T.C.
NECMETTİN ERBAKAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MİKROSKOBİK GÖRÜNTÜLER ÜZERİNDE DERİN ÖĞRENME ALGORİTMALARI KULLANARAK HASTALIKLI HÜCRELERİN OTOMATİK TANIMLANMASI

Mücahit BÜYÜKYILMAZ

YÜKSEK LİSANS TEZİ

Endüstri Mühendisliği Ana Bilim Dalı

Aralık-2017
KONYA

TEZ KABUL VE ONAYI

Mücahit Büyükyılmaz tarafından hazırlanan "Mikroskobik Görüntüler Üzerinde Derin Öğrenme Algoritmaları Kullanarak Hastalıklı Hücrelerin Otomatik Tanımlanması" adlı tez çalışması 25/12/2017 tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Ana Bilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Sabri KOÇER	……..
Danışman Yrd. Doç. Dr. Ali Osman ÇIBIKDİKEN	……..
Üye Yrd. Doç. Dr. Ahmet Ercan TOPCU	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Ahmet COŞKUN
FBE Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz olarak atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İmza

Mücahit Büyükyılmaz

25.12.2017

ÖZET

YÜKSEK LİSANS TEZİ

MİKROSKOBİK GÖRÜNTÜLER ÜZERİNDE DERİN ÖĞRENME ALGORİTMALARI KULLANARAK HASTALIKLI HÜCRELERİN OTOMATİK TANIMLANMASI

Mücahit BÜYÜKYILMAZ

Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Ana Bilim Dalı

Danışman: Yrd. Doç. Dr. Ali Osman ÇIBIKDİKEN

2017, 56 Sayfa

Jüri
Yrd. Doç. Dr. Ali Osman ÇIBIKDİKEN
Prof. Dr. Sabri KOÇER
Yrd. Doç. Dr. Ahmet Ercan TOPCU

Eimeria parazit türlerini otomatik olarak algılamak ve hastalıklı olup olmadığını tespit etmek amacıyla, Çok Katmanlı Sinir Ağı ve Konvolüsyonel Sinir Ağı derin öğrenme algoritması kullanılarak bir model mimarisi geliştirilmiştir. Modele girdi olarak tavuk ve tavşanlara ait Eimeria mikroskobik görüntüleri kullanılmıştır.

Görüntülerin işlenmesinde OpenCV kütüphanesi'nden faydalanılmıştır. Keras ve Theano kütüphaneleri kullanılarak farklı modeller oluşturulmuş ve testler yapılmıştır. Oluşturulan modeller eğitilerek elde edilen sonuçların test edilebilmesi için, yüklenen mikroskobik görüntüler için sınıflandırma yapan bir web uygulaması geliştirilmiştir. Model sonucunda tavuk veri seti için %87.75 doğruluk oranıyla, tavşan veri seti için %78.42 doğruluk oranıyla hastalıklı hücreler sınıflandırılabilmektedir.

Anahtar Kelimeler: Derin Öğrenme, Görüntü İşleme, Makine Öğrenmesi, Sınıflandırma

ABSTRACT

MS THESIS

AUTOMATED IDENTIFICATION OF DISEASED CELLS FROM MICROSCOPIC IMAGES USING DEEP LEARNING ALGORITHMS

Mücahit BÜYÜKYILMAZ

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF NECMETTIN ERBAKAN UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN INDUSTRIAL ENGINEERING

Advisor: Assistant Prof. Dr. Ali Osman ÇIBIKDİKEN

2017, 56 Pages

Jury
Assistant Prof. Dr. Ali Osman ÇIBIKDİKEN
Prof. Dr. Sabri KOÇER
Assistant Prof. Dr. Ahmet Ercan TOPCU

A model architecture has been developed using the Multi-Layer Neural Network and the Convolutional Neural Network deep learning algorithm to automatically detect Eimeria parasite species and determine if they are diseased. Eimeria microscopic images of chickens and rabbits were used as model inputs.

Images have been processed in the OpenCV library. Different models has been obtained using by Keras and Theano libraries. An application with web user interface has been developed that classifies the obtained model for the run and loaded the microscopic image. As a result of models, cell images can be classified with an accuracy of %87.75 for chicken dataset, %78.42 for rabbit dataset.

Keywords: Deep Learning, Image Processing, Machine Learning, Classification

ÖNSÖZ

Tez çalışmam sırasında yaptığı katkılar ve desteklerden dolayı danışmanım Necmettin Erbakan Üniversitesi Bilgisayar Mühendisliği öğretim üyesi Yrd. Doç. Dr. Ali Osman Çıbıkdiken'e, Harran Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Yrd. Doç. Dr. Mehmet Akif Nacar'a ve Northumbria Üniversitesi öğretim üyesi Prof. Dr. Hüseyin Şeker'e teşekkür ederim.

Mücahit BÜYÜKYILMAZ
KONYA-2017

İÇİNDEKİLER

KISALTMALAR

Kısaltmalar

- **CNN** : Convulational Neural Networks
- **MLP** : Multi Layer Perceptron
- **SGD** : Stochastic Gradient Descent
- **ReLU** : Rectified Linear Units
- **GPU** : Graphics Processing Unit
- **CPU** : Central Processing Unit
- **ANN** : Artificial Neural Networks
- **LMS** : Least Mean Square

1. GİRİŞ

Derin öğrenme, önceden bilinen veriler üzerinde pek çok katmanlı yapay sinir ağı ile model oluşturarak, bilinmeyen veriler için özelliklerin öğrenimi yardımıyla sınıflandırma ve analiz yapılmasına imkan sağlayan makine öğrenmesi algoritmalarını içeren yaklaşımdır.

En temel derin öğrenme algoritmaları çok katmanlı yapay sinir ağı ve konvolüsyonel sinir ağlarıdır. Özellikle görüntü işleme alanından bu algoritmalar yaygın biçimde kullanılmaktadır.

Bilgisayarların hızlarının artması ve öğrenme için gerekli veri miktarının fazlalaşması ile görüntü işlemede makine öğrenmesi algoritmalarının becerisi oldukça başarılı hale gelmiştir. Akademik ve ticari pek çok uygulama bu sayede yaygınlaşmıştır.

1.1. Çalışmanın Amacı

Temel derin öğrenme algoritmaları kullanarak, mikroskobik görüntüler üzerinde hastalık teşhisinde kullanılabilecek bir model ve bu modele bağlı bir uygulama geliştirmek çalışmanın temel amacını oluşturmaktadır.

Görüntü işleme için önerilen algoritma ile; resimler üzerinde ön işleme aşaması ve temel derin öğrenme algoritmalarının Python tabanlı Theano ve Keras kütüphaneleri kullanılarak model oluşturma ve uygulama geliştirme aşamaları gerçekleştirilecektir.

1.2. Çalışmanın Önemi

Kümes hayvanlarına ait hastalıkların tespiti ticari kayıplar açısından önemlidir. Görüntü işleme ve makine öğrenmesi ile otomatik tespit sistemi bu hastalıkların teşhisinde yaygın olarak kullanılan bir yöntemdir. Birleşmiş Milletler Tarım ve Gıda Örgütü'nün 2010 yılında yayımladığı rapora göre Eimeria paraziti hayvanlara bulaşmakta ve kısa sürede ölmelerine sebep olmaktadır. Eimeria protozoan paraziti tespiti zor ve Coccidiosis denilen bir salgın hastalığa sebep olan bir parazittir. Eimeria'nın her cins hayvan için birden fazla türü vardır. Bu hastalık hayvanların

arasında hızlıca yayılmakta ve ölümüne sebep olmaktadır. Farklı Eimeria türleri için dışkı örneklerinden alınan mikroskobik görüntü ile hastalık tespit edilebilmektedir.

Mikroskobik görüntülerin işlenmesi için geliştirilecek bir uygulama ile hastalık teşhisinde kullanılması ticari olarak önemli bir avantaj sağlayacaktır.

1.3. Tezin Yapısı

Tez beş bölümden oluşmaktadır.

Birinci bölümde tezin konusunu oluşturan derin öğrenme ve kümes hayvanlarının mikroskobik görüntü işleme problemi ele alınmıştır.

İkinci bölümde literatürdeki daha önceki çalışmalar incelenmiştir.

Üçüncü bölümde kullanılacak derin öğrenme algoritmaları ve özellikleri ile veri seti hakkında bilgi verilmiştir.

Dördüncü bölümde çok katmanlı yapay sinir ağı ve konvolüsyonel sinir ağı derin öğrenme algoritmaları, belirlenen tavuk ve tavşan veri setlerine uygulanmıştır. Sonuçlara ait bir uygulama geliştirilerek, web uygulaması üzerinden yüklenecek herhangi bir mikroskobik görüntünün hastalıklı olup olmadığı sonucunu veren çalışma da eklenmiştir.

Beşinci bölümde sonuçlar değerlendirilerek gelecek çalışmalar hakkında bilgi verilmiştir.

2. KAYNAK ARAŞTIRMASI

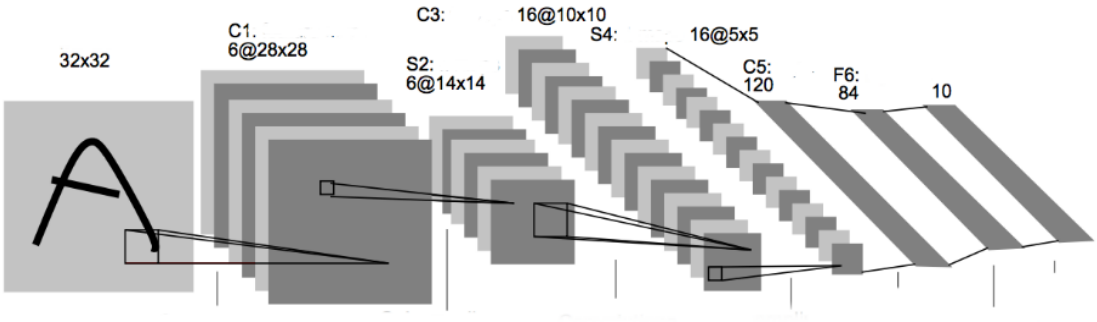
2.1. Derin Öğrenme

Derin öğrenme (*deep learning*); yüksek boyutlu verilerin düşük boyutlu dönüşümlerini oluşturmak için derin mimarileri kullanan çok sayıda algoritmayı kapsamaktadır. Kullanılan derin öğrenme mimarisi giriş verilerinin özelliklerini doğrudan hesaplamak yerine, farklı soyutlama düzeylerine sahip birden fazla hesaplama katmanından oluşur. Her katmanda, derin öğrenme algoritmasına bağlı olarak değişen belirli bir doğrusal olmayan dönüşüm fonksiyonu ile verilerin yeni bir temsilini hesaplar. Bir katmanın çıktısını bir sonraki katmanın girdisi olarak kullanarak, katman başına katmanın kademeli olarak azaldığı hiyerarşik bir ağ oluşturulur. Derin öğrenme kavramı insan beyninin çalışma mekanizmasına benzetilerek, aldığı yeni bilgilerin derin bir anlayışını elde etmek için farklı hiyerarşik soyutlama düzeylerini kullanmaktadır. Böylece her katmandaki soyutlamanın artmasıyla daha güçlü hale gelen yüksek boyutlu girdi verisinin üst düzey **özelliklerini** (*features*) öğrenmektedir (Bengio ve ark., 2014).

Derin öğrenme; hiyerarşik mimarilerdeki birçok bilgi işleme katmanından oluşan, desen sınıflandırması ve özellik çıkarımı için kullanılan makine öğrenme tekniklerini içerir. Grafiksel modelleme, optimizasyon, model tanıma, görüntü işleme, doğal dil işleme ve sinyal işleme gibi çeşitli araştırma alanlarında yoğun olarak kullanılmaya başlanmıştır.

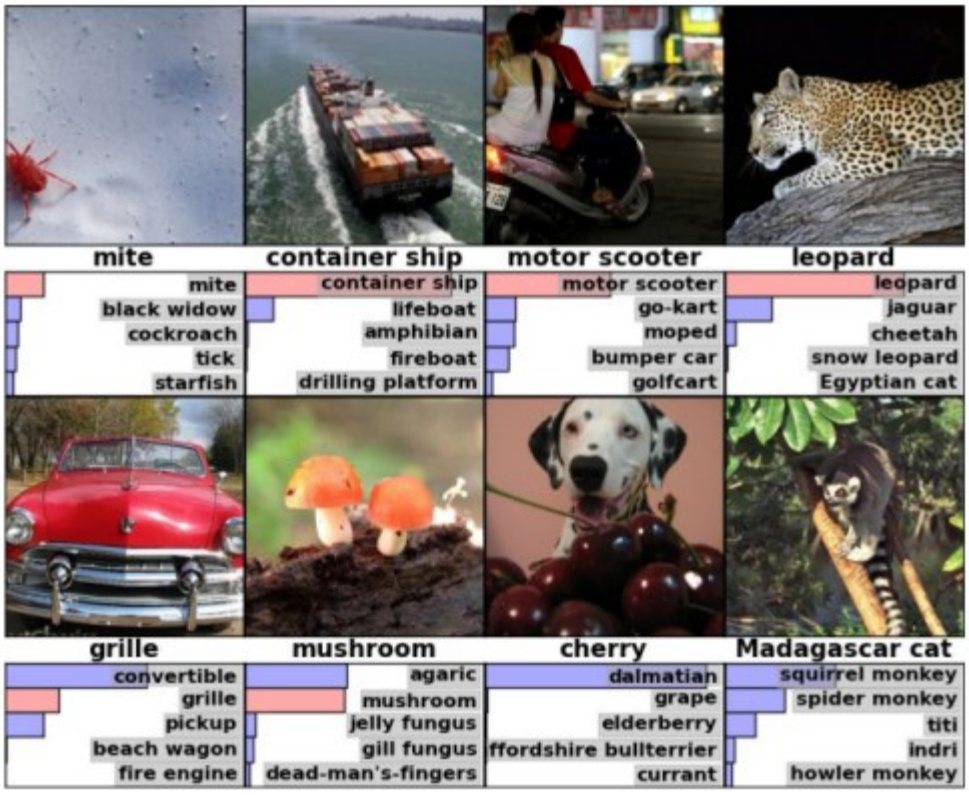
GPU (*Graphics Processing Unit*) ile gelen hesaplama güçlerinin artması, eğitim için kullanılabilecek veri setlerinin büyümesi, yeni ve daha güçlü algoritmaların geliştirilmesi ile birlikte son zamanlarda derin öğrenme algoritmaları yoğun olarak kullanılmaya başlanmıştır.

Derin öğrenme algoritmalarının başlıcalarından sayılan Konvolüsyonel Sinir Ağı kullanarak ilk defa Lecun ve ark. (1998) tarafından el yazısı tanımak için özellik öğrenme yöntemi olarak kullanmışlardır. Oluşturulan LeNet ağında konvolüsyon, alt örnekleme, tam bağlantı ve Gaussian bağlantı olmak üzere dört farklı katmandan faydalanılmıştır. Giriş değeri ve çıkış değeri bir vektör olarak tanımlanmıştır. (Şekil 2.1).



Şekil 2.1. El yazısı tanımak için Lecun tarafından oluşturulan LeNet ağı.

Görüntü işlemede büyük veri setlerinin kullanımı modelin oluşturulmasında büyük katkı sağlamaktadır. ImageNet üzerinde yapılan çalışmada Derin Konvolüsyonel Sinir Ağı kullanılarak, 1,000 sınıflandırılmış grup için 1.2 milyon resim işlenmiştir. Test ve doğrulama için 200,000 resim kullanılmıştır. 8 katmandan oluşturulan ağ üzerinde görüntüler için özellik öğrenme modeli ile 650 bin nöron kullanılarak 60 milyon parametre ve çıkarılmıştır (Krizhevsky ve ark., 2012). Kullanılan test resim verisi Şekil 2.2'de gösterilmiştir.



Şekil 2.2. ImageNet veri seti üzerinde çıkartılan derin öğrenme modeli için kullanılan test resim verisi.

Google'daki araştırmacılar tarafından, ImageNet Large-Scale Visual Recognition Challenge 2014 veri setinde öğrenme için 1.2 milyon, test için 100,000 ve doğrulama için 50,000 resim kullanılarak 22 katmanlı bir derin öğrenme mimarisi (GoogleNet) çalıştırılmıştır. Resimlerin içeriğinin sınıflandırılması ve tespiti için kullanılan mimari Şekil 2.3'de verilmiştir (Szegedy ve ark., 2015).

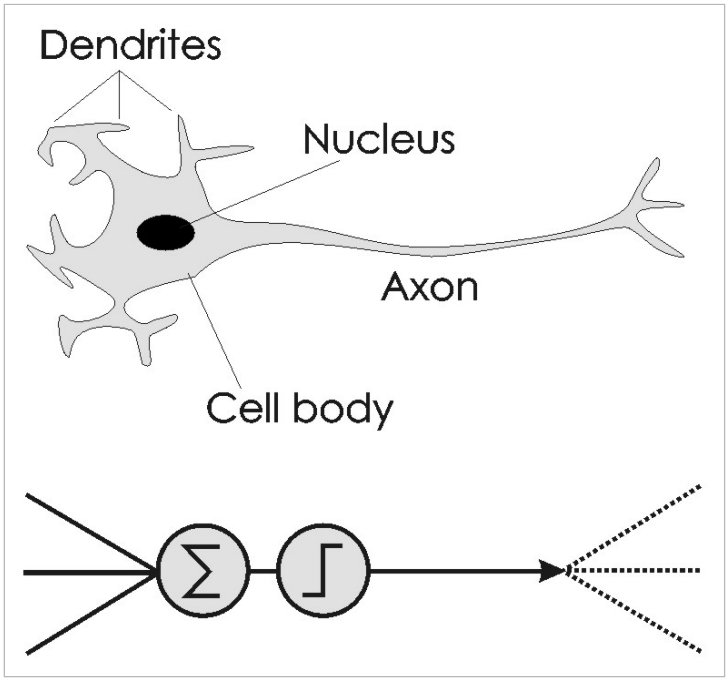
Şekil 2.3. GoogleNet derin öğrenme mimarisi.

2.1.1. Yapay Sinir Ağı

Zihinsel aktivitelerin, beyin hücrelerindeki ağların temelini oluşturan **nöron** (*neuron*) olarak adlandırılan yapının elektro kimyasal etkileşimlerden oluştuğu fikrinden yola çıkarak, McCulloch ve Pitts (1943) tarafından nöronun matematiksel modeli tanıtılmıştır. Biyolojik bir nöronun çalışma prensibi aşağıdaki bileşenler ile sağlanmaktadır (Seymour 1997);

- **Dendrit** (*Dendrite*): Sinir hücrelerinden gelen sinyalin sinir hücre çekirdeğine iletilmesinden sorumludur.
- **Hücre Gövdesi** (*Cell body*): Dendrit üzerinden gelen sinyalleri toplayan merkezdir.
- **Akson** (*Axon*): Çekirdekten alınan bilginin sonraki sinir hücresine iletilmesini sağlar.
- **Sinaps** (*Synapses*): Akson'dan gelen bilginin ön işlemden geçirdikten sonra diğer sinir hücrelerine iletilmesidir. Bu ön işlem, sinyalin belli bir eşik değerine göre değiştirilmesidir.

Nöronun bileşenleri ve yapay sinir arasında benzetim yapılarak bir ilişki kurulmuştur (Şekil 2.4). Böylece nöronun çalışma prensibi ile "öğrenmenin" yapay bir şekilde elde edilmesi sağlanmıştır.



Şekil 2.4. Nöron ile yapay sinir arasındaki benzetim ilişkisi.

Yapay sinir; birden fazla **giriş değeri** (*input*) ve bu giriş değerlerinin ağırlıklı toplamlarının belli eşik değerine ulaşması ile **çıkış değeri** (*output*) elde edilmesi yaklaşımı ile çalışır.

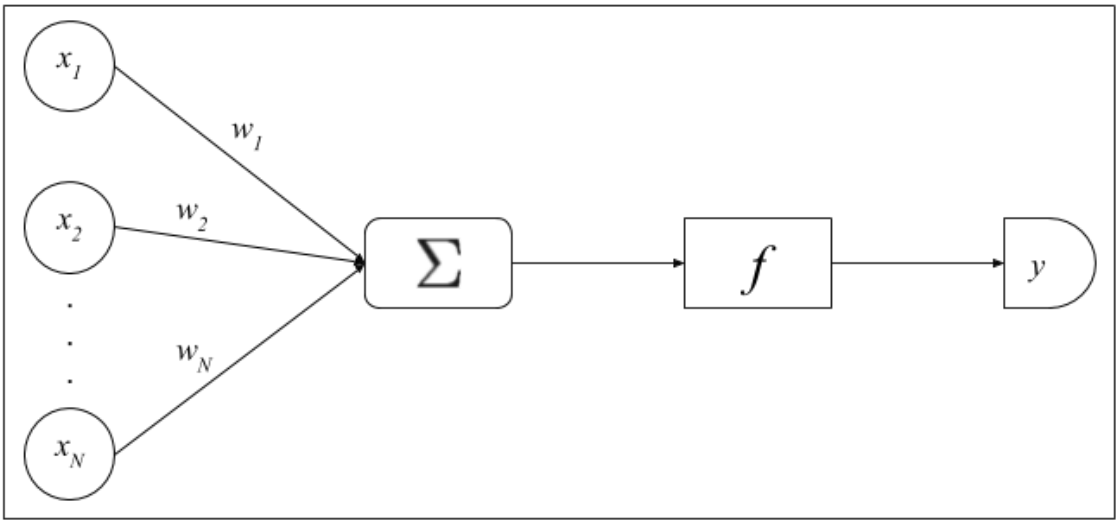
Yapay sinir yaklaşımı matematiksel olarak ifade edildiğinde;

- N : Giriş değeri sayısı
- x : Giriş değerleri
- w : Ağırlık (*weight*)
- b : Sapma (*bias*)
- f : Doğrusal olmayan aktivasyon fonksiyonu (*activation function*)
- y : Çıkış değeri

olmak üzere,

$$y(x) = f(\sum_{i=1}^{N} w_i x_i + b) \tag{1}$$

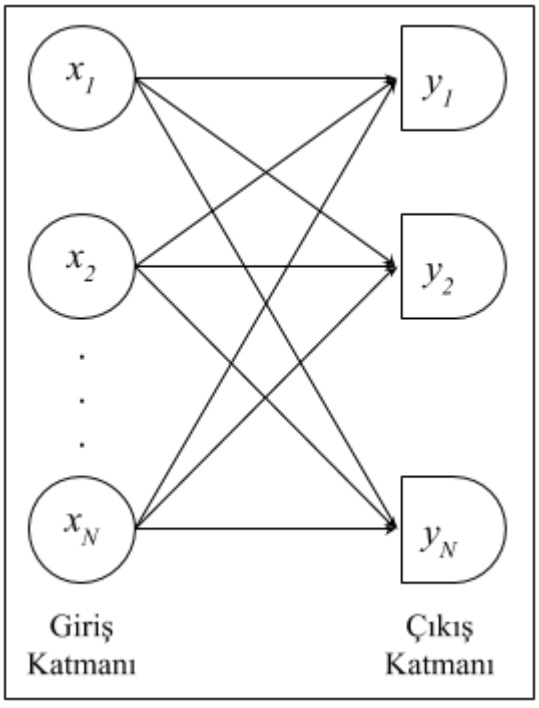
olarak gösterilebilir. Burada y ile oluşturan temel yapıya **algılayıcı** (*perceptron*) adı verilir (Şekil 2.5) (Rosenblatt, 1962).



Şekil 2.5. Algılayıcı matematiksel modeli.

2.1.2. Tek Katmanlı Yapay Sinir Ağı

Algılayıcı aynı zamanda **tek katmanlı yapay sinir ağı** (ANN-*Artificial Neural Networks*) olarak kabul edilebilir. Tek katmanlı yapay sinir ağında; (1) Bazı girdi verileri alınır, (2) girdi verileri üzerinde ağırlıklı toplamlar hesaplanır ve (3) hesaplanan dönüşüme doğrusal olmayan aktivasyon fonksiyon uygulanır (Şekil 2.6). Elde edilen sonuç bir sınıflandırma için değer üretir.



Şekil 2.6. Tek katmanlı yapay sinir ağı modeli.

Sınıflandırma için elde edilecek sonuç, aktivasyon fonksiyonuna göre değişebilir. Örneğin; eğer sigmoid fonksiyonu kullanılırsa sonuç için [0,1] gibi bir değer üretilir. Aktivasyon fonksiyonu olarak genellikle sigmoid, tanh ve ReLU kullanılmaktadır.

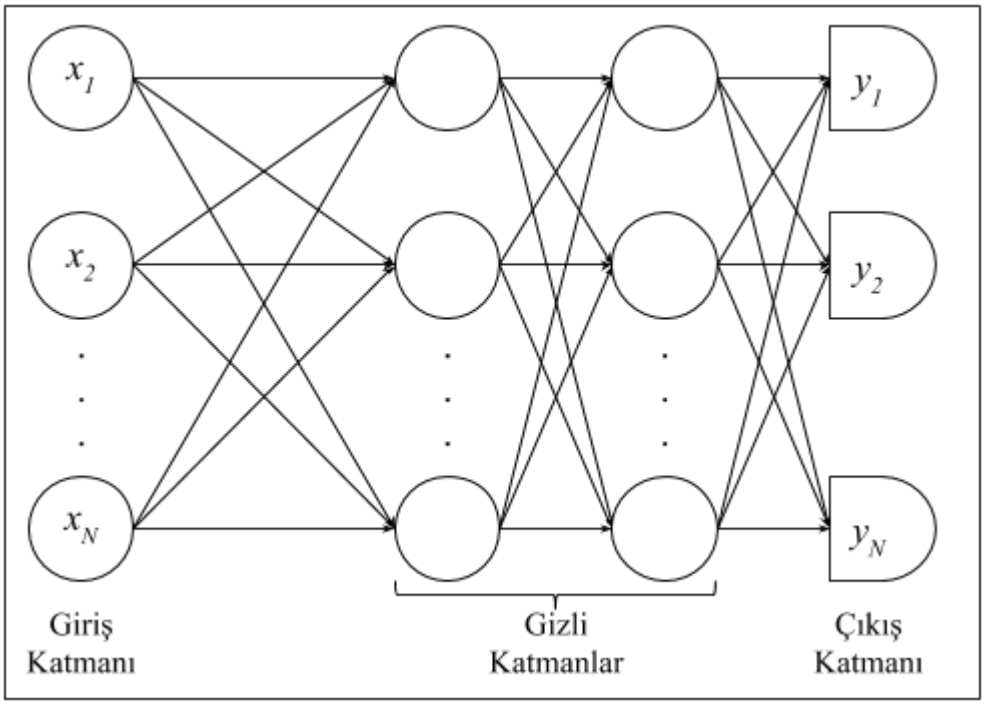
2.1.3. Çok Katmanlı Yapay Sinir Ağı

Yapay sinir ağında çıkış değeri olarak elde edilen sonuç yeniden giriş değeri alınarak işleme alındığında kullanılan katman sayısı artar. İşleme alınarak yeniden ağa girmesini sağlayan aradaki katmanlara **gizli katman** (*hidden layer*) adı verilir. Gizli katmanlar içeren ağ **çok katmanlı yapay sinir ağı** (MLP-*Multi Layer Perceptron*) olarak adlandırılmaktadır (Rumelhart ve ark., 1986).

Yeniden işleme alma öğrenmenin iyileştirilmesi amacıyla yapılır. Burada iki türlü işleme alma söz konusudur:

- İleri doğru işlem (*feed-forward*)
- Geriye yayılımlı işlem (*backpropagation*)

İleri doğru işlemde, bir önceki katmanda elde edilen çıktı değeri doğrudan bir sonraki katmana aktarılır ve tüm yapay sinir ağı işlemleri yeniden yapılır (Şekil 2.7).



Şekil 2.7. İleri doğru işlemli çok katmanlı yapay sinir ağı modeli.

Geriye yayılımlı işlemde, yapay sinir ağında elde edilen sınıflandırma sonucu, eldeki veri ile değerlendirildiğinde giriş değerlerinin ağırlıklarının yeniden hesaplanarak yeni bir giriş değeri oluşturulması öğrenmeyi iyileştirmektedir. Bu şekilde yapılan işlem ile yeni giriş değerleri tekrar yapay sinir ağında çalıştırılır. Burada; beklenen çıktı ile hesaplanan değer arasındaki hatanın minimize edilerek işleme alınması hedeflenmektedir. Hatanın minimize edilmesinde en küçük ortalama kareler (LMS-*least mean squares*) için "eğim iniş (*gradient descent*) metodu" kullanılır. Metod başlangıçtaki ağırlık ve sapma değerlerinin seçimi problemini çözmek için uygundur. En küçük ortalama kareler yöntemi ile ağırlıklar ***w***, hedeflenen değer ***t*** ve hesaplanan değer ***z*** olmak üzere arasındaki hata;

$$J(w) = \frac{1}{2}(t - z)^2 \tag{2}$$

ile bulunur. Ağırlıklar rastgele değerlerle başlatılır ve öğrenme oranı (*learning rate*) η olmak üzere hata aşağıdaki şekilde azaltılarak ilerlenir:

$$\Delta w = -\eta \frac{\partial J}{\partial w} \tag{3}$$

Elde edilen değer iterasyon yöntemi ile bir sonraki ağa aşağıdaki gibi aktarılır:

$$w(m + 1) = w(m) + \Delta w(m) \tag{4}$$

Böylece elde edilen sonuçlar bir sonraki gizli katman için giriş değeri olarak işleme alınır (Duda ve ark., 2012).

2.1.4. Konvolüsyonel Sinir Ağı

Konvolüsyonel sinir ağları (CNN-*Convolutional Neural Networks*), özellikle bilgisayarda görme ve görüntü işlemede kullanılacak derin öğrenme algoritmalarındandır. Beynin görsel bilgiyi işleme tekniğinden esinlenerek ortaya çıkarılmıştır (Bengio, 2009).

2.1.4.1. Konvolüsyon

Sinir ağına adını veren **konvolüsyon** (*convolution*); değişken büyüklükteki giriş verisini işleyebilen bir **doğrusal işlem**dir (*linear operation*). Giriş verisi genellikle iki boyutlu bir dizi ve **çekirdek** (*kernel*) öğrenilebilir parametreleri içeren bir iki boyutlu dizidir. Konvolüsyon işlemi matematiksel olarak;

$$s[i,j] = (I \times K)[i,j] = \sum_m \sum_n I[m,n]K[i-m,j-n] \tag{5}$$

şeklinde ifade edilir. Burada;

- ***I*** : m ve n boyutlu resim için giriş değeri
- ***K*** : Çekirdek
- ***s*** : *i* ve *j* koordinatlarına sahip sonuç değeri

olarak tanımlanmıştır.

Konvolüsyon işlemi giriş resmine uygulandığında, belirli özelliklerin girdide nerede göründüğünü gösteren 2 boyutlu özellik haritaları oluşturulur. Parametre paylaşımı katmanda dönüştürme işlemine eş değer olmasını sağlar, yani girdi değişirse çıkış aynı şekilde değişir. Girişteki nesne hareket ettiyse, onun gösterimi aynı miktarda çıkışta hareket ettirilir (Bengio ve ark., 2014).

2.1.4.2. Havuzlama

Havuzlama (*pooling*) katmanları çıkış birimlerinin komşu değerlerinin bir özetini çıkartarak çekirdek haritasındaki değerlerle değiştirilmesini sağlar. Genel olarak havuz katmanı belli aralıklara bölünmüş bir ızgara gibi düşünülebilir. Her bir ızgara havuzdaki $z \times z$ boyutlu merkezi olarak alınmış komşu değerini taşır. Eğer $s = z$ alınırsa bu ızgara değerleri çakışmaz.

Büyük boyutlu resimlerde havuz katmanlarına sahip özellikleri tanımlamak için çeşitli istatistiksel yaklaşımlar kullanılmaktadır. Bir görüntün bölgesinin değerini ortalaması ya da maksimum değerleri alınarak hesaplanabilir. Bu istatistiki yaklaşımlar nedeniyle, en yaygın kullanılan havuzlama işlemleri **maksimum havuzlama** (*max pooling*) ve **ortalama havuzlama** (*average pooling*) şeklinde adlandırılır (Barkmeyer, 2015; Bengio ve ark., 2014).

2.1.4.3. Öğrenme

Konvolüsyon sinir ağında öğrenme aşamasında, hata değerini minimize eden maliyet fonksiyonu için genellikle Stokastik Eğim Azalması (SGD-*Stochastic Gradient Descent*) yöntemi kullanılmaktadır (Bottou, 1991).

θ öğrenme parametresi (*learning parameter*) ve ε öğrenme oranı (*learning rate*) olmak üzere Stokastik Eğim Azalması;

$$\theta^{k+1} = \theta^{k} - \varepsilon_k \frac{\partial L(\theta^k, z)}{\partial \theta^k} \tag{6}$$

şeklinde gösterilir (Bengio ve ark., 2014).

2.1.4.4. ReLU

Bir yapay sinir ağında, nöronun çıktı değerini geçiren aktivasyon fonksiyonu için genellikle *tanh* ya da *sigmoid* fonksiyonları seçilmektedir. Ancak her iki doğrusal olmayan fonksiyon da yavaş çalışmaktadır. Bu nedenle Konvolüsyonel Sinir Ağı'nda

genellikle **Düzeltilmiş Doğrusal Birim** (ReLU- *Rectified Linear Unit*) fonksiyonu tercih edilmektedir. ReLU fonksiyonu;

$$f(x) = max(0, 9) \tag{7}$$

olarak gösterilir (Hinton ve ark., 2010).

2.1.4.5. Kaçınma

Kaçınma (*dropout*), gizli nöronların çıktılarını önceden belirlenmiş bir olasılık ile sıfıra yaklaştırarak **aşırı uyumsuzluğu** (*over fitting*) önlemeye yardımcı bir tekniktir. Literatürde bu değer genellikle 0.5 veya 0.7 olarak seçilmiştir (Krizhevsky ve ark. 2012, Szegedy ve ark., 2015).

2.1.4.6. Softmax

Resim sınıflandırmasında Konvolüsyonel Sinir Ağı'nın son katmanı genellikle Softmax katmanı olarak kullanılmaktadır (Bishop, 2006). Bu katmanda (8) ile, K boyutlu z vektörünün değerleri K boyutlu ve değerleri 0 ile 1 arasında değişen $\sigma(z)$ vektörüne dönüştürülür:

$$\sigma(z)_j = \frac{e_j^z}{\sum_{k=1}^{K} e_k^z} \tag{8}$$

2.2. Mikroskobik Görüntü ile Parazit Türleri Üzerinde Yapılan Çalışmalar

Birleşmiş Milletler Tarım ve Gıda Örgütü'nün 2010 yılında yayımladığı rapora (FAO, 2010) göre Eimeria (Kaupp, 1917) hayvanlara bulaşan ve kısa sürede ölmelerine sebep olan bir tür parazittir. Bu nedenle parazit türleri üzerinde görüntü işleme ile ilgili pek çok çalışma yapılmaya devam edilmektedir. Bu çalışmalarda hastalık tespitinde başarı oranının artırılması için farklı farklı algoritmalar denenmektedir.

Hücre görüntüleri üzerinde önce istatistiksel yöntemler kullanarak Eimeria türlerinin sınıflandırması üzerine çalışılmıştır (Kučera ve ark., 1991).

Farklı hayvan türleri üzerinde parazit tespiti için morfolojik algoritmalar kullanılmaya başlanmış, domuz Eimeria'larının ayrıştırılması ile ilgili çalışmalar yapılmıştır (Daugschiesa ve ark., 1999).

Castañón ve ark. (2007) hücre resimleri üzerinde Eimeria tespiti ile ilgili yaptıkları çalışmada başarı oranını artırarak %85.75 doğrulukta sonuç elde etmişlerdir. Bunun yanısıra insan yumurta hücrelerine ait mikroskobik resimler kullanılarak yapılan sınıflandırma çalışmalarında %97.70 başarım elde edilmiştir (D. Avci ve ark., 2009).

Mikroskobik resimler üzerinde insanlarda bağırsak parazitlerinin tespiti üzerine görüntü işleme yöntemi ile yapılan çalışmalarda elde edilen sonuçlar, tespitin tam otomasyonuna yönelik başarılı yaklaşımlar sağlandığını göstermektedir (Suzuki ve ark., 2013).

Hadipour ve ark. (2013) yerel tavuklar üzerinde Eimeria bulaşan tavuklar ve Eimeria türleri ile ilgili çalışmalar yapmışlardır. Yine tavuk resimlerinden oluşan veri seti ile K-Nearest Neighbour kullanarak yapılan çalışmada %82±3.44 başarı sağlanmıştır (Abdalla ve ark., 2015).

2.3. Kullanılan Teknolojiler

2.3.1. Python

Python platform bağımsız, nesne yönelimli üst seviye bir programlama dilidir. Dinamik bir programlama dili olan Python sade bir yazım kuralı sunmaktadır. Yazılan kodlar derleme ihtiyacı olmadan bir yorumlayıcı üzerinde çalışmaktadır. Bu sayede geliştirilen kod herhangi bir değişikliğe ihtiyaç duyulmadan farklı işletim sistemlerinde ve farklı mimarilerde çalıştırılabilmektedir. Açık kaynak kodlu olarak geliştirilmekte ve dağıtılmaktadır.

Yaygın geliştirici ve kütüphane desteği sayesinde Python kullanarak Web uygulamaları, masaüstü uygulamaları, bilimsel ve matematiksel hesaplamalar, eğitim ve iş dünyası uygulamarı başta olmak üzere her konuyla ilgili uygulama geliştirmek mümkündür. Uygulama için ihtiyaç olan kütüphanelerin yönetimi, pip adı verilen bir paket yönetici aracılığı ile yapılabilmektedir. Bu çalışmada geliştirilen uygulamaların tamamı Python programlama dili ve aşağıda ayrıntılı olarak bahsedilen Python paketleri

kullanılarak geliştirilmiştir. Geliştirilen uygulamalar Linux işletim sistemi üzerinde çalıştırılmış ve sonuçlar elde edilmiştir.

2.3.2. Scipy/Numpy

SciPy, Python dili için bilimsel hesaplamalarda kullanılmak üzere geliştirilmiş kütüphane setidir. İçerisinde SciPy Lib, NumPy, Mathplot, Sympy gibi paketler bulunmaktadır.

NumPy, *n* boyutlu dizilerde (matrislerde) yapılan matematiksel işlemleri kolaylaştırmak için geliştirilmiştir. NumPy kullanarak matris oluşturma, toplama ve çarpma gibi işlemler tanımlı fonksiyonlar aracılığı ile kolayca yapılabilmektedir. NumPy içerisindeki hesaplama fonksiyonları C, C++ gibi derlenebilen diller kullanılarak geliştirilmiştir. Bu sayede NumPy ile yapılan hesaplama işlemleri, Python temel fonksiyonları ile yapılan hesaplamalardan daha performanslı ve hızlı çalışmaktadır (Van Der Walt ve ark., 2011).

Bu çalışmada veri setinin oluşturulması, dönüştürülmesi ve elde edilen sonuçların analiz edilmesi aşamalarında NumPy dizileri ve fonksiyonları kullanılmıştır.

2.3.3. OpenCV

OpenCV akademik ve ticari ihtiyaçlar için kullanılan açık kaynak kodlu bir görüntü işleme kütüphanesidir. İçerisinde 2500'den fazla görüntü işleme algoritması bulunan kırk yedi binin üzerinde geliştirici tarafından C/C++ dilleri kullanılarak geliştirilmiştir. OpenCV kütüphanesi C++, C, Python, Java ve MATLAB gibi birçok dil içerisine kullanılabilmektedir. Bununla birlikte OpenCV; Windows, Linux, Mac OS ve Android platformları üzerinde çalışabilmektedir (Beyeler, 2015). Bu çalışamada ön işlem aşamasında OpenCV kütüphanesi kullanılmıştır.

2.3.4. Theano

Theano çok boyutlu diziler dahil olmak üzere matematiksel ifadeleri tanımlama, optimize etmeye ve çalıştırmaya imkan sağlayan bir Python kütüphanesidir. NumPy ile entegre olabilen Theano içerisinde NumPy ile oluşturulan diziler direkt kullanılabilmektedir. Theano kullanarak tanımlanan matematiksel işlemler GPU ve CPU üzerinde ek bir işleme gerek olmadan çalıştırılabilmektedir. Derin öğrenme ve diğer makine öğrenmesi algoritmalarının bilgisayar üzerinde matematiksel olarak tasarlanması ve çalıştırılması için arayüzler sunmaktadır. Theano daha iyi performans sağlamak için matematiksel ifadeleri sadeleştirmekte ve derlemektedir. Bu sayede daha doğru ve daha hızlı sonuçlar sunmaktadır (Bergstra ve ark., 2010). Bu çalışmada Keras kütüphanesiyle birlikte Theano kullanılmıştır.

2.3.5. Keras

Keras üst seviye bir yapay sinir ağı arayüz kütüphanesidir. Python dili ile geliştirilen Keras Theano veya TensorFlow kütüphanelerinin üstünde çalışabilir. Keras hızlı ve kolay bir şekilde yapay sinir ağları oluşturmak ve çalıştırmak için geliştirilmiştir. Theano etkileşimi sayesinde geliştirilen yapay sinir ağları CPU veya GPU üzerinde değişiklik yapılmadan çalıştırılabilir. Keras sıralı ağlar, çoklu giriş ve zaman serisi verileri üzerinde hızlı modelleme ve çalıştırma imkanı sunmaktadır. Ayrıca Keras geliştiricilerin kendi aktivasyon, öğrenme ve kayıp algoritmalarını modelleyip çalıştırmalarına imkan sağlayan bir kütüphane desteği sunmaktadır. Keras ile modellenen ve eğitilen ağların ağırlıkları HDF5 dosya formatında kaydedilebilmekte ve sonradan kullanılabilmektedir. Keras modellenen ağları görsel olarak çizebilmekte ve resim olarak kaydedilebilmesine imkan sağlamaktadır (Chollet, 2015).

3. MATERYAL VE YÖNTEM

3.1. Görüntü İşleme İçin Derin Öğrenme Algoritmaları

Görüntü işlemede en çok kullanılan Çok Katmanlı Yapay Sinir Ağı ve Konvolüsyonel Yapay Sinir Ağı algoritmalarının belirlenen veri setleri üzerinde **özellik öğrenme** yöntemi yardımıyla hastalık tespit modeli geliştirilmiştir.

3.2. Veri Seti

Bu çalışmada kullanılan veri seti São Paulo Biyomedikal Enstitüsü, Parazitoloji Bölümü, Coccidia Moleküler Biyoloji Laboratuarı (Laboratory of Molecular Biology of Coccidia at the Departament of Parasitology of the Institute of Biomedical Sciences in São Paulo) ve São Paulo Üniversitesi, São Carlos Fizik Enstitüsü, Sibernetik Vizyon Araştırma Grubu (The Cybernetic Vision Research Group at The Institute of Physics in São Carlos, University of São Paulo) tarafından hazırlanmış ve araştırmacıların kullanımına sunulmuştur (http://www.coccidia.icb.usp.br/imagedb/). Veri seti farklı **Eimeria** türleri için dijital mikroskobik resimler içermektedir. Veri seti tam mikroskobik görüntü ve hücre olarak parçalanmış resimler halinde indirilebilir. Bu çalışmada parçalanmış hazır resimler kullanılmıştır.

Tavuk veri seti yedi farklı **Eimeria** türü için yerel tavuklardan alınan örnekler kullanılarak oluşturulmuştur. Bu türler aşağıda belirtilmiştir;

- E. acervulina
- E. brunetti
- E. maxima
- E. mitis
- E. necatrix
- E. praecox
- E. tenella

Tavuk veri seti 4402 resim içermektedir. Her bir resimde tek bir hücre bulunmaktadır. Veri seti içerisindeki resimler incelendiğinde hücrelerin yönleri ve büyüklükleri birbirinden farklıdır. Veri seti içerisinde resimler farklı boyutlardadır. Veri seti içerisindeki resimler incelendiğinde en büyük genişliğe sahip dosya 534x432 piksel boyutlu "H_MAX185.jpg", en büyük yüksekliğe sahip dosya ise 447x642 piksel boyutlu "50_MAX612.jpg" dosyasıdır. En küçük genişliğe sahip dosya 177x225 piksel boyutlu "H_ACE477.jpg", en küçük yüksekliğe sahip dosya ise 231x177 piksel boyutlu "H_ACE439.jpg" dosyalarıdır. Her bir sınıfa ait resim sayıları Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1. Tavuk veri setinin sınıflara göre resim sayıları

Sınıf Kısaltması	Eimeria Sınıfı	Resim Sayısı
ACE	E. Acervulina	736
BRU	E. Brunetti	437
MAX	E. Maxima	353
MIT	E. Mitis	813
NEC	E. Necatrix	493
PRA	E. Praecox	887
TEN	E. Tenella	683
	Toplam	4402

Tavşan veri seti on bir farklı **Eimeria** türü için yerel hayvanlardan alınan örnekler kullanılarak oluşturulmuştur. Bu türler aşağıda belirtilmiştir;

- E. coecicola
- E. exigua
- E. flavescens
- E. intestinalis
- E. irresidua
- E. magna

- E. media
- E. perforans
- E. piriformis
- E. stiedai
- E. vejdovskyi

Tavşan veri seti 3237 resim içermektedir. Tavuk veri setine benzer şekilde her bir resimde tek bir hücre bulunmaktadır ve hücrelerin yönleri ve büyüklükleri birbirinden farklıdır. Veri seti içerisindeki resimler incelendiğinde en büyük genişliğe sahip dosya 582x423 piksel boyutlu "IRR128.jpg", en büyük yüksekliğe sahip dosya 390x564 piksel boyutlu "MAG703.jpg" dosyasıdır. En küçük genişliğe sahip dosya 198x216 piksel boyutlu "EXI354.jpg", en küçük yüksekliğe sahip dosya ise 219x198 piksel boyutlu "EXI332.jpg" dosyasıdır. Her bir sınıfa ait resim sayıları Çizelge 3.2'de gösterilmiştir.

Çizelge 3.2. Tavşan veri setinin sınıflara göre resim sayıları

Sınıf Kısaltması	Eimeria Sınıfı	Resim Sayısı
COE	E. coecicola	283
EXI	E. exigua	292
FLA	E. flavescens	521
INT	E. intestinalis	170
IRR	E. irresidua	213
MAG	E. magna	572
MED	E. media	315
PER	E. perforans	210
PIR	E. piriformis	136
STI	E. stiedai	229
VEG	E. vejdovskyi	296
	Toplam	3237

3.4. Geliştirilen Derin Öğrenme Algoritması ile Mikroskobik Görüntü İşleme

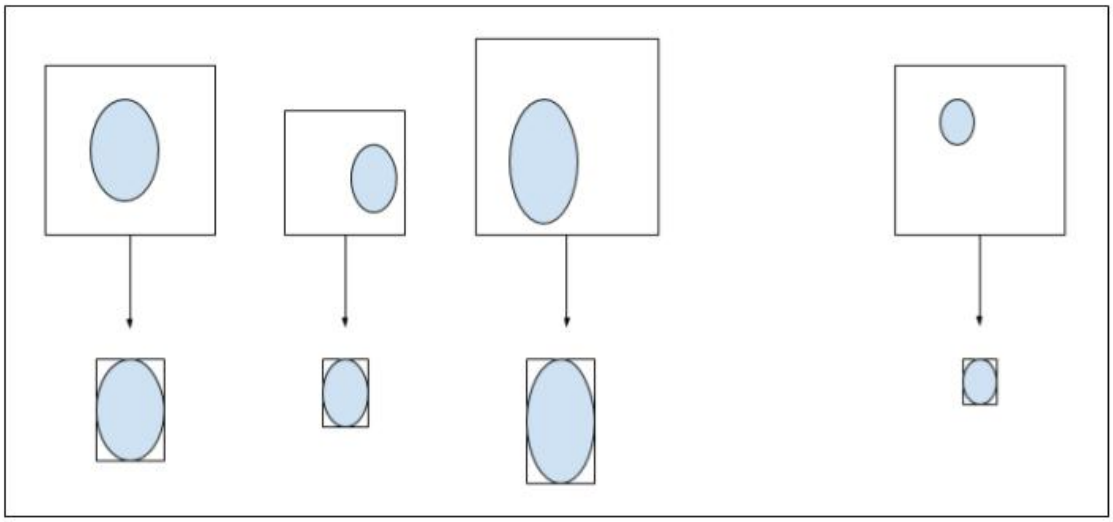
Bu çalışmada yapılan araştırmalar aşağıdaki adımlara bölünmüştür. Her bir adım için Python dili kullanılarak uygulama geliştirilmiştir. Uygulamalara ait açıklamalar ve algoritma alt başlıklarda ayrıntılı olarak verilmiştir. Geliştirilen uygulamalardaki ekran çıktıları, yorum satırları ve değişkenler İngilizce olarak yazılmıştır.

3.4.1. Ön İşlem

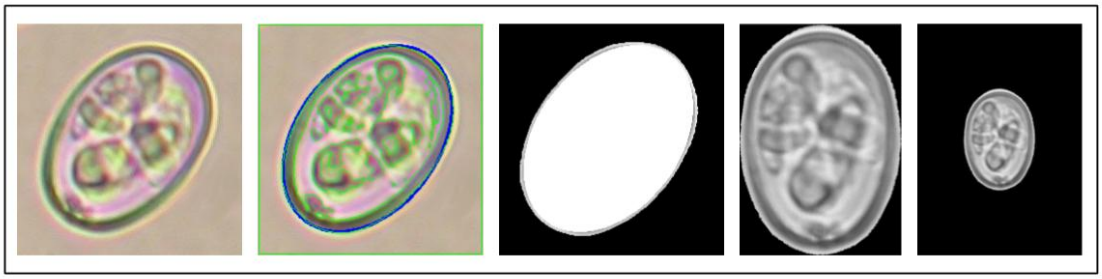
Resim boyutlarının farklı olması sebebiyle bir ön işlem yapılması zorunluluğu bulunmaktadır. Bu sebeple bir uygulama geliştirilmiş ve veri setindeki her bir resim için bazı işlemler yapılmıştır. Ön işlem adımında OpenCV kütüphanesi kullanılmıştır.

Öncelikle farklı boyutlardaki resimler disk üzerinden yüklenmiştir. Her bir resim öncelikle gri tonlamalı hale çevirilmiştir. Bu işlemin ardından hücre çevresi tespit edilerek hücre resimden ayrılmıştır. Tüm hücrelerin yönü yukarıdan aşağıya olacak şekilde transformasyon işlemi yapılmış ve hücreler döndürülmüştür. Bu işlemin ardından her bir hücrenin sığabileceği minimum resim boyutu hesaplanmıştır (Şekil 3.1). Bu boyutlar Tavuk veri seti için 360x504 piksel boyutunda, Tavşan veri seti için 616x644 piksel boyutundadır. Bu işlemin ardından yönleri düzeltilmiş olan hücreler belirtilen büyüklükteki resim içerisine yatay ve dikeyde ortalı olacak şekilde yerleştirilmiştir. İşlemlere ait adımların çıktıları aşağıda gösterilmiştir (Şekil 3.2).

Bu adımda Çizelge 3.3'de verilen **Algoritma 1** yardımıyla Python programlama dili kullanılarak geliştirilen kod çalıştırılmış; tüm veri seti arka planı siyah olarak temizlenmiş, hücre yönelimleri yukarıdan aşağıya doğru hizalanmış ve ortalanmış, gri tonlamalı resim haline dönüştürülmüştür.



Şekil 3.1. Hücrenin sığabileceği minimum resim boyutu bulma.



Şekil 3.2. Resim üzerinde yapılan ön işlem adımları.

Çizelge 3.3. Ön işlem algoritması.

Algoritma 1: (Ön İşlem)
Girdi : Veri seti içerisindeki N adet resim **Çıktı :** Gri tonlamalı N adet resim içeren veri seti
Adım 1. Başla **Adım 2.** Döngü (N tane resim) - Klasör içerisinden resim yükle. - Resmi gri tonlamalı hale dönüştür. - Gri tonlamalı resim üzerindeki alanların çevrelerini tespit et. - Çevresi en büyük olan alanı tespit et. - Resim ile aynı boyutta siyah bir resim oluşturup tespit edilen alanın içerisini beyazla boya. - Oluşturulan yeni resmi, gri tonlamalı resim üzerine maske olarak uygula.

- Hücrenin açısını tespit et.
- Resmi yukarıdan aşağıya gelecek şekilde döndür.
- Döndürülen hücrenin sığabileceği minimum alanı hesapla.

Adım 3. Veri seti içerisindeki tüm resimlerin sığabileceği minimum yükseklik ve genişliği kabul et.

Adım 4. Döngü (N tane resim)

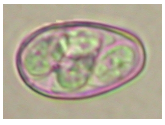
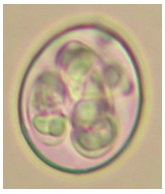
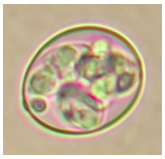
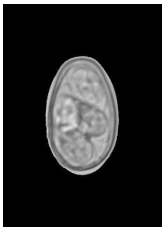
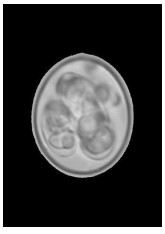
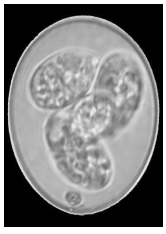
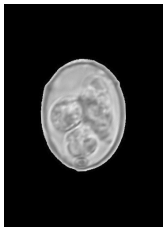
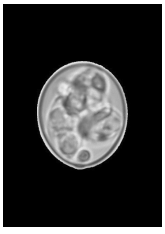
- Minimum yükseklik ve genişlikte bir resim oluşturarak hücreyi ortalı bir şekilde resme yerleştir.
- Resmi veri setindeki orjinal ismi ile yüklenenden farklı bir klasöre kaydet.

Adım 5. Bitir

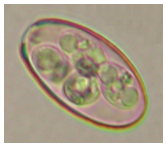
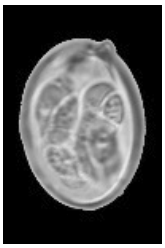
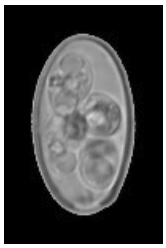
Ön işlem adımı sonunda hücrenin arka planı temizlenmiş, yukarıdan aşağıya hizalanmış ve tüm resimlerin aynı yükseklik ve genişlikte olduğu, gri tonlamalı resimler elde edilmiştir. Bu adım esnasında standart dışı bazı resimler bozulduğu için sonraki adımlarda kullanılmamak üzere elle tespit edilerek veri setinden çıkartılmıştır.

Ön işlem algoritması uygulanan tavuk veri seti için örnek bazı resimler Çizelge 3.4'de ve tavşan veri seti için örnek bazı resimler Çizelge 3.5'de gösterilmiştir.

Çizelge 3.4. Tavuk veri setinin sınıflara göre resim ön işlemesi

Resim	7_ACE123	BRU590	50_MAX576	103_NEC574	H_TEN326
Orijinal					
Düzeltilmiş					

Çizelge 3.5. Tavşan veri setinin sınıflara göre resim ön işlemesi

Resim	COE122	FLA517	IRR135	PER526	VEG153
Orijinal					
Düzeltilmiş					

3.4.2. Eğitim, Doğrulama ve Test

Eğitim (*training*), doğrulama (*validation*) ve test (*test*) adımları algoritması verilerek Python dilinde geliştirilen uygulama yapılmıştır. Eğitim için ön işlem sonucu oluşturulan resimler kullanılmaktadır. Test adımında sonuçların doğruluğunu kontrol edebilmek için "5-fold cross validation" yöntemi kullanılmıştır. Veri setindeki resimler rastgele karıştırılarak, resim veri seti her bir sınıf için 5'e bölünmektedir. Sırası ile her %20 lik grup test için, kalan %20'lik 4 grup ise eğitim için kullanılmıştır (Çizelge 3.6). Bu işlem 100 kere tekrarlanmış ve sonuçların ortalaması elde edilmiştir.

Çizelge 3.6. Resim veri setinin eğitim ve test amaçlı döngü kullanımı

	%20	%20	%20	%20	%20
Döngü 1	Test	Eğitim	Eğitim	Eğitim	Eğitim
Döngü 2	Eğitim	Test	Eğitim	Eğitim	Eğitim
Döngü 3	Eğitim	Eğitim	Test	Eğitim	Eğitim
Döngü 4	Eğitim	Eğitim	Eğitim	Test	Eğitim
Döngü 5	Eğitim	Eğitim	Eğitim	Eğitim	Test

Algoritma 2 ile eğitim, doğrulama ve test aşamaları bulunmaktadır (Çizelge 3.7.). Her bir döngüde test sonucunda çıkan sonuç ve test verisinin orijinal değerleri ve tahmin edilen değerler ekrana yazdırılmaktadır. Ekrana yazdırılan bu veriler "Sonuç Analiz" uygulaması tarafından okunabilir bir formatta ekrana yazdırılmaktadır.

Çizelge 3.7. Eğitim, doğrulama ve test algoritması.

Algoritma 2: (Eğitim, Doğrulama ve Test)

Girdi : Yeni oluşturulan veri seti içerisindeki N adet resim
Çıktı : Doğrulama ve test sonuçları ve model ağırlıkları

Adım 1. Başla
Adım 2. Derin öğrenme algoritması için ağ modelini oluştur.
Adım 3. Modelin başlangıç ağırlıklarını kaydet.
Adım 4. Döngü (100)

- Veri setindeki resimleri rastgele karıştır.
- Veri setini her bir sınıf için eşit 5 parçaya böl.
- Döngü (5)
    - Parçalanan veri setinin $i sıradaki parçasını test olarak seç
    - Geri kalan parçaları eğitim verisi olarak birleştir.
    - Verilerin daha sonra analiz edilebilmesi için kaçıncı turda ve veri setinin kaçıncı sırasında olduğunu ekrana yazdır.
    - Veri setindeki resimlerin pikselini 1 boyutlu diziye çevir (MLP için).
    - Veri setindeki resimlerin piksel değerlerini 0 ile 1 arasında float olarak dönüştür. (piksel/255)
    - Eğitim verilerini kullanarak modeli eğit.
    - Eğitilen model üzerinde test verilerini tahmin et.
    - Test verisinin değerlerini ve tahminleri ekrana yazdır.
    - Her bir tur için doğruluk ve kayıp değerlerini ekrana yazdır.
- Veri setini temizle.
- Modeli başlangıç ağırlıklarına döndür.

Adım 5. Bitir

Python programlama dilinde geliştirilen uygulamanın örnek ekran çıktısı Şekil 3.3'de verilmiştir.

```
Using Theano backend.
1 - PIR
2 - VEG
3 - IRR
4 - COE
5 - STI
6 - EXI
7 - PER
8 - MAG
9 - MED
10 - FLA
>Round=0|Cv=0
#Train Size=2289|Test Size=571
Epoch 1/40
3s - loss: 2.1586 - acc: 0.2512
...
Epoch 40/40
3s - loss: 0.2059 - acc: 0.9209
Generating test predictions...
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  9  0  0  0  0  1  1  1  9  1  1  0  1  1  1  1 10  0  1  1  1  1  1  1  1
  1  1  0  0  1  1  9  2  2  4  9  4  4  4  2  2  2  2  9  9  2  9  2  2  4
  2  9  2  4  2  2  4  4  4  9  4  2  2  2  2  2  2  2  2  5  2  2  2  2  1
  9  2  2  2  2  2  2  2  2  2  2  9  2  2  8  3  3  3  3  8  8  3  3  3  5
  3  3  8  3  8  3  3  3  3  8  8  3  3  3  3  8  8  3  3  8  3  3  3  8  3
  3  3  8  3  3  2  4  5  4 10  5  4  4  2  4  4  4  4  4  4  5  2  5  4  4
  5  4  4  4  4  5  4  4  5  2  2  5  4  4  5  4  9  4  4  4  4  4  2  4  4
  4  4  9  4  5  4  2 10  4  5  5  5  5  5  5  5  5  5  8  8  5  5  5  5  5
  5  2  4  5  5  5  5  5  5  5  5  2  5  2  5  5  8  5  5  5  2  5  5  5  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
  7  7  7  7  0  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  8  8  8  8
  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
 10  8  8  8  8  8  8  8  8  5  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
  8  8  8  8  5  8  8  8  8  8  4  8  8  8  8  8  8  8  8  8  3  8  8  8  8
  8  8  8  8  8  8  8  8  8  5  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9
  7  9  9  9  9  0  9  9  9  9  9  9  9  9 10  9  4  2  9  9  9  9  9  7  9
  2  9  9  9  2  9  9  9  4  9  9  2  9  9  0 10  9  9  2 10  8 10  8 10 10
  8  8 10  5 10 10  9 10  5 10 10 10  5 10 10  5 10 10 10 10 10 10 10  2 10
 10 10 10  5 10 10 10 10 10 10 10 10  1  9 10 10 10 10 10 10 10  8 10 10  5
 10  0 10  0  3  9 10 10 10 10 10 10  5 10 10  9 10 10 10 10 10]
[  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.
   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   1.   2.   2.   2.   2.
   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.
   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.
   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.   2.
   2.   2.   2.   2.   2.   2.   2.   2.   2.   3.   3.   3.   3.   3.   3.
   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.
   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.   3.
   3.   3.   3.   3.   3.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.
   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.
   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.
   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   4.   5.   5.   5.
   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.
   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.   5.
   5.   5.   5.   5.   5.   5.   5.   5.   5.   6.   6.   6.   6.   6.   6.
   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.
   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.
   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.   6.
   6.   6.   6.   6.   6.   6.   6.   7.   7.   7.   7.   7.   7.   7.   7.
   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.
   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.   7.
   7.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.
   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.
   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.
   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.
```

```
   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.
   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.   8.
   8.   8.   8.   8.   8.   8.   8.   8.   8.   9.   9.   9.   9.   9.   9.
   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.
   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.
   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.   9.  10.
  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.
  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.
  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.
  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.  10.
  10.]
----------------------------
//R/0/0/
  29   0   0   0   0   0   0   0   0   1   0
   4  20   0   0   0   0   0   0   0   1   1
   0   1  37   0  10   1   0   0   0   9   0
   0   0   0  28   0   1   0   0  12   0   0
   0   0   7   0  32  10   0   0   0   2   1
   0   0   4   0   2  32   0   0   3   0   1
   0   0   0   0   0   0  58   0   0   0   0
   1   0   0   0   0   0   0  38   0   0   0
   0   0   0   1   1   3   0   0  92   0   1
   2   0   5   0   2   0   0   2   0  37   2
   2   1   1   1   0   7   0   0   5   4  56
----------------------------
//HH/0/0/
{'acc':
[0.25120139828984422, 0.41983398929231808, 0.48623853091227015,
0.52337265116202880, 0.54259501919052699, 0.55220620274908316, 0.57273918637650245,
0.58453473194867256, 0.59196155461331756, 0.59676714766853600, 0.59982525255545716,
0.61074705173897403, 0.61817387437757942, 0.62822193196372933, 0.64394932420246465,
0.65093927614654501, 0.66317169168413126, 0.66754040980036988, 0.68152031553734227,
0.70423765896500956, 0.71690694613454642, 0.71384884252356562, 0.74442988319030334,
0.74355613755679850, 0.76802096897048588, 0.77282656111431836, 0.79467016205927232,
0.80471821834344226, 0.82394058506996048, 0.83792048914039852, 0.85233726611872751,
0.85583224117938173, 0.87330712158641421, 0.88772389580454558, 0.88903451327831795,
0.89340323415475420, 0.90956749747860843, 0.91087811346812353, 0.91961555136713513,
0.92092616975229347],
'loss': [2.1585833996130734, 1.6135653257369995, 1.38415737738511410,
1.25697394493314670, 1.18738451041525340, 1.14238465129157340, 1.08101125875900480,
1.08164438361838270, 1.04661731664766760, 1.02830203629414970, 1.00827013425506220,
0.99220429606289773, 0.96510566319894353, 0.93726421136843585, 0.90258975185750090,
0.89744710430092767, 0.86652757484561860, 0.84790626375265854, 0.80261158086473428,
0.76382530292812634, 0.75983135402853597, 0.72279922901776850, 0.67770571855021755,
0.65980489897696715, 0.61158725982568518, 0.59804961331393736, 0.56819483852636454,
0.53168223210431020, 0.48547146574489514, 0.42557381834979657, 0.41725984123295806,
0.38259094220522244, 0.35372575489961200, 0.31465169361008466, 0.31041827840855052,
0.30614564394315502, 0.26228090148140842, 0.24601270083310356, 0.22003218585640752,
0.20585755051458282]}
----------------------------
Test loss: 0.62394905761
Test accuracy: 0.803852889563
############################
Cleaned ....
```

Şekil 3.3. 1.Tur ve 1. Cross Validation adımı için Python programlama dilinde geliştirilen uygulamanın ekran çıktısı.

1. Tur ve 1. Cross Validation adımı için parametrelerin örnek çıktı açıklamaları aşağıda verilmiştir;

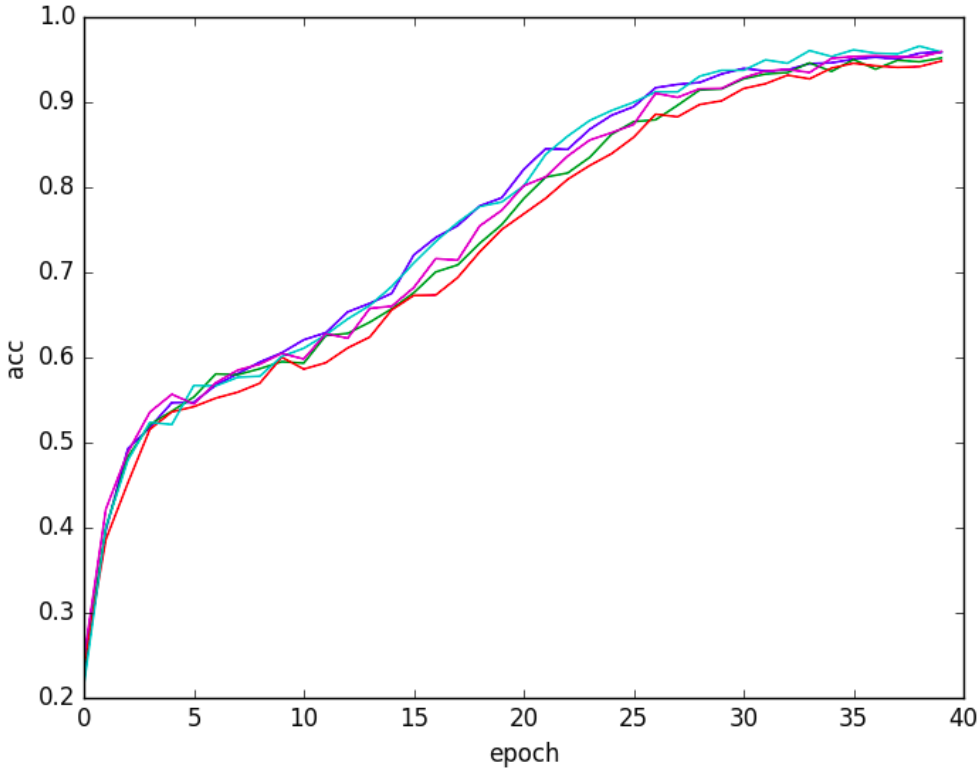
- **>Round=0|Cv=0** : Her bir cross validation (**Cv**) adımı için yazdırılmaktadır. Eğitim ve test aşamasının kaçıncı turda (**Round**) ve cross validation (**Cv**) un kaçıncı sırasında olduğu gösterir.
- **#Train Size=2289|Test Size=571** : Cross validation adımı için kaç resmin eğitim, kaç resmin test için kullanılacağını gösterir.
- **Generating test predictions… :** Bu mesajın hemen ardından gelen 2 dizinin ilki test veri setinin değerleri, ikinci dizi ise test veri setinin tahmin edilen değerleridir.
- **//R/0/0/ :** Her bir cross validation adımı için doğruluk tablosu yazdırılmaktadır. Satırlar orjinal değerleri, sütunlar ise tahmin edilen değerleri göstermektedir. **//R/** sonrasındaki ilk değer kaçıncı turda olduğunu, ikinci değer ise cross validationun kaçıncı adımında olduğunu göstermektedir.
- **//HH/0/0/ :** Her bir cross validation adımı için eğitim aşamasındaki doğruluk ve kayıp geçmişini yazdırmaktadır. Her bir epoch için gerçekleşen doğruluk ve kayıp dizi şeklinde yazdırılmaktadır.
- **Test loss ve Test accuracy:** Her bir cross validation adımı için test sonucundaki doğruluk (**accuracy**) ve kayıp (**loss**) yazdırılmaktadır.

3.4.3. Sonuç Analizi

Sonuç analiz uygulaması Python dili kullanılarak geliştirilmiştir. Uygulama parametre olarak test sonuçlarının yazılı olduğu dosya adını otomatik olarak almaktadır. Ardından dosyayı yükleyerek içerisindeki sonuçları ayrıştırmakta ve analiz etmektedir. Dosya analiz edildikten sonra 4 farklı işlem yapılabilir.

İşlem 1: Seçilen tur için doğrulama (*validation*) grafiğini çiz.
100 tur yapılan cross validation işleminin seçilen turuna ait eğitim aşamasının geri besleme adımları için, doğrulama (*validation*) sonuçları grafik olarak çizdirilmektedir.

Çizilen grafikte her bir cross validation adımı farklı renk olarak gösterilmektedir. Çizdirilen grafiklere ait bir örnek Şekil 3.4'de verilmiştir.



Şekil 3.4. Uygulama tarafından seçilen tur için çizdirilen doğrulama grafiği

İşlem 2: Seçilen tur için özet ekrana yazdır.

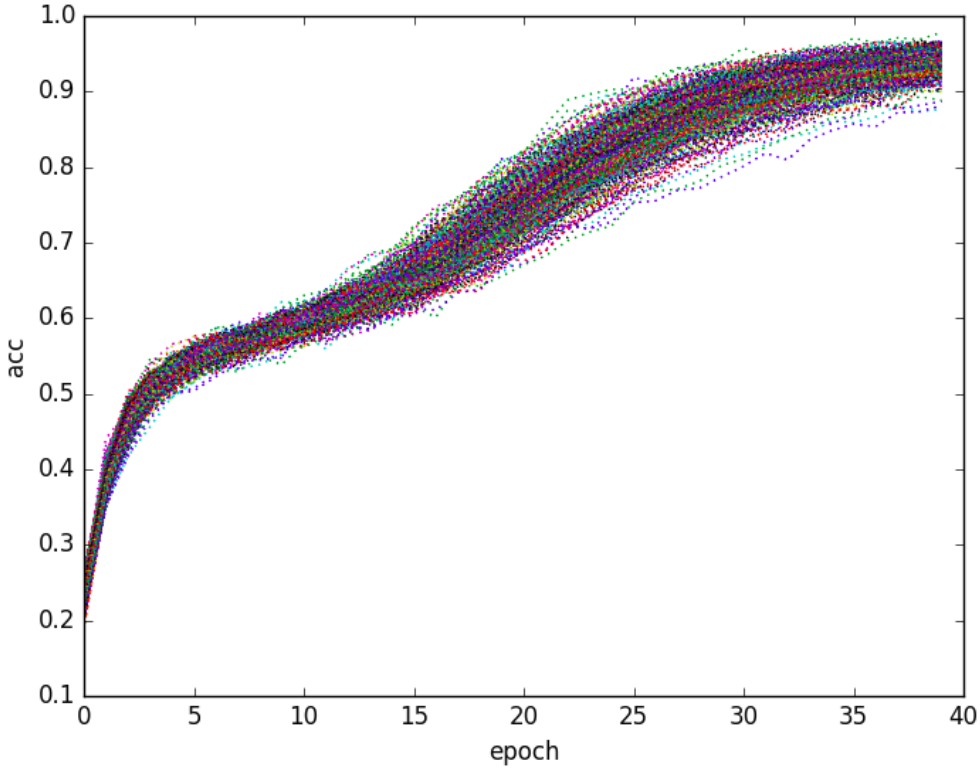
100 tur yapılan cross validation işleminin seçilen turuna ait sonuçların test sonuçlarının özet bilgisi ekrana yazdırılmaktadır. Yazdırılan örnek özet ekranı Şekil 3.5'te verilmiştir.

```
Choose round: 1
Total Acc     : 0.784265734266
Standart Dev : 0.00607749762732
Min Fold Acc : 0.76707530648
Max Fold Acc : 0.784265734266
```

Şekil 3.5. Uygulama tarafından seçilen tur için yazdırılan özet ekranı

İşlem 3: Tüm turlar için doğrulama grafiğini çiz.

100 tur yapılan cross validation işleminin tamamına ait geri besleme adımlarındaki doğrulama sonuçları grafik olarak çizdirilmektedir. Çizdirilen örnek doğrulama grafiği Şekil 3.6'da verilmiştir.



Şekil 3.6. Uygulama tarafından seçilen tur için çizdirilen doğrulama grafiği

İşlem 4: Tüm analizi yazdır.

Tüm öğrenme ve test adımlarına ait sonuçlar toplu olarak ekrana yazdırılmaktadır. Ortalama doğruluk, maksimum ve minimum doğruluk sonuçları ile her bir sınıfa ait ortalama doğruluk oranları yazdırılmaktadır. Yazdırılan örnek bir analiz ekran çıktısı Şekil 3.7'de verilmiştir.

```
Total Acc : 78.4153846154
Round Based Standart Dev : 0.527355264264
Min Round Acc : 77.2027972028
Max Round Acc : 79.7202797203
Min Fold Acc : 73.5551663748
Max Fold Acc : 83.2460732984
```

```
Class 0 Acc - 78.0821917808
Class 1 Acc - 72.8421052632
Class 2 Acc - 66.6040955631
Class 3 Acc - 86.1682692308
Class 4 Acc - 52.8538461538
Class 5 Acc - 57.7475247525
Class 6 Acc - 99.8938356164
Class 7 Acc - 99.3045685279
Class 8 Acc - 90.0691056911
Class 9 Acc - 70.2817460317
Class 10 Acc - 76.825974026

 78.08  7.82  1.93   0.1  1.23   0.0  0.05  2.26   0.0  6.75  1.77
 11.75 72.84  4.74   0.0  0.11  0.12   0.0  0.01  0.98  2.01  7.44
  0.96  1.25  66.6  0.16  16.4  3.38   0.0   0.0  0.21   9.2  1.84
   0.0   0.0  0.45 86.17  0.25  0.07   0.0   0.0 12.82   0.0  0.23
  0.55   0.1 23.08  1.18 52.85 13.12   0.0   0.0  0.93  6.67  1.53
  0.02  0.06  6.93  0.91  16.1 57.75   0.0   0.0  6.64  0.68 10.91
   0.0   0.0   0.0   0.0   0.0   0.0 99.89  0.11   0.0   0.0   0.0
   0.0   0.0   0.0   0.0   0.0   0.0  0.04  99.3   0.0  0.65   0.0
   0.0  0.02   0.1  3.97  0.25  0.97   0.0   0.0 90.07   0.0  4.63
  2.62  0.65   8.6   0.0  4.01  1.27   0.0  4.88  0.02 70.28  7.67
  1.32  2.18  1.83  0.14  0.47  4.25   0.0  0.02  8.76  4.22 76.83
---
 11400  1142   282    15   180     0     7   330     0   985   259
  1563  9688   630     0    14    16     0     1   131   267   990
   282   366 19515    47  4804   990     0     1    61  2695   539
     0     1    94 17923    52    15     0     0  2667     0    48
   143    27  6000   306 13742  3410     0     0   242  1733   397
     5    12  1399   183  3253 11665     0     0  1342   138  2203
     0     0     0     0     0     0 29169    31     0     0     0
     0     0     0     0     0     0     8 19563     0   129     0
     0     9    48  1954   123   475     0     0 44314     1  2276
   659   164  2168     0  1010   320     0  1230     5 17711  1933
   509   838   705    52   180  1637     0     7  3371  1623 29578
---
  3.14  2.13  1.01  0.26  0.65   0.0  0.17  0.99   0.0  1.95  0.87
  2.43  3.47  1.08   0.0  0.28  0.29   0.0  0.07  0.58  0.92   1.9
  0.46  0.52  3.51  0.17  3.03  1.02   0.0  0.03  0.26  1.33  0.71
   0.0  0.05  0.25  2.35  0.31  0.18   0.0   0.0  2.31   0.0  0.31
```

```
 0.37  0.19   3.9  0.28  4.08  2.16   0.0   0.0   0.5  1.24  0.59
 0.11  0.16  1.78  0.48  2.53  3.78   0.0   0.0  1.21   0.6  1.91
  0.0   0.0   0.0   0.0   0.0   0.0  0.22  0.22   0.0   0.0   0.0
  0.0   0.0   0.0   0.0   0.0   0.0   0.2  0.76   0.0  0.71   0.0
  0.0  0.06  0.14  0.83   0.2  0.44   0.0   0.0  1.47  0.02  1.06
  1.0  0.48  1.99   0.0  1.42  0.59   0.0  0.72  0.09   2.7  1.41
 0.56  0.61  0.55  0.19  0.33   1.0   0.0  0.07  1.54  0.85  2.46
```

Şekil 3.7. Uygulama tarafından tüm analiz için yazdırılan özet ekranı

Bu adımın çıktısında öncelikle doğruluk sonuçları ve standart sapması yazdırılmaktadır. Ardından minimum ve maksimum sonuç veren turlar ve arkasından minimum ve maksimum sonuç veren *cross validation* turları yazılmaktadır. Ardından her bir sınıf için hesaplanan ortalama doğruluk değerleri yazdırılmaktadır. Hemen sonrasında üç farklı matris yazdırılmaktadır. İlk matris yüzde olarak ortalama doğruluk değerlerini göstermektedir. ikinci matriste 100 tur sonucunda değer olarak tahmin ve doğruluk sonuçları gösterilmektedir. üçüncü matriste ise turların sonuçları arasındaki standart sapmalar gösterilmiştir.

3.5. Modelin Çalıştırılmasında Kullanılan Donanım

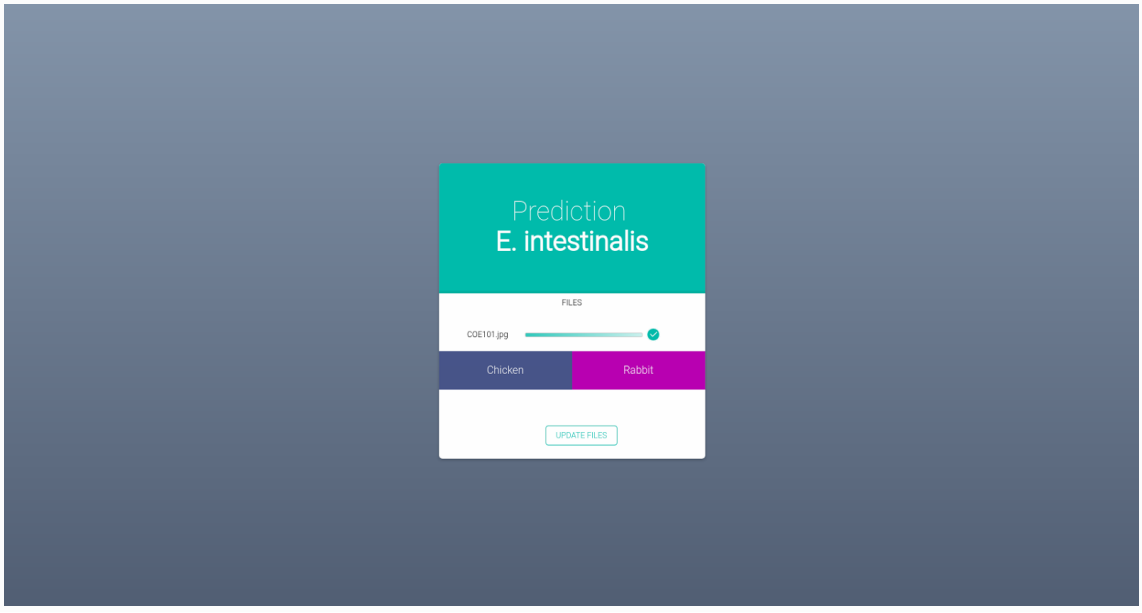
Derin öğrenme algoritmaları yüksek hesaplama gücü gerektirmektedir. Tezde yapılan çalışmalar için eğitim, doğrulama ve test adımları Harran Üniversitesi Yüksek Başarımlı Hesaplama Merkezi'nde yapılmıştır. Bu merkezde 4 adet GPU düğümü, 20 adet CPU düğümü araştırmacıların kullanımına sunulmaktadır. Tezde gerçekleştirilen çalışmalar GPU düğümleri üzerinde yapılmıştır. 1 GPU makinesine ait donanım bilgileri Çizelge 3.8 gösterilmiştir.

Çizelge 3.8. GPU düğümü donanım özellikleri

İşlemci	2 x Intel Xeon E5-2670 v3
Bellek	128 GB ECC DDR4 (8 x 16GB 2133 mHz DDR4)
Disk	2 x 400GB SSD 2,5″ SAS
GPU	2 x Nvidia Tesla K20x GPGPU
Ağ Bağlantısı	1 x Intel QDR Single-Port HCA

3.6. Geliştirilen Derin Öğrenme Algoritmasının Uygulama Web Sitesi

Tavuk ve tavşan veri setleri için eğitim sonucunda en başarılı tura ait model ağırlıkları HDF5 dosyası olarak kaydedilmiştir. Sonuçların test edilebilmesi için bir web sitesi geliştirilmiştir. Kullanıcılar siteye ön işleme tabi tutulmamış bir resmi yükleyerek hangi sınıfa ait olduğunu test edebilirler (Şekil 3.8). Web sitesinin linkine: *https://www.konya.edu.tr/acdal/projects/eimeria* adresinden ulaşılabilir.



Şekil 3.8. Hastalık tespiti için resim yüklemesi yapılan ve sonucun verildiği test web sitesi görüntüsü

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Geliştirilen algoritma adımları, tavuk ve tavşan veri setleri için ayrı ayrı uygulanmış ve sonuçları elde edilmiştir.

Çalışmalar yapılırken veri setindeki hücrelerin sığabileceği minimum yükseklik ve genişlik hesaplanmış ve herhangi bir küçültme işlemi yapılmadan tüm resimler aynı yükseklik ve genişliğe ayarlanmıştır. Bu resimler üzerinde MLP algoritması kullanılarak çalışma yapılmış ve sonuçları bu bölümde paylaşılmıştır. Aynı resimler üzerinde CNN algoritması da çalıştırılmak istenmiştir. Fakat CNN algoritmasının fazla kaynak ihtiyacı duyması ve resim boyutları yüksek olduğu için modelleme problemleri oluşmuştur. Bu yüzden veri seti içerisindeki resimler orantılı bir şekilde küçültülerek yeni bir veri seti oluşturulmuştur.

Yeni oluşturulan küçük resimli veri seti üzerinde MLP ve CNN algoritmaları ayrı ayrı çalıştırılmış ve sonuçlar bu bölümde paylaşılmıştır.

4.1. Ön İşlem Sonrası Elde Edilen Veri Setleri

4.1.1. Tavuk Veri Seti

Tavuk veri seti üzerinde ön işlem uygulaması çalıştırıldıktan sonra standart dışı bazı resimlerin bozulduğu tespit edilmiştir. Bu şekilde bozulan resimler elle tespit edilerek veri setinden çıkartılmıştır.

Sonuçların tespitinde kullanılan resimlerin sınıflara göre dağılımı Çizelge 4.1'de gösterilmiştir. Ön işlem sonrası tüm resimler **360x504 piksel** boyutuna getirilmiştir. Küçültülmüş resimlerin boyutları ise 100x140 pikseldir.

Her iki boyut için yapılan çalışmalar ve sonuçları alt başlıklarda listelenmiştir.

Çizelge 4.1. Tavuk veri seti sınıflara göre resim sayıları

Sınıf Kısaltması	Eimeria Sınıfı	Resim Sayısı	K. Resim Sayısı
ACE	E. Acervulina	736	731
BRU	E. Brunetti	437	437
MAX	E. Maxima	353	353
MIT	E. Mitis	813	809
NEC	E. Necatrix	493	488
PRA	E. Praecox	887	885
TEN	E. Tenella	683	677
	Toplam	4402	4380

4.1.2. Tavşan Veri Seti

Tavşan veri seti üzerinde ön işlem uygulaması çalıştırıldıktan sonra standart dışı bazı resimlerin bozulduğu fark edilmiştir. Bu şekilde bozulan resimler elle tespit edilerek veri setinden çıkartılmıştır. Sonuçların tespitinde kullanılan resimlerin sınıflara göre dağılımı Çizelge 4.2'de gösterilmiştir. Ön işlem sonrası tüm resimler **616x644 piksel** boyutuna getirilmiştir. Küçültülmüş resimlerin boyutları ise 100x150 pikseldir. Her iki boyut için yapılan çalışmalar ve sonuçları alt başlıklarda listelenmiştir.

Çizelge 4.2. Tavşan veri seti sınıflara göre kullanılabilen resim sayıları

Sınıf Kısaltması	Eimeria Sınıfı	Resim Sayısı	K. Resim Sayısı
COE	E. coecicola	283	260
EXI	E. exigua	292	292
FLA	E. flavescens	521	385
INT	E. intestinalis	170	146
IRR	E. irresidua	213	208
MAG	E. magna	572	492
MED	E. media	315	252
PER	E. perforans	210	197
PIR	E. piriformis	136	133
STI	E. stiedai	229	202
VEG	E. vejdovskyi	296	293
	Toplam	3,237	2,860

4.2. Çok Katmanlı Yapay Sinir Ağı (MLP)

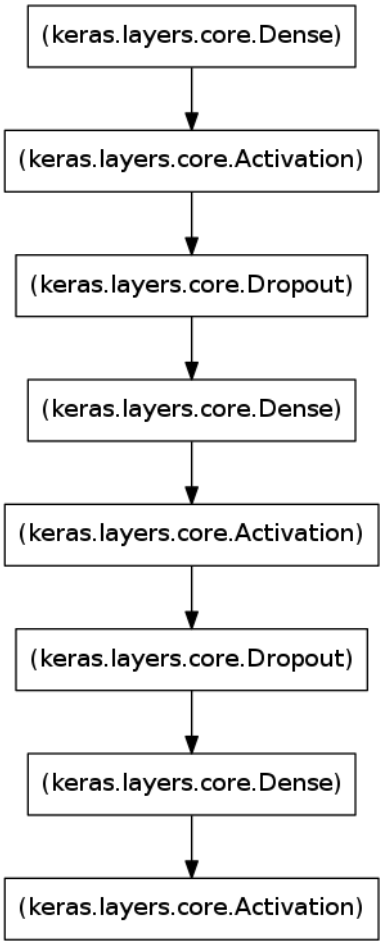
4.2.1. Tavuk Veri Seti 360x504 Piksel Boyutlu Resimler

MLP algoritması kullanılarak orijinal boyutlu resimler üzerinde yapılan çalışmaların ilk adımında 360x504 piksel boyutlu resimlerin piksel değerleri NumPy ile tek boyutlu diziye dönüştürülmüştür. Bu işlemin ardından NumPy ile 181440 elemanlı dizi elde edilmiştir. Keras kullanılarak oluşturulan modelde katmanlara göre yapılan işlemler şunlardır:

- **Giriş katmanı:** 181440 eleman ile bire bir bağlı 256 nöron bulunmaktadır. İkinci katman ile arasında aktivasyon fonksiyonu olarak "ReLU" ve aşırı uyumsuzluk problemini önlemek için 0.25 kaçınma değeri kullanılmıştır.
- **İkinci katman:** 256 nöron bulunmaktadır. Çıkış katmanı ile ikinci gizli katman arasında aktivasyon fonksiyonu olarak "ReLU" ve 0.25 kaçınma değeri kullanılmıştır.
- **Çıkış katmanı:** sınıf sayısına eşit şekilde 7 nöron bulundurmaktadır. Bu katmanda aktivasyon fonksiyonu olarak "Softmax" kullanılmıştır.

Öğrenme algoritması olarak Stokastik Eğim Azalması (SGD) seçilmiştir. Algoritma üzerinde öğrenme oranı olarak başlangıç değeri 0.1 olmasına rağmen 0.05 tercih edilmiştir. Bu şekilde öğrenme geç olmakta fakat algoritmanın minimum'u kaçırma ihtimali azalmaktadır.

Öğrenme esnasında geri besleme işlemi (*epoch*) 150 kez tekrarlanmıştır. Her bir geri besleme yaklaşık 4 dakika sürmektedir. Oluşturulan model Şekil 4.1'de verilmiştir.



Şekil 4.1. Tavuk veri seti, 360x504 piksel boyutlu resimler için elde edilen model

MLP algoritması kullanılarak 100 tur yapılan çalışmada toplamda % 83.75 başarım elde edilmiştir. En yüksek başarım % 99.08 ile E. Maxima türünde, en düşük başarım ise % 58.18 ile E. Necatrix türünde gerçekleşmiştir (Çizelge 4.3).

Sonuçlara ait doğruluk karşılaştırma tablosu Çizelge 4.4'de gösterilmiştir.

Çizelge 4.3. Tavuk veri seti, 360x504 piksel boyutlu resimler için MLP sonuçları

Ortalama Doğruluk	% 83.75
Standart Sapma	% 0.60
Minimum Doğruluk	% 81.83
Maksimum Doğruluk	% 85.05
BRU	% 90.70
PRA	% 78.65
TEN	% 72.66
MAX	% 99.08
ACE	% 95.14
NEC	% 58.18
MIT	% 93.30

Çizelge 4.4. Tavuk veri seti, 360x504 piksel boyutlu resimler için MLP doğruluk tablosu

		BRU	**PRA**	**TEN**	**MAX**	**ACE**	**NEC**	**MIT**
BRU	O.%	**90.70**	5.59	3.43	0.28	0.00	0.00	0.00
	Std.	**1.8**	1.77	0.64	0.29	0.0	0.0	0.0
PRA	O.%	3.49	**78.65**	6.82	0.00	0.28	4.63	6.13
	Std.	0.82	**3.11**	2.1	0.0	0.11	1.51	0.5
TEN	O. %	3.51	7.76	**72.66**	0.00	0.32	14.85	0.90
	Std.	0.5	2.52	**4.97**	0.0	0.08	3.58	0.19
MAX	O. %	0.91	0.01	0.00	**99.08**	0.00	0.00	0.0
	Std.	0.34	0.04	0.0	**0.34**	0.0	0.0	0.0
ACE	O. %	0.00	0.19	0.21	0.00	**95.14**	3.58	0.88
	Std.	0.0	0.18	0.23	0.0	**0.66**	0.64	0.26
NEC	O. %	0.34	10.97	19.80	0.00	7.18	**58.18**	3.53
	Std.	0.16	2.98	4.78	0.0	1.04	**6.16**	0.85
MIT	O. %	0.00	4.58	0.11	0.00	0.27	1.73	**93.30**
	Std.	0.0	0.86	0.14	0.0	0.17	0.55	**0.81**

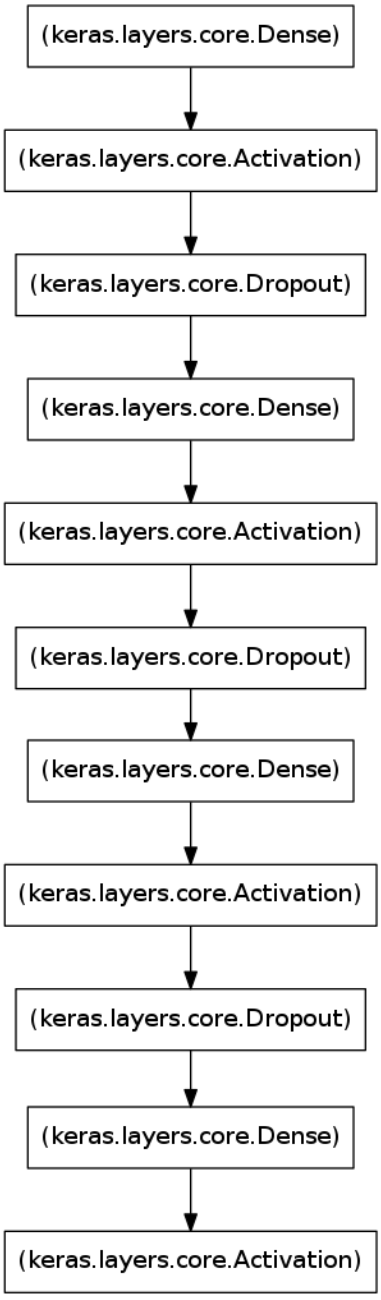
4.2.2. Tavuk Veri Seti 100x140 Piksel Boyutlu Resimler

MLP algoritması kullanılarak küçültülmüş boyutlu resimler üzerinde yapılan çalışmaların ilk adımında 100x140 piksel boyutlu resimlerin piksel değerleri NumPy ile tek boyutlu diziye dönüştürülmüştür. Bu işlemin ardından NumPy ile 14000 elemanlı dizi elde edilmiştir. Bu dizi, orijinal boyutlu resimlerde yapılan çalışmanın 12 de birine denk gelmektedir. Bu sayede eğitim aşaması çok daha hızlı gerçekleşmiştir. Keras kullanılarak oluşturulan modelde katmanlara göre yapılan işlemler şunlardır:

- **Giriş katmanı:** 14000 eleman ile bire bir bağlı 64 nöron bulunmaktadır. İkinci katman ile arasında aktivasyon fonksiyonu olarak "ReLU" ve aşırı uyumsuzluk problemini önlemek için 0.25 kaçınma değeri kullanılmıştır.
- **İkinci katman:** 128 nöron bulunmaktadır. İkinci ve üçüncü katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.25 kullanılmıştır.
- **Üçüncü katman:** 128 nöron bulunan bulunmaktadır. Çıkış katmanı ile üçüncü katman arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.25 kullanılmıştır.
- **Çıkış katmanı:** sınıf sayısına eşit şekilde 7 nöron bulundurmaktadır. Bu katmanda aktivasyon fonksiyonu olarak "Softmax" kullanılmıştır.

Öğrenme algoritması olarak Stokastik Eğim Azalması (SGD) seçilmiştir. Algoritma üzerinde öğrenme oranı olarak başlangıç değeri 0.1 olmasına rağmen 0.01 tercih edilmiştir. Bu şekilde öğrenme geç olmakta fakat algoritmanın minimum'u kaçırma ihtimali azalmaktadır.

Öğrenme esnasında geri besleme işlemi (*epoch*) 100 kez tekrarlanmıştır. Her bir geri besleme yaklaşık 1 sn sürmektedir. Oluşturulan model Şekil 4.2'de verilmiştir.



Şekil 4.2. Tavuk veri seti, 100x140 piksel boyutlu resimler için kullanılan model

MLP algoritması kullanılarak 100 tur yapılan çalışmada toplamda %82.81 başarım elde edilmiştir. En yüksek başarım %99.46 ile E. Maxima türünde, en düşük başarım ise %55.06 ile E. Necatrix türünde gerçekleşmiştir (Çizelge 4.5). Sonuçlara ait doğruluk karşılaştırma tablosu Çizelge 4.6'da gösterilmiştir.

Çizelge 4.5. Tavuk veri seti 100x140 piksel boyutlu resimler için MLP sonuçları

Ortalama Doğruluk	%82.81
Standart Sapma	% 0.81
Minimum Doğruluk	%81.23
Maksimum Doğruluk	%83.72
BRU	%89.22
PRA	%77.75
TEN	%70.75
MAX	%99.46
ACE	%95.37
NEC	%55.06
MIT	%93.37

Çizelge 4.6. Tavuk veri seti 100x140 piksel boyutlu resimler için MLP doğruluk tablosu

		BRU	**PRA**	**TEN**	**MAX**	**ACE**	**NEC**	**MIT**
BRU	O. %	**89.22**	7.09	3.02	0.66	0.00	0.00	0.00
	Std.	**3.13**	2.78	0.62	0.44	0.0	0.0	0.0
PRA	O. %	3.59	**77.75**	7.19	0.00	0.26	5.06	6.15
	Std. %	1.16	**3.30**	2.07	0.0	0.05	1.48	0.5
TEN	O. %	3.69	9.93	**70.43**	0.00	0.34	14.49	1.12
	Std.	0.54	3.52	**5.69**	0.0	0.06	3.55	0.16
MAX	O. %	0.54	0.00	0.00	**99.46**	0.00	0.00	0.0
	Std.	0.27	0.0	0.0	**0.27**	0.0	0.0	0.0
ACE	O. %	0.00	0.21	0.11	0.00	**95.38**	3.41	0.90
	Std.	0.0	0.27	0.21	0.0	**0.64**	0.54	0.26
NEC	O. %	0.43	10.97	19.80	0.00	7.93	**55.06**	3.63
	Std.	0.19	3.19	5.19	0.0	1.14	**6.96**	0.83
MIT	O. %	0.00	4.56	0.021	0.00	0.31	1.73	**93.37**
	Std.	0.0	0.66	0.05	0.0	0.19	0.57	**0.66**

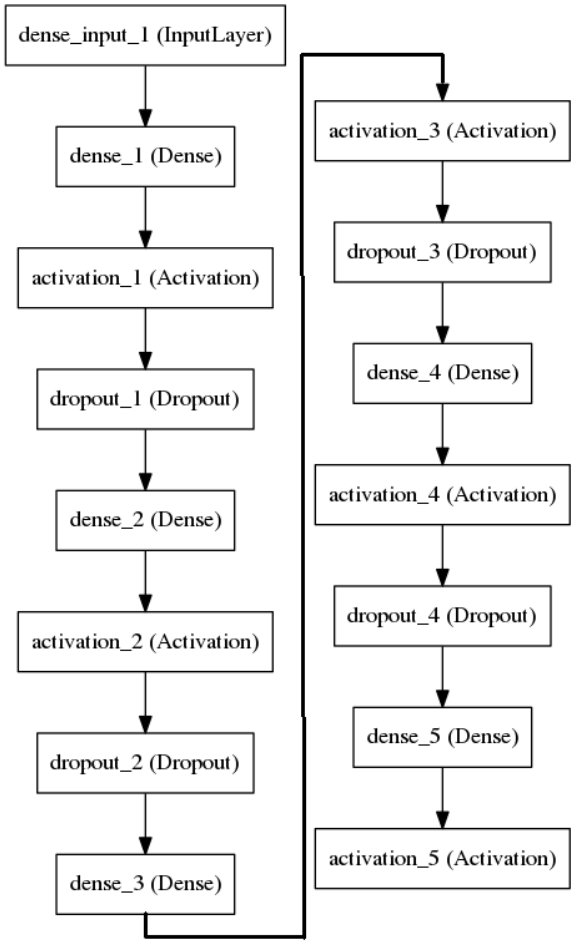
4.2.3. Tavşan Veri Seti 100x150 Piksel Boyutlu Resimler

MLP algoritması kullanılarak küçültülmüş resimler üzerinde yapılan çalışmaların ilk adımında 100x150 piksel boyutlu resimlerin piksel değerleri NumPy ile tek boyutlu diziye dönüştürülmüştür. Bu işlemin ardından NumPy ile 15000 elemanlı dizi elde edilmiştir. Keras kullanılarak oluşturulan modelde katmanlara göre yapılan işlemler şunlardır:

- **Giriş katmanı:** 15000 eleman ile bire bir bağlı 128 nöron bulunmaktadır. İkinci katman ile arasında aktivasyon fonksiyonu olarak "ReLU" ve aşırı uyumsuzluk problemini önlemek için 0.25 kaçınma değeri kullanılmıştır.
- **İkinci katman:** 256 nöron bulunmaktadır. İkinci ve üçüncü gizli katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.25 kullanılmıştır.
- **Üçüncü katman:** 128 nöron bulunmaktadır. Çıkış katmanı ile üçüncü gizli katman arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.25 kullanılmıştır.
- **Çıkış katmanı:** sınıf sayısına eşit şekilde 11 nöron bulundurmaktadır. Bu katmanda aktivasyon fonksiyonu olarak "Softmax" kullanılmıştır.

Öğrenme algoritması olarak Stokastik Eğim Azalması (SGD) seçilmiştir. Algoritma üzerinde öğrenme oranı olarak başlangıç değeri 0.1 olmasına rağmen 0.01 tercih edilmiştir. Bu şekilde öğrenme geç olmakta fakat algoritmanın minimum'u kaçırma ihtimali azalmaktadır.

Öğrenme esnasında geri besleme işlemi (*epoch*) 100 kez tekrarlanmıştır. Her bir geri besleme yaklaşık 2 sn sürmektedir. Oluşturulan model Şekil 4.3'de verilmiştir.



Şekil 4.3. Tavşan veri seti 100x150 piksel boyutlu resimler için kullanılan model

MLP algoritması kullanılarak 100 tur yapılan çalışmada toplamda %58.99 başarım elde edilmiştir. En yüksek başarım %99.97 ile E. Perforans türünde, en düşük başarım ise %2.67 ile E. Magna türünde gerçekleşmiştir (Çizelge 4.7). Sonuçlara ait doğruluk karşılaştırma tablosu Çizelge 4.8'de gösterilmiştir.

Çizelge 4.7. Tavşan veri seti 100x150 piksel boyutlu resimler için MLP sonuçları

Ortalama Doğruluk	% 58.99
Standart Sapma	% 0.70
Minimum Doğruluk	% 57.27
Maksimum Doğruluk	% 60.52
INT	% 86.41
PIR	% 64.40
VEG	% 44.19
IRR	% 56.54
COE	% 95.58
STI	% 19.94
EXI	% 42.60
PER	% 99.97
MAG	% 2.67
MED	% 38.96
FLA	% 65.02

Çizelge 4.8. Tavşan veri seti 100x150 piksel boyutlu resimler için MLP doğruluk tablosu

		INT	**PIR**	**VEG**	**IRR**	**COE**	**STI**	**EXI**	**PER**	**MAG**	**MED**	**FLA**
INT	O. %	**86.42**	0.67	0.0	0.0	0.0	0.53	0.04	0.0	0.32	6.66	5.35
	Std.	**4.3**	0.41	0.0	0.02	0.0	0.37	0.13	0.0	0.33	2.45	2.5
PIR	O. %	1.05	**64.4**	10.7	4.87	0.06	10.9	3.3	0.01	0.68	4.01	0.03
	Std.	0.39	**6.57**	2.71	1.65	0.13	5.49	1.33	0.1	1.07	1.59	0.09
VEG	O. %	0.31	15.1	**44.19**	18.01	7.07	4.48	7.08	0.0	0.04	3.71	0.0
	Std.	0.29	3.76	**5.35**	4.5	1.62	1.71	2.81	0.04	0.14	1.36	0.0
IRR	O. %	0.01	2.53	11.24	**56.54**	4.04	0.35	22.97	0.0	0.0	2.32	0.0
	Std.	0.07	1.77	4.47	**8.89**	1.59	0.47	7.98	0.0	0.0	2.31	0.0
COE	O. %	0.0	0.0	3.11	0.89	**95.58**	0.0	0.05	0.37	0.0	0.0	0.0
	Std.	0.0	0.0	1.35	0.63	**1.62**	0.0	0.18	0.47	0.0	0.0	0.0
STI	O. %	3.98	59.42	9.88	0.95	0.01	**19.95**	0.62	0.01	1.63	2.75	0.81
	Std.	1.41	8.16	2.57	0.58	0.05	**7.3**	0.46	0.08	1.55	1.06	0.47
EXI	O. %	2.53	1.92	4.53	23.69	0.28	0.02	**42.61**	0.0	0.07	24.35	0.0
	Std.	0.74	1.51	2.84	6.94	0.54	0.15	**9.38**	0.0	0.24	7.41	0.0
PER	O. %	0.0	0.0	0.0	0.0	0.02	0.0	0.0	**99.98**	0.0	0.0	0.0
	Std.	0.0	0.0	0.0	0.0	0.12	0.0	0.0	**0.12**	0.0	0.0	0.0
MAG	O. %	17.42	46.46	0.57	0.12	0.01	12.82	2.0	0.0	**2.67**	17.33	0.59
	Std.	3.61	6.72	0.56	0.41	0.11	5.36	1.22	0.05	**2.78**	3.06	0.63
MED	O. %	31.02	6.24	1.22	5.16	0.14	0.73	15.42	0.01	0.67	**38.96**	0.42
	Std.	5.14	1.52	0.76	1.71	0.23	0.52	4.26	0.05	0.57	**5.12**	0.31
FLA	O. %	34.05	0.23	0.0	0.0	0.0	0.22	0.0	0.0	0.15	0.33	**65.02**
	Std.	7.46	0.33	0.0	0.0	0.0	0.29	0.05	0.0	0.28	0.35	**7.29**

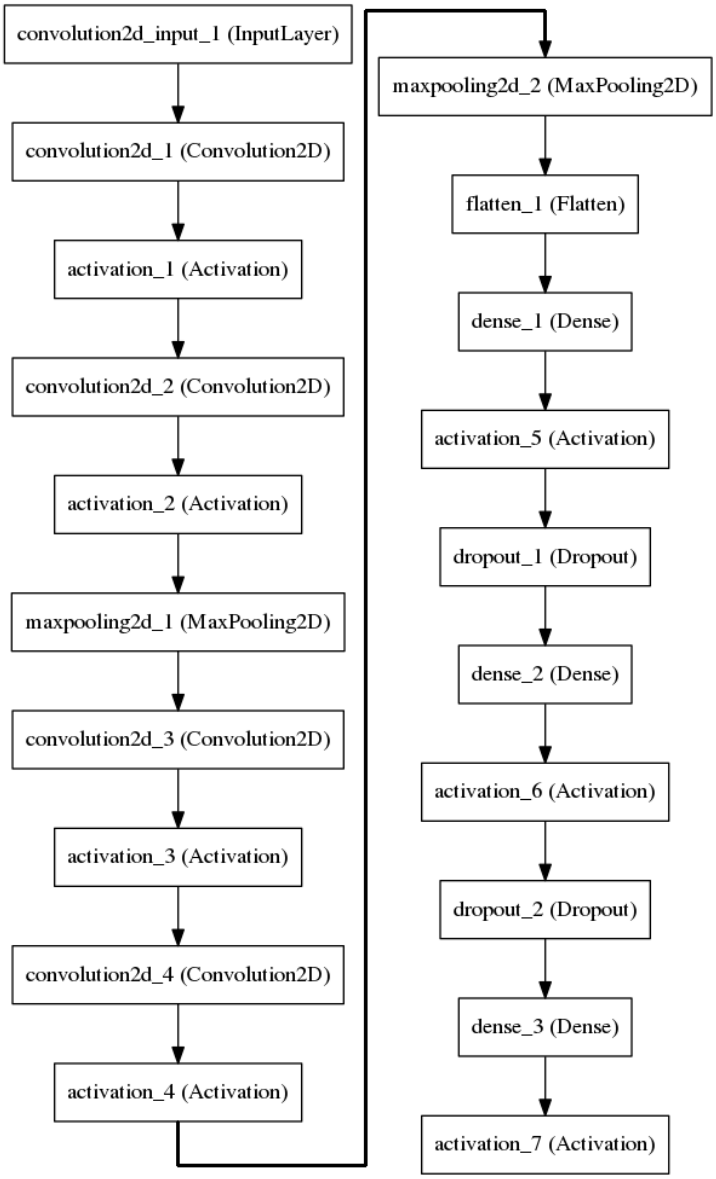
4.3. Konvolüsyonel Sinir Ağı (CNN)

4.3.1. Tavuk Veri Seti 100x140 Piksel Boyutlu Resimler

CNN algoritması kullanılarak ölçeklenmiş boyutlu resimler üzerinde yapılan çalışmaların ilk adımında 100x140 piksel boyutlu resimler kullanılmıştır. Yapılan testler sonucunda en başarılı model tespit edilmiştir. Keras kullanılarak oluşturulan modelde katmanlara göre yapılan işlemler şunlardır:

- **Giriş katmanı:** 100x140 piksel boyutlu resimler 3x3 boyutlu 16 tur konvolüsyon maskesine sokulmuştur. Giriş katmanında çerçeve türü "valid" seçilerek tüm pikseller üzerinde işlem yapması sağlanmıştır.
- **İkinci katman:** 3x3 16 tur konvolüsyon maskesi uygulanmıştır. Katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" kullanılmıştır.
- **Üçüncü katman:** 2x2 havuzlama işlemi yapılmıştır. Üçüncü ve dördüncü katmanlar arasında aktivasyon fonksiyonu "ReLU" ve kaçınma değeri 0.25 olarak kullanılmıştır.
- **Dördüncü katman:** 3x3 32 turluk konvolüsyon maskesi uygulanmıştır.
- **Beşinci katman:** 3x3 32 turluk konvolüsyon maskesi uygulanmıştır.
- **Altıncı katman:** "Flatten" katmanı kullanılarak veriler tek boyutlu dizi haline dönüştürülmüştür.
- **Yedinci katman**: 256 nöron bulunmaktadır. Yedinci ve sekizinci katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.5 kullanılmıştır.
- **Sekizinci katman:** 32 nöron bulunmaktadır. Çıkış katmanı ile sekizinci katman arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.5 kullanılmıştır.
- **Çıkış katmanı:** sınıf sayısına eşit şekilde 7 nöron bulundurmaktadır. Bu katmanda aktivasyon fonksiyonu olarak "Softmax" kullanılmıştır (Şekil 4.4).

Öğrenme algoritması olarak Stokastik Eğim Azalması (SGD) seçilmiştir. Algoritma üzerinde öğrenme oranı olarak başlangıç değeri 0.1 olmasına rağmen 0.05 tercih edilmiştir. Bu şekilde öğrenme geç olmakta fakat algoritmanın minimum'u kaçırma ihtimali azalmaktadır. Öğrenme esnasında geri besleme işlemi (*epoch*) 32 kez tekrarlanmıştır. Her bir geri besleme yaklaşık 4 saniye sürmektedir.



Şekil 4.4. Tavşan veri seti 100x150 piksel boyutlu resimler için kullanılan model

100x140 piksel resimler üzerinde CNN algoritması kullanılarak 100 tur yapılan çalışmada toplamda 87.44 % başarım elde edilmiştir. En yüksek başarım %99.04 ile E. Maxima türünde, en düşük başarım ise %68.75 ile E. Necatrix türünde gerçekleşmiştir (Çizelge 4.9). Sonuçlara ait doğruluk karşılaştırma tablosu Çizelge 4.10'da gösterilmektedir.

Çizelge 4.9. Tavuk veri seti 100x140 piksel boyutlu resimler için CNN sonuçları

Ortalama Doğruluk	% 87.44
Standart Sapma	% 0.36
Minimum Doğruluk	% 86.68
Maksimum Doğruluk	% 88.26
BRU	% 93.56
PRA	% 84.73
TEN	% 79.79
MAX	% 99.04
ACE	% 95.04
NEC	% 68.75
MIT	% 92.83

Çizelge 4.10. Tavuk veri seti 100x140 piksel boyutlu resimler için CNN doğruluk tablosu

		BRU	**PRA**	**TEN**	**MAX**	**ACE**	**NEC**	**MIT**
BRU	O.%	**93.56**	3.16	2.92	0.32	0.00	0.03	0.00
	Std.	**1.07**	0.98	0.64	0.24	0.0	0.08	0.0
PRA	O.%	1.98	**84.73**	3.92	0.00	0.26	3.61	5.51
	Std.	0.59	**1.78**	0.95	0.0	0.14	0.8	0.79
TEN	O.%	2.61	5.42	**79.79**	0.00	0.39	11.19	0.59
	Std.	0.41	1.25	**2.7**	0.0	0.15	2.19	0.17
MAX	O.%	0.96	0.00	0.00	**99.04**	0.00	0.00	0.0
	Std.	0.39	0.00	0.0	**0.39**	0.0	0.0	0.0
ACE	O.%	0.00	0.27	0.05	0.00	**95.04**	3.49	1.14
	Std.	0.0	0.18	0.12	0.0	**0.82**	0.72	0.45
NEC	O.%	0.52	7.54	13.95	0.00	5.73	**68.75**	3.51
	Std.	0.14	1.34	2.48	0.0	0.74	**3.11**	0.92
MIT	O.%	0.00	4.94	0.09	0.00	0.4	1.74	**92.83**
	Std.	0.02	0.85	0.10	0.0	0.18	0.52	**1.03**

4.3.2. Tavşan Veri Seti 100x150 Piksel Boyutlu Resimler

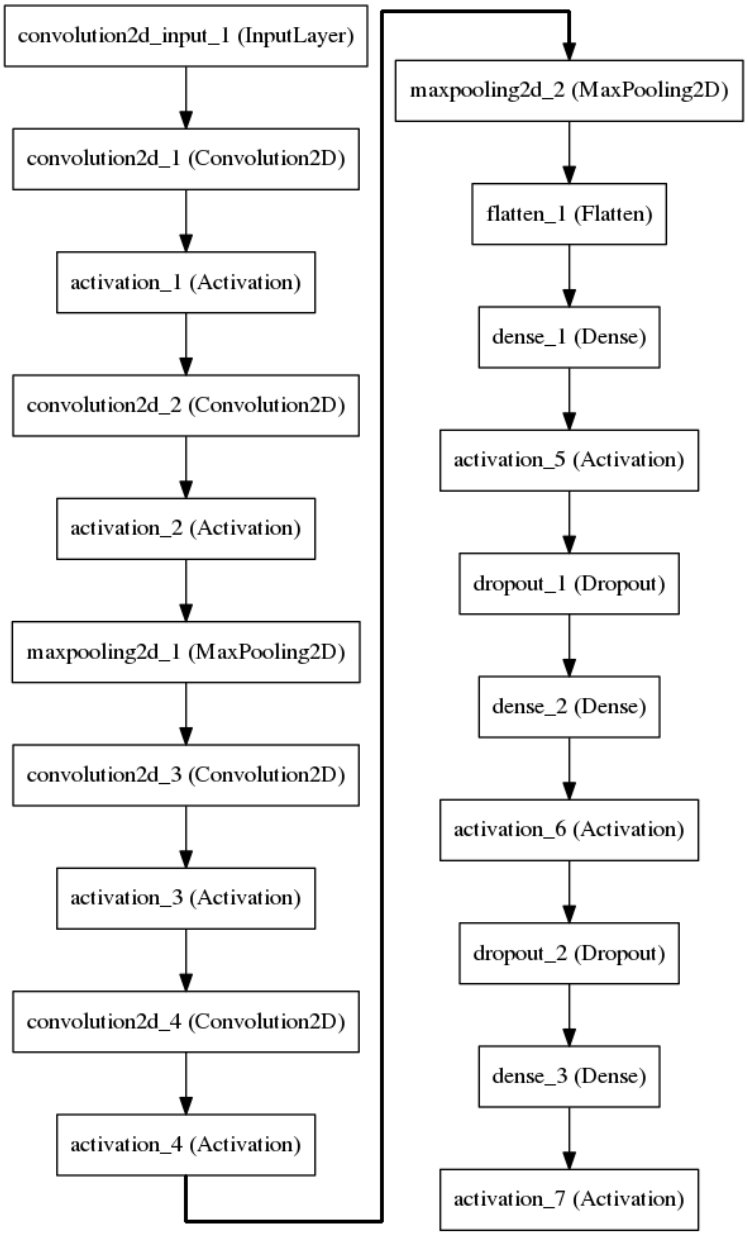
CNN algoritması kullanılarak ölçeklenmiş boyutlu resimler üzerinde yapılan çalışmaların ilk adımında 100x150 piksel boyutlu resimler kullanılmıştır. Yapılan testler sonucunda en başarılı model tespit edilmiştir. Keras kullanılarak oluşturulan modelde katmanlara göre yapılan işlemler şunlardır:

- **Giriş Katmanı:** 100x150 piksel boyutlu resimler 3x3 boyutlu 32 tur konvolüsyon maskesine sokulmuştur. Giriş katmanında çerçeve türü "valid" seçilerek tüm pikseller üzerinde işlem yapması sağlanmıştır. Katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" kullanılmıştır.
- **İkinci katman:** 3x3 32 tur konvolüsyon maskesi uygulanmıştır.
- **Üçüncü katman:** 2x2 havuzlama işlemi yapılmıştır. Üçüncü ve dördüncü katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" kullanılmıştır.
- **Dördüncü katman:** 3x3 64 turluk konvolüsyon maskesi uygulanmıştır. Katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" kullanılmıştır.
- **Beşinci katman:** 3x3 64 turluk konvolüsyon maskesi uygulanmıştır. Katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" kullanılmıştır.
- **Altıncı katman:** 2x2 havuzlama işlemi yapılmıştır.
- **Yedinci katman:** "Flatten" katmanı kullanılarak veriler tek boyutlu dizi haline dönüştürülmüştür.
- **Sekizinci katman:** 128 nöron bulunmaktadır. Sekizinci ve dokuzuncu katmanlar arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.4 kullanılmıştır.
- **Dokuzuncu katman:** 128 nöron bulunmaktadır. Çıkış katmanı ile dokuzuncu katman arasında aktivasyon fonksiyonu olarak "ReLU" ve kaçınma değeri olarak 0.4 kullanılmıştır.
- **Çıkış katmanı:** sınıf sayısına eşit şekilde 11 nöron bulundurmaktadır. Bu katmanda aktivasyon fonksiyonu olarak "Softmax" kullanılmıştır (Şekil 4.5).

Öğrenme algoritması olarak Stokastik Eğim Azalması (SGD) seçilmiştir. Algoritma üzerinde öğrenme oranı olarak başlangıç değeri 0.1 olmasına rağmen 0.01

tercih edilmiştir. Bu şekilde öğrenme geç olmakta fakat algoritmanın minimum'u kaçırma ihtimali azalmaktadır.

Öğrenme esnasında geri besleme işlemi (*epoch*) 40 kez tekrarlanmıştır. Her bir geri besleme yaklaşık 10 saniye sürmektedir.



Şekil 4.5. Tavşan veri seti 100x150 piksel boyutlu resimler için kullanılan model

100x150 piksel resimler üzerinde CNN algoritması kullanılarak 100 tur yapılan çalışmada toplamda %78.42 başarım elde edilmiştir. En yüksek başarım %99.89 ile

E.exigua türünde, en düşük başarım ise %52.85 ile E. coecicola türünde gerçekleşmiştir (Çizelge 4.11).

Sonuçlara ait doğruluk karşılaştırma tablosu Çizelge 4.12'de gösterilmektedir.

Çizelge 4.11. Tavşan veri seti 100x150 piksel boyutlu resimler için CNN sonuçları

Ortalama Doğruluk	% 78.42
Standart Sapma	% 0.53
Minimum Doğruluk	% 77.20
Maksimum Doğruluk	% 79.72
INT	% 78.08
PIR	% 72.84
VEG	% 66.60
IRR	% 86.17
COE	% 52.85
STI	% 57.75
EXI	% 99.89
PER	% 99.30
MAG	% 90.07
MED	% 70.28
FLA	% 76.83

Çizelge 4.12. Tavşan veri seti 100x150 piksel boyutlu resimler için CNN doğruluk tablosu

		INT	**PIR**	**VEG**	**IRR**	**COE**	**STI**	**EXI**	**PER**	**MAG**	**MED**	**FLA**
INT	O. %	**78.08**	7.82	1.93	0.10	1.23	0.0	0.05	2.26	0.0	6.75	1.77
	Std.	**3.14**	2.13	1.01	0.26	0.65	0.0	0.17	0.99	0.0	1.95	0.87
PIR	O. %	11.75	**72.84**	4.74	0.0	0.11	0.12	0.0	0.01	0.98	2.01	7.44
	Std.	2.43	**3.47**	1.08	0.0	0.28	0.29	0.0	0.07	0.58	0.92	1.9
VEG	O. %	0.96	1.25	**66.6**	0.16	16.40	3.38	0.0	0.0	0.21	9.20	1.84
	Std.	0.46	0.52	**3.51**	0.17	3.03	1.02	0.0	0.03	0.26	1.33	0.71
IRR	O. %	0.0	0.0	0.45	**86.17**	0.25	0.07	0.0	0.0	12.82	0.0	0.23
	Std.	0.0	0.05	0.25	**2.35**	0.31	0.18	0.0	0.0	2.31	0.0	0.31
COE	O. %	0.55	0.10	23.08	1.18	**52.85**	13.12	0.0	0.0	0.93	6.67	1.53
	Std.	0.37	0.19	3.9	0.28	**4.08**	2.16	0.0	0.0	0.5	1.24	0.59
STI	O. %	0.02	0.06	6.93	0.91	16.10	**57.75**	0.0	0.0	6.64	0.68	10.91
	Std.	0.11	0.16	178	0.48	2.53	**3.78**	0.0	0.0	1.21	0.6	1.91
EXI	O. %	0.0	0.0	0.0	0.0	0.0	0.0	**99.89**	0.11	0.0	0.0	0.0
	Std.	0.0	0.0	0.0	0.0	0.0	0.0	**0.22**	0.22	0.0	0.0	0.0
PER	O. %	0.0	0.0	0.0	0.0	0.0	0.0	0.04	**99.30**	0.0	0.65	0.0
	Std.	0.0	0.0	0.0	0.0	0.0	0.0	0.2	**0.76**	0.0	0.71	0.0
MAG	O. %	0.0	0.02	0.1	3.97	0.25	0.97	0.0	0.0	**90.07**	0.0	4.63
	Std.	0.0	0.06	0.14	0.83	0.2	0.44	0.0	0.0	**1.47**	0.02	1.06
MED	O. %	2.62	0.65	8.6	0.0	4.01	1.27	0.0	4.88	0.02	**70.28**	7.67
	Std.	1.0	0.48	1.99	0.0	1.42	0.59	0.0	0.72	0.09	**2.7**	1.41
FLA	O. %	1.32	2.18	1.83	0.14	0.47	4.25	0.0	0.02	8.76	4.22	**76.83**
	Std.	0.56	0.61	0.55	0.19	0.33	1.0	0.0	0.07	1.54	0.85	**2.46**

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Çalışmada mikroskobik görüntüler üzerinde derin öğrenme algoritmaları kullanılarak, tavuk ve tavşan veri setlerine uygulaması gerçekleştirilmiştir.

Konvolüsyonel Sinir Ağı algoritması Çok Katmanlı Yapay Sinir Ağı algoritmasına göre daha fazla kaynak ihtiyacı olduğu için büyük resimlerle Konvolüsyonel Sinir Ağı algoritmasını çalıştırmak mümkün olmamıştır.

Tavuk ve tavşan veri setleri için kullanılan resim boyutlarına göre MLP ve CNN derin öğrenme algoritması sonuçları Çizelge 5.1'de verilmiştir.

Çizelge 5.1. Tavuk ve tavşan veri setlerinin kullanılan resim boyutlarına göre MLP ve CNN sonuçları

		Tavuk 360x504 piksel	**Tavuk** 100x140 piksel	**Tavşan** 100x150 piksel
CNN	Ort. %	-	% 87.44	% 78.42
	Std.Sp.	-	0.36	0.53
MLP	Ort. %	% 83.75	% 82.81	% 58.99
	Std.Sp.	0.60	0.81	0.70

Konvolüsyonel Sinir Ağı ve Çok Katmanlı Yapay Sinir Ağı algoritmaları karşılaştırıldığında Konvolüsyonel Sinir Ağı algoritmasının kullanılan iki veri setinde de daha başarılı sonuç verdiği görülmektedir. Bununla birlikte Konvolüsyonel Sinir Ağı algoritmasının eğitim aşaması Çok Katmanlı Yapay Sinir Ağı algoritmasına göre daha fazla kaynak tüketmekte ve daha uzun sürmektedir. Tavuk veri setinde her bir geri besleme adımı Çok Katmanlı Yapay Sinir Ağı algoritması için 1 saniye sürerken, Konvolüsyonel Sinir Ağı algoritmasında ise 4 saniye sürmektedir.

Tavuk veri setine ait sonuçlar incelendiğinde Çok Katmanlı Yapay Sinir Ağı algoritması için orijinal boyutlu resimler (360x504) ile küçültülüş resimler (100x140) arasında % 1 den daha küçük bir fark bulunmaktadır. Sonuçlar arasındaki bu küçük farka rağmen eğitim süreleri arasında oldukça büyük farklar bulunmaktadır. Orijinal boyutlu resimlerdeki her bir geri besleme adımı 4 dakika sürerken, bu süre küçültülmüş resimler için 4 saniye olarak ölçülmüştür.

Tavşan veri setine ait sonuçlar incelendiğinde sonuçların tavuk veri setine göre daha düşük olduğu görülmektedir. Sonuçların düşük olmasının sebebinin her bir sınıf için az sayıda resim olması sebebiyle öğrenme için yeterli veri bulunmaması olduğu değerlendirilmektedir.

Tavuk veri setine ait sonuçlar değerlendirildiğinde yapılan her 3 çalışmadaki sonuçların benzer çalışmalarda elde edilen sonuçlardan daha başarılı olduğu görülmektedir. Tavuk veri seti için Abdalla ve ark. (2015) yılında yayınladıkları çalışmada %82 başarım elde etmişlerdir. Bu çalışmada elde edilen en başarılı sonuç Konvolüsyonel Sinir Ağı algoritması kullanılarak %0.36 standart sapma ile %87.44 olarak elde edilmiştir. En başarısız sonuç ise % 0.81 standart sapma ile %82.81 olarak elde edilmiştir.

Tavşan veri setine ait sonuçlar değerlendirildiğinde Konvolüsyonel Sinir Ağı algoritması kullanılarak yapılan çalışmadaki sonuçların benzer çalışmalarda elde edilen sonuçlar ile karşılaştırılabilir seviyelerde olduğu görülmektedir. Tavşan veri seti için Abdalla ve ark., 2017 yılında yayınladıkları çalışmada %1.60 standart sapma ile %82.83 başarım elde etmişlerdir. Bu çalışmada elde edilen en başarılı sonuç Konvolüsyonel Sinir Ağı algoritması kullanılarak %0.53 standart sapma ile %78.42 olarak elde edilmiştir. Elde edilen sonuçlar daha iyi standart sapma ile %4.41 daha az başarım göstermektedir.

5.2. Öneriler

Bu çalışmada Çok Katmanlı Yapay Sinir Ağı ve Konvolüsyonel Sinir Ağı algoritmaları iki ayrı veri seti üzerinde uygulanmış ve sonuçları incelenmiştir. Kullanılan veri setleri üzerinde farklı derin öğrenme algoritmaları ve parametreleri kullanılarak başarım oranı ve performans artırılabilecektir. Sonraki çalışmalarda yeni algoritmalar kullanılarak başarım oranının artırılması denenecektir. Sonraki çalışmalarda aynı veri seti üzerinde derin öğrenme algoritmaları kullanarak “Süper piksel” (Cong ve ark., 2015; Abdalla ve ark., 2017) tespiti için çalışmalar yapılacaktır.

KAYNAKLAR

Abdalla, M. A. E., Seker, H.,, 2017, Recognition of Protozoan Parasites from Microscopic Images: Eimeria species in Chickens and Rabbits as a case study, 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, pp. 1517-1520.

Avci, D., Varol, A., 2009, An expert diagnosis system for classification of human parasite eggs based on multiclass SVM, Expert Systems with Applications, vol. 36, pp. 43–48.

Barkmeyer, N., 2015, Deep learning: Convolutional neural networks for object recognition, TUM Advanced Seminar, Supervisor: Andreas Holzbach, TUM Institute for Cognitive Systems.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y., 2010, Theano: a CPU and GPU math expression compiler, In Proceedings of the Python for Scientific Computing Conference (SciPy)-June 2010.

Bengio, Y., 2008, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1– 127.

Bengio, Y., Goodfellow, I. J., Courville, A., 2014, Deep learning. Book in preparation for MIT Press.

Beyeler, M., 2015, OpenCV with Python Blueprints, Packt Publishing.

Bishop, C. M., 2006, Pattern Recognition and Machine Learning, Springer.

Bottou, L., 1991, Stochastic gradient learning in neural networks, In Proceedings of Neuro-Nime, EC2.

Castañón, C. A. B., Fraga, J. S., Fernandez, S., Gruber, A., 2007, Biological shape characterization for automatic image recognition and diagnosis of protozoan parasites of the genus Eimeria, Pattern Recognition, vol. 40(7), pp. 1899 – 1910.

Chollet, F., 2015, Keras, *https://github.com/fchollet/keras* [Ziyaret Tarihi: 1 Ekim 2017].

Cong, Y., Wang, S., Liu, J., Cao, J., Yang, Y., Luo, J., 2015, Deep sparse feature selection for computer aided endoscopy diagnosis, Pattern Recognition, Volume: 48, Issue: 3, p.907-917.

Daugschiesa, A., Imaromb, S., Bollwahnc, W., 1999, Differentiation of porcine Eimeria spp. by morphologic algorithms, Veterinary Parasitology, vol. 81, pp. 201-210.

Duda, R. O., Hart, P. E., Stork, D. G., 2012, Pattern classification. John Wiley & Sons.

Food and Agriculture Organization (FAO), 2010, Poultry meat & eggs agribusiness handbook, Rome, Italy.

Hadipour, M. M., Olyaie, A., Naderi, M., Azad, F., Nekouie, O., 2013, Prevalence of Eimeria Species in Scavenging Native Chickens of Shiraz, Iran, African Journal of Poultry Farming, vol. 1, pp. 034-036.

Hinton, G. E., Nair, V., 2010, Proceeding ICML'10 Proceedings of the 27th International Conference on International Conference on Machine Learning, Pages 807-814.

Kaupp, B. F., 1917, Animal Parasites and Parasitic Diseases. Chicago: Alexander Eger.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012, ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25 (NIPS2012), pages 1–9.

Kučera J., Režnický, M., 1991, Differentiation of species of Eimeria from the fowl using a computerized image-analysis system, Folia Parasitologica, vol. 38, pp. 107-113.

Lecun, Y., Bottou, L., Bengio, Y., Ha, P., 1998, Gradient-Based Learning Applied to Document Recognition. (November):1–46.

LeCun, Y., Bottou, L., Orr, G., Müller, K. R., 1998, Efficient backpropagation, In Neural networks: Tricks of the trade. Springer.

Rosenblatt, F. 1962. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, New York: Spartan.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1986, Learning Internal Representations by Error Propagation, Parallel distributed processing: Explorations in the microstructure of cognition, Vol: 1, MIT Press.

Seymour S., 1997, The Brain: Our Nervous System, New York: William Morrow.

Suzuki, C. T. N., Gomes, J. F., Falcao, A. X., Shimizu, S. H., Papa, J. P., 2013. Automated Diagnosis Of Human Intestinal Parasites Using Optical Microscopy Images, 10th International Symposium on Biomedical Imaging: From Nano to Macro, San Francisco, CA, USA, pp. 460-463.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015, Going Deeper With Convolutions, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 1–9.

Van Der Walt, S., Colbert, S. C., Varoquaux, G., 2011, The NumPy array: a structure for efficient numerical computation, Computing in Science & Engineering 13.2, p.22-30.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Mücahit Büyükyılmaz
Uyruğu	**:**	Türk
Doğum Yeri ve Tarihi	**:**	Selçuklu, 27.07.1990
Telefon	**:**	0544 4233582
Faks	**:**	
e-mail	**:**	mucahit@konya.edu.tr

EĞİTİM

Derece		**Adı, İl**	**Bitirme Yılı**
Lise	:	Karaman Fen Lisesi, Karaman	2008
Üniversite	:	Selçuk Üniversitesi, Bilgisayar Mühendisliği, Konya	2013
Yüksek Lisans	:		
Doktora	:		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2015 - devam	Necmettin Erbakan Üniversitesi, Bilgi İşlem Daire Başkanlığı	Uzman

UZMANLIK ALANI

- **Programlama Dilleri:** Python, Php, Javascript, C/C++
- **İşletim Sistemleri:** Linux, Unix

YABANCI DİLLER

- **İngilizce** (Okuma: İyi, Konuşma: İyi, Yazma: İyi)

BELİRTMEK İSTEĞİNİZ DİĞER ÖZELLİKLER

YAYINLAR

- Mucahit Buyukyilmaz, Ali Osman Cibikdiken, “Voice Gender Recognition Using Deep Learning”, Advances in Computer Science Research, Volume 58, 2017
- Ali Osman Cibikdiken, Mucahit Buyukyilmaz, Mehmet Akif Nacar, Ahmet Ercan Topcu, Yusuf Tambag, Abdullah Karadag, “Development of a New Platform through Distributed Storage System for the Bioinformatics Analysis”, 3rd International Conference on Artificial Intelligence and Industrial Engineering (AIIE2017), 2017.