak için cuDNN ve NCCL gibi hızlandırıcılar kullanmaktadır. En popüler derin öğrenme kütüphaneleri arasında TensorFlow (Google), CNTK (Microsoft), Theano, MXNet (Amazon), Caffe (Berkeley AI Research), PyTorch (Facebook), Caffe2 (Facebook-NVIDIA) sayılabilir. Kod yazma tarafında derin

öğrenme kütüphanelerini alt seviyede tutarak üst seviyede API olarak çalışan Keras kütüphanesi yaygın bir kullanıma sahiptir [66].

Tensorflow esnek mimarisi sayesinde birçok farklı platformda (CPU, GPU, TPU) ve masaüstü bilgisayarlar, sunucu kümeleri ile mobil cihazlarda hesaplamanın kolayca yapılmasını sağlamaktadır [67]. Tensorflow'un Keras ile kullanımının sağlamış olduğu kolaylık ve esneklikten dolayı tez çalışmasında üst seviye kütüphane olarak Keras, alt seviye kütüphane olarak Tensorflow kullanılmıştır.

3.3. Hiper Parametre Seçimi

Bu çalışmada olabildiğince farklı parametre ile eğitim yapılabilmesi için Çizelge 3.2 hiper parametre ve seçenekleri çıkarılmıştır. Bu çizelgedeki parametreler kullanıldığında toplamda 288 farklı eğitimin ortaya çıkmaktadır.

Çizelge 3.2. Hiper parametre ve seçenekler

Hiper Parametre	**Seçenek 1**	**Seçenek 2**	**Seçenek 3**
Paket	128	256	
Filtre Sayısı	32	64	
Filtre Boyutu	3	5	
Seyreltme Oranı	0.3	0.5	
Öğrenme Katsayısı	0.01	0.05	
Aktivasyon Fonksiyonu	ReLU	ELU	SELU
Eniyileme	SGD	Adam	RMSprop

Kullanılan hiper parametrelere göre ağın parametre sayısı değişmektedir. Bu çalışmada filtre sayısı ve boyutuna göre toplam parametre sayısı Çizelge 3.3'te gösterilmektedir.

Çizelge 3.3. Toplam parametre sayısı tablosu

S.No.	Filtre Sayısı	Filtre Boyutu	Toplam Parametre Sayısı
1	32	3x3	~12 Milyon
2	32	5x5	~20 Milyon
3	64	3x3	~33 Milyon
4	64	5x5	~69 Milyon

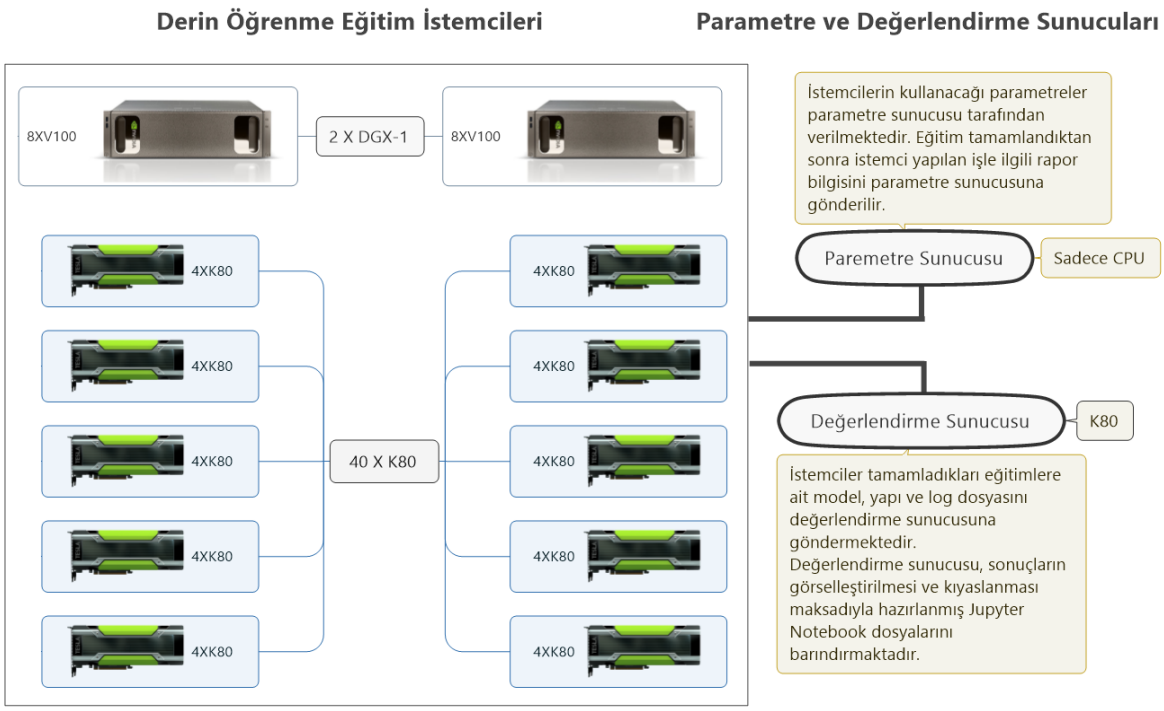
Öğrenme Katsayısı Kullanımı: Eğitimde öğrenme katsayısı başlangıç değeri olarak 0.01 ve 0.05 seçilmiştir. Öğrenme katsayısının başlangıç değerinin seçiminin eğitime etkisi ilk eğitim işlerinin sonuçlarına göre değerlendirip ilave öğrenme katsayıları belirlenmiştir.

Eğitime sokulacak verilerin çeşitli bozma ve dönüştürme işlemlerinden geçirilmesi eğitimin başarısını artırmaktadır. Keras, ImageDataGenerator sınıfı ile bu işlemlerin gerçek zamanlı yapılmasını mümkün kılmaktadır. ImageDataGenerator kullanmak eğitim başarısını olumlu katkı sağlamakla birlikte eğitim süresini uzatmaktadır.

288 farklı eğitimin 100 epok yapılması için bulutta aşağıda belirtildiği şekilde yapılanma kurulmuş ve sunucu kodlaması yapılmıştır.

3.4. Derin Öğrenme Mimarisi

Eğitim işlerinin yapılması, sonuçların toplanması ve incelenmesi sürecini doğru ve hızlı bir şekilde yapmak için bulutta süper bilgisayarların da dahil edildiği parametre ve değerlendirme sunucuları ile derin öğrenme eğitim istemcilerini barındıran mimari oluşturulmuştur. Şekil 3.4'te gösterilen bu mimarinin detayları aşağıdaki maddelerde açıklanmıştır.



Şekil 3.4. İstemci – Sunucu mimarisi

3.4.1. Parametre Sunucusu

Yapılacak tüm eğitimlerin parametrelerinin tutulduğu SQL (sqlite) ve web sunucusu (nginx) altyapısına sahiptir. Derin öğrenme eğitim istemcilerine, iş gönderilmesi ve biten işlerin raporlarının tutulması işlemlerini yapmaktadır.

3.4.2. Değerlendirme Sunucusu

Derin öğrenme eğitim istemcilerinin eğitim sonrası oluşturduğu model dosyası ve eklerini (gorev.log, gorev.json ve gorev.hdf5) toplamaktadır. GPU tabanlı bu sunucu, eğitim sonuçlarının görselleştirmesi ve kıyaslaması maksadıyla hazırlanmış kodları barındırmaktadır.

3.4.3. Derin Öğrenme Eğitim İstemcileri

Yapılacak görev sayısının ve tercih edilen paket sayılarının büyük olması nedeniyle GPU RAM ihtiyacı değişkenlik göstermiştir. Bu ihtiyacı karşılamak için 2 farklı mimariye sahip istemci yapısı kurulmuştur. Birinci grupta her biri 16 GB GPU belleğine sahip 8 adet V100'den oluşan 2 adet DGX-1 sunucu; ikinci grupta her biri 12 GB GPU belleğine sahip 4 adet K80'den oluşan 10 adet sunucu yer almaktadır. Eğitim işlemleri toplamda 40 adet K80 ve 16 adet V100 GPU üzerinde yapılmıştır. İstemciler, yapılacak eğitimlerle ilgili parametreleri, parametre sunucusuna bağlanarak almakta ve eğitim bittiğinde parametre sunucusuna görevle ilgili raporu, değerlendirme sunucusuna eğitimde oluşturulan modele ait dosyaları (gorev.log, gorev.json ve gorev.hdf5) göndermektedir.

3.5. Derin Öğrenme Eğitimi İş Akışı

Derin öğrenme eğitimi iş akışı aşağıdaki gibi gerçekleşmektedir.

1. Eğitim işlemine başlamadan önce performans problemi yaşamamak için GPU belleği boşaltılmaktadır.
2. İstemci, yapılacak eğitimle ilgili parametreleri parametre sunucusundan almaktadır.
3. Sunucudan alınan parametrelerle VGG-19 modeli oluşturulmaktadır.
4. Modele, paket sayısı kadar resim çekilmektedir.
5. Keras'a ait ImageDataGenerator fonksiyonu ile paket sayısı kadar alınan resimlere Çizelge 3.4'te belirtilen resim ön işleme argümanları uygulanmaktadır. Bu argümanların Şekil 3.5 (a) resme uygulanması sonucunda Şekil 3.5 (b)'deki görüntüler oluşmaktadır. Çıkan görüntülerin farklı olmasının nedeni ön işleme seçeneklerin rastgele uygulanmasıdır. Rastgele uygulaması, her epokta ağa giren resimlerin küçük değişikliklerde olsa farklılaşmasını sağlamaktadır. Tez çalışmasında farklı ön işlem uygulamalarının sonuçları da araştırılmıştır. Çizelge 3.4'e ek olarak Çizelge 3.5'te yer alan argümanların Şekil 3.5 (a) resmine uygulanması sonucunda

Şekil 3.5 (c)'deki görüntüler oluşmaktadır. Dikkat edilirse Şekil 3.5 (c)'deki görüntüler, Şekil 3.5 (b)'deki görüntülerden daha fazla bozulma içermektedir. Girdi resmine uygulanan ön işlem sonucunda resimlerin farklılaştırılmasına genişletme denmektedir. Kerasda kullanılabilecek tüm ön işleme argüman ve açıklamaları EK-1'de yer almaktadır.

Çizelge 3.4. Eğitimde uygulanan ön işleme argümanları

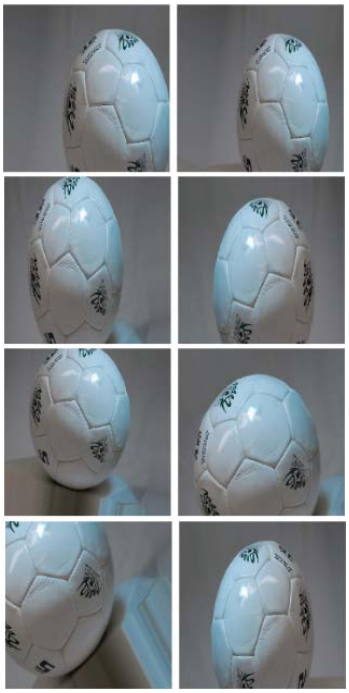
Resim İşleme Argümanı	**Açıklama**
rescale=1./255	Yeniden ölçeklendirme faktörü. Varsayılan değeri 'None' olarak ayarlanmıştır. Değer olarak None veya 0 verilirse yeniden ölçeklendirme uygulanmaz. Aksi takdirde bütün işlemlerden önce girdi verisi, verilen değerle çarpılır.
shear_range=0.2	Saat yönünün tersi yönde uygulanan, derece cinsinden shear açısı.
zoom_range=0.2	Rastgele yakınlaştırma aralığı. 1-zoom_range, 1+zoom_range yani 0.8-1.2 aralığında büyütülür yada küçültülür.
horizontal_flip=True	Girdileri rastgele yatayda çevirir.

Çizelge 3.5. İlave ön işleme argümanları

Resim İşleme Argümanı	**Açıklama**
rotation_range=40	Rastgele döndürme için derece aralığı.
width_shift_range=0.2	Yatayda rastgele kaydırma aralığı. Toplam genişlikle çarpılan değer.
height_shift_range=0.2	Dikeyde rastgele kaydırma aralığı. Toplam yükseklikle çarpılan değer.
fill_mode='nearest'	Çevirme, kırpma gibi resmin orijinal sınırlarını değiştiren işlemlerden sonra, değişen sınırların hangi yöntemle doldurulacağını belirtmek için kullanılır.

(a) Örnek resim

(b) Rastgele 4 parametre değeri ile ön işlemden geçirilmiş resimler



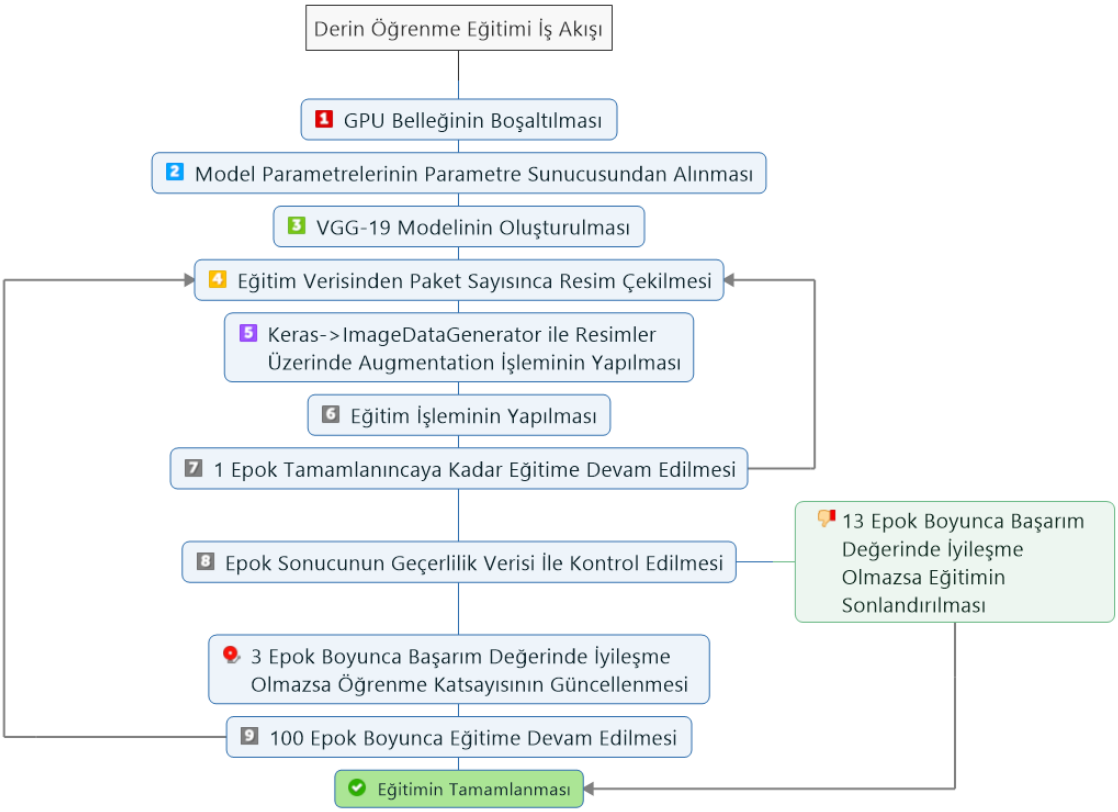
(c) Rastgele 8 parametre değeri ile ön işlemden geçirilmiş resimler

Şekil 3.5. Resim ön işleme örnekleri

6. Eğitim işlemi yapılmaktadır. Eğitim işleminde her iterasyonda eğitim verisinin doğruluk ve hata değerleri hesaplanmaktadır.
7. Bir epok tamamlanıncaya kadar 4. basamağa dönülerek işlemlere devam edilmektedir.
8. Bir epok tamamlandıktan sonra model ile geçerlilik verisi test edilerek geçerlilik doğruluğu ve hatası hesaplanmaktadır. Bu hesaplama sonucunda:
    a. 3 epok boyunca geçerlilik doğruluğunda iyileşme görülmezse öğrenme katsayısı faktör katsayısı ile çarpılmakta ve yeni öğrenme katsayısı belirlenmektedir.
    b. Öğrenme katsayısının güncellenmesine rağmen geçerlilik doğruluğunda 13 epok boyunca iyileşme görülmezse eğitim sonlandırılmaktadır.

9. Eğitimin sürekli iyi bir seyirde devam etmesi durumunda 100 epok eğitim yapılmaktadır.

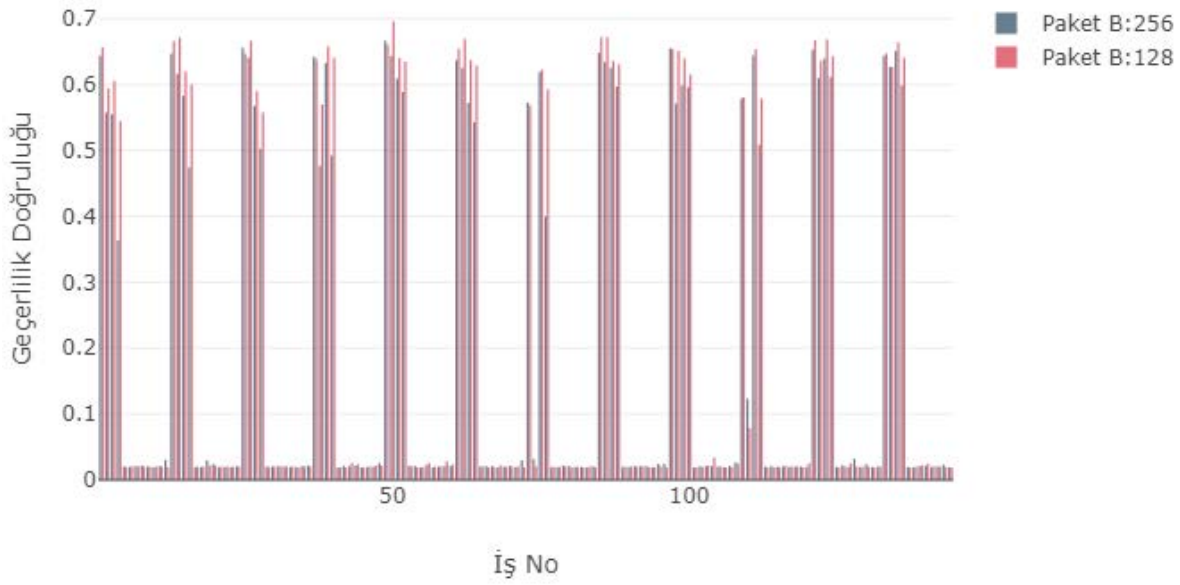
Eğitim tamamlandıktan sonra istemci yeni bir iş için parametreleri parametre sunucusundan alarak Şekil 3.6'daki süreci tekrar etmektedir.



Şekil 3.6. Derin öğrenme iş akışı

3.6. Eğitimin Genel Sonucu

Bu çalışmada hiper parametreler için belirlenen seçenekler ile 144 farklı iş 128'lik ve 256'lık paketler için toplamda 288 iş olacak şekilde eğitilmiştir. 288 işe ait hiper parametre ve başarım değerleri EK-2'de yer almaktadır. Bu eğitim sonrasında Şekil 3.7'deki başarım grafiği elde edilmiştir.



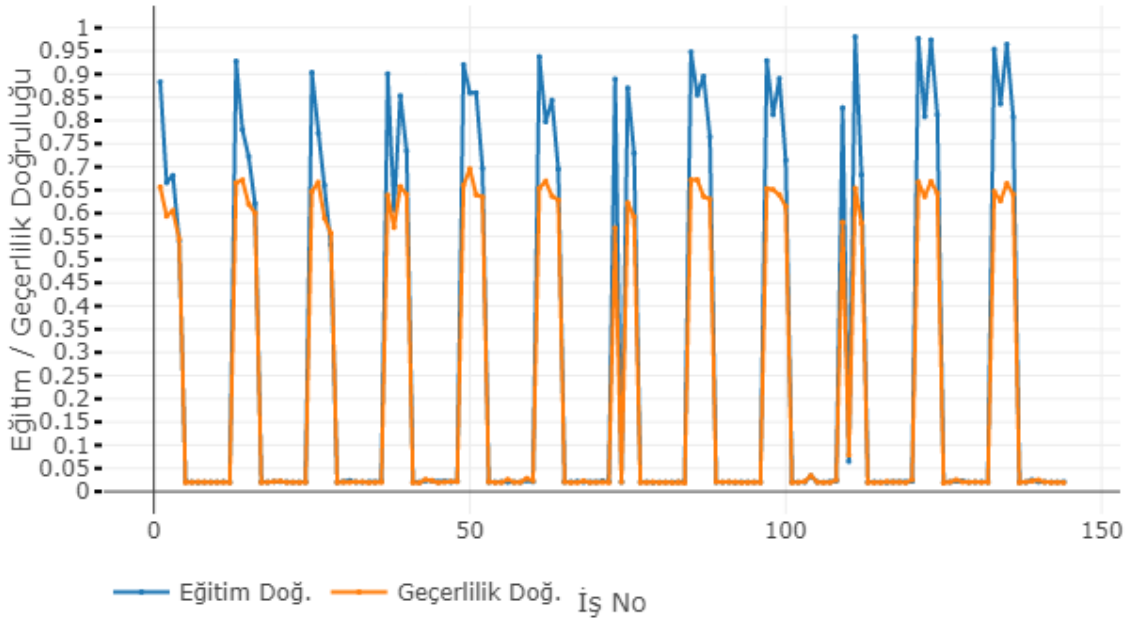
Şekil 3.7. 144 farklı işin paket boyutu 128 ve 256 için başarım grafiği

Şekil 3.7'de eniyileyici olarak SGD kullanılan işlerin başarımı iyi seviyede iken Adam ve RMSprop kullanılan görevlerin tamamı %3'ün altında kalmıştır. Eniyileyici olarak Adam ve RMSprop kullanılan eğitimlerin başarımı başlangıçta belirtilen hiper parametre değerleri ile artmadığı için erken dur (early stop) kuralına takılmış ve eğitim durdurulmuştur. Erken dur parametresi olarak 13 epok belirlenmiştir. Bu parametre sayesinde eğitilemeyen ağların sistem kaynaklarını tüketmesi ve zaman kaybının önüne geçilmiştir.

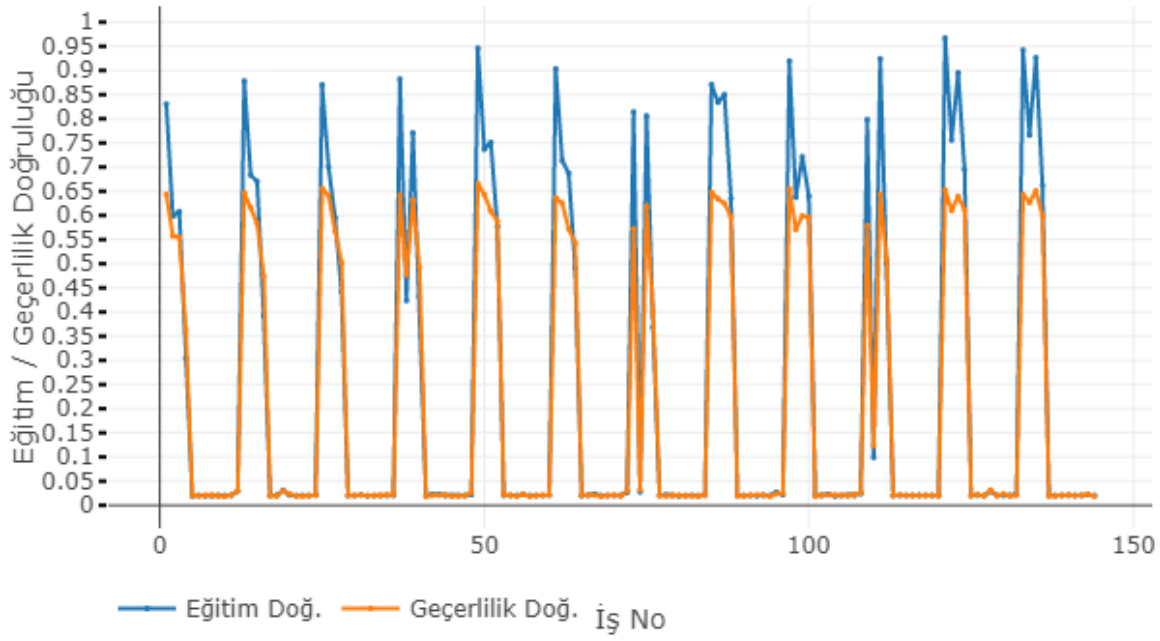
Adam ve RMSprop kullanılan işlerin erkenden sonlanmasının nedeni çok yüksek öğrenme katsayısıyla öğrenme sürecinin başlatılmasıdır.

İşlerin geneline bakıldığında Şekil 3.7'de aynı parametrelere sahip eğitimlerden paket boyutu 128 olan eğitimlerin paket boyutu 256 olan eğitimlerden daha başarılı sonuç elde ettiği görülmektedir.

Şekil 3.8 ve Şekil 3.9'da paket boyutu 128 ve 256 olan işlerin eğitim ve geçerlilik doğruluğu gösterilmektedir.



Şekil 3.8. Paket boyutu 128 olan işlerin eğitim ve geçerlilik doğruluğu



Şekil 3.9. Paket boyutu 256 olan işlerin eğitim ve geçerlilik doğruluğu

Toplamda yapılan 288 eğitim işinin en yüksek başarım değerine sahip 10 tanesinin, paket boyutu 128 ve paket boyutu 256 için sonucu Şekil 3.10'da, hiper parametre değerleri ise Çizelge 3.6'da gösterilmektedir.

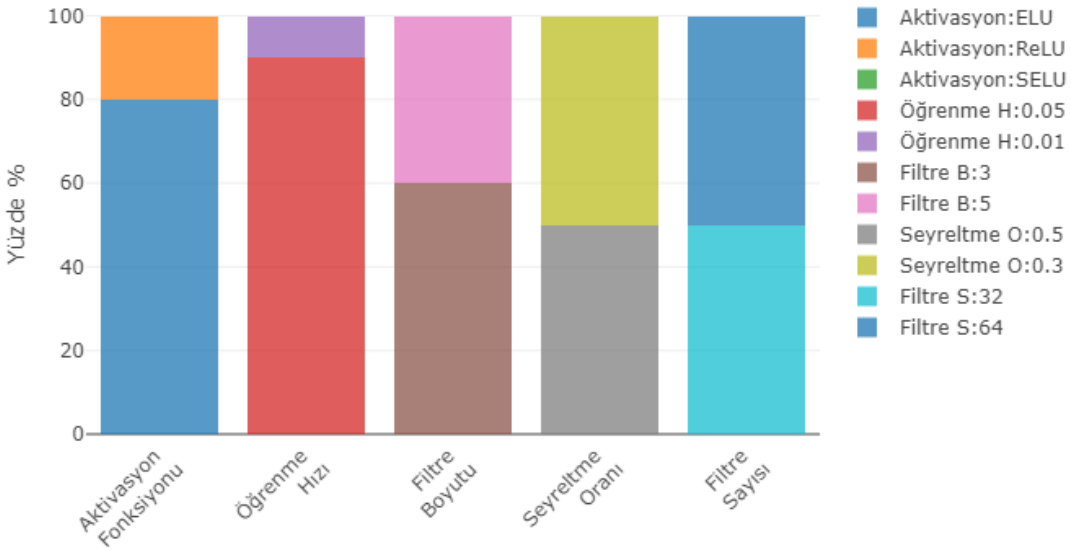
Şekil 3.10. En yüksek başarıma sahip 10 iş

Çizelge 3.6. En yüksek başarıma sahip işlerin hiper parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	PB	Epok	Başarım	Sıralama
13	sgd	0.05	32	3	0.3	elu	93	256	0.646701	
13	sgd	0.05	32	3	0.3	elu	100	128	**0.665707**	**10**
14	sgd	0.05	32	3	0.5	elu	100	256	0.616319	
14	sgd	0.05	32	3	0.5	elu	100	128	**0.671464**	**4**
26	sgd	0.05	32	3	0.5	selu	100	256	0.641059	
26	sgd	0.05	32	3	0.5	selu	100	128	**0.66653**	**8**
49	sgd	0.05	64	3	0.3	elu	100	256	**0.666233**	**9**
49	sgd	0.05	64	3	0.3	elu	91	128	0.660156	
50	sgd	0.05	64	3	0.5	elu	100	256	0.643229	
50	sgd	0.05	64	3	0.5	elu	100	128	**0.695724**	**1**
62	sgd	0.05	64	3	0.5	selu	100	256	0.625	
62	sgd	0.05	64	3	0.5	selu	100	128	**0.668997**	**5**
85	sgd	0.05	32	5	0.3	elu	80	256	0.648003	
85	sgd	0.05	32	5	0.3	elu	100	128	**0.671875**	**3**
86	sgd	0.05	32	5	0.5	elu	100	256	0.634115	
86	sgd	0.05	32	5	0.5	elu	100	128	**0.672286**	**2**
121	sgd	0.05	64	5	0.3	elu	94	256	0.652778	
121	sgd	0.05	64	5	0.3	elu	100	128	**0.667352**	**7**
123	sgd	0.01	64	5	0.3	elu	79	256	0.639323	
123	sgd	0.01	64	5	0.3	elu	99	128	**0.668586**	**6**

En başarılı 10 işte kullanılan parametrelerin yüzdelik dağılımları Şekil 3.11'de, parametre sayıları Çizelge 3.7'de gösterilmektedir. Şekil 3.11 incelendiğinde dikkat çekici oranda aktivasyon fonksiyonu olarak ELU, öğrenme katsayısı başlangıç değeri olarak 0.05 kullanıldığı anlaşılmaktadır. Diğer parametrelerin ise ortalama

olarak dağıldığı söyleyebilir. Bu sonuçlar eniyileyici olarak SGD kullanıldığında geçerlidir.

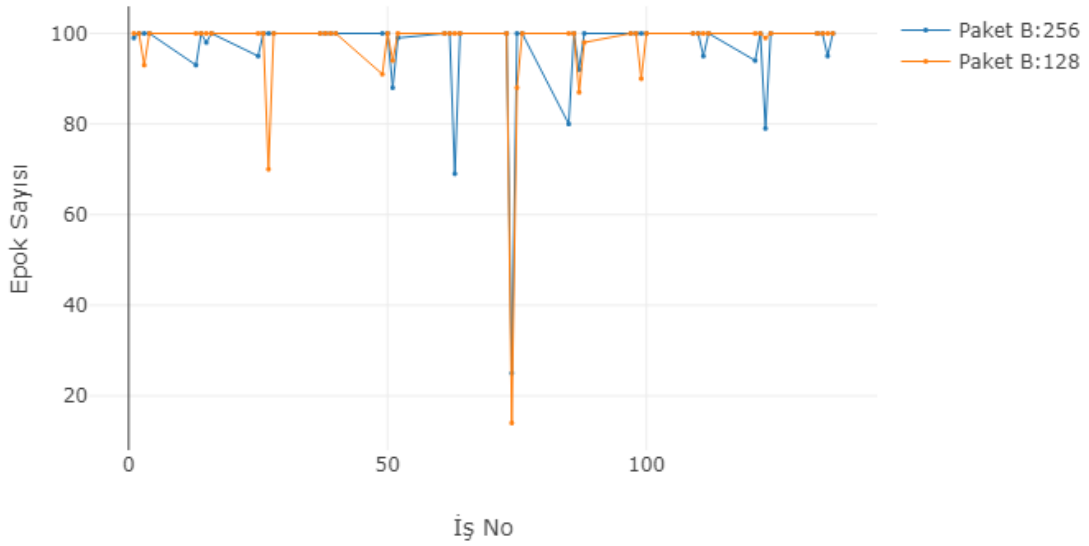


Şekil 3.11. En iyi 10 başarıma sahip işte kullanılan parametrelerin yüzdelik dağılımı

Çizelge 3.7. En iyi 10 başarıma sahip işte kullanılan parametreler

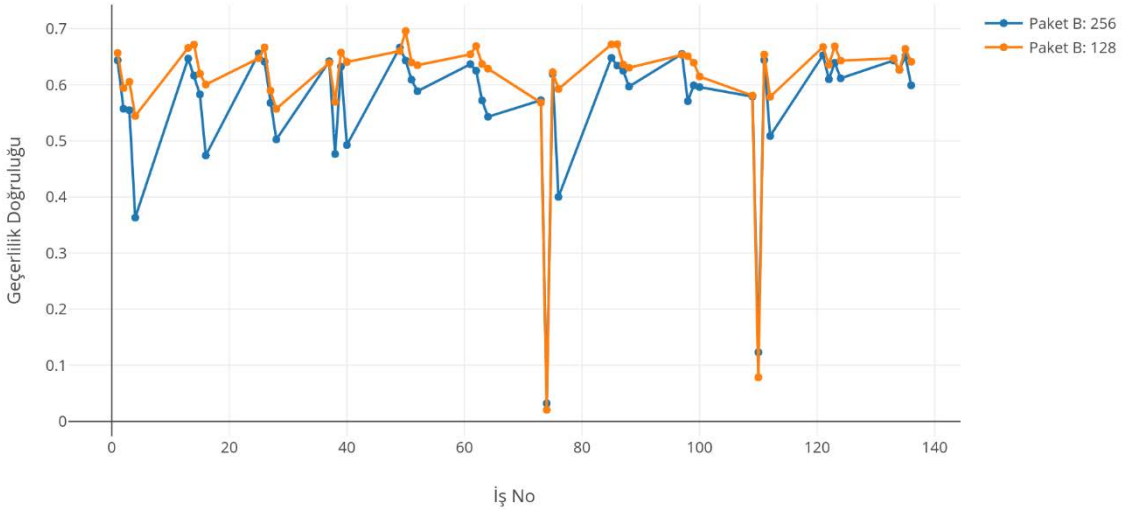
	Parametre	**Adet**
Aktivasyon Fonksiyonu	Relu	0
	Elu	**8**
	Selu	2
Öğrenme Katsayısı	**0.05**	**9**
	0.01	1
Filtre Boyutu	**3**	**6**
	5	4
Seyreltme Oranı	0.3	5
	0.5	5
Filtre Sayısı	32	5
	64	5

Adam ve RMSprop kullanılan işlerde eğitim başarımı artmadığı için erkenden dur özelliği devreye girerek eğitimler en fazla 34 epokta sonlanmıştır. SGD kullanılan eğitimlerin geneli Şekil 3.12'de görüleceği üzere 100 epoğa kadar devam etmiştir.



Şekil 3.12. Eniyileyici olarak SGD kullanılan işlerdeki epok sayısı

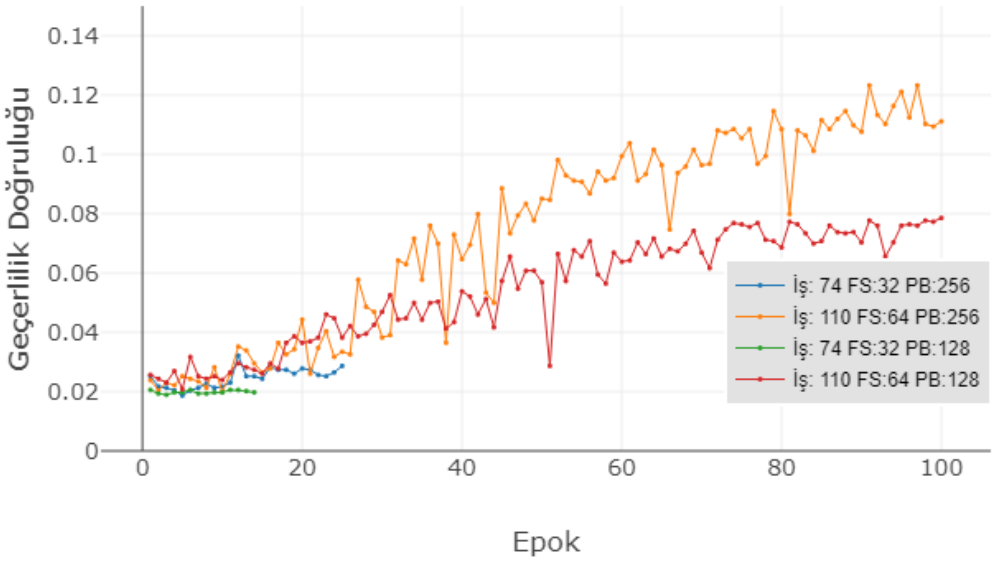
Şekil 3.13'te eniyileyici olarak SGD olan sonuçların başarımı gösterilmektedir. Şekil incelendiğinde 74. ve 110. işlerin geçerlilik değerinin hem paket boyutu 128 hem de 256 için çok düşük olduğu görülmektedir.



Şekil 3.13. Eniyileyici olarak SGD kullanılan işlerin başarımı

Şekil 3.14'te 74. işin eğitilemediği için erken dura takılarak paket boyutu 128 için 14, paket boyutu 256 için 25 epokta sonlandığı; 110. işin ise çok yavaş ilerlediği ve eğitim sonuna kadar devam ettiği anlaşılmaktadır.

Çizelge 3.8'de en düşük başarıma sahip bu iki eğitimin parametreleri incelendiğinde filtre sayıları haricinde tüm değerlerinin aynı olduğu görülmektedir.



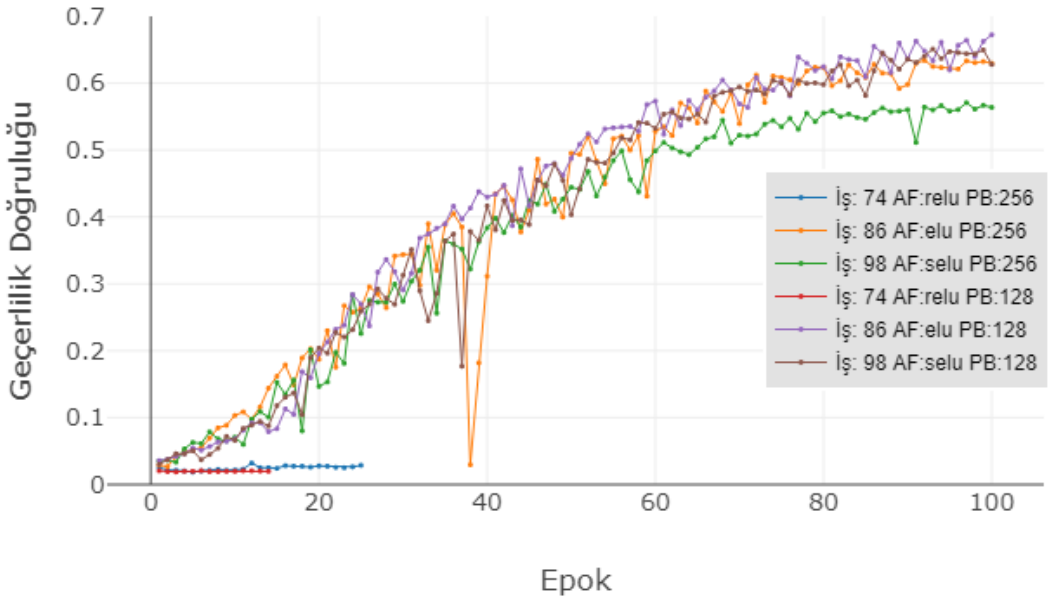
Şekil 3.14. 74. ve 110. işlerin başarımı

Çizelge 3.8. 74. ve 110. işlere ait parametreler

İş	Eİ	ÖK	FS	FB	SO	AF	PB	Epok	Başarım
74	sgd	0.05	**32**	5	0.5	relu	128	14	0.020559
74	sgd	0.05	**32**	5	0.5	relu	256	25	0.032118
110	sgd	0.05	**64**	5	0.5	relu	128	100	0.078559
110	sgd	0.05	**64**	5	0.5	relu	256	100	0.123264

3.6.1. 74. İşin Parametre Analizi

Şekil 3.15'te, 74. iş ile aktivasyon fonksiyonu dışında aynı parametrelerin kullanıldığı 86. ve 98. işler ile karşılaştırıldığında, ReLU ile eğitilemeyen ağ, ELU fonksiyonu ile %67, SELU fonksiyonu ile %65 başarım elde etmiştir. 74., 86. ve 98. işlerin parametre ve başarım değerleri Çizelge 3.9'da yer almaktadır.

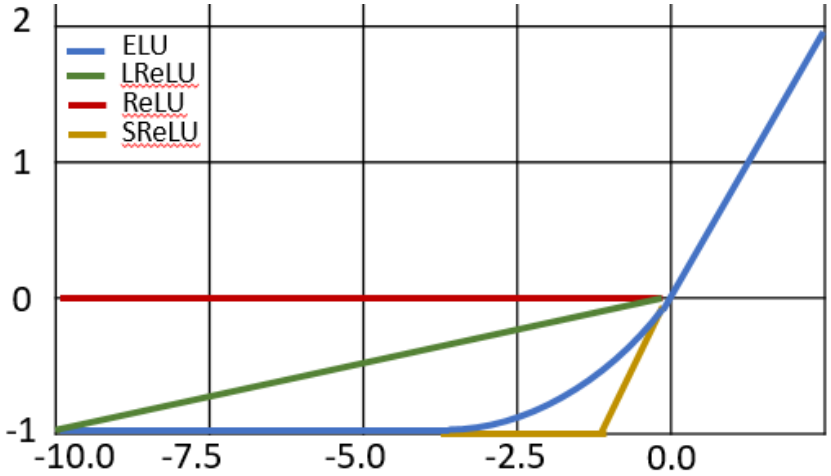


Şekil 3.15. 74. eğitimin aktivasyon fonksiyonu hariç tüm parametreleri aynı olan eğitimlerle kıyaslanması

Çizelge 3.9. 74., 86. ve 98. işlerin parametreleri

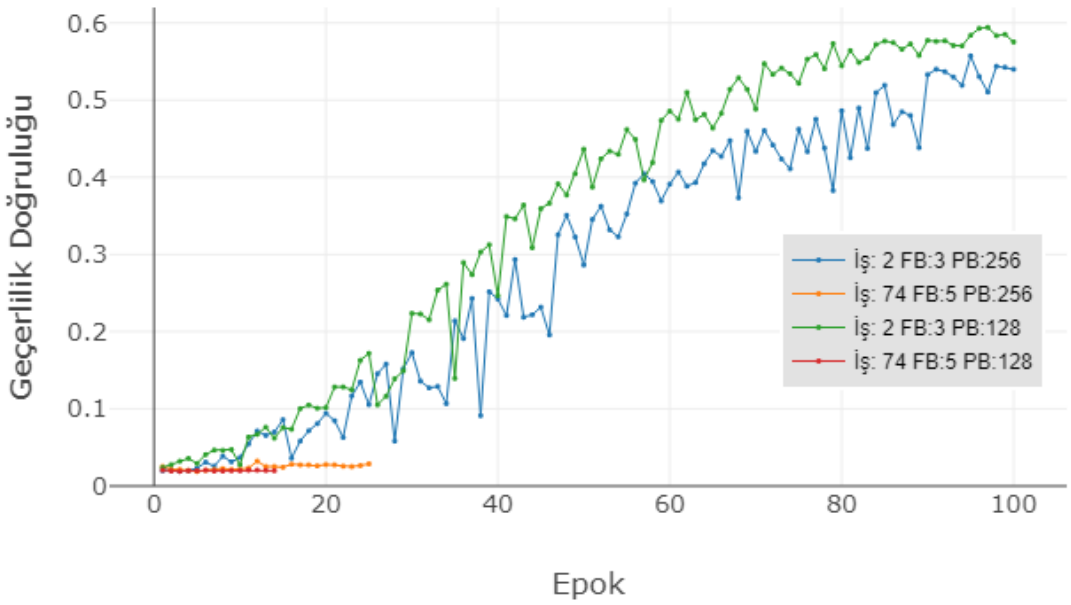
İş	Eİ	ÖK	FS	FB	SO	AF	PB	Epok	Başarım
74	sgd	0.05	32	5	0.5	**relu**	128	14	0.020559
74	sgd	0.05	32	5	0.5	**relu**	256	25	0.032118
86	sgd	0.05	32	5	0.5	**elu**	128	100	0.672286
86	sgd	0.05	32	5	0.5	**elu**	256	100	0.634115
98	sgd	0.05	32	5	0.5	**selu**	128	100	0.650905
98	sgd	0.05	32	5	0.5	**selu**	256	100	0.570747

Clevert ve arkadaşları [68], Şekil 3.16'da fonksiyon grafiği yer alan ELU fonksiyonunun ReLU ve SELU'dan daha başarılı olduğunu belirtmiştir.



Şekil 3.16. ELU, ReLU, SReLU ve LReLU karşılaştırması

Şekil 3.17'de 74. eğitim işi ile filtre boyutu hariç tüm parametreleri aynı olan 2. eğitim işinin sonuçları gösterilmiştir. Filtre boyutu 5x5 olduğu zaman eğitilemeyen ağ, 3x3 olduğunda %59 başarım göstermiştir. Filtre boyutunun küçük olması, ağın resimler hakkında daha küçük detayları da öğrenebilmesini sağlamıştır. 2. ve 74. işlerin parametre ve başarım değerleri Çizelge 3.10'da yer almaktadır.

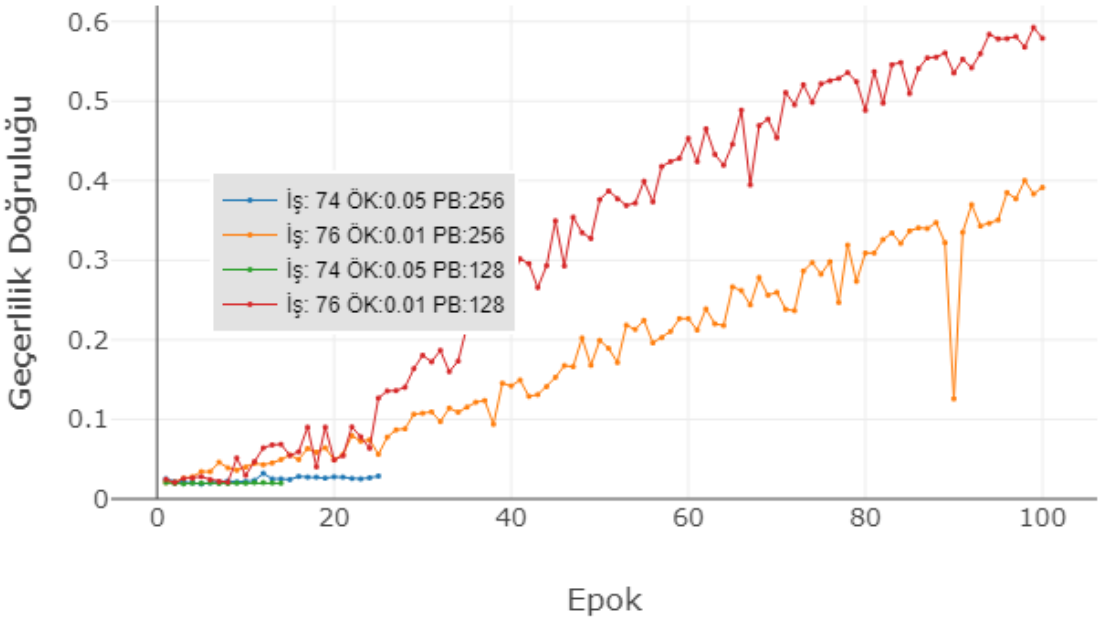


Şekil 3.17. 74. eğitimin filtre boyutu hariç tüm parametreleri aynı olan eğitimle kıyaslanması

Çizelge 3.10. 2. ve 74. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	PB	Epok	Başarım
2	sgd	0.05	32	**3**	0.5	relu	256	100	0.557292
2	sgd	0.05	32	**3**	0.5	relu	128	100	0.594161
74	sgd	0.05	32	**5**	0.5	relu	256	25	0.032118
74	sgd	0.05	32	**5**	0.5	relu	128	14	0.020559

Şekil 3.18'de, 74. iş ile öğrenme katsayısı dışında aynı parametreler kullanan 76. iş incelediğinde, eğitim katsayısı 0.05 olduğunda eğitilemeyen ağın, 0.01 olduğunda %59 başarım sağladığı görülmüştür. 74. ve 76. işlerin parametre ve başarım değerleri Çizelge 3.11'de yer almaktadır.

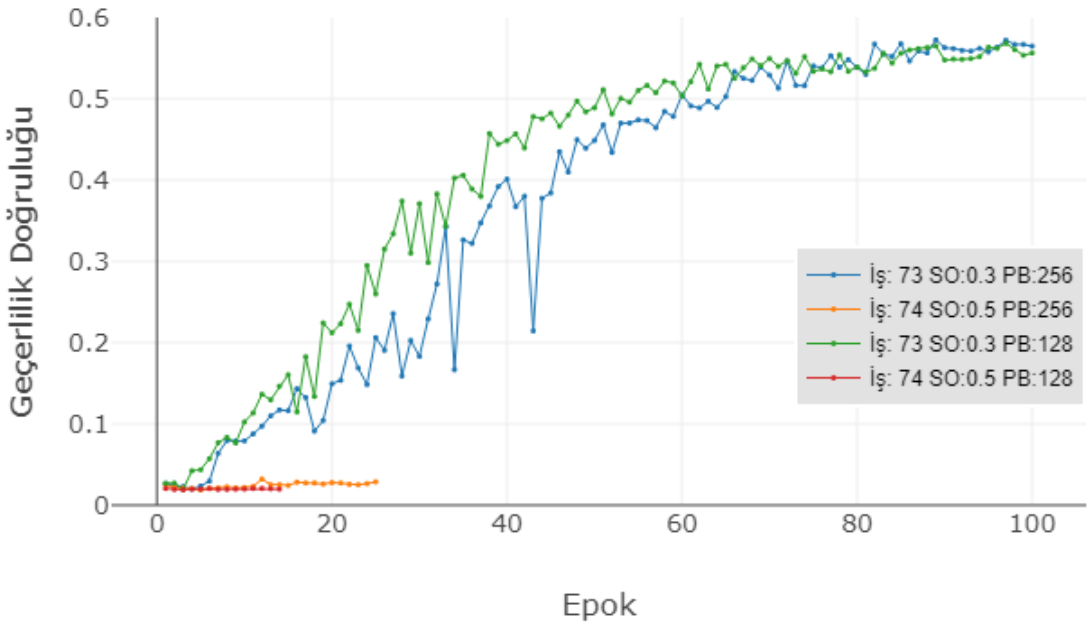


Şekil 3.18. 74. eğitimin öğrenme katsayısı hariç tüm parametreleri aynı olan eğitimle kıyaslanması

Çizelge 3.11. 74. ve 76. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	PB	Epok	Başarım
74	sgd	**0.05**	32	5	0.5	relu	256	25	0.032118
74	sgd	**0.05**	32	5	0.5	relu	128	14	0.020559
76	sgd	**0.01**	32	5	0.5	relu	256	100	0.400174
76	sgd	**0.01**	32	5	0.5	relu	128	100	0.592516

Şekil 3.18'de, 74. iş ile seyreltme oranı dışında aynı parametreler kullanan 73. iş incelediğinde, seyreltme oranı 0.5 olduğunda eğitilemeyen ağın, 0.3 olduğunda %56 başarım sağladığı görülmüştür. 74. ve 73. işlerin parametre ve başarım değerleri Çizelge 3.12'de yer almaktadır.



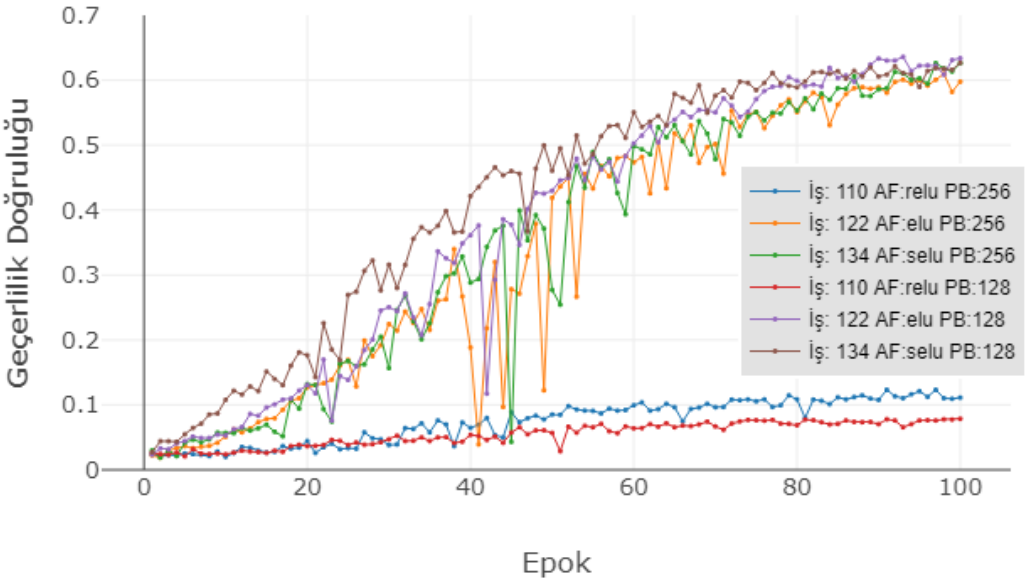
Şekil 3.19. 74. eğitimin seyreltme oranı hariç tüm parametreleri aynı olan eğitimle kıyaslanması

Çizelge 3.12. 74. ve 73. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
74	sgd	0.05	32	5	**0.5**	relu	128	14	0.019736
74	sgd	0.05	32	5	**0.5**	relu	256	25	0.028645
73	sgd	0.05	32	5	**0.3**	relu	128	100	0.556332
73	sgd	0.05	32	5	**0.3**	relu	256	100	0.564670

3.6.2. 110. İşin Parametre Analizi

Şekil 3.20'de, 110. iş ile aktivasyon fonksiyonu dışında aynı parametreler kullanan diğer işler (122, 134) karşılaştırıldığında, ReLU ile eğitilemeyen ağ ELU fonksiyonu ile %63, SELU fonksiyonu ile %62 başarım elde etmiştir. 110., 122. ve 134. işlerin parametre ve başarım değerleri Çizelge 3.13'te yer almaktadır.

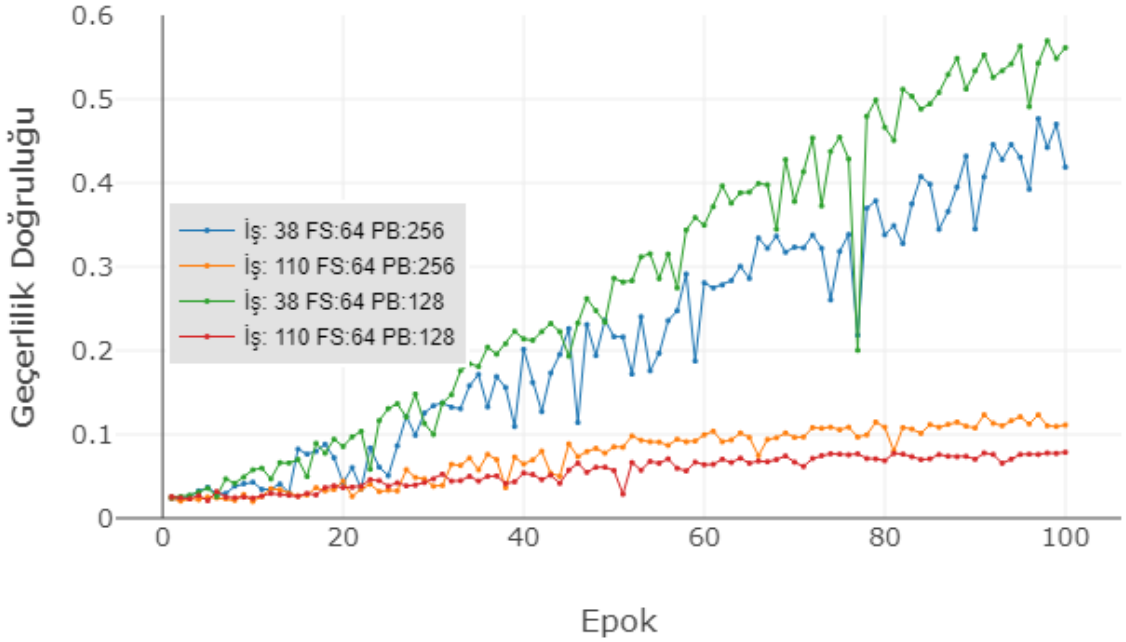


Şekil 3.20. 110. eğitimin aktivasyon hariç tüm parametreleri aynı olan eğitimlerle kıyaslanması

Çizelge 3.13. 110., 122. ve 134. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
110	sgd	0.05	64	5	0.5	**relu**	128	100	0.078559
110	sgd	0.05	64	5	0.5	**relu**	256	100	0.123264
122	sgd	0.05	64	5	0.5	**elu**	128	100	0.635851
122	sgd	0.05	64	5	0.5	**elu**	256	100	0.609809
134	sgd	0.05	64	5	0.5	**selu**	128	100	0.626645
134	sgd	0.05	64	5	0.5	**selu**	256	100	0.626736

Şekil 3.21'de 110. eğitim işi ile filtre boyutu hariç tüm parametreleri aynı olan 38. eğitim işinin sonuçları gösterilmiştir. Filtre boyutu 5x5 olduğu zaman eğitilemeyen ağ, 3x3 olduğunda %56 başarım göstermiştir. 38. ve 110. işlerin parametre ve başarım değerleri Çizelge 3.14'te yer almaktadır.

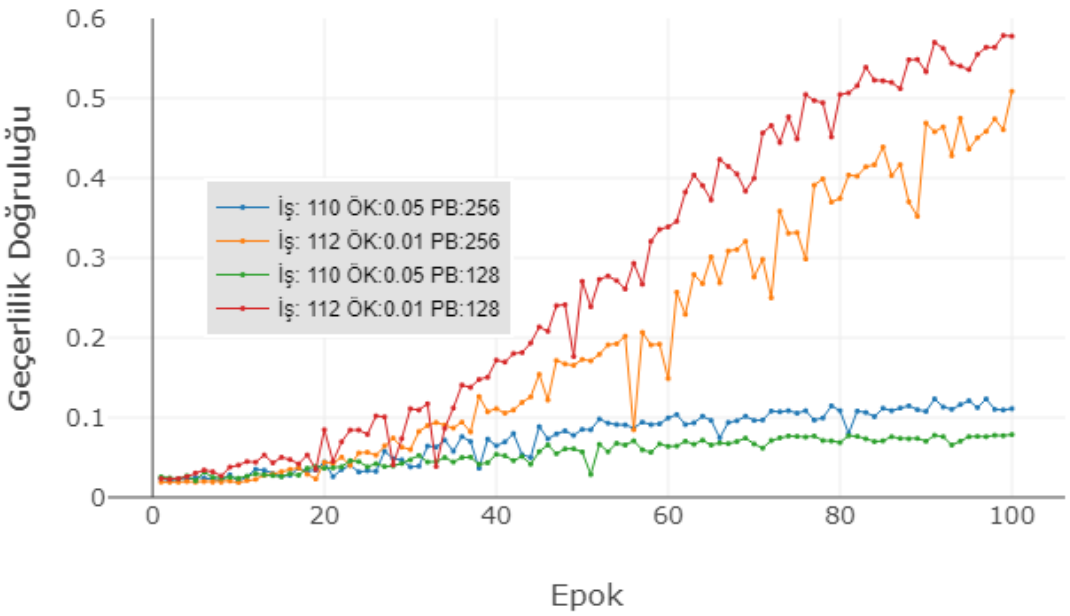


Şekil 3.21. 110. eğitimin filtre boyutu hariç tüm parametreleri aynı olan eğitimlerle kıyaslanması

Çizelge 3.14. 38. ve 110. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
38	sgd	0.05	64	3	0.5	relu	256	100	0.476562
38	sgd	0.05	64	3	0.5	relu	128	100	0.569490
110	sgd	0.05	64	5	0.5	relu	256	100	0.123264
110	sgd	0.05	64	5	0.5	relu	128	100	0.078559

Şekil 3.22'de, 110. iş ile öğrenme katsayısı dışında aynı parametreler kullanan 112. iş incelediğinde, eğitim katsayısı 0.05 olduğunda eğitilemeyen ağın, 0.01 olduğunda %57 başarım sağladığı görülmüştür. 110. ve 112. işlerin parametre ve başarım değerleri Çizelge 3.11'de yer almaktadır.

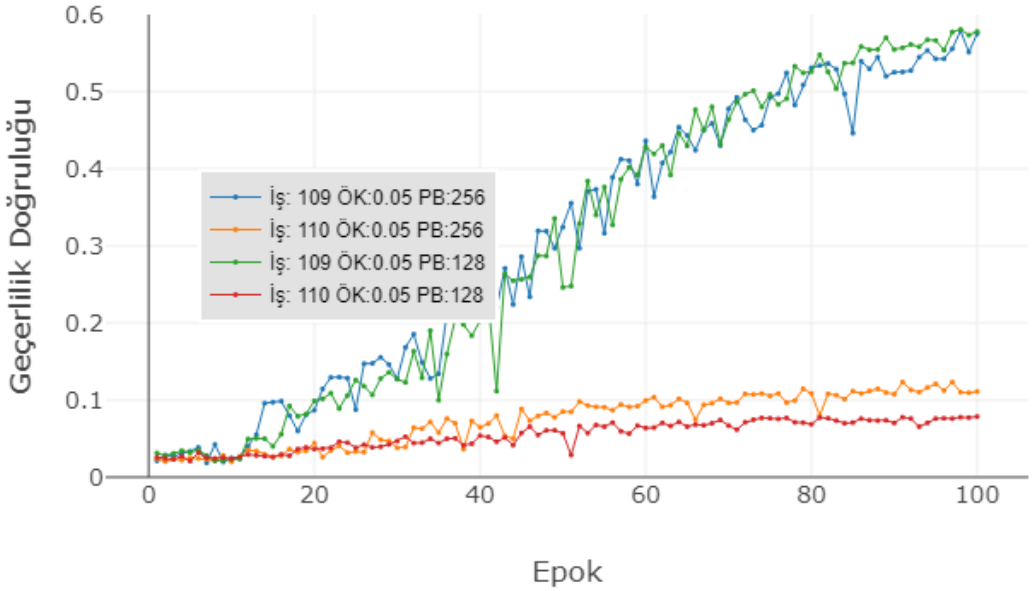


Şekil 3.22. 110. eğitimin öğrenme katsayısı hariç tüm parametreleri aynı olan eğitimle kıyaslanması

Çizelge 3.15. 110. ve 112. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
110	sgd	0.05	64	5	0.5	relu	256	100	0.123264
110	sgd	0.05	64	5	0.5	relu	128	100	0.078559
112	sgd	0.01	64	5	0.5	relu	256	100	0.508681
112	sgd	0.01	64	5	0.5	relu	128	100	0.578536

Şekil 3.23'te, 110. iş ile seyreltme oranı dışında aynı parametreler kullanan 109. iş incelediğinde, seyreltme oranı 0.5 olduğunda %12 başarım gösteren ağın, 0.3 olduğunda %58 başarım sağladığı görülmüştür. 109. ve 110. işlerin parametre ve başarım değerleri Çizelge 3.16'da yer almaktadır.



Şekil 3.23. 110. eğitimin seyreltme oranı tüm parametreleri aynı olan eğitimle kıyaslanması

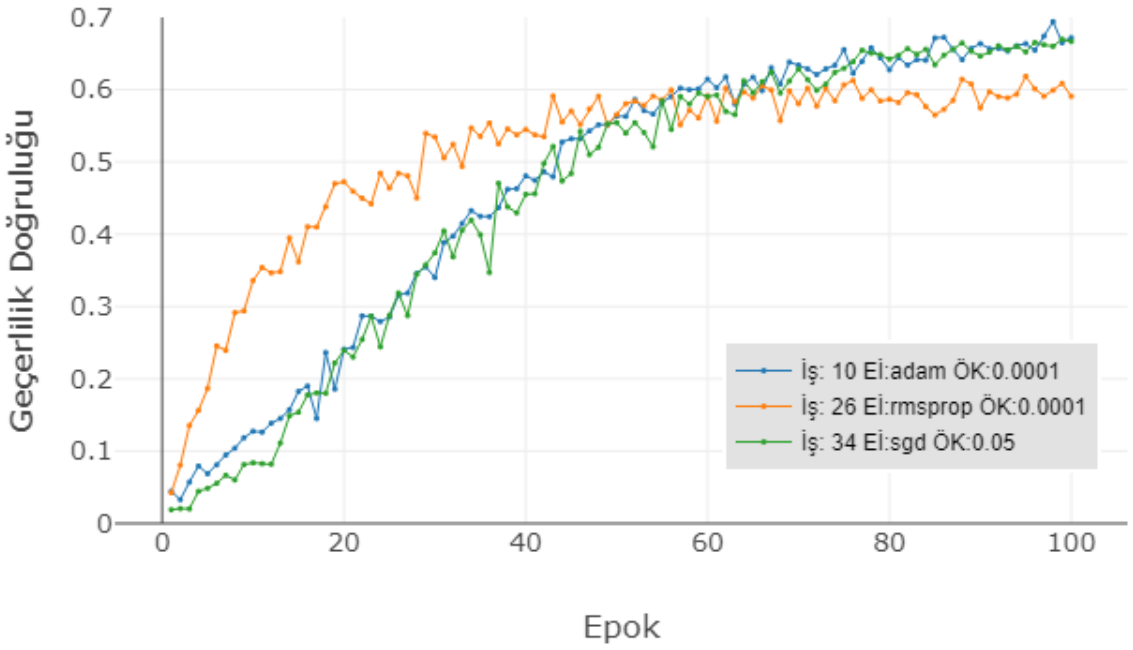
Çizelge 3.16. 109. ve 110. İşlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
109	sgd	0.05	64	5	0.3	relu	256	25	0.578993
109	sgd	0.05	64	5	0.3	relu	128	14	0.580729
110	sgd	0.05	64	5	0.5	relu	256	100	0.123264
110	sgd	0.05	64	5	0.5	relu	128	100	0.078559

3.7. Öğrenme Katsayısı Kıyaslaması

144 farklı işin paket boyutu 128 ve 256 olarak eğitilmesi sonucunda eniyileme fonksiyonlarından sadece SGD kullanılan eğitimler eğitme işlemini başarıyla gerçekleştirmiştir. Adam ve RMSprop kullanılan işlerdeki eğitim başarılı olmamıştır. Bu sonucun nedeninin 144 işi oluştururken seçilen öğrenme hızlarından (0.01 ve 0.05) kaynaklandığı değerlendirilmiş ve ilave bir iş planlaması yapılmıştır. İlave iş planlamasında 144 iş arasında en yüksek başarım değeri elde edilen 50., 86. ve 85. (başarım sırasına göre) işlerin parametreleri kullanılarak her bir eniyileme fonksiyonu ve 5 farklı öğrenme hızı (0.01, 0.05, 0.001, 0.0001, 0.00001) ile eğitim yapılmıştır. Daha önce öğrenme olmaması durumunda kullanılan erken dur işlemi burada kullanılmamış, eğitime 100 epok boyunca devam edilmiştir. Yapılan ilave işlerin hiper parametre ve başarım değerleri EK-3'te yer almaktadır.

Eğitim sonrasında her bir eniyileme fonksiyonunun belirlenen öğrenme katsayıları içerisinde ulaştığı en yüksek başarım grafiği Şekil 3.24'te, parametre ve başarım değerleri Çizelge 3.17'de yer almaktadır. Daha önce Adam ve RMSprop ile eğitilemeyen işler, öğrenme katsayısı 0.0001 yapıldığında başarıyla eğitilmiştir.



Şekil 3.24. Her bir eniyileme fonksiyonunun ulaştığı en yüksek başarım

Çizelge 3.17. 10., 26. ve 34. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	PB	Epok	Başarım
10	adam	0.0001	64	3	0.5	elu	128	100	0.694079
26	rmsprop	0.0001	32	5	0.3	elu	128	100	0.618421
34	sgd	0.05	64	3	0.5	elu	128	100	0.669819

İlave yapılan eğitim işlerinin sonucu incelendiğinde öğrenme katsayısı seçiminin eğitim başarımını yüksek oranda etkilediği tespit edilmiştir. Öğrenme katsayısının eğitim başarımını ne oranda etkilediğini eniyileme fonksiyonları grafiklerinden görülebilir.

5 farklı öğrenme katsayısı ile eniyileme fonksiyonu olarak Adam kullanılan eğitimlerin sonuçları Şekil 3.25'te, parametre ve başarım değerleri Çizelge 3.18'de yer almaktadır. Grafiğe göre en iyi başarım, öğrenme katsayısı 0.0001 olduğunda ortaya çıkmıştır. Daha sonra sırasıyla 0.00001 ve 0.001 değerlerinde eğitim başarılı olmuştur. 0.01 ve 0.05 değerlerinde ise eğitim gerçekleşmemiştir. Keras kütüphanesinde, Adam için varsayılan öğrenme katsayısı 0.001 olarak atanmıştır

[66]. Bu tez çalışmasında tespit edilen en iyi öğrenme katsayısının başarımı ile Adam varsayılan değeri kullanıldığında elde edilen başarım arasında %16'lık bir fark bulunmuştur.

Şekil 3.25. Adam ile 5 farklı öğrenme katsayısı eğitim sonucu

Çizelge 3.18. 1., 4., 8., 10. ve 14. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
1	adam	0.01	64	3	0.5	elu	128	100	0.027138
4	adam	0.05	64	3	0.5	elu	128	100	0.020148
8	adam	0.001	32	5	0.3	elu	128	100	0.533717
10	adam	0.0001	64	3	0.5	elu	128	100	0.694079
14	adam	1.0e-05	32	5	0.3	elu	128	100	0.584293

5 farklı öğrenme katsayısı ile eniyileme fonksiyonu olarak RMSprop kullanılan eğitimlerin sonuçları Adam sonuçlarına benzer bir şekilde gerçekleşmiştir. RMSprop ile yapılan eğitimlerin grafiği Şekil 3.26'da, parametre ve başarım değerleri Çizelge 3.19'da yer almaktadır. Grafiğe göre en iyi başarım, öğrenme katsayısı 0.0001 olduğunda ortaya çıkmıştır. Daha sonra sırasıyla 0.00001 ve 0.001 değerlerinde eğitim başarılı olmuştur. 0.01 ve 0.05 değerlerinde ise eğitim gerçekleşmemiştir. Keras kütüphanesinde, RMSprop için varsayılan öğrenme katsayısı Adam'da olduğu gibi 0.001 olarak atanmıştır [66]. Bu tez çalışmasında tespit edilen en iyi

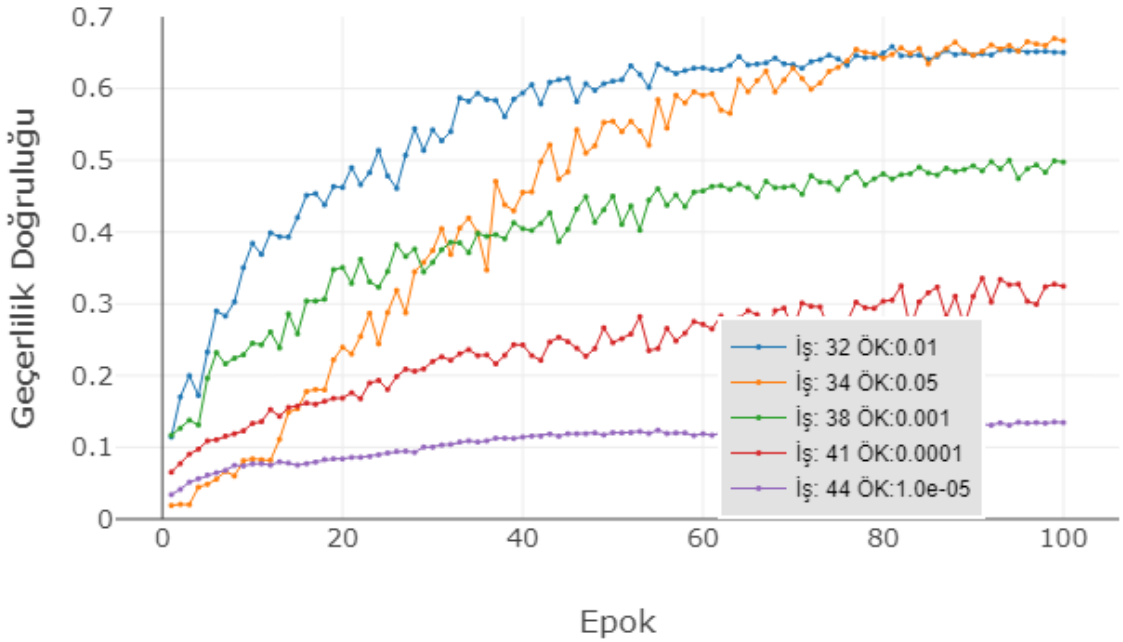
öğrenme katsayısı başarımı ile RMSprop varsayılan değeri kullanıldığında elde edilen başarım arasında %27'lik bir fark bulunmaktadır.

Şekil 3.26. RMSprop ile 5 farklı öğrenme katsayısı eğitim sonucu

Çizelge 3.19. 16., 19., 23., 26. ve 29. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
16	rmsprop	0.01	64	3	0.5	elu	128	100	0.025905
19	rmsprop	0.05	64	3	0.5	elu	128	100	0.020559
23	rmsprop	0.001	32	5	0.3	elu	128	100	0.348273
26	rmsprop	0.0001	32	5	0.3	elu	128	100	0.618421
29	rmsprop	1.0e-05	32	5	0.3	elu	128	100	0.571957

İlk yapılan eğitimlerde SGD ile öğrenme katsayısı olarak 0.01 ve 0.05 kullanıldığında eğitimler başarılı sonuç vermişti. İkinci eğitimde 5 farklı öğrenme katsayısı ile yapılan eğitim sonucunda en yüksek başarım birinci eğitim işlerinde olduğu gibi 0.05 değeri kullanıldığında gerçekleşmiştir. Öğrenme katsayısı büyüdükçe eğitim başarımı düşmüştür. SGD ile 5 farklı öğrenme katsayısı kullanılan eğitimlerin grafiği Şekil 3.27'de, parametre ve başarım değerleri Çizelge 3.20'de yer almaktadır.



Şekil 3.27. SGD ile 5 farklı öğrenme katsayısı eğitim sonucu

Çizelge 3.20. 32., 34., 38., 41. ve 44. işlerin parametreleri

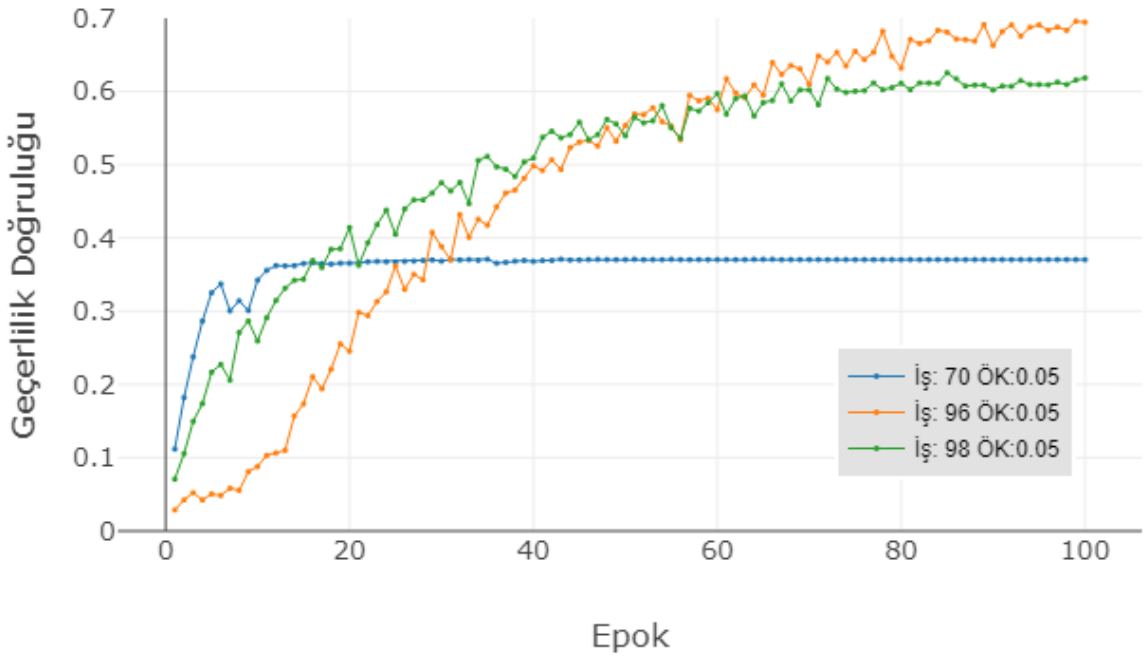
İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım
32	sgd	0.01	32	5	0.3	elu	128	100	0.658306
34	sgd	0.05	64	3	0.5	elu	128	100	0.669819
38	sgd	0.001	32	5	0.3	elu	128	100	0.500000
41	sgd	0.0001	32	5	0.3	elu	128	100	0.335526
44	sgd	1.0e-05	32	5	0.3	elu	128	100	0.135280

100 epok boyunca öğrenme olmaması durumunda, öğrenme katsayısının değişimi EK-4'te yer almaktadır.

3.8. Ön İşleme Kıyaslaması

Ön işleme, eğitime girecek resimlere belirlenen parametreler çerçevesinde rastgele bozma işlemi yapılmasıdır. Bu çalışma kapsamında eğitime girecek tüm resimler Çizelge 3.4'te gösterilen 4 argümanla ön işleme tabi tutulmuştur. Ön işlemenin katkısını daha iyi anlamak için ilk eğitimdeki başarımı en yüksek olan 50. iş, yeniden ölçeklendirme hariç ön işleme yapılmaksızın ve Çizelge 3.4 ile Çizelge 3.5'de gösterilen toplam 8 argümanla ön işleme yapılarak 100 epok boyunca eğitilmiştir. Şekil 3.28'de ön işleme yapılmayan (sadece yeniden ölçeklendirme yapılmış) 70. eğitimde, öğrenmenin 16. epoğa kadar devam ettiği, sonrasında yatay seyir izlediği

görülmektedir. 4 argümanla ön işleme yapılan 96. eğitimde ise, öğrenme eğitim sonuna kadar devam etmiş ve eğitim başarısı %32 oranında artış göstermiştir. Toplam 8 argümanla ön işleme yapılan 98. eğitimin başarımı, 4 argümanla ön işleme yapılan 96. eğitime göre daha düşük seviyede kalmıştır. Ön işlemede kullanılan argüman sayısının artmasının Şekil 3.5'te de görüleceği gibi resimlerde bozulmayı arttırdığı ve başarımı olumsuz yönde etkilediği değerlendirilmektedir. Her üç eğitime ait başarım ve süre değerleri Çizelge 3.21'de yer almaktadır.



Şekil 3.28. Ön işleme yapılan ve yapılmayan eğitimin karşılaştırılması

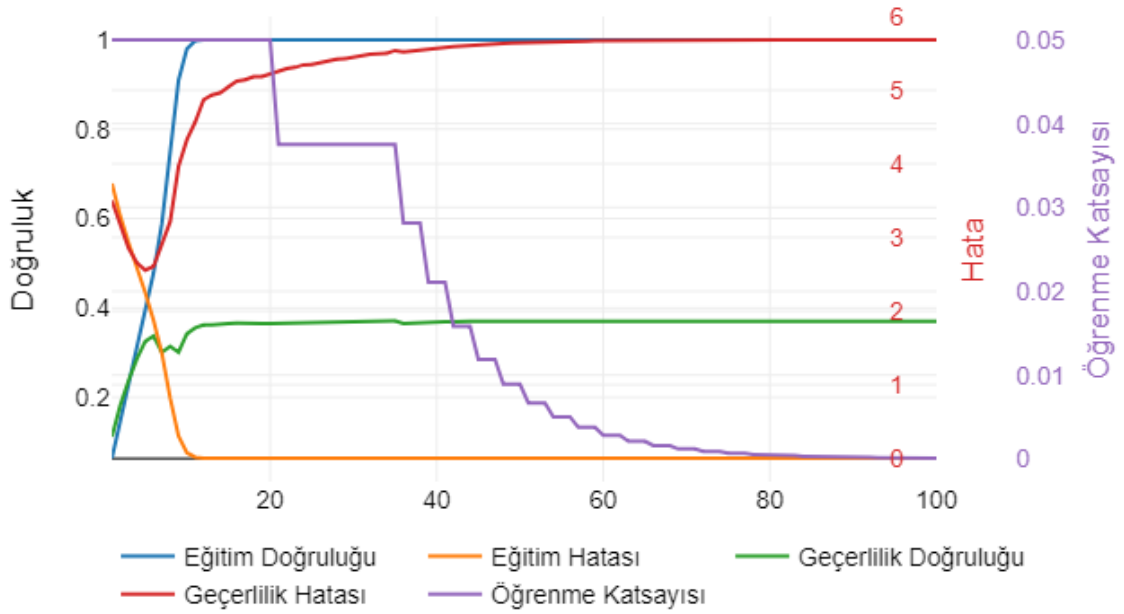
Çizelge 3.21. 50. işin ön işleme yapılan ve yapılmayan sonuçları

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım	Süre(sn)
70	sgd	0.05	64	5	0.3	elu	128	100	0.695723	57292
96	sgd	0.05	64	5	0.3	elu	128	100	0.370888	36513
98	sgd	0.05	64	5	0.3	elu	128	100	0.625411	56463

Şekil 3.29'da ön işleme yapılmamış 70. işin detaylı grafiği yer almaktadır. Bu grafikte, eğitim doğruluğunun 11. epokta %99'u geçtiği, eğitim hatasının ise 0'a yakınsadığı görülebilir. Eğitim doğruluğu bu kadar mükemmel iken geçerlilik doğruluğu ancak %37 seviyene ulaşabilmiştir. Geçerlilik hatası ise başlangıçta düşme eğilimindeyken eğitim ilerledikçe yükselmiştir. Öğrenme katsayısı, 32.

epoktan sonra öğrenme olmadığı için 3 epokta bir güncelleme sürecine girmiştir. Grafiğe yukarıdaki bilgiler ışığında bakıldığında, ağın ezber yaptığı öğrenmeyi gerçekleştiremediği söylenebilir.

Ezberlemeyi önlemenin yöntemlerinden birisi veri çoğaltma ve veri genişletmedir. Ön işleme, eğitime sokulan her bir paket resme rastgele uygulandığından resimler küçük seviyede de olsa farklılıklar göstermekte ve eğitim uzun süre başarıyla devam etmektedir. Ön işlemenin katkısı yanında maliyeti de oluşmaktadır. Ön işleme yapılan iş ön işleme yapılmayan işleme göre 1.57 kat daha uzun sürmektedir.

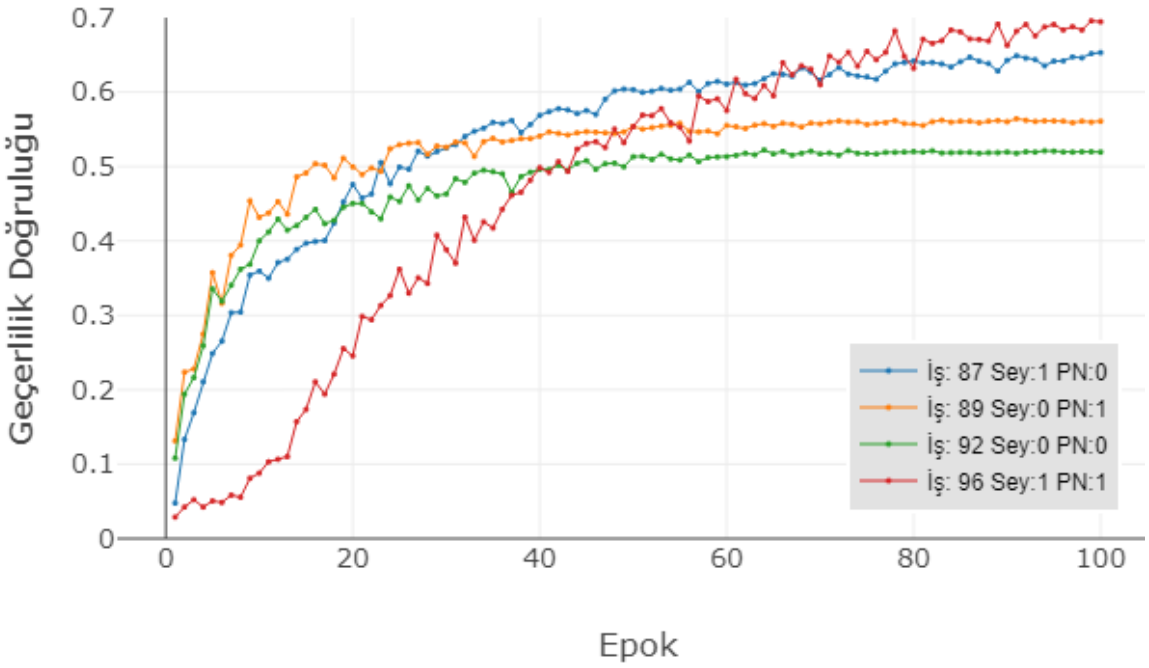


Şekil 3.29. Ön işleme yapılmayan 96. işin grafiği

3.9. Paket Normalizasyonu ve Seyreltme İşlemi

Eğitim işi planlamalarında tüm işlerde paket normalizasyonu ve seyreltme (dropout) işlemi (0.3, 0.5) yapılmış ve en yüksek başarım %69 seviyesinde çıkmıştır. Paket normalizasyonu ve seyreltme işleminin eğitime katkısını ortaya çıkarmak için 144 iş arasında en iyi sonucun elde edildiği 3 eğitimin parametreleri kullanılarak bu iki özelliğin ayrı ayrı ve beraber olmadığı 9 ayrı eğitim yapılmıştır. Bu eğitimler içinde 3 farklı durum için en iyi sonuçlar Şekil 3.30'da, parametre ve başarım değerleri Çizelge 3.22'de gösterilmiştir. Eğitim sonunda 87. ve 89. işler kıyaslandığında, seyreltme işleminin kullanıldığı paket normalizasyonunun kullanılmadığı eğitimin başarımı, seyreltme işleminin kullanılmadığı paket normalizasyonunun kullanıldığı

eğitimden daha yüksek çıkmıştır. Paket normalizasyonu ve seyreltme işleminin beraber kullanılmadığı 92. işin başarımı ise en düşük seviyede çıkmıştır.



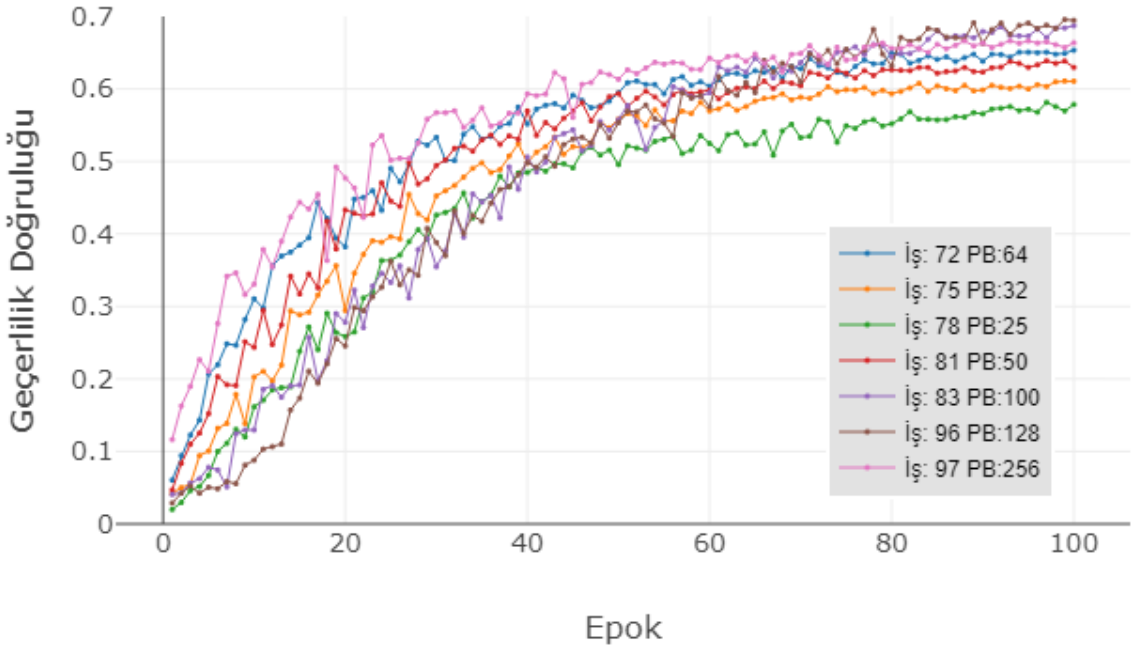
Şekil 3.30. Paket normalizasyonu ve seyreltme işlemi kıyaslaması

Çizelge 3.22. 87., 89., 92 ve 96. işlerin parametreleri

İş	PN	Sİ	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım	Süre(sn)
87	0	1	sgd	0.05	32	5	0.3	elu	100	128	0.652960	52515
89	1	0	sgd	0.05	64	3	0.5	elu	100	128	0.560855	56089
92	0	0	sgd	0.05	64	3	0.5	elu	100	128	0.519325	55045
96	1	1	sgd	0.05	64	3	0.5	elu	100	128	**0.695724**	57292

3.10. Paket Sayısı Kıyaslaması

İlk analizlerde paket boyutu olarak 128 ve 256 kullanılmıştır. Bu eğitimlerin başarıma göre sıralanması yapıldığında 128'lik paketlerin daha başarılı olduğu görülmüştür. İlk 144 iş arasında en başarılı sonucun elde edildiği 3 iş, paket boyutu 25, 32, 50, 64 ve 100 olacak şekilde eğitilmiştir. Eğitim sonrasında en yüksek başarıma, paket boyutu 128 olan eğitimde ulaşılmıştır. Diğer paket boyutlarının kullanıldığı başarımı en yüksek eğitimlerin grafiği Şekil 3.31'de, parametre ve başarım değerleri Çizelge 3.23'te yer almaktadır.



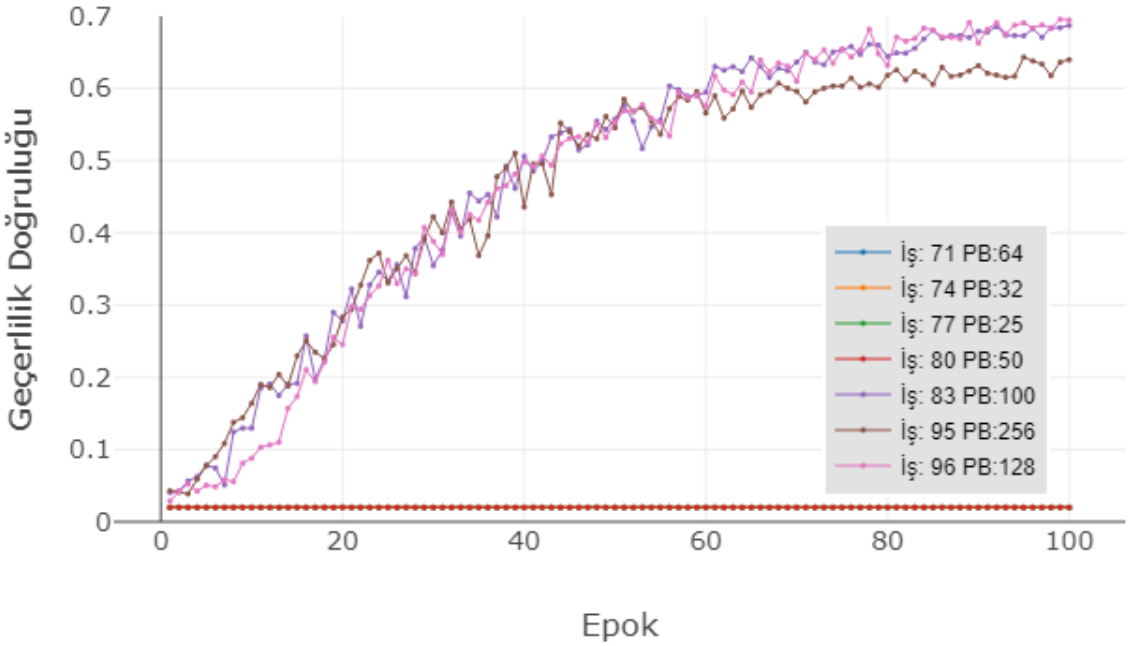
Şekil 3.31. En başarılı 3 işin farklı paket boyutlarındaki başarımı

Çizelge 3.23. 72., 75., 78., 81.,83., 96. ve 97. işlerin parametreleri

İş	Eİ	ÖK	FS	FB	SO	AF	Paket	Epok	Başarım	Süre(sn)
78	sgd	0.05	32	5	0.3	elu	**25**	100	0.581200	63570
75	sgd	0.05	32	5	0.3	elu	**32**	100	0.610978	66742
81	sgd	0.05	32	5	0.3	elu	**50**	100	0.638800	61496
72	sgd	0.05	32	5	0.3	elu	**64**	100	0.653446	59639
83	sgd	0.05	64	3	0.5	elu	**100**	100	0.687200	58154
96	sgd	0.05	64	3	0.5	elu	**128**	100	**0.695724**	57292
97	sgd	0.05	64	3	0.3	elu	**256**	100	0.666233	56376

En yüksek başarıma sahip 50. işin paket boyutu 25, 32, 50, 64 ve 100 için başarım grafiği Şekil 3.32'de, parametre ve başarım değerleri Çizelge 3.24'te görülmektedir. Grafiğe göre paket boyutu 25, 32, 50 ve 64 olduğunda ağ eğitilememiştir.

İdeal paket boyutunu bulmaya yönelik yapılan ilave 15 eğitim sonrasında süre sistem kaynağı kullanımı ve başarım açısından en iyi paket boyutu 100 ve 128 olarak tespit edilmiştir. Daha küçük paket boyutlarında 50. İş örneğinde olduğu gibi ağın eğitilememe durumu ortaya çıkabilir.



Şekil 3.32. 50. işe ait parametrelerin farklı paket boyutlarında kullanılarak eğitilmesi

Çizelge 3.24. 50. işin farklı paket boyutu için parametreleri

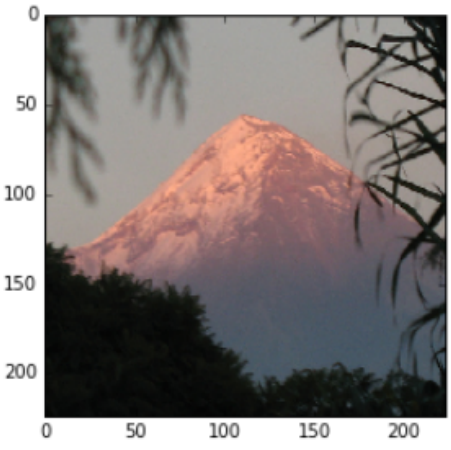
İş	**Eİ**	**ÖK**	**FS**	**FB**	**SO**	**AF**	**Paket**	**Epok**	**Başarım**	**Süre(sn)**
77	sgd	0.05	64	3	0.5	elu	**25**	100	0.020000	68403
74	sgd	0.05	64	3	0.5	elu	**32**	100	0.020032	61403
80	sgd	0.05	64	3	0.5	elu	**50**	100	0.020000	61407
71	sgd	0.05	64	3	0.5	elu	**64**	100	0.020032	63343
83	sgd	0.05	64	3	0.5	elu	**100**	100	0.687200	58154
96	sgd	0.05	64	3	0.5	elu	**128**	100	**0.695724**	57292
95	sgd	0.05	64	3	0.5	elu	**256**	100	0.643229	55934

4. ÇIKARIM BAŞARIMI

Eğitilen bir ağın kullanılabilmesi için hem başarım değerinin hem de çalışma hızının yüksek olması hedeflenmektedir. ESA başarımını gözlerken geçerlilik verisi ile etiket değeri kontrol edilmektedir. Elde edilen sonuç çerçevesinde gerektiğinde öğrenme katsayısı güncellenmekte ve model eğitimi gerçekleşmektedir. Eğitilen model kendisine verilen 1 resimle ilgili tahmin yaparken eğitimde kullanılan tüm sınıflar için değer üretmektedir. Bu çalışmada 50 sınıflı bir yapı kullanıldığı için model, girdiyle ilgili 50 sınıfa ait sonuç döndürmektedir. Döndürdüğü sonuçlardan en yüksek değere sahip olan sınıfın girdinin gerçek değeriyle aynı olması durumu gözlenmek istendiğinde Top-1 değerine ulaşılmış olur. Modelin ürettiği sonuçlar arasında en yüksek olasılığa sahip ilk 5 çıkarım sonucu içerisinde veri etiketinin olması durumu kontrol edildiğinde Top-5 değerine ulaşılır. Şekil 4.1'de etiket değeri "volcano" olan bir resim için modelin ürettiği 5 sonuç görülmektedir.

```
display(HTML('<a id="1"></a>'))
show_result(random.randint(0,2500-1))
```

```
Resim ID=1235
Gerçek değer= volcano
Sonuçlar=
(0, 45, 'volcano', 0.99911267)
(1, 1, 'alp', 0.00087207207)
(2, 43, 'valley', 1.0505081e-05)
(3, 36, 'space_shuttle', 2.319661e-06)
(4, 42, 'umbrella', 1.2085169e-06)
```

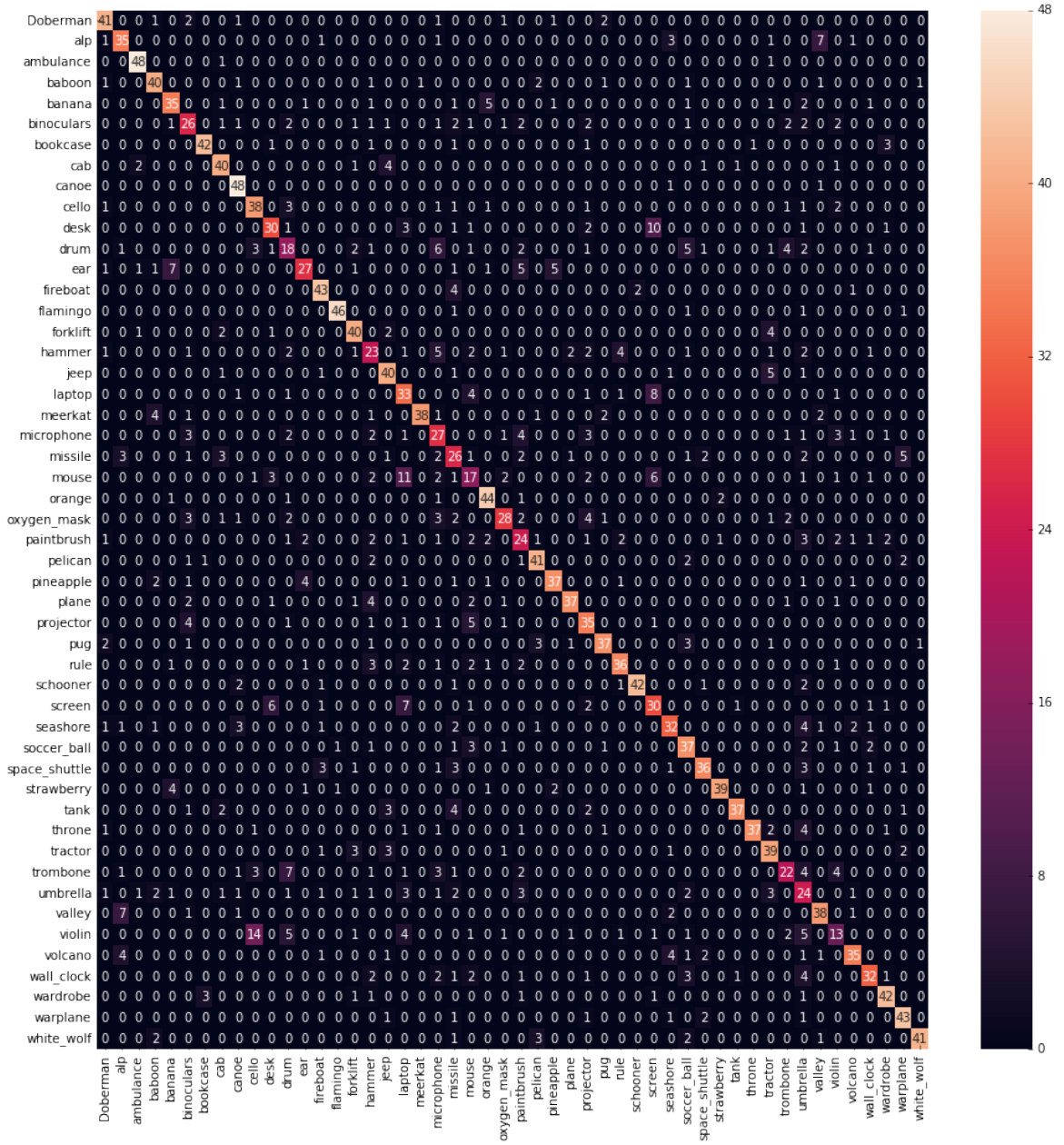


Şekil 4.1. Top-5 değeri

IMAGE-NET yarışmasında algoritmaların Top-1 ve Top-5 başarım değerleri verilmektedir. Bu çalışmada başarımı en yüksek olan modelin Top-1 değeri %69, Top-5 değeri %90 olarak bulunmuştur.

Eğitim sonrasında oluşturulan modelin tüm sınıflardaki başarımını görmenin yollarından birisi hata matrisi oluşturulmasıdır.

Şekil 4.2'de geçerlilik verisi kullanılarak hazırlanmış hata matrisi yer almaktadır. Hata matrisi incelendiğinde en yüksek başarım değerinin 48/50 oranında "ambulance" ve "canoe" sınıflarında olduğu görülebilir. "ambulance" sınıfında yer almasına rağmen iki resim "cab" ve "tractor" olarak tahmin edilmiştir.



Şekil 4.2 Hata matrisi

Eğitimin yanında eğitim sonrasında elde edilen model dosyasının kullanıldığı çıkarım işlemini hızlandırmaya yönelik çalışmalar da yapılmaktadır. Bu alanda yapılan çalışmalardan birisi NVIDIA'nın geliştirdiği TensorRT kütüphanesidir.

NVIDIA TensorRT, resim sınıflandırma, segmentasyon ve nesne tanıma gibi derin öğrenme uygulamaları için geliştirilmiş, düşük gecikme ve yüksek verimlilik sağlayan yüksek performanslı bir çıkarım iyileştiricisidir. TensorRT, çevrimiçi ağ, mobil uygulama, gömülü sistemler ve kendi kendine süren sistemler için eğitilmiş bir ağı eniyileyerek eğitilen modelin gerçek zamanlı ve GPU desteği ile yüksek başarımlı çalışmasını sağlar [69].

TensorRT, gerçek zamanlı çalışabilmek için ağ mimarisi, eğitilmiş ağırlık değerleri ve etiket dosyasına gereksinim duymaktadır. Ek olarak paket boyutu ve çıktı katmanının da tanımlanması gereklidir [69].

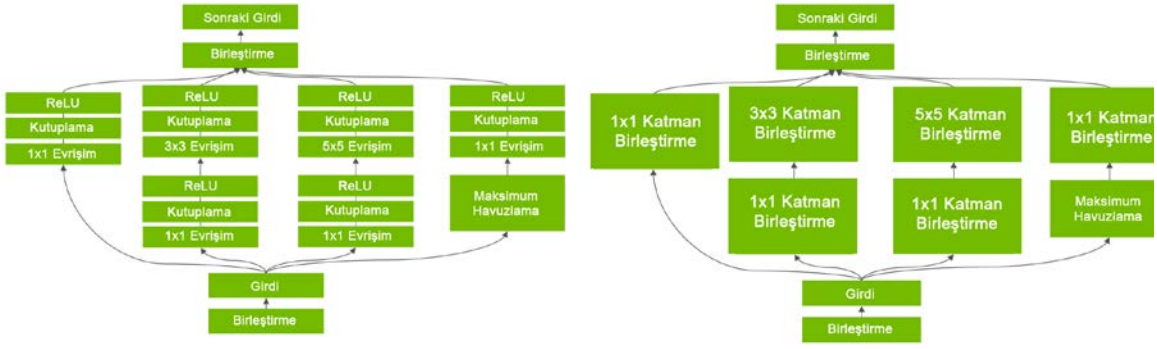
TensorRT Çizelge 4.1'de verilen katman türlerini desteklemektedir [69]:

Çizelge 4.1. TensorRT'de desteklenen katmanlar

Katman	Açıklama
Activation	ReLU, tanh ve sigmoid aktivasyon katmanı
BatchGemm	Matris çarpım katmanı
Concatenation	Birleştirme katmanı
Constant	-
Convolution	Evrişim katmanı
Deconvolution	Dekonvolüzasyon katmanı
ElementWise	Toplama, çıkarma, çarpma, bölme, maksimumu seçme, üs alma işlemleri katmanı
Flatten	Paket normalizasyonunu korurken girdiyi düzleştiren katman
FullyConnected	Tam bağlaşımlı katman
Gather	Bir veri tensörü, bir indis tensörü ve bir veri tensörü ekseni giriş olarak alır ve indeks tensörünü kullanarak verilen eksen boyunca veri tensörünü yeniden dizer
LRN	Lokal cevap normalizasyonu
MatrixMultiply	Matris çarpımı
Padding	Dolgu
Plugin	TensorRT'nin desteklemediği bir katmanı

	eklenmesini sağlar
Pooling	Havuzlama katmanı
Ragged SoftMax	Kanallar arası softmax işlemi uygular
Reduce	Boyut düşürme
RNN	Tekrarlayan sinir ağı katmanı
RNNv2	Tekrarlayan sinir ağı katmanı
Scale	Dönüşüm işlemlerinin uygulandığı katman
Shuffle	Tensoru karıştıran katman
SoftMax	Softmax işlemi
Squeeze	-
TopK	-
Unary	Noktasal tekli işlemleri destekler.

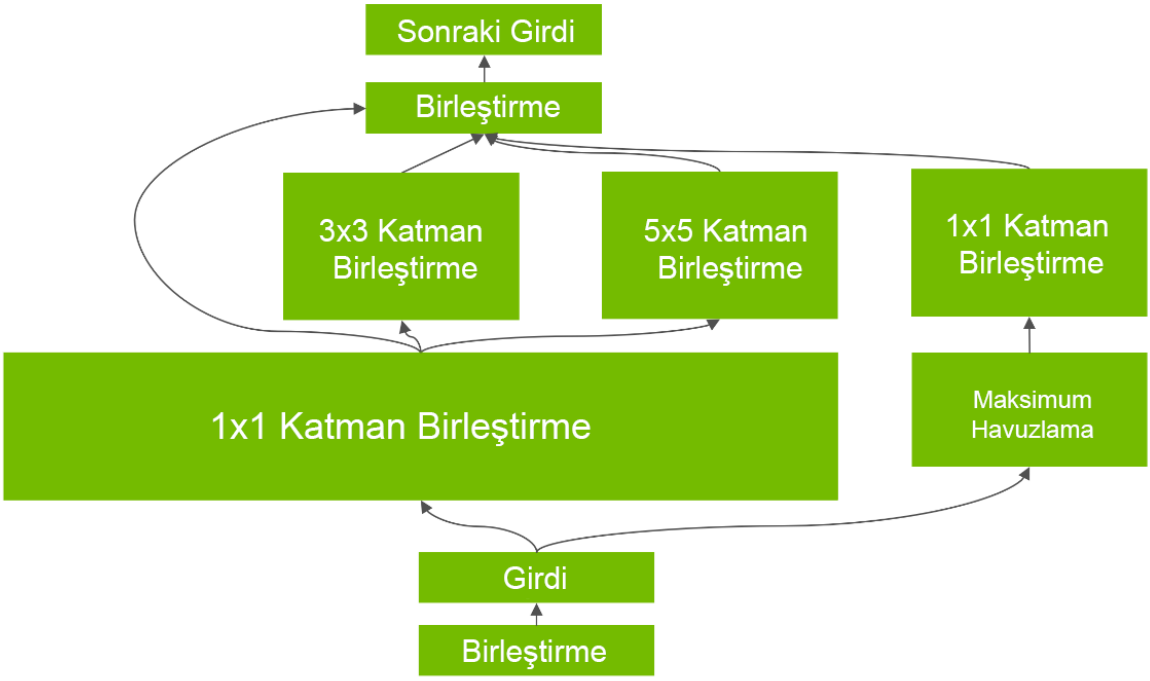
TensorRT, sinir ağında birçok eniyileme ve dönüşüm yapar. İlk olarak gereksiz hesaplamadan kaçınmak için çıktıları kullanılmayan katmanlar elimine edilir. Daha sonra uygun olan yerlerde evrişim, ReLU ve kutuplama katmanları tek bir katman haline getirilir. Şekil 4.3. (a), Şekil 4.3. (b)'de ağın bu birleştirme işleminden sonraki sonucunu göstermektedir. Birleştirilmiş katmanlar CBR etiketiyle gösterilmiştir [69].



(a) Örnek bir ağ (b) Dikey katman birleştirmenin sonucu

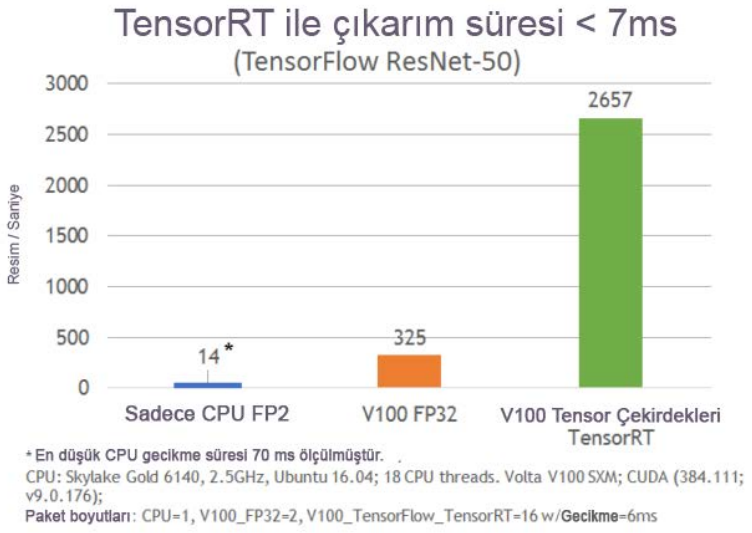
Şekil 4.3. TensorRT dönüşümü

Bir diğer dönüşüm ise yatay katmanları birleştirme veya katmanları toplamadır. Bu işlem, girdi olarak aynı tensoru alan ve benzer parametrelerle aynı işlemler uygulayan katmanları birleştirerek performansı arttırır ve hesaplama verimliliği için tek bir büyük katman oluşturur. Şekil 4.4'teki 1x1 CBR katmanı, Şekil 4.3 (b)'deki üç tane 1x1 CBR katmanını birleştirilerek tek bir katman hale getirilmiş halidir. Bu katmanın çıktısının orijinal giriş grafiğinden sonraki katmanlara beslenmesi için ayrıştırılması gerekmektedir [69].



Şekil 4.4. Yatay katman birleştirme

NVIDIA, yayındalığı TensorRT 4 sürümünde, daha yüksek doğruluk ile INT8 ve FP16 hesaplama imkanı sunmaktadır. Bu alanda gelişmeyi daha da kolaylaştırmak için NVIDIA ve Google mühendisleri ortak çalışarak bu sürüm ile birlikte, TensorRT, Tensorflow 1.7 sürümüne entegre edilmiştir. Şekil 4.5'te Tensorflow-TensorRT entegrasyonu ile ResNet-50 modeli için çıkarım performansı, 8 kat daha hızlı ve gecikme süresi 7 milisaniyenin altında sonuç vermiştir [70].



Şekil 4.5. Tensorflow-TensorRT entegrasyonu ile ResNet-50 [70]

5. SONUÇ

Evrişimli sinir ağında kullanılan hiper parametrelerin etkisinin ölçülebilmesi için bu çalışmada birçok eğitim işi planlanmış ve çıkan sonuçlar analiz edilmiştir. Analizler sonucunda ilk aşamada başarımı düşük çıkan işlerin parameterleri değiştirilerek yüksek başarımlı sonuçlar elde edilmiştir. Evrişimli sinir ağının bir veri setinde kullanılırken, veri seti özelliklerine göre seçilecek hiper parametrelerin farklı olabileceği değerlendirilmiştir.

Eğitilen ilk 144 işte eniyileyici olarak Adam ve RMSprop kullanılan işler paket boyutu 128 ve 256 için beklenen başarımı sağlamamıştır. Bunun üzerine daha düşük değere sahip öğrenme katsayıları ile eğitim tekrar edilmiş ve eniyileyici olarak SGD kullanılan eğitimlerin başarımına yakın sonuçlar elde edilmiştir. Özellikle Adam ve RMSprop için tavsiye edilen 0.001 öğrenme katsayısı yetersiz başarım göstermiş, 0.0001 öğrenme katsayısı ise yüksek başarım seviyesine ulaşmayı sağlamıştır. Öğrenme katsayısı belirlenirken tercih edilen eniyileyiciye göre tek bir değere bağlı kalınmaması, farklı değerler ile eğitim başarımının ölçülmesi önemlidir. Bu çalışma da gerek SGD gerekse Adam ve RMSprop için tavsiye edilen öğrenme katsayılarından farklı değerlerin başarımı artırdığı tespit edilmiştir. Uygun öğrenme katsayısı seçilmesi durumunda SGD ve Adam için başarım seviyesinin aynı olduğu görülmüştür.

Standart VGG-19 ağında havuzlama katmanından önce eklenen paket normalizasyon katmanının eğitim başarımını %4,2 kadar arttırdığı, bunun yanında eğitim süresinde anlamlı bir değişiklik oluşturmadığı; seyreltme kullanımının başarımı %13,3 artırdığı ve eğitim süresini kısalttığı görülmüştür. Her iki özelliğin beraber kullanılması durumunda eğitim başarımı %17,6 artış göstermiştir.

Paket boyutu için en iyi değerlerin sırasıyla 128 ve 100 olduğu görülmüştür. Paket boyutunun yüksek tercih edilmesinin dezavantajı GPU RAM ihtiyacının paket boyutu ile doğru orantılı olmasıdır. Yeterli GPU RAM olmaması durumunda daha düşük tercih edilecek paket boyutunun ağın gerçek başarımını etkileyeceği tespit edilmiştir. Bu durum 144 iş arasında en iyi başarıma sahip olan 50. işin, paket boyutu 25, 32, 50 ve 64 kullanılarak eğitildiğinde, başarımın çok düşük seviyede kalması ile tespit edilmiştir.

Tez çalışmasında planlanan 144 işte 4 argümanla ön işleme kullanılmıştır. Ön işlemenin eğitime katkısı ve maliyetini ortaya çıkarmak için başarımı en yüksek iş

ile ön işleme olmaksızın eğitim yapılmıştır. Yapılan bu eğitim sonrasında ön işleme yapılmayan eğitimin başarımının %37 olduğu görülmüştür. Ön işleme yapılmasının dezavantajı eğitim süresini 1.57 kat oranında artırmasıdır. Ön işleme için ne kadar argüman kullanılması gerektiği sorusunun cevabını bulmak için argüman sayısı 8'e çıkarılıp tekrar eğitim yapılmış ve eğitim başarımının düştüğü görülmüştür. Şekil 3.5'te bir resme 4 ve 8 argümanla ön işleme yapıldığında resimde bozulmanın nasıl bir sonuç ortaya çıkardığı gösterilmiştir. Yapılan eğitim sonuçlarına göre ön işlemenin başarıma katkısının, seçilen ön işleme argüman sayısı ve değerleriyle ilişkili olduğu söylenebilir.

Filtre boyutu olarak 3 veya 5 tercih edilmesinin eğitim başarımı açısından çok farkının olmadığı görülmüştür. Yapılan literatür taramasında filtre boyutu olarak 3'ün kullanılmasının hem eğitim başarımı hem de parametre sayısını azaltması nedeniyle eğitimi hızlandırmasından dolayı tercih edildiği anlaşılmıştır.

Yapılan çalışmanın en büyük kazanımlarından birisi yüksek miktarda eğitim işinin otomasyon ortamına dökülmesidir. Oluşturulan istemci, parametre ve değerlendirme sunucu yapısı eğitimlerin ölçeklenebilmesini mümkün hale getirmiştir. Raporlama kısmında kullanılan grafik ara yüzü ile sonuçlar anlamlı bir şekilde gösterilmiştir.

Yeni bir veri setinin eğitimde kullanılacak uygun hiper parametre değerlerinin tespit edilebilmesi için, bu çalışma sonunda elde edilen başarımı en yüksek işlere ait hiper parametre değerleri ile ızgara arama yöntemine göre eğitim işi varyasyonlarının oluşturulup, eğitim yapılmasının uygun olacağı değerlendirilmektedir.

KAYNAKÇA

[1] J. Brownlee, What is the Difference Between a Parameter and a Hyperparameter?, https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/ (Mayıs, **2018**).

[2] I. Amazon Web Services, Amazon SageMaker, https://docs.aws.amazon.com/sagemaker/latest/dg/IC-Hyperparameter.html (Mayıs, **2018**).

[3] D. Webb, Guideline to select the hyperparameters in Deep Learning, https://stats.stackexchange.com/q/95821 (Mayıs, **2018**).

[4] W. Fu, V. Nair, and T. Menzies, "Why is Differential Evolution Better than Grid Search for Tuning Defect Predictors?," ArXiv e-prints, **2016**, https://ui.adsabs.harvard.edu/#abs/2016arXiv160902613F

[5] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *NIPS*, **2012**.

[6] J. Ward, S. Andreev, F. Heredia, B. Lazar, and Z. Manevska, "Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster," *Eindhoven University of Technology, The Netherlands,* vol. 12, **2011**.

[7] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychol Rev,* vol. 65, no. 6, pp. 386-408, Nov **1958**.

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks,* vol. 2, no. 5, pp. 359-366, **1989**.

[9] S. S. Haykin, *Neural Networks and Learning Machines*, 3rd ed ed. New York: Pearson Education, p. 906 S., **2009**.

[10] S. University, CS231n: Convolutional Neural Networks for Visual Recognition, http://cs231n.github.io/neural-networks-2 (Mayıs, **2018**).

[11] Learning with large datasets, http://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html (Nisan, **2018**).

[12] R. Zadeh, The Hard Thing about Deep Learning, https://www.matroid.com/blog/post/the-hard-thing-about-deep-learning (Mayıs, **2016**).

[13] G. S. John McGonagle, Andrew Hsu, Jimin Khim,Christopher Williams, Backpropagation, https://brilliant.org/wiki/backpropagation/.

[14] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289,* **2015**.

[15] T. U. Günter Klambauer, Andreas Mayr,Sepp Hochreiter, "Self-Normalizing Neural Networks," *1706.02515v5,* p. 102, **2017**.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *2015 IEEE International Conference on Computer Vision (ICCV),* pp. 1026-1034, **2015**.

[17] S. Ruder, "An overview of gradient descent optimization algorithms," ArXiv e-prints, vol. 1609, **2016**, http://adsabs.harvard.edu/abs/2016arXiv160904747R

[18] D. Soni, Improving Vanilla Gradient Descent, https://towardsdatascience.com/improving-vanilla-gradient-descent-f9d91031ab1d (Mayıs, **2018**).
[19] G. B. Orr, Momentum and Learning Rate Adaptation, https://www.willamette.edu/~gorr/classes/cs449/momrate.html (Mayıs, **1999**).
[20] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in Optimizing Recurrent Networks," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 8624-8628, December 01, 2012 **2012**.
[21] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The Expressive Power of Neural Networks: A View from the Width," in *NIPS*, **2017**.
[22] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on Learning Theory*, pp. 907-940, **2016**.
[23] J. Bergstra, G. Desjardins, P. Lamblin, and Y. Bengio, "Quadratic polynomials learn better image features," Technical Report 1337, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, **2009**.
[24] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet Training in 24 Minutes," **2017**.
[25] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *NIPS*, **2017**.
[26] Y. You, I. Gitman, and B. Ginsburg, "Large Batch Training of Convolutional Networks," *ArXiv e-prints,* Publication 08/2017 **2017**.
[27] P. Goyal *et al.*, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *arXiv preprint arXiv:1706.02677,* **2017**.
[28] P. Murugan and S. Durairaj, "Regularization and Optimization strategies in Deep Convolutional Neural Network," ArXiv e-prints, **2017**, https://ui.adsabs.harvard.edu/#abs/2017arXiv171204711M
[29] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, **2014**.
[30] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 1058-1066, **2013**.
[31] K. Gustafson and G. Sartoris, "Assigning Initial Weights in Feedforward Neural Networks," *IFAC Proceedings Volumes,* vol. 31, no. 20, pp. 1053-1058, 1998/07/01/ **1998**.
[32] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, pp. 2377-2385, **2015**.
[33] F. Chollet, Keras Usage of initializers, https://keras.io/initializers/ (**2015**).
[34] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120,* **2013**.
[35] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*: Springer, pp. 9-48, **2012**.

[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249-256, **2010**.
[37] M. Peemen, R. Shi, S. Lal, B. Juurlink, B. Mesman, and H. Corporaal, "The neuro vector engine: Flexibility to improve convolutional net efficiency for wearable vision," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pp. 1604-1609: IEEE, **2016**.
[38] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation,* vol. 1, no. 4, pp. 541-551, **1989**.
[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (Adaptive computation and machine learning). Cambridge, Massachusetts: The MIT Press, pp. xxii, 775 pages, **2016**.
[40] J. I. Jackson, C. H. Meyer, D. G. Nishimura, and A. Macovski, "Selection of a convolution function for Fourier inversion using gridding (computerised tomography application)," *IEEE transactions on medical imaging,* vol. 10, no. 3, pp. 473-478, **1991**.
[41] M. De los Santos, J. Martınez, and S. Cardona, "A convolution application to determine the dynamic response of a railway track," *Mechanical systems and Signal processing,* vol. 9, no. 6, pp. 707-708, **1995**.
[42] S. H. Ahn, "Convolution," ed. http://www.songho.ca, **2014**.
[43] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *CoRR,* vol. abs/1603.07285, **2016**.
[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818-2826, **2016**.
[45] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR,* vol. abs/1409.1556, **2014**.
[46] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," ArXiv e-prints, **2015**, https://ui.adsabs.harvard.edu/#abs/2015arXiv150203167I
[47] B. Velioğlu, "Multi-Subject Brain Decoding Using Deep Learning Techniques," **2016**.
[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097-1105, **2012**.
[49] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818-833: Springer, **2014**.
[50] C. Szegedy *et al.*, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 1-9, **2015**.
[51] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML*, **2015**.
[52] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, vol. 4, p. 12, **2017**.
[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, **2016**.

[54] ImageNet, ImageNet Large Scale Visual Recognition Challenge 2016 Teams, http://image-net.org/challenges/LSVRC/2016/results#team. (Nisan, **2018**).
[55] P.-L. Pröve, Squeeze-and-Excitation Networks, https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eacd7 (20 April, **2017**).
[56] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," ArXiv e-prints, **2017**, https://ui.adsabs.harvard.edu/#abs/2017arXiv170901507H
[57] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision,* vol. 115, no. 3, pp. 211-252, **2015**.
[58] A. Zhai, "Going Deeper on the Tiny Imagenet Challenge," Stanford, **2016**.
[59] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740-755: Springer, **2014**.
[60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, **1998**.
[61] S.-H. Lim, S. R. Young, and R. M. Patton, "An analysis of image storage systems for scalable training of deep neural networks," *system,* vol. 5, no. 7, p. 11, **2016**.
[62] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3642-3649: IEEE, **2012**.
[63] S. Tabik, D. Peralta, and A. Herrera, "A snapshot of image pre-processing for convolutional neural networks: case study of mnist," *Int J Comput Intell Syst,* vol. 10, pp. 555-568, **2017**.
[64] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," **2009**.
[65] L. Li, Classifying Caltech 101 categories of images, https://lihan.me/2018/01/vgg19-caltech101-classification/ (Nisan, **2018**).
[66] F. Chollet, Keras, https://keras.io (Mayıs, **2018**).
[67] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," in *OSDI*, vol. 16, pp. 265-283, **2016**.
[68] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," ArXiv e-prints, **2015**, https://ui.adsabs.harvard.edu/#abs/2015arXiv151107289C
[69] C. G. Allison Gray, Ryan Olson, Shashank Prasanna "Deploying Deep Neural Networks with NVIDIA TensorRT," **2017**.
[70] J. B. Sami Kama, Siddharth Sharma, TensorRT Integration Speeds Up TensorFlow Inference, https://devblogs.nvidia.com/tensorrt-integration-speeds-tensorflow-inference/ (Mayıs, **2018**).

EKLER

EK-1: KERAS RESİM ÖN İŞLEME ARGÜMANLARI

featurewise_center: Boolean veri tipi. Girdi değerini veri seti üzerinde 0 olarak ayarlar.

samplewise_center: Boolean veri tipi. Her örnek ortalamasını 0 olarak ayarlar.

featurewise_std_normalization: Boolean veri tipi. Girişleri data setin std değeri ile böler.

samplewise_std_normalization: Boolean veri tipi. Her girişi std ile böler.

zca_epsilon: zca beyazlama için varsayılan değer 1e-6

zca_whitening : zca beyazlatma uygular.

rotation_range: Tamsayı veri tipi. Rastgele dönüşler için değer aralığı belirler.

width_shift_range: Kesirli veri tipi. Rastgele yatay kaymalar için menzil belirler.

height_shift_range : Kesirli veri tipi. Rasgele dikey geçişler için menzil belirler.

shear_range : Kesirli veri tipi. Kesme şiddeti (Kesim açısı, saat yönünün tersi yönündedir)

zoom_range : Kesirli veri tipi. Rastgele odaklama için aralık [alt,üst].

[lower, upper] = [1-zoom_range, 1+zoom_range]

channel_shift_range: Kesirli veri tipi. Rastgele kanal geçişleri için aralık.

fill_mode: Girdi sınırlarının dışındaki noktalar, belirtilen moda göre doldurulur. varsayılan değer "nearest" yani en yakındır.

'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)

'nearest': aaaaaaaa|abcd|dddddddd

'reflect': abcddcba|abcd|dcbaabcd

'wrap': abcdabcd|abcd|abcdabcd

cval: Kesirli veya tamsayı veri tipi. fill_mode = "constant" iken sınır dışındaki noktalar için kullanılan değer.

horizontal_flip: Mantıksal veri tipi. Girişleri yatay olarak rastgele çevirir.

vertical_flip: Mantıksal veri tipi. Girişleri dikey olarak rastgele çevirir.

rescale: Yeniden ölçeklendirme faktörüdür. Varsayılan değer “None”dır. Varsayılan değer 0 veya başka bir değer ise yeniden ölçeklendirme uygulanmaz. Veriler başka bir dönüşüm uygulamadan önce verilen değerle çarpılır.

preprocessing_function: Girdiye uygulanan işlemdir. Görüntü yeniden boyutlandırıldıktan ve büyütüldükten sonra çalışır.

data_format: ["Channels_first", "channels_last"]. "channels_last" modu, görüntülerin şekilleri (örnekler, yükseklik, genişlik, kanallar) olması gerektiği ve "channels_first" modunun olması gerektiği anlamına gelir.
Keras config dosyasında bulunan “image_data_format” değerine varsayılan olarak “~ / .keras / keras.json” asla ayarlamazsanız, o zaman "channels_last" olacaktır.
validation_split: Kesirli veri tipi. Geçerlilik için ayrılmış görüntülerin kesiti kesinlikle 0-1 aralığında olmak zorundadır.

EK-2: 288 EĞİTİM İŞİNİN HİPER PARAMETRE, EĞİTİM SÜRESİ VE BAŞARIMI

Tablo başlığı açıklamaları:

PN : Paket Normalizasyonu (1: Var, 0: Yok)
Eİ : Eniyileme
FS : Filtre Sayısı
SO : Seyreltme Oranı
PB : Paket Boyutu
Sİ : Seyreltme İşlemi (1: Var, 0: Yok)
ÖK : Öğrenme Katsayısı
FB : Filtre Boyutu
AF : Aktivasyon Fonksiyonu

İş	PN	Sİ	Eİ	ÖK	FS	FB	SO	AF	Epok	PB	Süre(sn)	Başarım
1	1	1	sgd	0.05	32	3	0.3	relu	99	256	48849.0	0.643663
2	1	1	sgd	0.05	32	3	0.5	relu	100	256	49221.0	0.557292
3	1	1	sgd	0.01	32	3	0.3	relu	100	256	50007.0	0.554688
4	1	1	sgd	0.01	32	3	0.5	relu	100	256	49507.0	0.363281
5	1	1	adam	0.05	32	3	0.3	relu	14	256	7956.0	0.020399
6	1	1	adam	0.05	32	3	0.5	relu	14	256	8027.0	0.019965
7	1	1	adam	0.01	32	3	0.3	relu	18	256	9720.0	0.021267
8	1	1	adam	0.01	32	3	0.5	relu	16	256	8656.0	0.021701
9	1	1	rmsprop	0.05	32	3	0.3	relu	14	256	6895.0	0.020399
10	1	1	rmsprop	0.05	32	3	0.5	relu	14	256	6894.0	0.018663
11	1	1	rmsprop	0.01	32	3	0.3	relu	14	256	6908.0	0.021267
12	1	1	rmsprop	0.01	32	3	0.5	relu	24	256	11790.0	0.030382
13	1	1	sgd	0.05	32	3	0.3	elu	93	256	46377.0	0.646701
14	1	1	sgd	0.05	32	3	0.5	elu	100	256	49474.0	0.616319
15	1	1	sgd	0.01	32	3	0.3	elu	98	256	47521.0	0.582899
16	1	1	sgd	0.01	32	3	0.5	elu	100	256	49009.0	0.473958
17	1	1	adam	0.05	32	3	0.3	elu	14	256	6826.0	0.019965
18	1	1	adam	0.05	32	3	0.5	elu	14	256	6853.0	0.019531
19	1	1	adam	0.01	32	3	0.3	elu	18	256	8769.0	0.029514
20	1	1	adam	0.01	32	3	0.5	elu	17	256	8333.0	0.023872
21	1	1	rmsprop	0.05	32	3	0.3	elu	14	256	6923.0	0.019097
22	1	1	rmsprop	0.05	32	3	0.5	elu	14	256	6922.0	0.019531
23	1	1	rmsprop	0.01	32	3	0.3	elu	14	256	6923.0	0.019531
24	1	1	rmsprop	0.01	32	3	0.5	elu	14	256	6977.0	0.020833
25	1	1	sgd	0.05	32	3	0.3	selu	95	256	47079.0	0.655816
26	1	1	sgd	0.05	32	3	0.5	selu	100	256	49878.0	0.641059

27	1	1	sgd	0.01	32	3	0.3	selu	100	256	50285.0	0.567708
28	1	1	sgd	0.01	32	3	0.5	selu	100	256	49475.0	0.502604
29	1	1	adam	0.05	32	3	0.3	selu	14	256	6988.0	0.020399
30	1	1	adam	0.05	32	3	0.5	selu	18	256	8923.0	0.020399
31	1	1	adam	0.01	32	3	0.3	selu	14	256	7094.0	0.021267
32	1	1	adam	0.01	32	3	0.5	selu	16	256	7966.0	0.020399
33	1	1	rmsprop	0.05	32	3	0.3	selu	14	256	7105.0	0.019531
34	1	1	rmsprop	0.05	32	3	0.5	selu	14	256	7065.0	0.019965
35	1	1	rmsprop	0.01	32	3	0.3	selu	14	256	7180.0	0.021267
36	1	1	rmsprop	0.01	32	3	0.5	selu	17	256	8550.0	0.021701
37	1	1	sgd	0.05	64	3	0.3	relu	100	256	43511.0	0.641927
38	1	1	sgd	0.05	64	3	0.5	relu	100	256	43708.0	0.476563
39	1	1	sgd	0.01	64	3	0.3	relu	100	256	43674.0	0.632378
40	1	1	sgd	0.01	64	3	0.5	relu	100	256	54267.0	0.492622
41	1	1	adam	0.05	64	3	0.3	relu	16	256	8897.0	0.019097
42	1	1	adam	0.05	64	3	0.5	relu	14	256	7657.0	0.020833
43	1	1	adam	0.01	64	3	0.3	relu	15	256	8233.0	0.020833
44	1	1	adam	0.01	64	3	0.5	relu	29	256	15639.0	0.021701
45	1	1	rmsprop	0.05	64	3	0.3	relu	14	256	6215.0	0.019097
46	1	1	rmsprop	0.05	64	3	0.5	relu	14	256	6023.0	0.019531
47	1	1	rmsprop	0.01	64	3	0.3	relu	14	256	6312.0	0.019965
48	1	1	rmsprop	0.01	64	3	0.5	relu	26	256	11240.0	0.026042
49	1	1	sgd	0.05	64	3	0.3	elu	100	256	56377.0	0.666233
50	1	1	sgd	0.05	64	3	0.5	elu	100	256	55934.0	0.643229
51	1	1	sgd	0.01	64	3	0.3	elu	88	256	48078.0	0.608941
52	1	1	sgd	0.01	64	3	0.5	elu	99	256	42885.0	0.588542
53	1	1	adam	0.05	64	3	0.3	elu	14	256	7686.0	0.021701
54	1	1	adam	0.05	64	3	0.5	elu	17	256	9308.0	0.020833
55	1	1	adam	0.01	64	3	0.3	elu	14	256	7610.0	0.019097
56	1	1	adam	0.01	64	3	0.5	elu	14	256	7727.0	0.023003
57	1	1	rmsprop	0.05	64	3	0.3	elu	14	256	5962.0	0.018663
58	1	1	rmsprop	0.05	64	3	0.5	elu	14	256	6154.0	0.020399
59	1	1	rmsprop	0.01	64	3	0.3	elu	14	256	5899.0	0.020833
60	1	1	rmsprop	0.01	64	3	0.5	elu	17	256	7556.0	0.021267
61	1	1	sgd	0.05	64	3	0.3	selu	100	256	43542.0	0.636719

62	1	1	sgd	0.05	64	3	0.5	selu	100	256	43598.0	0.625
63	1	1	sgd	0.01	64	3	0.3	selu	69	256	29475.0	0.572049
64	1	1	sgd	0.01	64	3	0.5	selu	100	256	42575.0	0.542969
65	1	1	adam	0.05	64	3	0.3	selu	14	256	6051.0	0.020399
66	1	1	adam	0.05	64	3	0.5	selu	14	256	6011.0	0.021267
67	1	1	adam	0.01	64	3	0.3	selu	17	256	7349.0	0.021267
68	1	1	adam	0.01	64	3	0.5	selu	14	256	6098.0	0.018663
69	1	1	rmsprop	0.05	64	3	0.3	selu	14	256	5973.0	0.020399
70	1	1	rmsprop	0.05	64	3	0.5	selu	14	256	6101.0	0.021267
71	1	1	rmsprop	0.01	64	3	0.3	selu	14	256	6242.0	0.019531
72	1	1	rmsprop	0.01	64	3	0.5	selu	34	256	14710.0	0.029514
73	1	1	sgd	0.05	32	5	0.3	relu	100	256	49833.0	0.572483
74	1	1	sgd	0.05	32	5	0.5	relu	25	256	12456.0	0.032118
75	1	1	sgd	0.01	32	5	0.3	relu	100	256	50731.0	0.619358
76	1	1	sgd	0.01	32	5	0.5	relu	100	256	50423.0	0.400174
77	1	1	adam	0.05	32	5	0.3	relu	14	256	7009.0	0.019965
78	1	1	adam	0.05	32	5	0.5	relu	14	256	7029.0	0.019531
79	1	1	adam	0.01	32	5	0.3	relu	18	256	8996.0	0.021701
80	1	1	adam	0.01	32	5	0.5	relu	14	256	7027.0	0.021267
81	1	1	rmsprop	0.05	32	5	0.3	relu	14	256	7044.0	0.019965
82	1	1	rmsprop	0.05	32	5	0.5	relu	14	256	7058.0	0.019965
83	1	1	rmsprop	0.01	32	5	0.3	relu	14	256	7162.0	0.019097
84	1	1	rmsprop	0.01	32	5	0.5	relu	15	256	7579.0	0.021267
85	1	1	sgd	0.05	32	5	0.3	elu	80	256	39864.0	0.648003
86	1	1	sgd	0.05	32	5	0.5	elu	100	256	49999.0	0.634115
87	1	1	sgd	0.01	32	5	0.3	elu	92	256	45886.0	0.625
88	1	1	sgd	0.01	32	5	0.5	elu	100	256	49855.0	0.596788
89	1	1	adam	0.05	32	5	0.3	elu	14	256	7080.0	0.019965
90	1	1	adam	0.05	32	5	0.5	elu	14	256	7093.0	0.019531
91	1	1	adam	0.01	32	5	0.3	elu	15	256	7592.0	0.021267
92	1	1	adam	0.01	32	5	0.5	elu	15	256	7525.0	0.021267
93	1	1	rmsprop	0.05	32	5	0.3	elu	14	256	7027.0	0.021267
94	1	1	rmsprop	0.05	32	5	0.5	elu	14	256	7151.0	0.018663
95	1	1	rmsprop	0.01	32	5	0.3	elu	18	256	9459.0	0.024306
96	1	1	rmsprop	0.01	32	5	0.5	elu	19	256	10386.0	0.02474

97	1	1	sgd	0.05	32	5	0.3	selu	100	256	51333.0	0.654948
98	1	1	sgd	0.05	32	5	0.5	selu	100	256	51707.0	0.570747
99	1	1	sgd	0.01	32	5	0.3	selu	100	256	50941.0	0.598958
100	1	1	sgd	0.01	32	5	0.5	selu	100	256	51123.0	0.59592
101	1	1	adam	0.05	32	5	0.3	selu	14	256	7348.0	0.019097
102	1	1	adam	0.05	32	5	0.5	selu	14	256	7277.0	0.021267
103	1	1	adam	0.01	32	5	0.3	selu	14	256	7186.0	0.021267
104	1	1	adam	0.01	32	5	0.5	selu	19	256	9944.0	0.021267
105	1	1	rmsprop	0.05	32	5	0.3	selu	14	256	7269.0	0.020399
106	1	1	rmsprop	0.05	32	5	0.5	selu	14	256	7166.0	0.019965
107	1	1	rmsprop	0.01	32	5	0.3	selu	14	256	7214.0	0.021267
108	1	1	rmsprop	0.01	32	5	0.5	selu	17	256	8799.0	0.026476
109	1	1	sgd	0.05	64	5	0.3	relu	100	256	59647.0	0.578993
110	1	1	sgd	0.05	64	5	0.5	relu	100	256	44579.0	0.123264
111	1	1	sgd	0.01	64	5	0.3	relu	95	256	59325.0	0.644097
112	1	1	sgd	0.01	64	5	0.5	relu	100	256	44828.0	0.508681
113	1	1	adam	0.05	64	5	0.3	relu	14	256	8393.0	0.020399
114	1	1	adam	0.05	64	5	0.5	relu	14	256	8380.0	0.021267
115	1	1	adam	0.01	64	5	0.3	relu	14	256	8450.0	0.020399
116	1	1	adam	0.01	64	5	0.5	relu	14	256	5861.0	0.020399
117	1	1	rmsprop	0.05	64	5	0.3	relu	14	256	6108.0	0.019531
118	1	1	rmsprop	0.05	64	5	0.5	relu	15	256	6666.0	0.020399
119	1	1	rmsprop	0.01	64	5	0.3	relu	14	256	6108.0	0.020399
120	1	1	rmsprop	0.01	64	5	0.5	relu	14	256	6198.0	0.019965
121	1	1	sgd	0.05	64	5	0.3	elu	94	256	57406.0	0.652778
122	1	1	sgd	0.05	64	5	0.5	elu	100	256	42501.0	0.609809
123	1	1	sgd	0.01	64	5	0.3	elu	79	256	33029.0	0.639323
124	1	1	sgd	0.01	64	5	0.5	elu	100	256	41547.0	0.611545
125	1	1	adam	0.05	64	5	0.3	elu	14	256	8529.0	0.019965
126	1	1	adam	0.05	64	5	0.5	elu	16	256	7160.0	0.022135
127	1	1	adam	0.01	64	5	0.3	elu	14	256	5738.0	0.019531
128	1	1	adam	0.01	64	5	0.5	elu	16	256	7135.0	0.031684
129	1	1	rmsprop	0.05	64	5	0.3	elu	14	256	6128.0	0.019965
130	1	1	rmsprop	0.05	64	5	0.5	elu	14	256	6345.0	0.023437
131	1	1	rmsprop	0.01	64	5	0.3	elu	14	256	6166.0	0.019531

132	1	1	rmsprop	0.01	64	5	0.5	elu	16	256	7034.0	0.020399
133	1	1	sgd	0.05	64	5	0.3	selu	100	256	42987.0	0.643229
134	1	1	sgd	0.05	64	5	0.5	selu	100	256	44384.0	0.626736
135	1	1	sgd	0.01	64	5	0.3	selu	95	256	41162.0	0.651042
136	1	1	sgd	0.01	64	5	0.5	selu	100	256	44565.0	0.598958
137	1	1	adam	0.05	64	5	0.3	selu	14	256	6029.0	0.019531
138	1	1	adam	0.05	64	5	0.5	selu	14	256	6067.0	0.019097
139	1	1	adam	0.01	64	5	0.3	selu	14	256	6053.0	0.020833
140	1	1	adam	0.01	64	5	0.5	selu	17	256	7297.0	0.021267
141	1	1	rmsprop	0.05	64	5	0.3	selu	14	256	6162.0	0.019965
142	1	1	rmsprop	0.05	64	5	0.5	selu	14	256	6087.0	0.020833
143	1	1	rmsprop	0.01	64	5	0.3	selu	14	256	6111.0	0.023003
144	1	1	rmsprop	0.01	64	5	0.5	selu	14	256	6084.0	0.019097
145	1	1	sgd	0.05	32	3	0.3	relu	100	128	50719.0	0.656661
146	1	1	sgd	0.05	32	3	0.5	relu	100	128	50632.0	0.594161
147	1	1	sgd	0.01	32	3	0.3	relu	93	128	45625.0	0.605263
148	1	1	sgd	0.01	32	3	0.5	relu	100	128	52995.0	0.544408
149	1	1	adam	0.05	32	3	0.3	relu	14	128	8165.0	0.019737
150	1	1	adam	0.05	32	3	0.5	relu	21	128	11369.0	0.020148
151	1	1	adam	0.01	32	3	0.3	relu	14	128	8321.0	0.020559
152	1	1	adam	0.01	32	3	0.5	relu	19	128	10281.0	0.020559
153	1	1	rmsprop	0.05	32	3	0.3	relu	14	128	7319.0	0.020148
154	1	1	rmsprop	0.05	32	3	0.5	relu	14	128	7384.0	0.020148
155	1	1	rmsprop	0.01	32	3	0.3	relu	14	128	7803.0	0.020148
156	1	1	rmsprop	0.01	32	3	0.5	relu	14	128	7610.0	0.019326
157	1	1	sgd	0.05	32	3	0.3	elu	100	128	50703.0	0.665707
158	1	1	sgd	0.05	32	3	0.5	elu	100	128	50673.0	0.671464
159	1	1	sgd	0.01	32	3	0.3	elu	100	128	50912.0	0.619655
160	1	1	sgd	0.01	32	3	0.5	elu	100	128	51257.0	0.600329
161	1	1	adam	0.05	32	3	0.3	elu	14	128	7143.0	0.019737
162	1	1	adam	0.05	32	3	0.5	elu	14	128	7215.0	0.020148
163	1	1	adam	0.01	32	3	0.3	elu	27	128	13779.0	0.021382
164	1	1	adam	0.01	32	3	0.5	elu	15	128	7656.0	0.022204
165	1	1	rmsprop	0.05	32	3	0.3	elu	14	128	7225.0	0.020148
166	1	1	rmsprop	0.05	32	3	0.5	elu	14	128	7236.0	0.020148

167	1	1	rmsprop	0.01	32	3	0.3	elu	14	128	7166.0	0.019737
168	1	1	rmsprop	0.01	32	3	0.5	elu	14	128	7184.0	0.020148
169	1	1	sgd	0.05	32	3	0.3	selu	100	128	51848.0	0.647204
170	1	1	sgd	0.05	32	3	0.5	selu	100	128	52400.0	0.66653
171	1	1	sgd	0.01	32	3	0.3	selu	70	128	36580.0	0.589638
172	1	1	sgd	0.01	32	3	0.5	selu	100	128	53220.0	0.557155
173	1	1	adam	0.05	32	3	0.3	selu	14	128	8137.0	0.019737
174	1	1	adam	0.05	32	3	0.5	selu	15	128	8503.0	0.020559
175	1	1	adam	0.01	32	3	0.3	selu	14	128	7312.0	0.020148
176	1	1	adam	0.01	32	3	0.5	selu	15	128	7956.0	0.020559
177	1	1	rmsprop	0.05	32	3	0.3	selu	14	128	7315.0	0.020559
178	1	1	rmsprop	0.05	32	3	0.5	selu	14	128	7292.0	0.019326
179	1	1	rmsprop	0.01	32	3	0.3	selu	14	128	7296.0	0.020148
180	1	1	rmsprop	0.01	32	3	0.5	selu	15	128	7938.0	0.020148
181	1	1	sgd	0.05	64	3	0.3	relu	100	128	56476.0	0.63898
182	1	1	sgd	0.05	64	3	0.5	relu	100	128	56081.0	0.56949
183	1	1	sgd	0.01	64	3	0.3	relu	100	128	56048.0	0.657484
184	1	1	sgd	0.01	64	3	0.5	relu	100	128	57342.0	0.640625
185	1	1	adam	0.05	64	3	0.3	relu	14	128	7980.0	0.019326
186	1	1	adam	0.05	64	3	0.5	relu	14	128	7989.0	0.018914
187	1	1	adam	0.01	64	3	0.3	relu	23	128	12996.0	0.025905
188	1	1	adam	0.01	64	3	0.5	relu	20	128	11350.0	0.023849
189	1	1	rmsprop	0.05	64	3	0.3	relu	14	128	8137.0	0.019326
190	1	1	rmsprop	0.05	64	3	0.5	relu	14	128	8009.0	0.020559
191	1	1	rmsprop	0.01	64	3	0.3	relu	16	128	9101.0	0.021793
192	1	1	rmsprop	0.01	64	3	0.5	relu	14	128	7958.0	0.021382
193	1	1	sgd	0.05	64	3	0.3	elu	91	128	38886.0	0.660156
194	1	1	sgd	0.05	64	3	0.5	elu	100	128	57293.0	0.695724
195	1	1	sgd	0.01	64	3	0.3	elu	94	128	54097.0	0.639803
196	1	1	sgd	0.01	64	3	0.5	elu	100	128	56378.0	0.634868
197	1	1	adam	0.05	64	3	0.3	elu	14	128	8015.0	0.020559
198	1	1	adam	0.05	64	3	0.5	elu	14	128	8019.0	0.019326
199	1	1	adam	0.01	64	3	0.3	elu	14	128	8042.0	0.019737
200	1	1	adam	0.01	64	3	0.5	elu	14	128	8022.0	0.025905
201	1	1	rmsprop	0.05	64	3	0.3	elu	14	128	8014.0	0.020148

202	1	1	rmsprop	0.05	64	3	0.5	elu	14	128	8022.0	0.020148
203	1	1	rmsprop	0.01	64	3	0.3	elu	25	128	14286.0	0.028372
204	1	1	rmsprop	0.01	64	3	0.5	elu	14	128	8015.0	0.023437
205	1	1	sgd	0.05	64	3	0.3	selu	100	128	58675.0	0.654194
206	1	1	sgd	0.05	64	3	0.5	selu	100	128	58245.0	0.668997
207	1	1	sgd	0.01	64	3	0.3	selu	100	128	58736.0	0.636924
208	1	1	sgd	0.01	64	3	0.5	selu	100	128	60169.0	0.628701
209	1	1	adam	0.05	64	3	0.3	selu	14	128	8364.0	0.020559
210	1	1	adam	0.05	64	3	0.5	selu	14	128	8440.0	0.019326
211	1	1	adam	0.01	64	3	0.3	selu	14	128	8373.0	0.019326
212	1	1	adam	0.01	64	3	0.5	selu	15	128	9057.0	0.022615
213	1	1	rmsprop	0.05	64	3	0.3	selu	14	128	8375.0	0.019737
214	1	1	rmsprop	0.05	64	3	0.5	selu	14	128	8518.0	0.020148
215	1	1	rmsprop	0.01	64	3	0.3	selu	14	128	8474.0	0.020559
216	1	1	rmsprop	0.01	64	3	0.5	selu	14	128	8467.0	0.019326
217	1	1	sgd	0.05	32	5	0.3	relu	100	128	52351.0	0.568257
218	1	1	sgd	0.05	32	5	0.5	relu	14	128	7385.0	0.020559
219	1	1	sgd	0.01	32	5	0.3	relu	88	128	46697.0	0.622533
220	1	1	sgd	0.01	32	5	0.5	relu	100	128	52349.0	0.592516
221	1	1	adam	0.05	32	5	0.3	relu	14	128	7391.0	0.019737
222	1	1	adam	0.05	32	5	0.5	relu	14	128	7427.0	0.019737
223	1	1	adam	0.01	32	5	0.3	relu	18	128	9657.0	0.020559
224	1	1	adam	0.01	32	5	0.5	relu	14	128	7395.0	0.019326
225	1	1	rmsprop	0.05	32	5	0.3	relu	14	128	7399.0	0.019737
226	1	1	rmsprop	0.05	32	5	0.5	relu	14	128	7481.0	0.019326
227	1	1	rmsprop	0.01	32	5	0.3	relu	19	128	10023.0	0.020559
228	1	1	rmsprop	0.01	32	5	0.5	relu	14	128	7403.0	0.018914
229	1	1	sgd	0.05	32	5	0.3	elu	100	128	53524.0	0.671875
230	1	1	sgd	0.05	32	5	0.5	elu	100	128	52604.0	0.672286
231	1	1	sgd	0.01	32	5	0.3	elu	87	128	46246.0	0.636102
232	1	1	sgd	0.01	32	5	0.5	elu	98	128	51515.0	0.630345
233	1	1	adam	0.05	32	5	0.3	elu	14	128	7453.0	0.020559
234	1	1	adam	0.05	32	5	0.5	elu	14	128	7448.0	0.020148
235	1	1	adam	0.01	32	5	0.3	elu	15	128	8085.0	0.020559
236	1	1	adam	0.01	32	5	0.5	elu	17	128	9025.0	0.020559

237	1	1	rmsprop	0.05	32	5	0.3	elu	14	128	7469.0	0.019326
238	1	1	rmsprop	0.05	32	5	0.5	elu	14	128	7550.0	0.019737
239	1	1	rmsprop	0.01	32	5	0.3	elu	14	128	7458.0	0.020148
240	1	1	rmsprop	0.01	32	5	0.5	elu	15	128	8002.0	0.019326
241	1	1	sgd	0.05	32	5	0.3	selu	100	128	54344.0	0.653372
242	1	1	sgd	0.05	32	5	0.5	selu	100	128	53502.0	0.650905
243	1	1	sgd	0.01	32	5	0.3	selu	90	128	48364.0	0.639391
244	1	1	sgd	0.01	32	5	0.5	selu	100	128	54229.0	0.61472
245	1	1	adam	0.05	32	5	0.3	selu	14	128	7595.0	0.019326
246	1	1	adam	0.05	32	5	0.5	selu	14	128	7603.0	0.019737
247	1	1	adam	0.01	32	5	0.3	selu	20	128	10800.0	0.021382
248	1	1	adam	0.01	32	5	0.5	selu	27	128	14574.0	0.034128
249	1	1	rmsprop	0.05	32	5	0.3	selu	14	128	7687.0	0.020148
250	1	1	rmsprop	0.05	32	5	0.5	selu	14	128	7747.0	0.019326
251	1	1	rmsprop	0.01	32	5	0.3	selu	14	128	7602.0	0.019737
252	1	1	rmsprop	0.01	32	5	0.5	selu	20	128	10835.0	0.024671
253	1	1	sgd	0.05	64	5	0.3	relu	100	128	44534.0	0.580729
254	1	1	sgd	0.05	64	5	0.5	relu	100	128	43307.0	0.078559
255	1	1	sgd	0.01	64	5	0.3	relu	100	128	60791.0	0.653783
256	1	1	sgd	0.01	64	5	0.5	relu	100	128	60525.0	0.578536
257	1	1	adam	0.05	64	5	0.3	relu	14	128	8937.0	0.019326
258	1	1	adam	0.05	64	5	0.5	relu	14	128	8881.0	0.019326
259	1	1	adam	0.01	64	5	0.3	relu	14	128	8759.0	0.019326
260	1	1	adam	0.01	64	5	0.5	relu	19	128	8450.0	0.021701
261	1	1	rmsprop	0.05	64	5	0.3	relu	14	128	8912.0	0.020559
262	1	1	rmsprop	0.05	64	5	0.5	relu	14	128	8850.0	0.020559
263	1	1	rmsprop	0.01	64	5	0.3	relu	14	128	8750.0	0.019326
264	1	1	rmsprop	0.01	64	5	0.5	relu	21	128	9389.0	0.025174
265	1	1	sgd	0.05	64	5	0.3	elu	100	128	62765.0	0.667352
266	1	1	sgd	0.05	64	5	0.5	elu	100	128	43422.0	0.635851
267	1	1	sgd	0.01	64	5	0.3	elu	99	128	62059.0	0.668586
268	1	1	sgd	0.01	64	5	0.5	elu	100	128	62133.0	0.643092
269	1	1	adam	0.05	64	5	0.3	elu	14	128	6269.0	0.018663
270	1	1	adam	0.05	64	5	0.5	elu	15	128	9374.0	0.020559
271	1	1	adam	0.01	64	5	0.3	elu	25	128	15370.0	0.025082

272	1	1	adam	0.01	64	5	0.5	elu	15	128	9305.0	0.020559
273	1	1	rmsprop	0.05	64	5	0.3	elu	14	128	8946.0	0.019737
274	1	1	rmsprop	0.05	64	5	0.5	elu	14	128	6451.0	0.020399
275	1	1	rmsprop	0.01	64	5	0.3	elu	14	128	8748.0	0.019326
276	1	1	rmsprop	0.01	64	5	0.5	elu	14	128	8679.0	0.020148
277	1	1	sgd	0.05	64	5	0.3	selu	100	128	45078.0	0.647135
278	1	1	sgd	0.05	64	5	0.5	selu	100	128	66310.0	0.626645
279	1	1	sgd	0.01	64	5	0.3	selu	100	128	67407.0	0.664063
280	1	1	sgd	0.01	64	5	0.5	selu	100	128	62518.0	0.641036
281	1	1	adam	0.05	64	5	0.3	selu	14	128	9277.0	0.019326
282	1	1	adam	0.05	64	5	0.5	selu	14	128	9299.0	0.020148
283	1	1	adam	0.01	64	5	0.3	selu	17	128	11253.0	0.022615
284	1	1	adam	0.01	64	5	0.5	selu	15	128	9842.0	0.024671
285	1	1	rmsprop	0.05	64	5	0.3	selu	14	128	6105.0	0.020399
286	1	1	rmsprop	0.05	64	5	0.5	selu	14	128	9026.0	0.019737
287	1	1	rmsprop	0.01	64	5	0.3	selu	18	128	11313.0	0.019737
288	1	1	rmsprop	0.01	64	5	0.5	selu	14	128	9006.0	0.019326

EK-3: 98 İLAVE EĞİTİM İŞİNİN HİPER PARAMETRE, EĞİTİM SÜRESİ VE BAŞARIMI

Tablo başlığı açıklamaları:

PN : Paket Normalizasyonu (1: Var, 0: Yok)
Eİ : Eniyileme
FS : Filtre Sayısı
SO : Seyreltme Oranı
PB : Paket Boyutu
Sİ : Seyreltme İşlemi (1: Var, 0: Yok)
ÖK : Öğrenme Katsayısı
FB : Filtre Boyutu
AF : Aktivasyon Fonksiyonu

İş	PN	Sİ	Eİ	ÖK	FS	FB	SO	AF	Epok	PB	Süre(sn)	Başarım
1	1	1	adam	0.01	64	3	0.5	elu	100	128	61310.0	0.027138
2	1	1	adam	0.01	32	5	0.3	elu	100	128	57220.0	0.020559
3	1	1	adam	0.01	32	5	0.5	elu	100	128	57572.0	0.020559
4	1	1	adam	0.05	64	3	0.5	elu	100	128	61272.0	0.020148
5	1	1	adam	0.05	32	5	0.3	elu	100	128	57265.0	0.020148
6	1	1	adam	0.05	32	5	0.5	elu	100	128	58803.0	0.019737
7	1	1	adam	0.001	64	3	0.5	elu	100	128	59728.0	0.020559
8	1	1	adam	0.001	32	5	0.3	elu	100	128	58055.0	0.533717
9	1	1	adam	0.001	32	5	0.5	elu	100	128	58860.0	0.020559
10	1	1	adam	0.0001	64	3	0.5	elu	100	128	61672.0	0.694079
11	1	1	adam	0.0001	32	5	0.3	elu	100	128	55763.0	0.661595
12	1	1	adam	0.0001	32	5	0.5	elu	100	128	52558.0	0.648438
13	1	1	adam	1.0e-05	64	3	0.5	elu	100	128	56234.0	0.571135
14	1	1	adam	1.0e-05	32	5	0.3	elu	100	128	52296.0	0.584293
15	1	1	adam	1.0e-05	32	5	0.5	elu	100	128	53558.0	0.546875
16	1	1	rmsprop	0.01	64	3	0.5	elu	100	128	57273.0	0.025905
17	1	1	rmsprop	0.01	32	5	0.3	elu	100	128	52585.0	0.020559
18	1	1	rmsprop	0.01	32	5	0.5	elu	100	128	53140.0	0.020148
19	1	1	rmsprop	0.05	64	3	0.5	elu	100	128	55849.0	0.020559
20	1	1	rmsprop	0.05	32	5	0.3	elu	100	128	52643.0	0.019737
21	1	1	rmsprop	0.05	32	5	0.5	elu	100	128	52285.0	0.019737
22	1	1	rmsprop	0.001	64	3	0.5	elu	100	128	56061.0	0.344161
23	1	1	rmsprop	0.001	32	5	0.3	elu	100	128	52559.0	0.348273
24	1	1	rmsprop	0.001	32	5	0.5	elu	100	128	52943.0	0.023437
25	1	1	rmsprop	0.0001	64	3	0.5	elu	100	128	56455.0	0.612664
26	1	1	rmsprop	0.0001	32	5	0.3	elu	100	128	52384.0	0.618421
27	1	1	rmsprop	0.0001	32	5	0.5	elu	100	128	53272.0	0.598273
28	1	1	rmsprop	1.0e-05	64	3	0.5	elu	100	128	56500.0	0.564145
29	1	1	rmsprop	1.0e-05	32	5	0.3	elu	100	128	53345.0	0.571957
30	1	1	rmsprop	1.0e-05	32	5	0.5	elu	100	128	52318.0	0.539474

31	1	1	sgd	0.01	64	3	0.5	elu	100	128	56108.0	0.622944
32	1	1	sgd	0.01	32	5	0.3	elu	100	128	52714.0	0.658306
33	1	1	sgd	0.01	32	5	0.5	elu	100	128	53410.0	0.633224
34	1	1	sgd	0.05	64	3	0.5	elu	100	128	56161.0	0.669819
35	1	1	sgd	0.05	32	5	0.3	elu	100	128	52441.0	0.652549
36	1	1	sgd	0.05	32	5	0.5	elu	100	128	53162.0	0.620477
37	1	1	sgd	0.001	64	3	0.5	elu	100	128	56373.0	0.409128
38	1	1	sgd	0.001	32	5	0.3	elu	100	128	52391.0	0.5
39	1	1	sgd	0.001	32	5	0.5	elu	100	128	53337.0	0.370888
40	1	1	sgd	0.0001	64	3	0.5	elu	100	128	56521.0	0.150905
41	1	1	sgd	0.0001	32	5	0.3	elu	100	128	52792.0	0.335526
42	1	1	sgd	0.0001	32	5	0.5	elu	100	128	52453.0	0.114309
43	1	1	sgd	1.0e-05	64	3	0.5	elu	100	128	57562.0	0.050576
44	1	1	sgd	1.0e-05	32	5	0.3	elu	100	128	52525.0	0.13528
45	1	1	sgd	1.0e-05	32	5	0.5	elu	100	128	53233.0	0.048109
46	1	1	sgd	0.01	64	3	0.5	elu	100	64	60060.0	0.673878
47	1	1	sgd	0.01	32	5	0.3	elu	100	64	56965.0	0.670673
48	1	1	sgd	0.01	32	5	0.5	elu	100	64	58354.0	0.647837
49	1	1	sgd	0.01	64	3	0.5	elu	100	32	62689.0	0.689503
50	1	1	sgd	0.01	32	5	0.3	elu	100	32	63448.0	0.66266
51	1	1	sgd	0.01	32	5	0.5	elu	100	32	63357.0	0.651843
52	1	1	sgd	0.01	64	3	0.5	elu	100	25	69917.0	0.6796
53	1	1	sgd	0.01	32	5	0.3	elu	100	25	63079.0	0.6636
54	1	1	sgd	0.01	32	5	0.5	elu	100	25	66027.0	0.644
55	1	1	sgd	0.01	64	3	0.5	elu	100	50	64463.0	0.6716
56	1	1	sgd	0.01	32	5	0.3	elu	100	50	60533.0	0.6532
57	1	1	sgd	0.01	32	5	0.5	elu	100	50	62318.0	0.6488
58	1	1	sgd	0.01	64	3	0.5	elu	100	125	56415.0	0.652
59	1	1	sgd	0.01	32	5	0.3	elu	100	125	52951.0	0.6508
60	1	1	sgd	0.01	32	5	0.5	elu	100	125	53055.0	0.6416
61	0	1	sgd	0.01	64	3	0.5	elu	100	128	55416.0	0.627878
62	0	1	sgd	0.01	32	5	0.3	elu	100	128	51856.0	0.602796
63	0	1	sgd	0.01	32	5	0.5	elu	100	128	51814.0	0.639391
64	1	0	sgd	0.01	64	3	0.5	elu	100	128	56160.0	0.551398
65	1	0	sgd	0.01	32	5	0.3	elu	100	128	53212.0	0.55551
66	1	0	sgd	0.01	32	5	0.5	elu	100	128	52495.0	0.551809
67	0	0	sgd	0.01	64	3	0.5	elu	100	128	56431.0	0.450247
68	0	0	sgd	0.01	32	5	0.3	elu	100	128	52020.0	0.442845
69	0	0	sgd	0.01	32	5	0.5	elu	100	128	52692.0	0.428865
70	1	1	sgd	0.05	64	3	0.5	elu	100	128	36513.0	0.370888
71	1	1	sgd	0.05	64	3	0.5	elu	100	64	63344.0	0.020032

72	1	1	sgd	0.05	32	5	0.3	elu	100	64	59639.0	0.653446
73	1	1	sgd	0.05	32	5	0.5	elu	100	64	58791.0	0.653045
74	1	1	sgd	0.05	64	3	0.5	elu	100	32	61403.0	0.020032
75	1	1	sgd	0.05	32	5	0.3	elu	100	32	66743.0	0.610978
76	1	1	sgd	0.05	32	5	0.5	elu	100	32	68238.0	0.020032
77	1	1	sgd	0.05	64	3	0.5	elu	100	25	68404.0	0.02
78	1	1	sgd	0.05	32	5	0.3	elu	100	25	63570.0	0.5812
79	1	1	sgd	0.05	32	5	0.5	elu	100	25	63713.0	0.02
80	1	1	sgd	0.05	64	3	0.5	elu	100	50	61408.0	0.02
81	1	1	sgd	0.05	32	5	0.3	elu	100	50	61497.0	0.6388
82	1	1	sgd	0.05	32	5	0.5	elu	100	50	62028.0	0.6056
83	1	1	sgd	0.05	64	3	0.5	elu	100	100	58155.0	0.6872
84	1	1	sgd	0.05	32	5	0.3	elu	100	100	54522.0	0.6596
85	1	1	sgd	0.05	32	5	0.5	elu	100	100	55636.0	0.6656
86	0	1	sgd	0.05	64	3	0.5	elu	100	128	56273.0	0.096628
87	0	1	sgd	0.05	32	5	0.3	elu	100	128	52515.0	0.652961
88	0	1	sgd	0.05	32	5	0.5	elu	100	128	51584.0	0.020559
89	1	0	sgd	0.05	64	3	0.5	elu	100	128	56089.0	0.564145
90	1	0	sgd	0.05	32	5	0.3	elu	100	128	52909.0	0.538651
91	1	0	sgd	0.05	32	5	0.5	elu	100	128	53135.0	0.554276
92	0	0	sgd	0.05	64	3	0.5	elu	100	128	55045.0	0.522204
93	0	0	sgd	0.05	32	5	0.3	elu	100	128	52279.0	0.019737
94	0	0	sgd	0.05	32	5	0.5	elu	100	128	52243.0	0.145148
95	1	1	sgd	0.05	64	3	0.5	elu	100	256	55934.0	0.643229
96	1	1	sgd	0.05	64	3	0.5	elu	100	128	57292.0	0.695724
97	1	1	sgd	0.05	64	3	0.3	elu	100	256	56376.0	0.666233
98	1	1	sgd	0.05	64	3	0.5	elu	100	128	56463.0	0.625411

EK-4: ÖĞRENME KATSAYISI GÜNCELLEME

S.No.	Epok	Öğrenme Katsayısı (Güncelleme sayısı: 0.75)
1	1	0.05
2	4	0.0375
3	7	0.028125
4	10	0.02109375
5	13	0.015820313
6	16	0.011865235
7	19	0.008898926
8	22	0.006674195
9	25	0.005005646
10	28	0.003754235
11	31	0.002815676
12	34	0.002111757
13	37	0.001583818
14	40	0.001187864
15	43	0.000890898
16	46	0.000668174
17	49	0.000501131
18	52	0.000375848
19	55	0.000281886
20	58	0.000211415
21	61	0.000158561
22	**64**	**0.000118921**
23	67	0.000089191
24	70	0.000066893
25	73	0.00005017
26	76	0.000037628
27	79	0.000028221
28	82	0.000021166
29	85	0.000015875
30	88	0.000011906
31	91	0.00000893
32	94	0.000006698
33	97	0.000005024
34	100	0.000003768

Öğrenme olmaması durumunda her 3 epokta bir öğrenme katsayısı 0.75 katsayısı ile çarpılarak güncellenmektedir. Yukarıdaki çizelge incelendiğinde Adam ve RMSprop için ideal öğrenme katsayısı başlangıç değerine 64. epokta ulaşıldığı görülmektedir.

EK-5: SÖZLÜK

Tezde olabildiğince Türkçe terminoloji kullanılmıştır. Çevrilen İngilizce kelimelerin Türkçe karşılıkları aşağıda yer almaktadır.

İngilizce	Türkçe
Adaptive Moment Estimation	Adaptif Moment Tahmin
Augmentation	Genişletme
Batch Gradient Descent	Paket Eğim Düşümü
Bias	Kutuplama
Classification	Sınıflandırma
Dropconnect	Ağırlık sıfırlama
Dropout	Seyreltme
Feature	Öznitelik
Fully connected layer	Tam bağlaşımlı katman
Gradient	Eğim
Gradient descent	Eğim düşümü
Grid	Izgara
Identity	Birim
Kernel	Filtre
Local Response Normalization (LRN)	Lokal Cevap Normalizasyonu
Loss	Maliyet
Manual	Elle
Mini-Batch Gradient Descent	Mini-Paket Eğim Düşümü
Nesterov Accelerated Gradient	Nesterov Hızlandırılmış Eğim
Orthogonal	Dik
Padding	Dolgu
Photometric distortions	Fotometrik bozunmalar
Pooling	Havuzlama
Region-Based Convolutional Neural Networks (R-CNN)	Bölge-Bazlı Evrişimli Sinir Ağları
Regularization	Düzenlileştirme
Regularization	Düzenlileştirme
Residual Network	Kalıntı Ağ
Squeeze	Sıkıştırma
Stochastic Gradient Descent	Olasalıksal Eğim Düşümü
Stride	Adım sayısı
Stride	Yürüme adımı
Symmetry breaking	Simetri kırılması
Uniform	Düzgün
Uniform distribution	Düzgün dağılım
Universal approximators	Evrensel yaklaşıklayıcı
Vanishing gradient	Kaybolan eğim
Wide Resiudal Network	Geniş Kalıntı Ağ

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Ferhat Kurt

Doğum Yeri : Yalova

Medeni Hali : Evli

E-posta : fkurt@openzeka.com

Adresi : Üniversiteler Mah., 1605. Cad. No:3/1-Z04 Çankaya/Ankara

Eğitim

Lise : Kuleli Askerî Lisesi

Lisans : Kara Harp Okulu

Yüksek Lisans : Hacettepe Üniversitesi

Yabancı Dil ve Düzeyi

İngilizce **:** İleri Düzey

İş Deneyimi

TSK (2003-2016),

Open Zeka Bilgi Teknolojileri Tic. Ltd. Şti. Kurucu (2016-Devam),

NVIDIA Derin Öğrenme Enstitüsü Eğitmeni (2017-Devam)

Deneyim Alanları

Yapay Zeka Uygulamaları,

Proje Yönetimi,

Tezden Üretilmiş Projeler ve Bütçesi

-

Tezden Üretilmiş Yayınlar

-

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

-

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
YÜKSEK LİSANS/DOKTORA TEZ ÇALIŞMASI ORJİNALLİK RAPORU

HACETTEPE ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI BAŞKANLIĞI'NA

Tarih:31/05/2018

Tez Başlığı / Konusu: Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi

Yukarıda başlığı/konusu gösterilen tez çalışmamın a) Kapak sayfası, b) Giriş, c) Ana bölümler d) Sonuç kısımlarından oluşan toplam 93 sayfalık kısmına ilişkin 31/05/2018 tarihinde ~~şahsım~~/tez danışmanım tarafından *Turnitin* adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %2'dir.

Uygulanan filtrelemeler:

1- Kaynakça hariç
2- Alıntılar hariç/~~dâhil~~
3- 5 kelimeden daha az örtüşme içeren metin kısımları hariç

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Uygulama Esasları'nı inceledim ve bu Uygulama Esasları'nda belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini saygılarımla arz ederim.

Tarih ve İmza

Adı Soyadı: Ferhat KURT
Öğrenci No: N13226449
Anabilim Dalı: Bilgisayar Mühendisliği
Programı: Bilgisayar Mühendisliği – Yüksek Lisans
Statüsü: ☒ Y.Lisans ☐ Doktora ☐ Bütünleşik Dr.

31.05.2018

DANIŞMAN ONAYI

UYGUNDUR.

Prof. Dr. Mehmet Önder EFE
(Unvan, Ad Soyad, İmza)