FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

EL YAZISI KARAKTER TANIMA VE RESİM SINIFLANDIRMADA DERİN ÖGRENME YAKLAŞIMLARI

YÜKSEK LİSANS TEZİ

Aoudou SALOUHOU

Anabilim Dalı: Bilgisayar Mühendisliği

EYLÜL 2019

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

EL YAZISI KARAKTER TANIMA VE RESİM SINIFLANDIRMADA DERİN ÖGRENME YAKLAŞIMLARI

YÜKSEK LİSANS TEZİ

Aoudou SALOUHOU
(160221013)

Anabilim Dalı: Bilgisayar Mühendisliği
Düzeltilmiş Tez

Tez Danışmanı:
Dr. Öğr. Üyesi Berna KİRAZ

Teslim Tarihi: 05.08.2019

TEZ ONAY SAYFASI

FSMVÜ, Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği yüksek lisans programı 160221013 numaralı öğrencisi Aoudou SALOUHOU'nun ilgili yönetmeliklerin belirlediği tüm şartları yerine getirdikten sonra hazırladığı **"El Yazısı Karakter Tanıma ve Resim Sınıflandırmada Derin Öğrenme Yaklaşımları"** başlıklı tezi aşağıda imzaları olan jüri tarafından **05.09.2019** tarihinde oybirliği ile kabul edilmiştir.

Tez Danışmanı

Dr. Öğr. Üyesi Berna KİRAZ .....
Fatih Sultan Mehmet Vakıf Üniversitesi

Jüri Üyeleri

Dr. Öğr. Üyesi Berna KİRAZ .....
Fatih Sultan Mehmet Vakıf Üniversitesi

Dr. Öğr. Üyesi Ayla GÜLCÜ .....
Fatih Sultan Mehmet Vakıf Üniversitesi

Dr. Öğr. Üyesi Fatma CORUT ERGİN .....
Marmara Üniversitesi

Teslim Tarihi : 05.08.2019
Savunma Tarihi: 05.09.2019

DÜZELTME METNİ

1. Çalışmamızda yer alan “Derin Öğrenme’yi Kullanarak Osmanlı El Yazısı Karakteri Tanıma” başlığı “El Yazısı Karakter Tanıma ve Resim Sınıflandırmada Derin Öğrenme Yaklaşımları” ile değiştirilmiştir.
2. Tezin formatı düzeltilmiştir.
3. Tezin yazım ve dilbilgisi hataları düzenlenmiştir.
4. Özet ve Abstract bilgiler eklenmiştir.
5. Çalışmamızda yer alan “Algoritmalar ve İlgili Çalışmalar” başlığı düzenlenerek Birinci Bölüm ile yeniden terkip yapılmıştır.
6. İkinci bölüm “Derin Öğrenme Algoritmaları” başlığı ile alt bölümlere ayrılmış ve yeniden oluşturulmuştur.
7. Üçüncü bölümdeki “Modellerin Eğitilmesi, Sonuçlandırılması ve karşılaştırılması ” başlığı “Deneysel Çalışmalar ve Bulgular” olarak değiştirilerek alt bölümleri yeniden düzeltilmiştir.
8. Tezin “Deneysel Çalışmalar ve Bulgular” başlığında, her modelin de oluşumunda kullanılan değerler tablolar haline getirilerek sunulmuştur.
9. Teze “Önerilen Yöntem” başlılığı Dördüncü bölüm olarak eklenerek alt bölümleri oluşturulmuştur.
10. Tezin Beşinci Bölümü olan “Sonuç” başlığına yeni bilgiler eklenmiştir.
11. “Kaynakça” kısmında bazı düzenlemeler yapılarak uygun formatta düzeltilmiştir.
12. Özgeçmiş yeniden düzenlenmiştir.

BEYAN BİLDİRİM

Bu tezin yazılmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, tezin herhangi bir kısmının bağlı olduğum üniversite veya bir başka üniversitedeki başka bir çalışma olarak sunulmadığını beyan ederim.

Aoudou SALOUHOU

EL YAZISI KARAKTER TANIMA VE RESİM SINIFLANDIRMADA DERİN ÖĞRENME YAKLAŞIMLARI

ÖZET

Geçtiğimiz yıllarda, yapay zeka (AI) ve makine öğrenmesi alanlarında muhteşem bir araştırma yapılmıştır. Bununla birlikte, geçmiş ve son yıllarda, araştırma, yaklaşımı, makine öğreniminin ileri alanı olan 'Derin Öğrenme' alanına yoğunlaşmıştır. Bu yeni araştırma alanı, algoritmalarını görüntü tanıma, görüntü sınıflandırma, konuşma tanıma gibi problemlere uygularken daha iyi sonuçlar vererek çok ilgi çekici bir araştırma alanı haline gelmiştir. Bu çalışmada, Derin Sinir Ağları (Deep Neural Network - DNN) veya Çok Katmanlı Algılayıcı (Multi-Layer Perceptron - MLP), Evrişimsel Sinir Ağı (Convolutional Neural Network - CNN) ve Uzun Kısa Süreli Bellek (Long Short Term Memory - LSTM) adı verilen özel Reccurent Sinir Ağı (Recurrent Neural Network - RNN) olmak üzere üç Derin Öğrenme algoritması kullanarak MNIST, FASHION-MNIST, CIFAR-10 ve ARAPÇA veri setleri aracılığıyla el yazı karaktere tanıma ve resim sınıflandırma problemlere uygulandıktan sonra, ilk aşamada modellerimizin sonuçları karşılaştırılmıştır. İkinci aşama ise kullanılan algoritmalar literatürde önerilen benzer modellerin sonuçlarıyla kıyaslamıştır. Bu araştırmada, gerekli yöntem ve ortam hazırladıktan sonra DNN, CNN ve RNN modelleri oluşturarak hiper-parametrelerini belirlenmiştir. Deneysel bölümünde, kullanılan veri setlerine göre, önerilen modeller test verilerindeki doğruluk (accuracy) ve kayıp (loss) değerleri bakımından alınan sonuçları Tensorboard ortamında optimum dönem (epoch) sayıları bazında modellerin test tamamladığında davranışları grafiksel olarak gösterilmiştir. Modellerin kıyaslaması için yine veri setlerine göre sütun grafikleri çizilmiştir. Kıyaslamadan el yazı karakter ve resim sınıflandırmada, CNN modeli en iyi olduğunu kaydedilmiştir. RNN ve CNN açısından, aynı veri seti kullanılan benzer çalışmalar karşılaştığında, bu tezde oluşturulan modellerin daha iyi sonuç verdikleri izlenmiştir. Deneysel çalışma

sonunda, tüm modellerimiz RNN (LTSM) Cifar-10 veri seti hariç, doğruluk değerlerinin gittikçe arttıkları ve kayıp değerlerinin gittikçe azaldıkları kaydedilerek modellerin iyi eğitmeleri ve test etmelerini ortaya çıkmıştır. RNN (LTSM) modeli, iyi sonuçları vermesine rağmen karakter tanıma ve resim sınıflandırmada pek uygun olmadığını kaydedilmiştir. Bu çalışmanın ardından Zıt Konumluluk Kavramı kullanılarak İkili Parçacık Sürü Optimizasyonu bir algoritması önerilerek DNN modeli Arapça veri seti üzerinde doğruluk ve kayıp değerleri değerlendirilmiştir.

Anahtar kelimerler; Derin Öğrenme, Derin Sinir Ağı, Evrişimsel Sinir Ağı, Uzun Kısa Vadeli Bellek, Hiper-parametre, Seyreltme.

DEEP LEARNING APPROACHES IN HANDWRITTING CHARACTER RECOGNITION AND IMAGE CLASSIFICATION

ABSTRACT

Through the past years, a marvelous research has been done on the fields of artificial intelligence or AI and machine learning. However, in the past and recent years, research has concentrated to the deep learning area which is the approach of AI, the advanced field of machine learning. This new field of reseach has given the better results while applying its algoritms to the problems such images recognation, images classification, speech recognation and that made it become a very interesting field of research. In this work, we apply three of Deep Learning algorithms which are Deep Neural Networks (DNN) or Multi Layers Perceptron (MLP), Convulutional Neural Network (CNN) and special type of Reccurent Neural Network (RNN) called Long Short Term Memory (LSTM). Those algorithms are applies to the problems such image recognition and image classification through MNIST, FASHION-MNIST, CIFAR-10 and ARABIC datasets in order to compare our models results between them at the first time, and secondly with others models that used the similar models in solving problems. In this study, after preparing the necessary methods and environment, DNN, CNN and RNN models were created and hyper-parameters were determined. In the experimental part, the results of the proposed models in terms of accuracy and loss values in the test data according to the data sets used are shown graphically when the models complete the test on the basis of the optimum epoch numbers in the Tensorboard environment. For comparison of the models, column chart were drawn according to the data sets. From that comparison, CNN model is recorded as the best one. In terms of RNN and CNN, it was observed that the models produced in this thesis yielded better results when compared to similar studies using the same data set. At the end of the experimental study, Except RNN (LTSM) model when tested with Cifar-10 dataset, it was recorded that the accuracy values were increasing and the loss values were decreasing gradually up to their optimum epoch values which mean

the models are well trained and tested. Although good results were noted with RNN (LTSM) model, it seems not to be a very suitable model in charater recognition and image classification. Following this study, Binary Particle Swarm Optimization together with Opposition-Based Learning was proposed to evaluate the accuracy and loss values on the Arabic data set of DNN model.

Keywords; Deep learning, Deep neural network, Convolutional neural network, Long short term memory, Hyper-parameter, Dropout.

ÖNSÖZ

Derin öğrenme, çok ilgi çekmesinin yanı sıra popüler bir araştırma alanı olmuştur. Bu çalışmanın amacı da, Derin, Evrişimsel ve Yinelemeli Sinir Ağları Derin Öğrenme Algoritmaları, el yazı karakteri tanıma ve resim sınıflandırma gibi problemlerin performansları arttırmak için ileri düzeydeki iyileme fonksiyon ve optimum hiper-parametreleri belirleyerek elde edilecek sonuçları önce modellerin sonradan benzer yapılan çalışma aynı veri seti ve aynı model göz önünde bulundurarak karşılaştırmaktır.

Bu tez Fatih Sultan Mehmet Vakıf Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Programı'nda hazırlanmıştır.

Öncelikle tez konusunu seçerken isteklerimi göz önünde bulundurup çalışmanın hazırlanma sürecinin her aşamasında bilgilesini, tecrübesini ve değerli zamanını esirgemeyerek, bana her fırsatta yardımcı olan değerli hocam Sayın Dr. Öğr. Üyesi Berna KİRAZ teşekkürlerimi sunarım. Tez sürecinde benden desteğini bir an için bile esirgemeyen değerli arkadaşım, Ayşenur ERDOĞAN'a, tüm eğitim hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen her zaman yanımda olan sevgili aileme, adlarını zikretmediği arkadaşlarıma, kurum ve kuruluşlara teşekkürlerimi bir borç bilirim.

Aoudou SALOUHOU

İstanbul, Eylül 2019

İÇİNDEKİLER

SEMOBLLER

$\boldsymbol{\theta}$	**:** İyileştirici Güncellenecek parametre vektörü
$\boldsymbol{\eta}$	: Öğrenme oranı
$\boldsymbol{\varepsilon}, \boldsymbol{\gamma}$	**:** 0 ile 1 arasında sabit katsayılar
$\boldsymbol{\beta_1}, \boldsymbol{\beta_2}$	**:** Adam iyileştirici hiper-parametreleri
$\boldsymbol{m_t}, \boldsymbol{v_t}$	**:** Birinci ve ikinci momentum tahminleri olan gradyanın üstel hareketli ve kare gradyanının ortalamaları
$\widehat{\boldsymbol{m_t}}, \widehat{\boldsymbol{v_t}}$	**:** Birinci ve ikinci momentum olan gradyanın üstel hareketli ve kare gradyanı ortalama değerleri

ÇİZELGE LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR

AI	Yapay Zeka (Artificial Intelligence)
API	Uygulama Programlama Arayüzü (Application Programming Interface)
BTT	Zaman Üzerinde Geri yayılım (Backpropagation Through Time)
CNN	Evrişimsel Sinir Ağı (Concolutıonal Neural Network)
DNN	Derin Sinir Ağı (Deep Neural Network)
CPU	Merkezi İşleme Ünitesi (Central Processing Unit)
GPU	Grafik İşleme Ünitesi (Graphics Processing Unit)
GRU	Geçitli Yinelemeli Ünite (Gated Recurrent Unit)
MNIST	Değiştirilmiş Ulusal Standartlar ve Teknoloji Enstitüsü (Modified National Institute of Standards and Technology)
OBL	Zıt Konumluluk Kavramı (Opposition-Based Learning)
PIDDA	Pima Hint diyabet veri kümesi (Pima Indian Diabetes Dataset)
BPSO	İkili Parçacık Sürü Optimizasyonu (Binary Particle Swarm Optimization)
RNN	Yinelemeli Sinir Ağı (Recurrent Neural Network)

1. GİRİŞ

Eski Yunanistan zamanına dayanan bir arzu, insanoğluna düşünen bir makine ortaya çıkarmayı hayal ettirmiştir. Programlanabilir bilgisayarların ilk kez tasarlanması ile birlikte, insanlar bu makinelerin akıllı hale gelip gelmeyeceğini merak etmişlerdir. Devamında ise yapay zekanın (Artificial Intelligence - AI) ortaya çıkmasıyla hayaller gerçek olmaya başlamıştır.

Yapay zeka birçok pratik uygulama ve aktif araştırma konuları ile gelişen bir alandır. Öyle ki makineye kendi kendine bazı problem çözebilme kabiliyeti kazandırılmasıyla 'makine öğrenmesi' kavramı ortaya çıkmıştır. Makine öğrenmesi, en basit seviyesinde, her türlü bilgisayar programını ifade eden ve bir insan tarafından açıkça programlanmasına gerek kalmadan kendi başına "öğrenebilen" bir çalışma alanıdır[1].

Son yıllarda, yapay zeka üzerinde yapılan araştırmalar sonucunda makine öğrenmesine bağlı olarak 'Derin Öğrenme' adıyla yeni bir alan gün ışığına çıkmıştır. Derin Öğrenme, özellikle bazı zor bilgisayar problemlerinin çözümünü hızlandırdığı için oldukça rağbet görmektedir[2]. Derin öğrenme, temelde birden fazla katmana sahip olan yapay sinir ağıdır ve örnekler üzerinden öğrenirler. Nitekim şoförsüz araçların arka planında yer alan derin öğrenme, bu araçların dur işaretini tanımalarını veya bir yayayı elektrik direğinden ayırmalarını sağlar. Dahası akıllı telefonlar, tabletler, akıllı TV'ler ve eller serbest (hands-free) hoparlörler gibi tüketici cihazlarında ses kontrolünün de anahtarıdır. Derin öğrenme çok dikkat çekici bir alan olup daha önce mümkün olmayan sonuçlara ulaşılmasına bir sebep olmuştur. Derin öğrenmede bir bilgisayar modeli, bir tanıma veya sınıflandırma problemlerinin doğrudan görüntülerden, metinlerden veya seslerden nasıl gerçekleştireceğini öğrenir. Derin öğrenme modelleri ile bazen insan seviyesindeki performansı aşarak, en gelişmiş doğruluğu (accuracy) ve en düşük kayıp (loss) değeri elde edebilir. Modeller, çok katmanlı, geniş bir etiketli veri seti (labeled data) sinir ağı mimarisi ile eğitilir[3].

Bu çalışmanın amacı; Derin, Evrişimsel ve Yinelemeli Sinir Ağları Derin Öğrenme algoritmalarının el yazı karakteri tanıma ve resim sınıflandırma gibi problemlere uygulanması ve performanslarının karşılaştırılmasıdır. Öncelikle her bir algoritma için en iyi performası veren modellerin ve optimum hiper-parametlerin belirlenmesi için çeşitli deneyler yapılmıştır. Modeller ve parametre ayarlamaları yapıldıktan sonra algoritmaların performasları dört farklı veri seti üzerinde karşılaştırılmıştır. Son zamanlarda, derin öğrenme alanında yeni ve gelişmiş çözümler bulmayı ve üzerinde çalışmayı önemli kılmıştır. Bir modelin daha iyi sonucu vermesi için büyük veri seti ihtiyacı duyulmuştur. Bu araştırmanın diğer amacı da çok çeşit ve büyük veri setleri ile oluşturularak modellerin eğitim ve testlerini gerçekleştirmektir. Bunlar için çalışmada, derin öğrenme algoritmaları olarak, Derin Sinir ağı (Deep Neural Network - DNN), Evrişimsel sinir ağı (Convolutinal neural network - CNN) ve Yinelemeli Sinir Ağı (Reccurent Neural network - RNN) olan Uzun Kısa Süreli Bellek (Long Short Term Memory - LSTM) kullanılmıştır.

Toplam beş bölümden oluşan bu tezde birinci bölümünde Makine Öğrenmesi ve ona bağlı olarak ortaya çıkan Derin Öğrenme anlatılmış, ayrıca bu tezin amacı açıklanmıştır. İkinci bölümde ise kullanılan derin öğrenme algortimalarının detayları ve bu algoritmaların bileşenleri olan algılayıcı ve lineer fonksiyonlar, kayıp fonksiyonu (loss function) ve eniyileme algoritmaları (optimization algorithms) sunulmuştur. Devamında gelen üçüncü bölümde, deneysel çalışmalar ve bulgurlardan bahsedilmiştir. Kullanılan veri kümeleri, performans metrikleri, kullanılan araçlar ve kütüphaneler, oluşturulan modeller ve hiper-parametre ayarlamaları, veri kümelerine göre algoritmaların karşılaştırılması, bunun sonucu elde edilen bulgular yine bu bölümde verilmiştir. Dördüncü bölümde önerilen yöntem, metotları ve onunla ilgili deneysel çalışması sunulmuştur. Beşinci bölümde ise sonuçlarlardan ve gelecek çalışmalardan bahsedilmiştir.

.

2. DERİN ÖĞRENME ALGORİTMALARI

Bu tezde literatürde iyi olarak bilinen ve en çok kullanılan üç adet derin öğrenme algoritması kullanılmıştır: Derin Sinir Ağı (Deep Neural Network - DNN), Evrişimsel Sinir Ağı (Convolutional Neural Network - CNN) ve Yinelemeli Sinir Ağı (Recurrent Neural Network - RNN) algoritmaları. Ayrıca yine bu bölümde, bu algoritmaların bileşenleri olan algılayıcı ve lineer fonksiyonlar, kayıp fonksiyonu ve eniyileme algoritmaları, aktivasyon fonksiyonları; DNN, CNN ve RNN algoritmaların detayları ve literatürde var olan çalışmalar ile ilgili bilgi verilmiştir.

2.1 Algılayıcı ve Lineer Fonksiyonlar

Matematikte **f(x) = a.x + b** şeklinde gösterilen fonksiyona lineer fonksiyon denilmektedir. Derin öğrenme veya yapay sinir ağlarında bir benzeri **y = W.x + b** lineer fonksiyonu olarak kullanılmaktadır. Burada **y** değeri ve **x** değeri bağımsız değişken olarak bağlıdır. Diğer **W** ve **b** ise fonksiyonun parametreleri tanımlanan değerleridir[4].

Yapay sinir ağlarında kullanılan bir başka fonksiyon ise algılayıcı (perceptron) olup, **y = W.x + b** olarak gösterilir. Derin öğrenme ağında girdi, gizli ve çıktı katmanlardan oluşturmaktadır. Gizli katmandaki çember ile gösterilen bir algılayıcı temsil eder. Bir ağ modelinde, algılayıcılar en küçük öğrenme birimi olarak tanımlanmaktadır. Dile getirdiğimiz fonksiyonda, **W** ağırlık parametre**, x** girdi**, b** yanlılık (bias) ve **y** çıktı değerler ifade etmektedir. Bu değerler daha iyi anlayabilmek için bu örneği göze alalım. Girdi olarak x = köpek resimlerini ait bir matrisi ağa ve öğrenme modeli besliyorsak, y ise köpek resimlerinin ne kadar doğru olduğunu değerleri verir. Parametrelerimiz W ve b ise y'den elde edilen değerler iyileştirmek için kullanılmaktadır. Buradan yola çıkarak derin öğrenme ağlarında veya çok katmanlı ağlarda temel bir şekilde yaptığımız işlemler daha iyi bir sonuç alabilmek için W ve b değerlerini hesaplanmasıdır [4].

2.2 Kayıp Fonksiyonu ve Eniyileme Algoritmaları

Derin öğrenmede optimum ve daha hızlı bir çözüm elde edebilmek için kayıp fonksiyonu ve optimizasyon algoritmaları seçmeleri büyük bir rolü oynamaktadır. Bu kısmında hata fonksiyonu ve eniyileme (optimization) algoritmalarından bahsedilecektir.

2.2.1 Kayıp Fonksiyonu

Hata fonksiyonu, Derin Öğrenme modellerinin hata oranı ve ne kadar başarı olduklarını ölçebilmek için kullanılan bir fonksiyondur. Çok büyük bir fayda sağlayan bu yapı, derin öğrenme ağlarının son katmanı hata fonksiyonu ile oluşturmaktadır. Bu fonksiyon, hata hesaplarken, eniyileme (optimization) problemini dönüştürerek modeli iyileştirmektedir. Bu fonksiyonun diğer adı ise maliyet fonksiyonu olup, görevi tasarlanan modelin tahmin değeri ve gerçek değeri arasındaki farkı hesaplamaktır. İyi bir model oluşturulmuş ise hata oranı düşük değil aksine yüksek olmalıdır [4].

2.2.2 Eniyileme Algoritmaları

Eniyileme algoritmaları, yitim fonksiyonunu (loss function) en aza indirgemek için ağırlık parametrelerini güncelleyen algoritmalardır. Yitim fonksiyonu, küresel minimum seviyesine ulaşmak için doğru yönde hareket edip etmediğini söyleyen bir fonksiyondur. Bu bölümde, Rasgele Gradyan İnişi (Stochastic Gradient Descent - SGD), Uyarlamalı Gradyan (Adaptive Gradient-Adagrad), Uyarlamalı Delta (Adaptive Delta - Adadelta), Ortalama Karekökü Yayılımı (Root Mean Square Propagation - Rmsprop) ve Uyarlamalı Momentum (Adaptive Moment - Adam) hakkında genel bilgiler verilmiştir.

- **Rasgele Gradyan İnişi (Stochastic Gradient Descent - SGD)**

Gradyan iniş, sinir ağlarında θ parametrelerine bağlı maliyet veya yitim fonksiyonunun azaltmak için yinelemeli bir yapay zeka, makine öğrenmesi veya derin öğrenmeside kullanıllan eniyileme algoritmasıdır. Bu algoritma, modellerin doğru tahminler yapmasına yardımcı olmaktadır. Her eğitim örneği için parametre güncellemesi yapan Rasgele Gradyan İnişi iyi bilinen gradyan iniştir. Genellikle çok daha hızlı bir tekniktir. Denklem 2.1'de güncelleme ifadesi gösterilmektedir.

$$\theta = \theta - \eta.\nabla \mathrm{J}(\theta;\ x(i);\mathrm{y}(i)) \qquad 2.1$$

θ, güncellenecek parametre vektörü, $\{x\,(\mathrm{i}), y\,(i)\}$ eğitim örnekleri, η ise öğrenme oranı ifade etmektedir.

Ancak SGD'nin büyük sorunu, sık sık yapılan güncellemeler ve dalgalanmalar nedeniyle, sonuçta yakınsamayı tam minimum seviyeye zorlaştırarak aşınmaya devam etmektedir [5].

- **Uyarlamalı Gradyan (Adaptive Gradient – Adagrad)**

Adagrad, uyarlanabilir bir öğrenme oranı yöntemidir. Bu yöntem, η öğrenme oranın parametrelere dayanarak adapte olmasını sağlamaktadır. AdaGrad seyrek parametreler için büyük güncellemeler yaparken sık parametreler için daha küçük güncellemeler yapmaktadır. Bu açıdan resim tanıma gibi seyrek veriler için daha uygundur. AdaGrad'da her parametrenin kendi öğrenme hızına sahip olmasından algoritmanın özelliklerine bağlı olarak öğrenme oranı giderek azalmaktadır. Bu yüzden öğreneme oranı giderek azalarak zamanın belli bir noktasında eğitilen modeli öğrenmeyi bırakmaktadır. Denklem 2.2'de θ parametre güncelleme ifadesi verilmektedir [6].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot g_t \qquad 2.2$$

G_t, tüm θ parametrelere geçmişteki gradyanların kare toplamıdır.

Adagrad'ın en büyük yararı, öğrenme oranını elle ayarlanmasına gerek olmamasıdır. Dezavantajı ise η öğrenme oranının daima azalmasıdır [5].

- **Uyarlamalı Delta (Adaptive Delta – Adadelta)**

Adadelta, Adagrad'ın bir uzantısıdır. Bu iyileştirici, Adagrad'ın eğitim süresince sürekli öğrenme oranlarının düşmesi ve elle seçilen küresel öğrenme oranı ortadan kaldırmaktır. Parametre güncellemeleri Denklem 2.3, 2.4 ve 2.5'te gösterilmiştir [6], [5].

$$\theta_{t+1} = \theta_t + \Delta\theta_t \qquad 2.3$$

$$\Delta\theta = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g_t]} \cdot g_t \quad (2.4)$$

$$RMS[g_t] = \sqrt{G_t + \varepsilon} \quad (2.5)$$

- **Ortalama Karekök Yayılımı (Root Mean Square propagation-RMSprop)**

RMSProp, Adagrad'ın köklü biçimde azalan öğrenme oranlarını, kare gradyanın hareketli bir ortalamasını kullanarak çözmeye çalışmaktadır. Gradyan normalleştirmek için son gradyan inişlerinin büyüklüğünü kullanmaktadır. RMSProp'ta, öğrenme hızı otomatik olarak ayarlanır ve her parametre için farklı bir öğrenme hızı seçmektedir. RMSProp öğrenme oranı, üstel kare gradyanların sönüm ortalamasına bölmektedir. Güncellenen θ değeri Denklem 2.6'te gösterilmektedir [5].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1-\gamma)g_{t-1}^2 + \gamma g_t + \varepsilon}} \cdot g_t \quad (2.6)$$

γ, 0 ile 1 arasında sönüm değeri g_t ise kare gradyanların hareketli ortalamasıdır.

- **Uyarlamalı Momentum (Adaptive Moment - Adam)**

Uyarlamalı Momentum Tahmini anlamına gelen Adam, her parametre için bireysel uyarlamalı öğrenme oranını, gradyanların birinci ve ikinci momentlerinin tahminlerinden hesaplayan başka bir eniyileştiricidir. Ayrıca Adagrad'ın köklü biçimde azalan öğrenme oranlarını da azaltmaktadır. Adam, Adagrad ve RMSprop'un bir kombinasyonu olarak düşünebilmektedir. Adam hesaplama açısından verimli ve çok az bellek gereksinimine sahip ve en popüler gradyan iniş eniyileme algoritmalarından biridir [7], [5].

Adam algoritması ilk olarak, Denklem 2.7 ve 2.7'te gösterildiği gibi, birinci ve ikinci momentum tahminleri olan gradyanın üstel hareketli (m_t) ve kare gradyanı (v_t) ortalamalarını güncellemektedir.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (2.7)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad (2.8)$$

Hiper parametreleri olan ***β1*** ve ***β2*** sırasıyla 0.9 ve 0.999 varsayılan değerlerini almaktadır.

θ parametresi güncellenebilime için m_t ve v_t bağlı olarak, birinci ve ikinci momentum olan gradyanın üstel hareketli ($\widehat{m_t}$) ve kare gradyanı ($\widehat{v_t}$) ortalama değerleri Denklem 2.9 ve 2.10'da verilmektedir.

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \quad 2.9$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t} \quad 2.10$$

θ parametresi güncelleme son hali Denklem 2.11'de gösterilmiştir.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}}+\varepsilon} \cdot \widehat{m_t} \quad 2.11$$

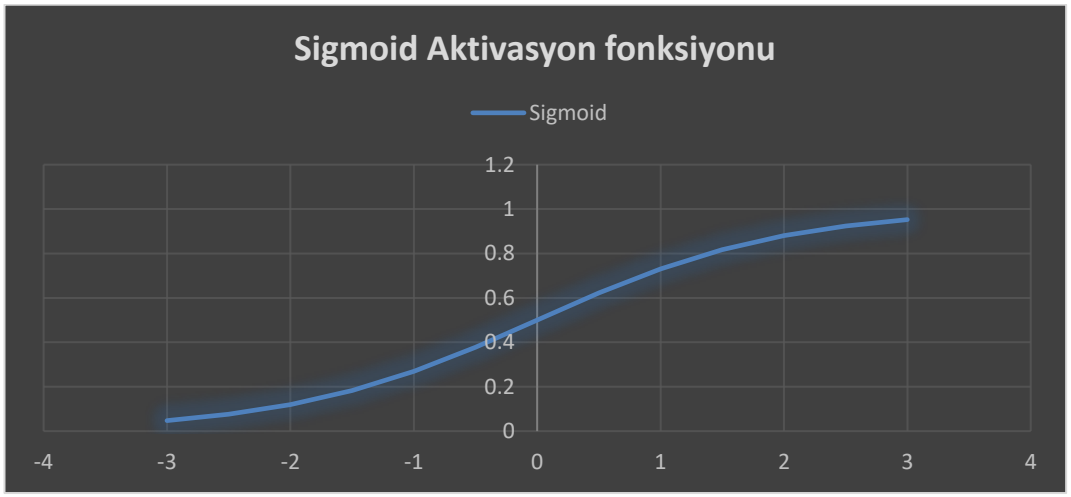
2.3 Aktivasyon Fonksiyonları

Aktivasyon fonksiyonu, yapay zeka, derin öğrenme modellerinde sinir olarak kullanılan bir fonksiyondur. Temel amacı ise bir nöronun bir giriş sinyalini bir çıkış sinyaline dönüştürmektir. Elde edilen çıkış sinyali, ağ modelindeki sonraki katmanda bir girdi olarak kullanılabilmektir. Sigmoid, Doğrultulmuş Doğrusal Üniteler veya Rectified Linear Units (ReLU) ve hiperbolik teğet fonksiyonları gibi birçok aktivasyon fonksiyonuna sahiptir. İfade ettiğimiz bu fonksiyonları kısaca aşağıdaki gibi açıklayabiliriz.

2.3.1 Sigmoid Fonksiyonu

Sigmoid aktivasyon fonksiyonu, f (x) = 1/1 + exp (-x) formunun aktivasyon fonksiyonudur (bkz. Şekil 2.1). Bu fonksiyon, 0 ile 1 arasında tanımlıdır. 'S' şeklinde bir eğridir. Anlamak ve uygulamak açısında kolay bir fonksiyondur ancak bunun popülerlikten düşmesinde iki neden vardır [8]: Birincisi Vanishing gradyan problemi, yani Sigmoidler gradyanları doyurarak öldürmektedir. Sigmoid nöronun çok istenmeyen bir özelliği, nöronun aktivasyonunun, ya 0 ya da 1 kuyruğundaki doygunluğa ulaştığında, bu bölgelerdeki gradyanın neredeyse sıfır olmaktadır. Geri yayılma sırasında, yerel gradyanın, tüm hedef için çıktı kapıdaki gradyana çarpılacaktır. Bu nedenle, eğer yerel gradyan çok küçükse, gradyanı etkili bir şekilde "öldürecek" ve neredeyse hiç bir sinyal nörondan ağırlığına ve tekrarlı olarak kendi verilerine akacaktır. Ek olarak, doygunluğu önlemek için sigmoid nöronlarının

ağırlıklarını başlatırken ekstra dikkat etmelidir. Örneğin, başlangıçtaki ağırlıklar çok büyükse, çoğu nöron doymuş hale gelir ve ağ zorlukla öğrenmektedir [9], [10]. İkincisi ise Sigmoid çıkışları sıfır merkezli olmamasıdır. Bir sinir ağında, daha sonraki işleminde, katmanlarındaki nöronların sıfır merkezli olmayan verileri alacağı için ortaya istenmeyen bir durum çıkmaktadır. Bu da sıkıntı söz konusudur, ancak yukarıdaki bahsedilen gradyan problemine kıyaslayacak olursak daha az ciddi etkileri vardır [11], [12].



Şekil 2.1: Sigmoid aktivasyon fonksiyonu

2.3.2 Hiperbolik Teğet Fonksiyonu

Hiperbolik teğet fonksiyonu veya Tanh, Matematikal formülü f (x) = 1 - exp (-2x) / 1 + exp (-2x) şeklindedir (bkz. Şekil 2.2). Bu fonksiyon, Sigmoid fonksiyonu aksine çıktısı sıfır merkezli -1 ile 1 arasında bir değer aldığından yani *-1 <çıkış <1*. Tanh fonksiyonu ile eniyileme işi daha kolaydır. Bu nedenle pratikte Sigmoid fonksiyonuna göre her zaman tercih edilmektedir. Ama yine de Vanishing gradyan problemi vardır [11], [12].

2.3.3 ReLU Fonksiyonu

Doğrultulmuş doğrusal üniteler (ReLU) fonksiyonu (bkz. Şekil 2.3), geçtiğimiz yıllarda çok popüler olduğundan son zamanlarda Tanh fonksiyonundan yakınsamada 6 kat iyileşme olduğu kanıtlanmıştır ve bugünlerde hemen hemen tüm derin öğrenme modelleri ReLU kullanmaktadır. Matematik ifadesi ise şu şekildedir: F (x) = maks (0,

x) yani eğer x <0, F (x) = 0 ve x> = 0 ise, F (x) = x. Dolayısıyla, bu matematiksel ifadelerden ReLU fonksiyonu çok basit ve etkili olduğunu anlamak mümkündür.

Şekil 2.2: Tanh aktivasyon fonksiyonu

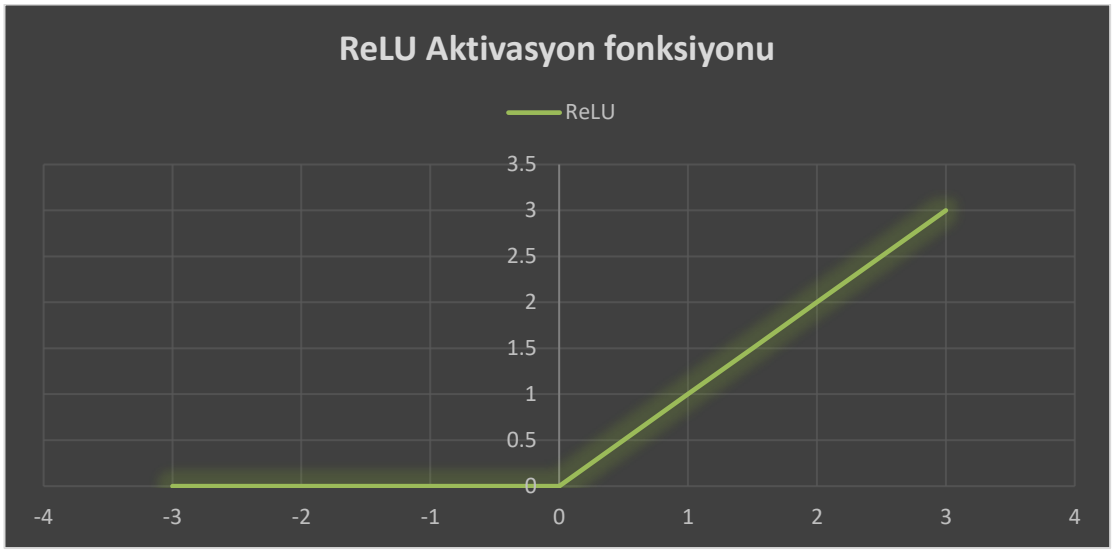
Makine öğrenme ve bilgisayar bilimi alanlarında en basit, tutarlı tekniklerin ve yöntemlerin daha çok tercih edildiği dikkat çekmektedir. Vanishing Gradyan probleminin önlenme ve düzeltilmesinde bu fonksiyon kullanılabilmektedir [11],[12].

2.4 Derin Sinir Ağı (Deep Neural Network - DNN)

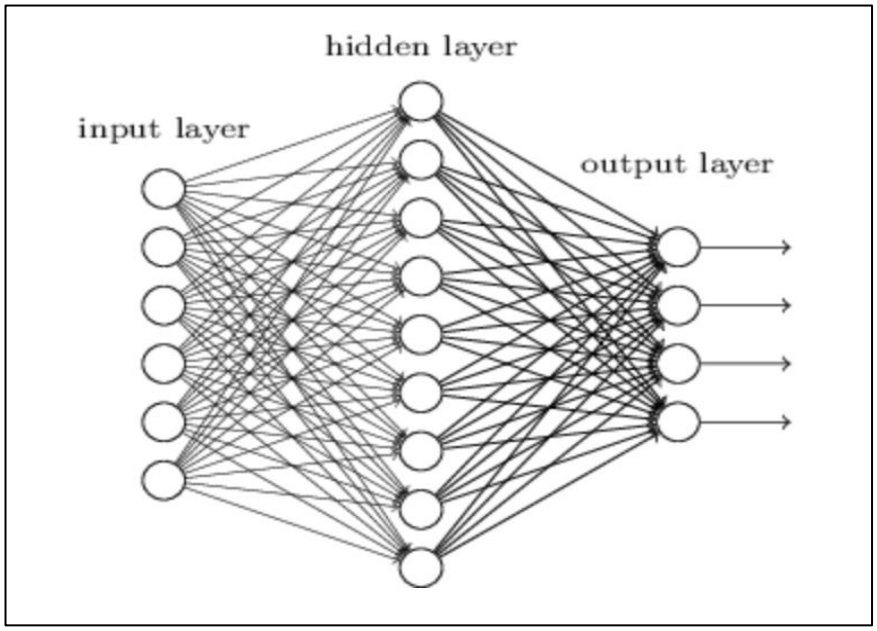
Derin öğrenme ağından bahsedeceğimiz bu bölümde konunun daha iyi anlaşılabilmesi için derin öğrenme ağının; mimari, algılayıcı, lineer fonksiyon, kayıp fonksiyonu, aktivasyon fonksiyonları ve diğer modelleri ayrıntılı olarak incelenecektir.

2.4.1 Derin Sinir Ağı Mimarisi

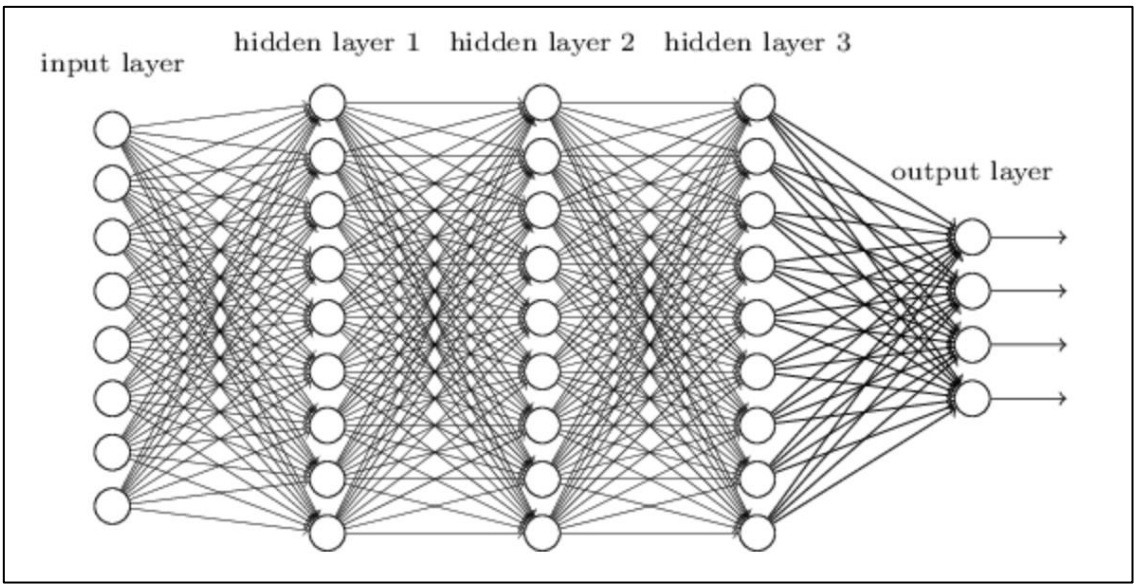
Derin sinir ağı (DNN) ağırlıkları tamamen bağlı ve sıklıkla kullanılan ön eğitim tekniğine göre başlatılan birçok gizli katmana sahip çok katmanlı, algılayıcı ağdır [11]. Genel olarak ifade etmek gerekirse Şekil 2.4 ve Şekil 2.5'te görüleceği üzere bir giriş katmanı ve çıkış katmanından oluşup aralarında en az bir gizli katman bulunan ağlardır.



Şekil 2.3: ReLU aktivasyon fonksiyonu



Şekil 2.4: DNN tek katmanı



Şekil 2.5: DNN çok katmanlı

Çok katmanlı ağlarda Şekil 2.4 ve Şekil 2.5'te gösterildiği gibi tek bir nörona bağlı, birden fazla girdi (input layer) vardır. Bu aynı zamanda ve seviyede birden fazla nöronda kullanılabileceğimizi göstermektedir. Nöronlar peş peşe katmanlar şeklinde de sıralanabilmektedir; bu tip ağ yapısı "Çok Katmanlı Ağ" olarak adlandırılmaktadır. Çok katmanlı ağlarında aradaki katmanlar gizli katman (hidden layer1, hidden layer2, hidden layer3) olarak isimlendirilmektedir. Son katman ise çıktı katmanıdır (output layer).

2.4.2 Katmanlar

- **Giriş katmanı**

Derin Sinir Ağında dış dünyadan girdilerin ilk geldikleri katman giriş katmanını ifade etmektedir. Bu katmanda girdi sayısı ve nöron sayısı aynı olması gerekmektedir. Bu katmana gelen girdiler hiç bir işlem gerçekleştirmeden bir sonraki katmalara iletilmektedir.

- **Gizli katman**

Giriş katmandan çıkan girdilerin aktığı 'Gizli Katman' sayısı ağdan ağa değişebilen bir yapıdır. Genelde bir ağa 'derin' denilmesi için en az iki gizli katmanı içermesi gerekmektedir. Gizli katmanlardaki nöron sayıları giren-çıkan nöron

sayısından tamamen bağımsızdır. Aynı zamanda da her gizli katmandaki nöron sayısı diğer ara katmanlardaki nöron sayısından farklı olabilmektedir. Bu katmanlarda, gizli katmanlar ve nöronların sayısının artması işlemin karmaşıklığının ve süresini arttırması anlamına gelmektedir. Bu yapı karmaşık problemlere çözüm bulabilmek için kullanılmaktadır.

- **Çıktı katmanı**

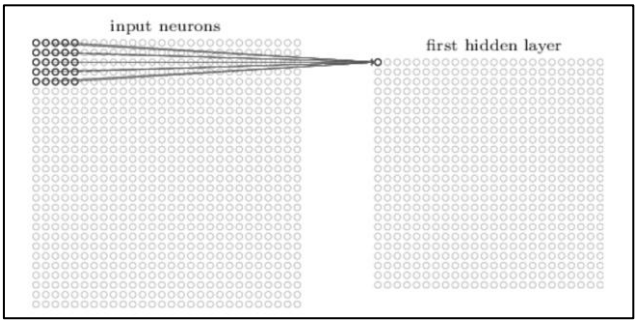
Gizli katmanlardan gelen çıktıların işlenip yeniden dış dünyaya aktırılmasını sağlayan katmandır.

2.5 Evrişimsel Sinir Ağı (Convolutional Neural Network - CNN)

Evrişimsel sinir ağı, derin öğrenmenin bir ağ mimarisidir. Doğrudan giriş verilerinden (görüntü, el yazısı veya sesler) öğrenen, denetimli modunda eğitilen ileri besleme, çok katmanlı ve özellikleri çıkartabilen bir ağ türüdür [11], [12]. Evrişimsel sinir ağı, bir çıktı üretmek için veri girişinin işlendiği ve dönüştürüldüğü bir takım katmanlardan oluşmaktadır. Bu ağ; sahne sınıflandırması, nesne algılama segmentasyonu ve görüntü işleme dahil olmak üzere görüntü analizi görevlerini yapmak için eğitebilen bir yapıdır [13]. Bu çalışmanın amacı evrişimsel sinir ağları nasıl çalıştıklarını anlamak için ağın tüm yapısından bahsederken kullanılan standart katmanları ve aktivasyon fonksiyonlarını ortaya koyabilmektir.

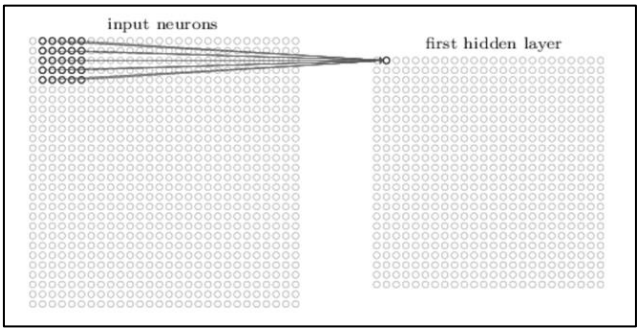
2.5.1 Evrişimsel Sinir Ağı Mimarisi

Evrişimsel sinir ağının genel yapısını açıklamadan önce yerel alıcı alanları ve paylaşılan ağırlık ve yanlılık (bias) kavramları üzerinde durmakta fayda vardır.



Şekil 2.6: İlk gizli katmandaki birinci bölgesel nöron karşılığı

Tipik bir sinir ağında, giriş katmanındaki her bir nöron, gizli katmanındaki bir nörona bağlıdır. Bununla birlikte, evrişimsel sinir ağında sadece giriş katmanda bulunan küçük bölgesel nöron, aşağına yer alan Şekil 2.6 ve Şekil 2.7'de [12] gösterildiği üzere gizli katmanındaki nöronlara bağlanır. Bu bölgelere ayrıca yerel alıcı alanlar denmektedir.



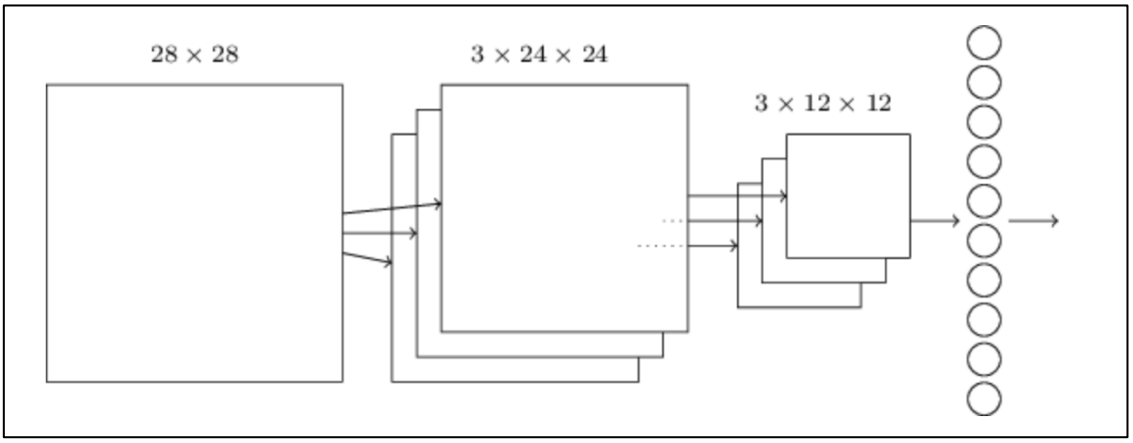
Şekil 2.7: İlk gizli katmandaki ikinci bölgesel nöron karşılığı

Şekillerde bulunan yerel alıcı alanları açıklanması gerekirse bunun için öncelikle 28x28 bir resimdeki nöron veya pixel temsil eden değerleri veri girişi olarak varsayalım. Birinci gizli katmandaki her bir nöron, giriş nöronlarının küçük bir bölgesine karşı düşmektedir. Örneğin, 25 giriş pikseline karşılık gelen 5x5 bir bölgeye bağlanacaktır (bkz. Şekil 2.6). Ardından ikinci bir gizli nörona bağlanmak için yerel alıcı alanı bir pikselden sağa kaydırmaktadır (bkz. Şekil 2.7). Bu işlem ilk gizli katmanı oluşuna kadar devam etmektedir.

Evrişimsel sinir ağında paylaşılan ağırlık ve yanlılıklar; herhangi standart bir ağ yapısında olduğu gibi bir evrişimsel sinir ağı, nöronların ağırlığı ve yanlılık değerleri şeklinde açıklanabilir. Model, eğitim sırasında bu değerleri öğrenir ve sürekli eğitim örneği ile güncellenir. Bununla birlikte evrişimsel sinir ağlarında bulunan katmanlardaki tüm gizli nöronlar da aynı ağırlık ve yanlılık değerlerine sahiptir. Bu da tüm gizli nöronların, görüntünün farklı bölgelerindeki kenar veya bir imge bölgesi (blob) gibi aynı özelliği algıladığı anlamına gelmektedir. Bu, ağın bir görüntüdeki nesnelerin çevirisine toleranslı olmasını sağlamaktadır. Örneğin, köpeği tanıyan bir ağ, köpeğinin görüntüde olduğu her yerde tanıma görevi yapabilmektedir. Bu açıdan baktığımızda giriş katmanından gizli katmana oluşturulan haritayı "özellik haritası"

olarak çağırmaktadır. Özellik haritasını tanımlayan ağırlıklara "paylaşılan ağırlıklar", yine özellik haritasını tanımlayan yanlılıklara ise "paylaşılan yanlılıklar" denilmektedir. Paylaşılan ağırlıklar ve yanlılıkların genellikle bir çekirdek veya filtreyi tanımladığını söylemek mümkündür.

Evrişimsel sinir ağı; giriş, ara ve çıkış katmalarını içermektedir. Ara katmanı ise genel olarak evrişimsel (convolutional), havuzlama (pooling) ve tam bağlı (fully-connected) katmanlardan oluşturmaktadır. Giriş katmanında x_1, x_2, ..., x_n, görüntü veya ses olabilecek verilerdir. Formal olarak, evrişim katmanına giriş verisinin M x M x C görüntüsüdür. M görüntünün yüksekliği ve genişliği ise M x M resimdeki piksel sayısı ve C ise piksel başına kanal sayısıdır. Gri tonlamalı görüntü için bir kanal $C = 1$ ancak *RGB* görüntüsü üç kanal $C = 3$'e sahiptir. evrişim katmanında, N x N x R boyutlu K filtreleri (çekirdeklerine) bulunacaktır. Burada 'N' filtrenin (çekirdeklerin) yüksekliği ve genişliğidir, 'R' ise 'C' görüntünün kanal sayısı eşit veya daha azdır; her filtre için farklı olabilmektedir [14] . Aşağıda yer alan Şekil 2.8'de [12] evrişimsel sinir ağı genel yapısı gösterilmektedir.



Şekil 2.8: Evrişimsel sinir ağı genel yapısı

Şekil 2.8'de, 28×28 pikseli bir görüntü yani ağının giriş katmanına 784 nöron ile beslenmektedir. Ardında, 5×5'lik yerel alıcı alan ve 3 filtre oluşturan bir evrişimsel katmanı gelmektedir. Evrişimsel katmandaki işlerden sonra bir 3×24×24 gizli özellikli nöronların katmanı elde edilmektedir. Bir sonraki adım, özellik haritalarının her birinde 2×2 bölgeye uygulanan bir havuzlama katmanı oluşturmaktadır. Bu katmanın

işlem sonu bir 3×12×12 gizli özellikli nöronların bir katmanı ortaya çıkmaktadır. Son katmanı, tam bağlı katman olarak tanımlamaktır. Bu katman, havuzlama katmandaki her bir nöronu bağlanmaktadır. Şekil 2.8'gösterişinde evrişimsel sinir ağının genel tanımına göre belirlenen parametreler $M = 28$, $N = 24$ ve $C = R = 3$ değerler almaktadırlar.

2.5.2 Katmanlar

- **Evrişimsel katmanı**

Evrişimsel katman, evrişimsel sinir ağının ana yapısı olarak düşünülebilir. Girdilerin özellerini algılamakta görevlidir. Yani resimdeki köşeler, kenarlar ve uç noktalarda göze çarpan nitelikleri ortaya çıkartan bir özellik çıkarıcısıdır. Bu özellik çıkartma işi de filtreleme işlemi uygulanarak gerçekleşmektedir. Örneğin Şekil 1.7'de, Evrişimsel katmanında 3 filtre uygulandığını görülmektedir. İlk filtre uygulandığında bir özellik haritası oluşturarak özellik türü tespit edilmektedir. Ardından, kullanılan diğer filtreler, başka özellik türlerini algılayarak yeni özellik haritaları elde etmektedir.

- **Havuzlama katmanları**

Evrişim işleminden sonra, evrişim katmanından küçük bloklar alarak tek bir çıktının üretilmesi için yapılan örneklemeden oluşan havuzlama katmanı gelir. Havuzlama katmanı, görüntünün çözünürlüğünü azaltarak çevirinin (kayma ve bozulma) etkisini azaltır. Birçok havuzlama işlemleri vardır ancak yaygın olarak maksimum havuzlama kullanılmaktadır. Diğer kullanılan çeşitli algoritmaları ise ortalama havuzlama veya bloktaki nöronların öğrenilmiş doğrusal bir kombinasyonlarıdır. Yine de Şekil 1.7'ye bakıldığında evrişim katmandan elde edilen sonuca *2x2* boyutunda bir filtreyi uygulandığında *3x12x12* sonucu bulunacağı görülmektedir.

- **Tam bağlı katmanı**

Havuzlama işleminden sonra 'Tam Bağlı Katmanı' bir gelmektedir. Tamamen bağlı katmanı, önceki katmandaki tüm nöronları alarak sahip olduğu her nörona bağlamaktadır. Tamamen bağlanmış katmanlar artık boşlukta lokalize olmadığından, tam bağlı bir katmandan sonra herhangi bir evrişim katmanı olamaz.

2.6 Yinelemeli Sinir Ağı (Recurrent Neural Network - RNN)

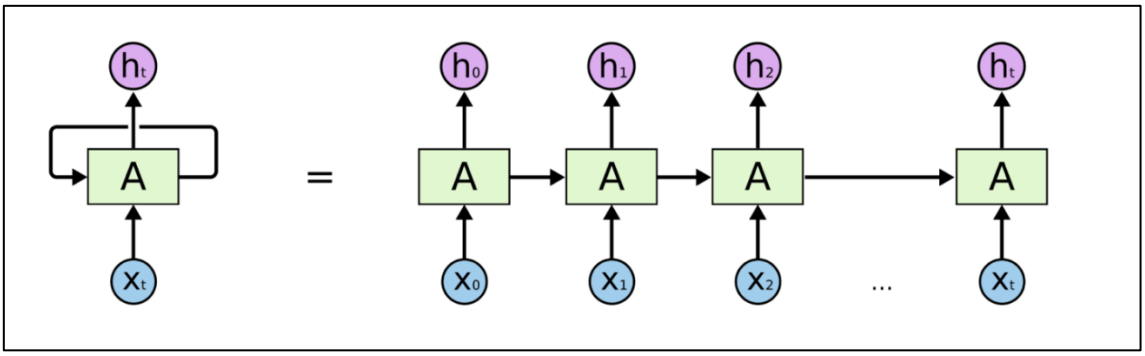
Bu kısmında RNN modelleri ve özellikle Uzun Kısa Süreli Bellek olan ağının çalışma mantığından bahsedilmiştir.

2.6.1 Yinelemeli Sinir Ağı Tanımı ve Yapısı

Yinelemeli Sinir Ağları (RNN'ler), gizli düğümler arasındaki döngüsel bağlantılara sahip olan sinir ağlardır. Bu döngüsel bağlantılar, RNN'leri hafızayı kodlama yeteneği sunar ve bu tür ağlar, başarılı bir şekilde eğitilmişse, sıralı öğrenme uygulamaları için uygundur [15].

Yinelemeli Sinir Ağları, metin dizileri, genomlar, el yazısı, konuşulan sözcük veya sensörler, borsalar, hava durumu ve devlet kurumlarından kaynaklanan dijital veri serilerinin tanıması için tasarlanan yapay sinir ağı türüdür[16]. Bu tür ağları kendi aldıkların veri girdileriyle beslenmektedir.

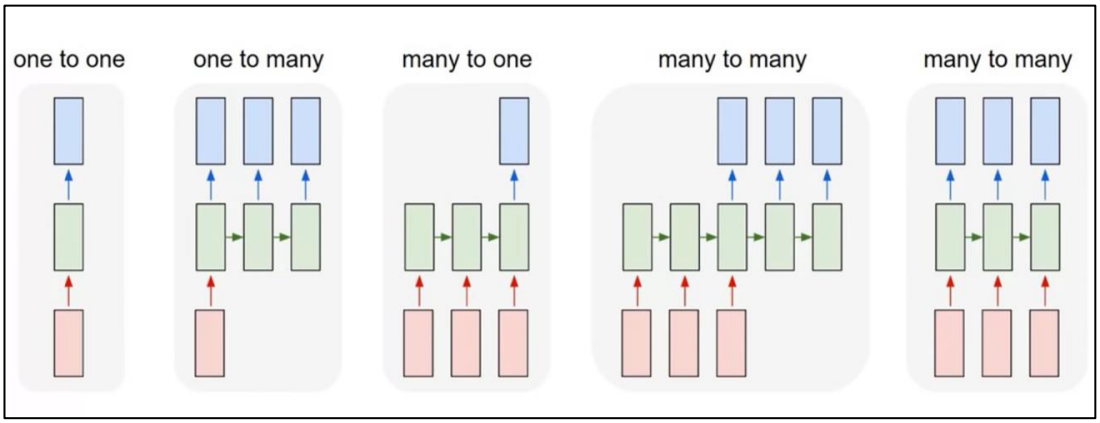
Veriler ağa girildiğinde, alınan çıkışı tekrar giriş olarak kullanılmaktadır. Şekil 2.9'da [17] gösterildiği üzere bu uygulama, sinir ağında bir döngü oluşturarak hafızanın korunmasına yardımcı olmaktadır. Bu, dijital veri serilerini tahmin etmek için kullanılmaktadır ki, böylelikle önceki çıktı veya önceki girdi, bir sonraki çıktıyı tahmin etmede çok iyi bir ağırlık yaşına sahiptir.



Şekil 2.9: RNN giriş döngü şeması

Yinelemeli Sinir Ağlarının farklı bir biçimi ya da yapıları (bkz. Şekil 2.10) vardır [18]. Belirli bir amaç için veya uygun bir durum için kullanılabilmektedir. Örneğin bire bir (1-1) yapısı, resim sınıflandırma; bire çok (1-N) yapısı, resimdeki

yazısı; çoğa bir (N-1) biçimi, tanıma eylemleri; Çoğa çok (N-N) yapısı ise dil çevirisi veya video özetleme için kullanılmaktadır.

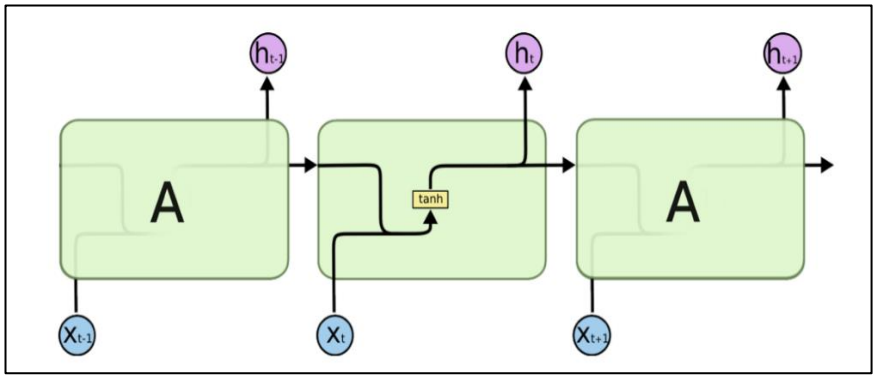


Şekil 2.10: RNN çoklu yapılar

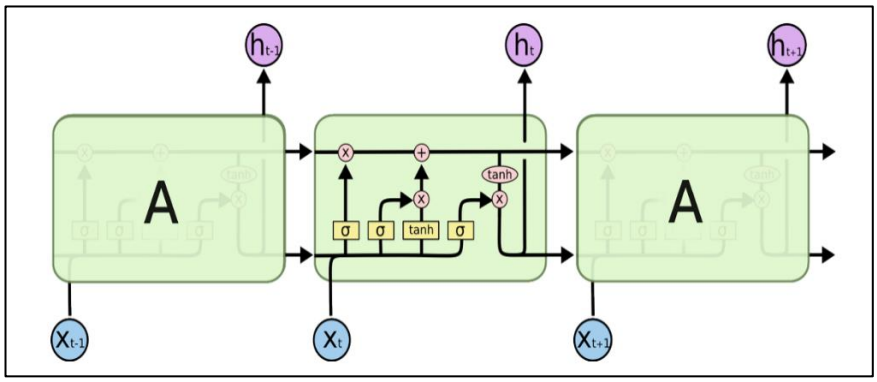
Bazen derin öğrenme problemlerinin çözülmesine rağmen, Yinelemeli Sinir Ağları, sistemi eğitmek için zaman boyunca geri yayılımı (Backpropagation Through Time (BTT) olarak bilinen geri yayılım algoritmasının kullanılmasından dolayı ufukta gradyanı, patlatma gradyanı ve uzun süreli bağımlılık problemleri gibi sorunlar ortaya çıkabilmektedir. Bu sorunların üstesinden gelmek için, Kesik (Truncated BTT), eşikteki Klip gradyanı, öğrenme hızını ayarlamak için RMSprop, ReLU aktivasyon fonksiyonu, LSTM'ler ve GRU'lar gibi bazı çözümler uygulanmaktadır. Bu çözümlerden, Yinelemeli Sinir Ağı tekniğinde en çok uygulanan LSTM'nin daha ayrıntılı olarak açıklanması hedeflenmektedir.

2.6.2 LSTM Ağları

LSTM'nin açılımı Long Short Term Memory olup Türkçe ifade ile "Uzun Kısa Süreli Bellek Ağları" anlamına gelmektedir. Yinelemeli Sinir Ağıların standartlarında, ağlar tek bir sinir ağ tanh katmanı basit bir yapı ile tekrar eden bir modül oluşturmaktadır (bkz. Şekil 2.11 [17]). Fakat LTSM'lerde tekrar eden modül, belirli bir şekilde dört etkileşimli sinir ağı olup karmaşık bir yapıya sahiptir (bkz. Şekil 2.12 [17]).



Şekil 2.11: Tek katmanlı standart RNN

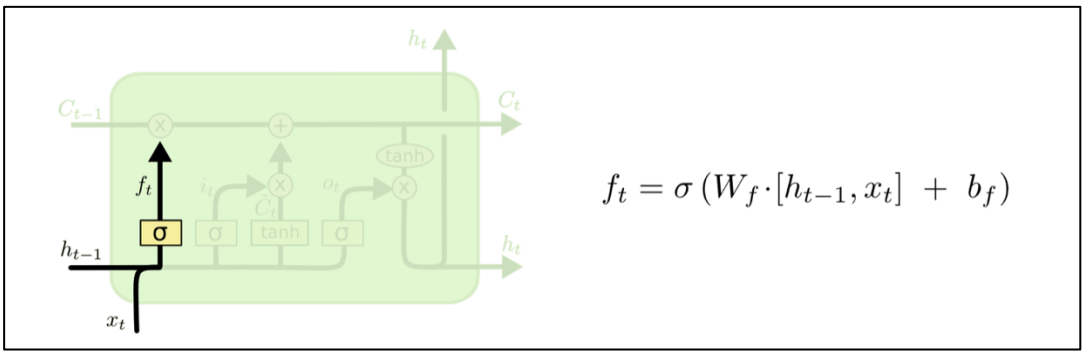


Şekil 2.12: Dört etkileşim katmanı olan LSTM'ler

LSTM işlemi dört adımda gerçekleşmekte. LSTM'lerin nasıl çalıştığını anlamak için, bu adımları incelemesi gerekmektedir:

- **Birinci adım**

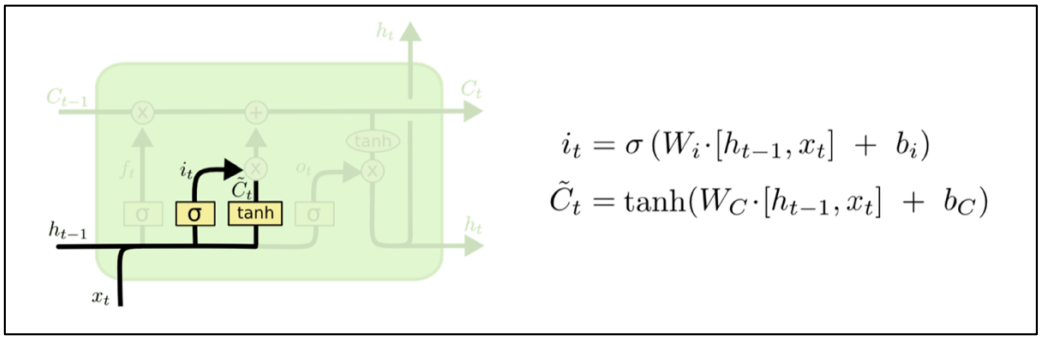
Bu adımda, ağ hangi bilginin gerekli olmadığını ve hücreden atılacağını belirlemektedir. Bu kararı vermek için, unutma geçidi katmanı (forget gate layer) adı verilen sigmoid fonksiyon katmanı aşağıdaki yer alan Şekil 2.13'te [17] gösterildiği gibi kullanılmaktadır.



Şekil 2.13: Unutma geçidi katmanı (Forget Gate Layer)

- **İkinci adım**

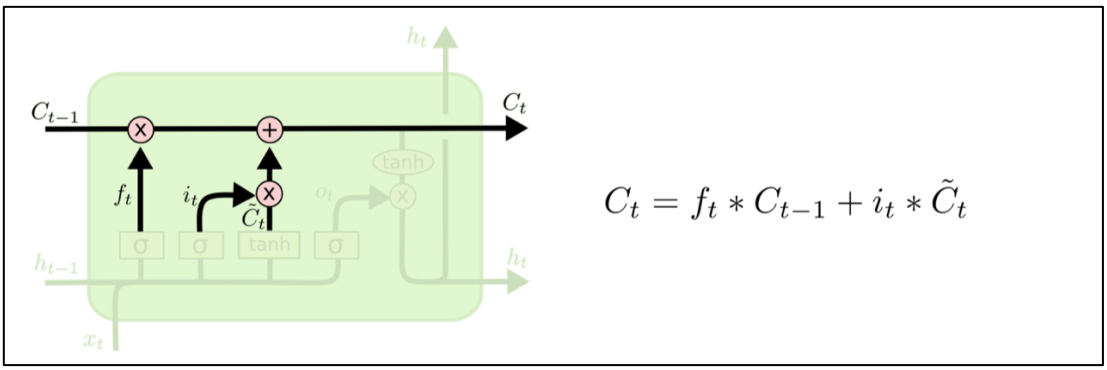
Burada, ağ hangi bilginin hücre durumuna (cell state) depolanacağına karar verilmektedir. Bütün bu süreç takip adımlarından taviz vermektedir. Hangi girişlerin güncelleneceğini "giriş kapısının katmanı(input gate layer)" olarak adlandırılan bir sigmoid katmanı belirlemektedir. Ardından, duruma eklenebilecek yeni aday değerlerin bir vektörünü oluşturan *tanh* katmanı Şekil 2.14'te [17] gösterildiği gibi kullanılmaktadır.



Şekil 2.14: Giriş geçidi ve yeni aday değer vektörü (Input Gate and New Candidate Value Vector)

- **Üçüncü adım**

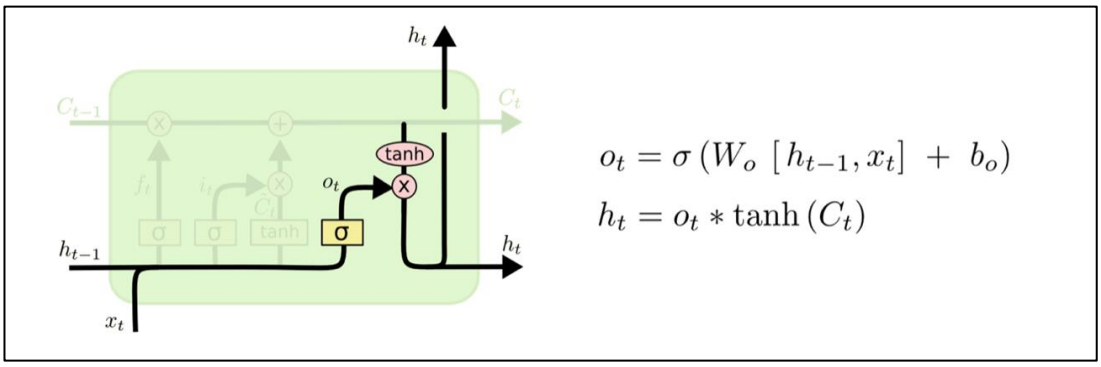
Bu kısmında, eski hücre durumu C_{t-1}, yeni hücre durumu C_t 'ye güncellenmektedir. İlk olarak, eski durumu (old state), daha önce unutmaya karar verilen bilgileri unutarak f_t ile çarpılmaktadır. Daha sonra, sonuçların her bir durumunun değerinin güncellemeye karar vermesiyle ölçeklenen yeni aday değerlerini veren $i_t * C_t$ eklenmektedir (bkz. Şekil 2.15[17]).



Şekil 2.15: Yeni hücre durum (New Cell State)

- **Dördüncü adım**

Bu son adımda, hücre durumunun (cell state) hangi bölümlerinin çıktı olacağına karar vererek sigmoid katmanı çalıştırılmaktadır. Daha sonra, hücre durumu *tanh*'tan geçirilerek sigmoid kapısının çıktı ile çarpılmaktadır ve böylece sadece ihtiyaç duyulan parçaları çıktı olarak verilecektir (bkz. Şekil 2.16 [17]).



Şekil 2.16: Çıkış geçidi ve yeni bilgi (Output gate and New Candicate Value)

2.7 İlgili Çalışmalar (Related Works)

Günümüzde, İngilizce, Arapça, Çince, Devanagari, Telugu vb. farklı dillerin el yazısının tanıması için çeşitli teknikler geliştirilmiştir ve yüksek tanıma oranları elde edilmiştir. Derin Sinir Ağı, Evrişimsel Sinir Ağı ve Yinelemeli Sinir Ağı gibi farklı yöntemler kullanarak yüksek orandaki sonuçlar elde edilmiştir. Son yıllarda, birçok uygulama belirtilen yöntemleri kullanmaktadır. Ruben vd [19] Yinelemeli Sinir Ağları (RNN'ler) temelli yeni bir yazar-bağımlı çevrimiçi imza doğrulama sistemleri

önermiştir. Sistemlerini iki RNN algoritması, Uzun Kısa Dönemli Bellek (LTSM) ve Siyam mimarisine sahip Gated Recurrent Unit (GRU) kullanarak tasarladılar. Deneysel çalışmalarında ise sistem performansının eşit hata oranları (% EER) cinsinden elde edilmesi için 11200 imzası olan BiosecureID veri tabanını kullanmışlardır [19].

Ahmet vd. [14] evrişimsel sinir ağı (CNN) kullanarak Arapça el yazısının karakterlerinin tanıması bir çalışma yayınlanmışlardır. El yazısıyla yazılmış Arapça karakterlerin oluşturulan veri kümesi, eğitim için 13440 görüntü ve test edilmesine yönelik 3360 görüntü içeren 16800 görüntüye sahip oldukları bir eğitim ve test veri setini hazırlanmıştır. Onların yapmış oldukları tanıma oranları % 94.9'dur [14]. Durjoy vd. [20] çalışmalarında, standart bir 50-sınıf Bangla temel karakter veri tabanını tanımak için bir CNN eğitimi aldıkları Çoklu Komutların El ile Yazılan Karakter tanımalarına Bir CNN Tabanlı Ortak Yaklaşım sunulmuşlardır ve aynı zamanda İngilizce, Devanagari, Bangla, Telugu ve Oriya 'nın 5 farklı 10-sınıf dijital tanıma için özellikler çıkarılmışlardır. Bu çalışmayı yapmalarındaki amaç, herhangi bir karakter tanıma konusuna uygulanabilecek bir özellik çıkarma stratejisi uygulamaktı. Deneylerinin sonuçları, Bangla temel karakterleri için % 95.6, Bangla dijital için % 98.375, Devanagari dijital için % 98.54, Oriya dijital için % 97.2, Telugu dijital için % 96.5 ve % 99.10 İngilizce dijital oranları elde etmişlerdir [20]

Dan Cireşan vd.[21] Alman trafik işareti tanıma kriterlerini kazandıran trafik işaretleri sınıflandırması için birden çok sütun derin sinir ağ üzerinde çalışmışlardır. Çalışmalarında, resimlerin 48 x 48 piksele yeniden boyutlandırıldığı tek bir trafik işaretinin bulunduğu resimler içeren veri tabanı hazırlamışlardır. Bu veri tabanı, eğitim için 39209 görüntü ve test için 12630 görüntüden oluşmaktadır. Girdi verilerini eğitmek için, tek bir GPU tarafından 87 görüntünün işlendiği dört Grafik İşlem Birimi (GPU) üzerinde 25 sütun içeren bir sistem oluşturmuşlardır. Veri kümeleri için 5 tane rastgele başlatılmış ağ ile 25 ağ eğitmişlerdir. Alman Trafik işareti tanıma kriterlerini kazanmalarını sağlayan % 99.46'lık bir ortalama tanıma oranına sahiplerdir [21].

Xu Yao vd.[22] Yinelenen Sinir Ağı (RNN) kullanarak, uçtan uca sisteme dayanan bir sistem, hem Çince karakterlerini tanıma hem de çizim için ayırıcı ve üretken modeller

olarak sunmuştur. Çerçeve sıralı yapı ile başa çıkmak için tasarlanmıştır ve herhangi bir etki alanına özel bilgi gerektirmez. Fark gözeten modelinde, çift yönlü RNN tekniği, tanıma işlemi için hem LSTM hem de GRU ile bütünleşmiştir. Çince karakterler üretirken, farklı el yazısı stilleri oluştururken çeşitliliği garanti eden kalem yönünü modellemek için Gaussian karışım modelini (GMM) kullanmışlardır. Eğitim ve test için, çevrimiçi el yazısıyla yazılmış Çince karakter tanıma yönteminin ICDAR-2013 Competition veri tabanını kullanmışlardır. Deneyim sonunda ise % 95.19 - % 98.15 arasında değişen farklı veri kümesi için ölçüm sonuçlarını almışlar [22].

Graves vd. [23] çok boyutlu Yinelemeli Sinir Ağları ile çevrimdışı el yazısı tanıma sunmuştur. Geliştirilen çerçeve ham pikselleri girdi olarak alır ve herhangi bir alfabeye özgü ön hazırlığı gerektirmeyecek şekilde tasarlanmıştır ve herhangi bir dil değiştirmeden kullanılabilmektedir. Oluşturdukları sistem, güçlü bir çevrimdışı el yazısı tanıyıcı oluşturmak için çok boyutlu Uzun-Kısa Vadeli Belleği (LSTM) ile bağlantıyı geçici bir sınıflandırma ve bir hiyerarşik yapıyı birleştirmiştir. Sisteme İngilizce ve Arapça veri tabanı uygulanarak harika sonuçlar elde etmişlerdir. Örnek olarak, Uluslararası Arapça tanıma Yarışması'nda sistem % 91.4 doğruluğu vermiştir. Aslında, ağların boyutsallığı uygulanan verilere uyum sağlamak için değiştirilebilir ve böylece çerçeve herhangi bir denetlenen dizi etiketleme görevi için kullanılabilir [23].

3. DENEYSEL ÇALIŞMALAR VE BULGULAR

Cifar-10, Fashion-Mnist, Mnist ve Arapça Kullanılan veri kümeleri; doğruluk ve kayıp performans metrikleri; tensorBoard, tensorflow ve keras kullanılan araçlar ve kütüphaneler; DNN, CNN ve RNN oluşturulan modeller ve hiper-parametre ayarlamaları; veri setlere göre algoritmaların karşılaştırılması; benzer çalışmalarla sonuçlarımız kıyaslaması ve bulgurlar detaylı ile bilgiler sunulmuştur. Algoritmalar ilk adımlarda CPU, intel i5 ve 8 RAM'lı bir MAC bilgisayar çalıştırılmıştır. Bazı modellerin eğitimleri daha uzun süre sürdüklerinden **NVIDIA GeForce GTX TITANX** 12 GB Hafıza, GDDR5 tipinde bir GPU'lu makine üzerinde çalıştırılmıştır.

3.1 Kullanılan Veri Kümeleri

Çalışmada kullanılan veri kümeleri ve özellikleri bu bölümde verilmiştir.

3.1.1 Cifar-10 Veri Kümesi

Cifar-10, küçük resim alt kümeleri olarak etiketlenmiş bir veri setidir. Veri seti, Alex Krizhevsky, Vinod Nair ve Geoffrey Hinton tarafından toplanmışlardır.

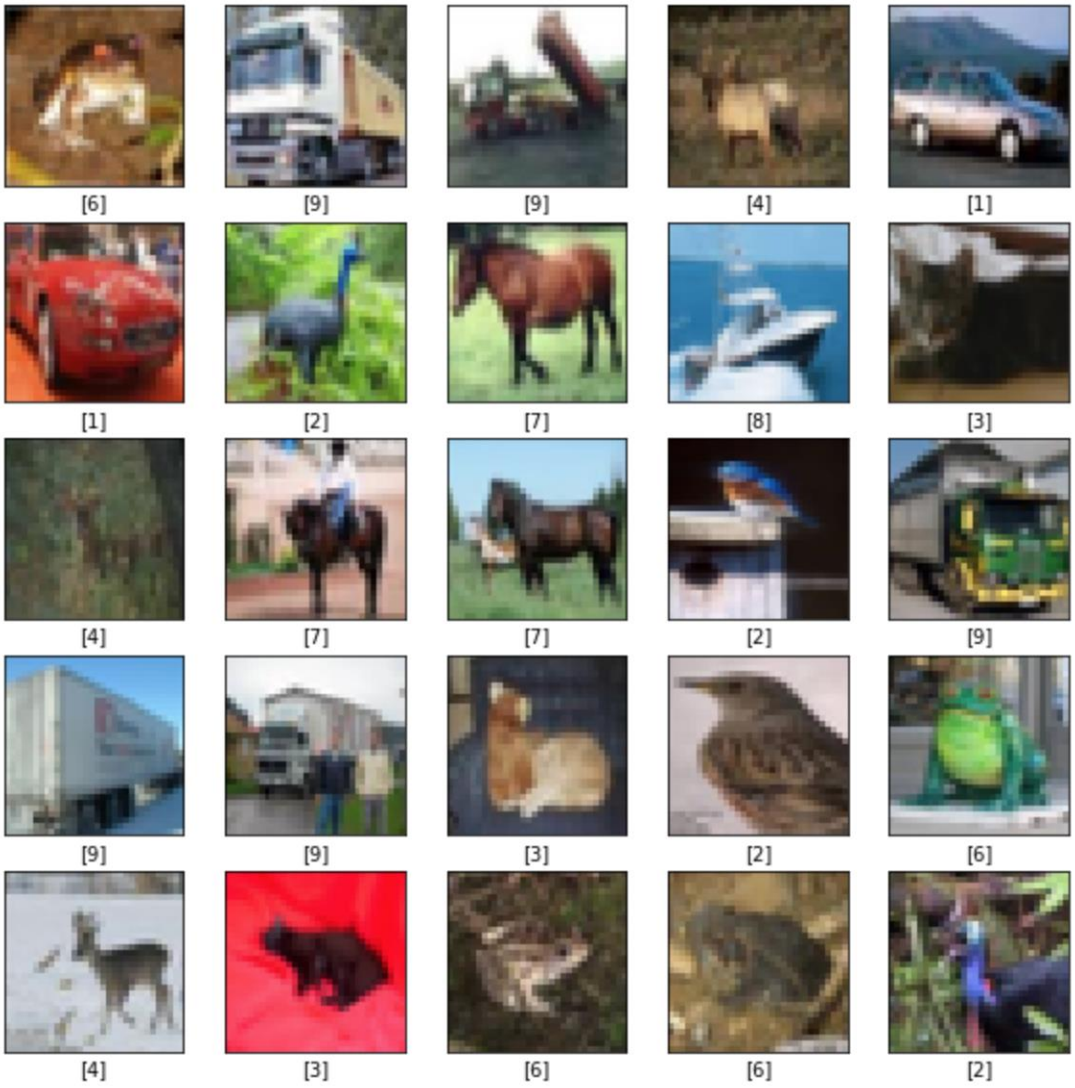
CIFAR-10 veri seti, sınıf başına 6000 görüntü içermekte, 10 sınıf olan ve toplamda 60000 32x32 renkli görüntüden oluşmaktadır. 50000 eğitim görüntüsü ve 10000 test görüntüsü vardır.

Veri kümesi, her biri 10000 görüntü içeren beş eğitim (data_batch_1, data_batch_2,…, data_batch_5) ve bir test (test_batch) kümesine ayrılmıştır. Test kümesi, her sınıftan tam olarak rastgele seçilen 1000 görüntü içermektedir. Eğitim kümeleri kalan görüntüleri rastgele sırayla oluşmaktadır, ancak bazı eğitim veri setleri bir sınıftan diğerine göre daha fazla görüntü içerebilmektedir. Aralarında, eğitim kümeleri her bir sınıftan tam olarak 5000 görüntü içermektedir[24]. Aşağıdaki bulunan Şekil 3.1'de veri kümesinden rastgele uçak, araba, kuş, kedi, geyik, köpek, kurbağa, at, gemi ve kamyon resimleri görülmektedir.

3.1.2 Fashion-Mnist Veri Kümesi

Fashion-Mnist, Zalando [25]'nun 60000 örnek eğitim setinden ve 10000 örnek test setinden oluşan makale görüntülerinin bir veri kümesidir. Zalando, Berlin

merkezli bir Avrupa e-ticaret şirketidir. Şirket, ayakkabı, moda ve güzellik ürünleri satan çapraz platformlu bir çevrimiçi mağazaya sahiptir. O şirketin resim ürünlerinden



Şekil 3.1: Cifar-10 veri setinden görüntü örnekleri

bir veri seti oluşturulmuştur. Veri seti içeren örnekler, 10 sınıftan etiketle ilişkilendirilmiş 28 × 28 gri tonlamalı görüntülerdir. Fashion-Mnist, makine öğrenme algoritmalarının kıyaslama için orijinal Mnist veri setinin doğrudan düşürülmesi yerine hizmet etmeyi amaçlamaktadır[25]. Bu veri seti, ['Tişört / üst', 'Pantolon', 'Kazak', 'Elbise', 'Ceket', 'Sandal', 'Gömlek', 'Sneaker', 'Çanta', 'Ayak Bileği Çizme']10 farklı sınıf veya kategorilerden oluşturmaktadır. Şekil 3.2'de anlaşılacağı üzere,

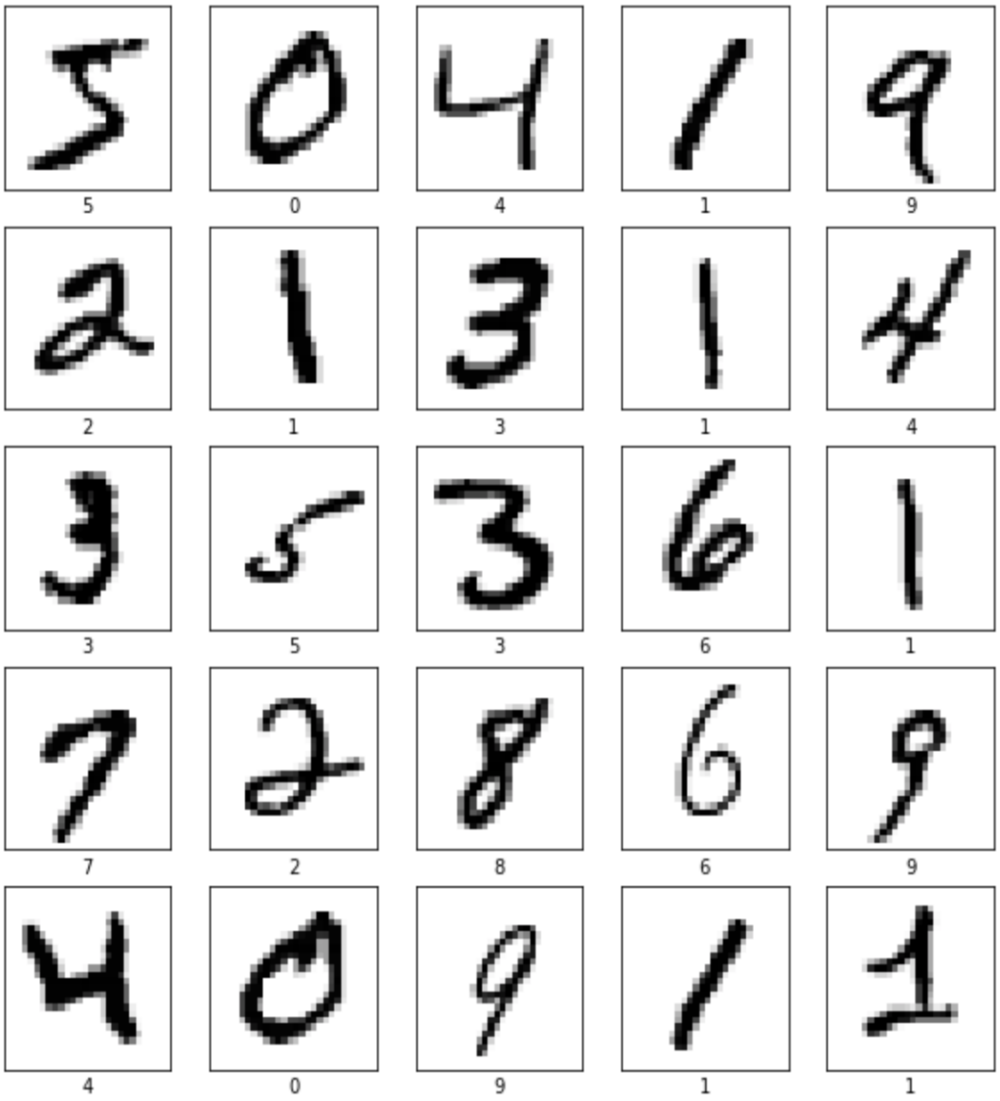
Fashion-Mnist veri setinden her sınıftan rasgele resimlerin genel bakışını gösterilmektedir.

Şekil 3.2: Fashion-Mnist veri setinden görüntü örnekleri

3.1.3 Mnist Veri Kümesi

Mnist veri tabanı açılımı "Modified National Institute of Standards and Technology Database" veya Türkçe olarak "Ulusal Standartlar ve Teknoloji Enstitüsü Veri Tabanı" İngilizce açılımın kelimelerinin baş harflerinden oluşan bir isimdir, çeşitli görüntü işleme sistemlerini eğitmek için yaygın olarak kullanılan el yazısı rakamlarından oluşan geniş bir veri tabanıdır. MNIST veri tabanı, Amerikan Sayım Bürosu çalışanlarından 60000 eğitim görüntüsü ve Amerikan lise öğrencilerinden

alınan 10000 test görüntüsünü içermektedir[26]. Ayrıca bu veri tabanı; yapay zeka, derin öğrenme, makine öğrenmesi alanlarında eğitim ve test yapmak için yaygın olarak kullanılmaktadır. Şekil 3.3'te bu veri tabandaki bazı örnekler gösterilmektedir.



Şekil 3.3: Mnist veri setinden görüntü örnekleri

Mnist veri tabanı NIST (National Institute of Standards and Technology)'in Özel Veri Tabanı 3 ve el yazısı rakamların ikili resimlerini içeren Özel Veri Tabanı 1'den oluşturulmuştur. NIST başlangıçta eğitim seti olarak SD-3'ü ve test seti olarak SD-1'i belirlemiştir. Ancak, SD-3 SD-1'den daha temiz ve kolay anlaşılmıştır. Bunun nedeni, SD-3'ün Sayım Bürosu (Census Bureau) çalışanları arasında, SD-1'in lise öğrencileri arasında toplanması gerçeğine dayanmıştır. Öğrenme deneylerinden mantıklı sonuçlar

çıkarmak, sonucun eğitim seti seçiminden bağımsız olmasını ve tüm örneklem grubunda test yapılmasını gerektirmektedir. Bu nedenle, NIST'in veri kümelerini karıştırarak yeni bir veri tabanı oluşturmasını ihtiyaç duyulmuştur [26].
MNIST eğitim seti, SD-3'ten 30000 kalıp ve SD-1'den 30000 kalıptan oluşmaktadır. Test setimiz SD-3'ten 5000 desen ve SD-1'den 5000 desenden oluşmuştur. Bu 60000 örüntü eğitim seti, yaklaşık 250 yazardan örnekler içermiştir. Eğitim seti ve test setinin yazar setlerinin birbirinden ayrık olduğundan emin olunmuştur [26].

3.1.4 Arapça Veri Kümesi

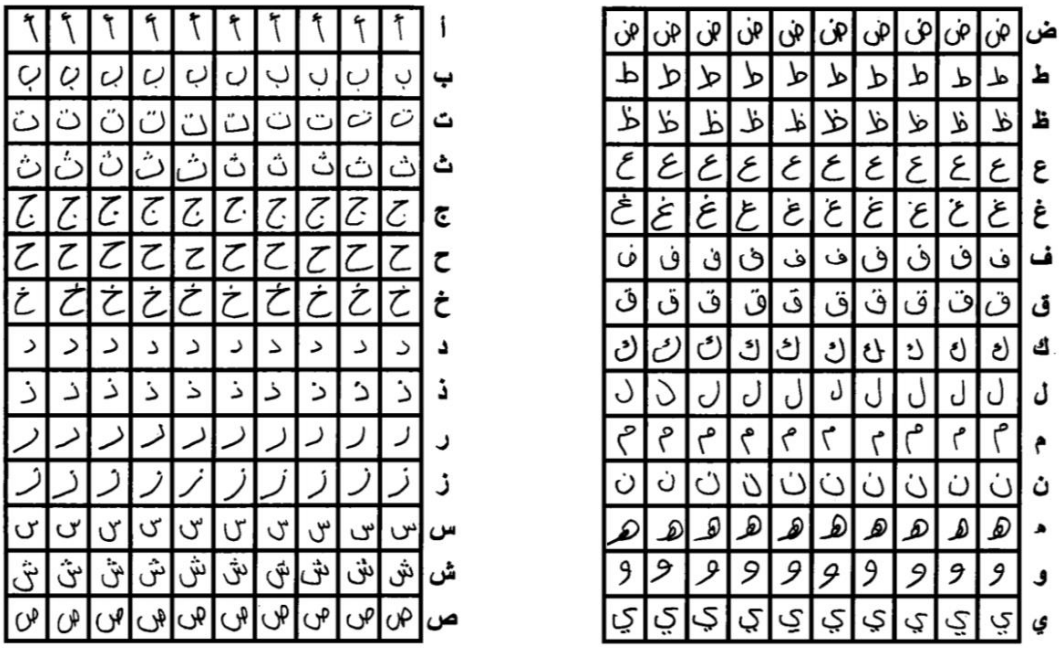
Arapça, Orta Doğu ülkelerinde milyonlarca insanın ana dili olarak kullanılan bir tür semitik dildir. Genel olarak, Arap alfabesi karakterleri, Şekil 3.4'de [27] gösterilen yirmi sekiz adet alfabe karakterinden oluşmaktadır.

خ	ح	ج	ث	ت	ب	أ
khah	hah	jeem	theh	teh	beh	alef
ص	ش	س	ز	ر	ذ	د
sad	sheen	seen	zain	reh	thal	dal
ق	ف	غ	ع	ظ	ط	ض
qaf	feh	ghain	ain	zah	tah	dad
ي	و	ه	ن	م	ل	ك
yeh	waw	heh	noon	meem	lam	kaf

Şekil 3.4: Arap alfabesi karakterleri

Araştırmamız için Ahmet vd.[14]'nın mevcut Arapça veri setini kullanmaktadır. Onların hazırlandıkları veri seti 60 katılımcı tarafından yazılan 16.800 karakterden oluşmakta olup, yaş aralığı 19 ile 40 yıl arasındadır ve katılımcıların% 90'ı sağ elini kullananlardır. Her katılımcı, aşağıda Şekil 3.5'te [27] gösterildiği gibi her bir karakteri (' alef'den 'yeh' e) 10 kez yazdırılmıştır. Her blok Matlab 2016a kullanılarak

otomatik olarak bölünmüştür. Veri tabanı iki kümeye ayrılmıştır: bir eğitim seti 13440 karakter ve bir test seti 3360 karakter oluşmaktadır.



Şekil 3.5: Arapça veri seti oluşumunda

Bu veri seti, 28 sınıftan bir etiketle ilişkilendirilmiş 32x32 gri tonlamalı bir görüntüdür. Çalışmamızda csv formatına dönüştürülmüş hali kullanılmıştır.

3.2 Performans Metrikleri

Bu çalışmada oluşturulan modellerin değerlendirilmesi iki önemli performans metrikleri olan doğruluk ve kayıp hesaplanması gerekmektedir.

Öncelik olarak kayıp değerini elde etmek için modelleri derlenmektedir. Kayıp, öngörülen çıktı ile beklentideki gerçek çıktı arasındaki fark veya sapma nicel ölçümüdür. Bize, çıktının öngörülmesinde ağ tarafından yapılan hataların ölçüsünü vermektedir[28]. Başka bir deyişle, kayıp değeri, modelin test aşamasında ne kadar iyi çalıştığının bir ölçüsüdür. Düşük kayıplı bir değer, modelin iyi olduğu anlamına gelmektedir. Bu çalışmada, kayıp fonksiyonu olarak ikili çapraz entropi (binary cross entropy) ve seyrek kategorik çapraz entropi (sparse categorical entropy) kullanılmıştır.

- **İkili çapraz entropi**

İkili çapraz entropi, veri etiketlerin 0 veya 1 değerlerini aldığı varsayılan fonksiyondur. Çapraz entropi, iki olasılık dağılımı arasındaki farkı ölçer, eğer çapraz entropi büyükse iki dağıtım arasındaki fark büyüktür, çapraz entropi küçük ise, bu iki dağılımın birbirine benzer olduğu anlamına gelmektedir. İkili çapraz entropi hesaplanabilmesi için Denklem 3.1'de verilmiştir.

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n}(\, y_i \log(\widehat{y_i}) + \, (1 - y_i)\log(1 - \, \widehat{y_i})) \qquad 3.1$$

- w, model parametreleri, sinir ağının yanlıkları veya ağırlıkları gibi
- y_i , gerçek etikettir
- $\widehat{y_i}$, öngörülen etikettir

- **Seyrek kategorik çapraz entropi**

Seyrek kategorik çapraz entropi, sınıf sayısı 2'den büyük olduğunda çapraz-entropinin tanımıdır. Değerini hesaplanması için Denklem 3.2'te tanımlanmıştır.

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n}(\, y_i \log(\widehat{y_i})) \qquad 3.2$$

Kayıp değerini elde etmek için, iyileştiriciye ihtiyaç vardır.

Uygulamamızda, Bölüm 2.2.2 ifade edilen Adam, Rmsprop, Sgd iyileştiricileri uygulanmıştır[36]. Kullanılacak olan eniyileme algoritması belirlendikten sonra, 'fit' metoduyla, tüm veri setinin ne kadar süre boyunca eğitileceğini belirten kesin dönemleri (epoch) kullanarak eğitme için çağrılmaktadır. Modellerin sonuçlar değerlendirebilmesi için her dönemden sonra model.evaluate metodu çağrılması gerekmektedir. Bu metot doğruluk ve kayıp değerleri vermektedir. Doğruluk hesaplaması için Denklem 3.3'te anlaşılacağı üzere, doğru öngörülen sınıf bölüm toplam test sınıfıdır.

$$Doğrulama = \frac{Doğru\ \ öngörülen\ sınıfı}{Toplam\ test\ sınıfı} * 100 \qquad 3.3$$

3.3 Kullanılan Araçlar ve Kütüphaneler

Veri setlerimiz işlenmesi, modellerimiz oluşturulması, eğitmesi ve test edilmesi Python programlama dili ile gereken ortam ve kütüphaneler

kullanarak kodlanmıştır. Burada, ortam olarak anaconda navigatör'dan jupyter notebook (6.0.0) sürümlü, kütüphane ise TensorFlow ve Keras kullanılmıştır. Modellerimizi eğitmek için, ilk yapılması iş veri setleri ayrılmasıdır. Çalışmamızda (x_train, y_train) veya (train_images, train_labels) eğitim, eğitim setlerinin belli bir kısmı doğrulama ve (x_test, y_test) veya (test_images, test_labels) test için ayrılmıştır. Bu çalışmada kullanılan Fashion-Mnist ve Mnist veri setleri her biri 60000 eğitim ve 10000 test, Cifar-10; 50000 eğitim ve 10000 test, Arapça veri seti ise 13440 eğitim ve 3360 test görüntüleri içermektedir.

Eğitim görüntüler, keras kütüphanesinden validation metoduyla ikiye ayrılmıştır. Cifar-10, Fashion-Mnist ve Mnist veri kümeleri %80 eğitim ve %20 doğrulama, Arapça veri kümesi ise %90 eğitim ve %10 doğrulama olarak ayrılmıştır.

3.3.1 Tensorflow

TensorFlow, araştırma ve üretim için açık kaynaklı bir makine öğrenme kütüphanesidir. Masaüstü, mobil, web ve bulut bilişim geliştirmek için yeni başlayanlar ve uzmanlar için API'ler sunmaktadır [29]. Makine öğrenme ve derin sinir ağları araştırması için Google'ın makine zeka araştırma teşkilatı içerisinde Google Beyin Ekibi tarafından geliştirilmiştir. Sistem esnektir ve derin sinir ağı modelleri için eğitim ve çıkarım algoritmaları dahil olmak üzere çok çeşitli algoritmaların ifade etmek için kullanılabilir ve araştırma yapılabilmektir. Konuşma tanıma, bilgisayar vizyonu, robotik, bilgi alma, doğal dil işleme, coğrafi bilgi çıkarma ve bilgisayarlı ilaç keşfi dahil olmak üzere bir düzineden fazla bilgisayar bilimi ve diğer alanlarda araştırma yapmak ve makine öğrenme sistemlerinin üretiminde yerleştirmek için kullanılmıştır [30]. Tensorflow kullanması, yüklemesi, hangi API'leri içerdiğini ve gereken öğretim bilgileri kendi web sitesinden bulunmaktadır. TensorFlow kütüphaneleri ile eğitilen modellerin görselleri izlemek için Tensorboard kullanılmaktadır.

TensorBoard, TensorFlow çalıştırılmış modelinin verileri ve nasıl davrandığını görselleştirebilmek, grafikleri incelemek ve anlamak için olanak tanıyan bir yardımcı

programdır. TensorBoard, skalerler, grafik, dağıtımlar, histogramlar, projektör, resim, ses ve metin farklı görünümler, farklı formatların girişlerini alınarak farklı şekilde görüntülemektedir.

- **Skaler** - Sınıflandırma doğruluğu veya hata oranı gibi skaler değerleri görselleştirmektedir.
- **Grafik** - Sinir ağı modeli gibi modellerin hesaplama grafiğini görselleştirmektedir.
- **Dağılımlar** - Bir sinir ağının ağırlıkları gibi verilerin zaman içinde nasıl değiştiğini gözünüzde canlandırabilmesidir.
- **Histogramlar** - Dağılımları 3 boyutlu bir perspektifte gösteren dağılımın meraklısı görüntüsüdür.
- **Projektör** - Kelime gömme işlemlerini görselleştirmek için kullanılabilmektedir. (yani kelime gömme, anlamsal ilişkilerini yakalayan kelimelerin sayısal temsilidir).
- **Resim** - Resim verilerini görselleştirmektedir.
- **Ses** - Ses verilerini görselleştirmektedir.
- **Metin** - Metin (dize) verilerini görselleştirmektedir.

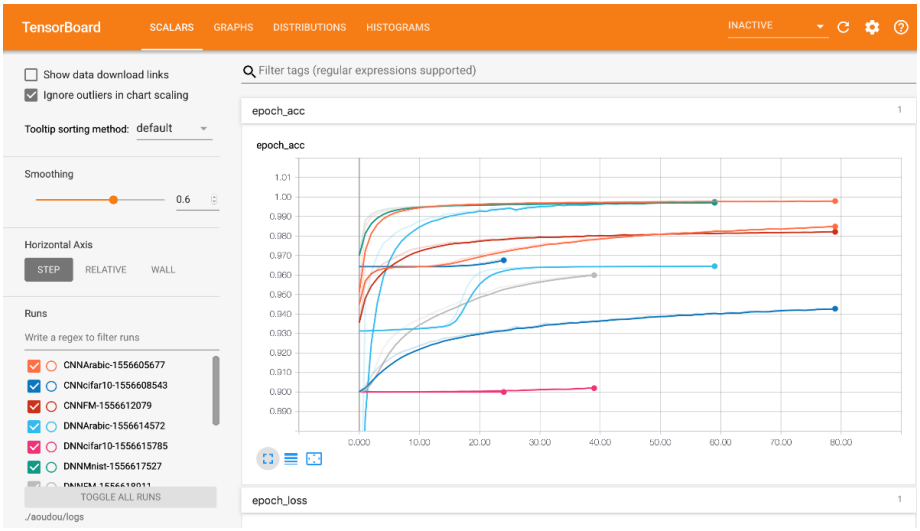
TensorFlow'da büyük bir derin sinir ağını eğitmek için kullanacağı hesaplamalar oldukça karmaşık ve kafa karıştırıcı olabilmekte, TensorBoard bunu TensorFlow programları anlaması, hata ayıklaması ve optimize etmesi çok daha kolay hale getirecektir[31], [32].

Şekil 3.6'da görsel kurduğumuz tensorboard ara yüzünü gösterilmektedir.

3.3.2 Keras

Keras Python ile yazılmış açık kaynaklı ve üst düzey bir sinir ağı kütüphanesidir (API). Tensorflow, Microsoft Cognitive Toolkit (CNTK) ve Theano üzerinde çalışabilmektedir. Derin sinir ağları ile hızlı deneyler sağlamak için tasarlanmıştır. Ana fikri, kullanıcı dostu, modüler ve genişletilebilir olmaktır. Çeşitli geliştiriciler tarafından geliştirilerek ilk sürümü Mart 2015'te sunulmuştur. Hem Merkezi işlem birimi (Central Processing Unit - CPU) hem de Grafik İşlem Birimi

(Graphics Processing Unit - GPU) üzerinde çalışıp birçok algoritmayı desteklemektedir [33].



Şekil 3.6: Tensorboard arayüzü

Keras, Tensorflow ile daha ayrıntılı yöntemleri kullanmak zorunda kalmadan, derin öğrenme modelleri oluşturmasını kolaylaştıran basit ve harika bir kütüphanedir.

3.4 Oluşturulan Modeller ve Hiper-Parametre Ayarlamaları

Bu kısmında modellerin oluşmaları incelenecektir. Yani modeller giriş-çıkış katmanları haricinde kaç gizli katman Derin Sinir Ağı ve Yinelemeli Sinir Ağı için içermeli, Evrişimsel modeli ise kaç evrişim, havuzlama ve tam bağlı katmanı olması belirlenmektedir. Tüm modeller için dönem (epoch) sayısı bir, toptan boyutu (batch-size) 128, eniyileme algoritması (optimizer) ise 'Adam' varsayılan değerler olarak kabul etmiştir. Ardında, modellerin hiper-parametre ayarlamaları için her bir veri kümesi için ayrı ayrı test yapılmıştır.

3.4.1 Derin Sinir Ağı (Deep Neural Network - DNN)

Derin Sinir Ağı modelinin oluşturabilmesi için bir giriş, gizli ve çıkış katmanlardan olması gerekmektedir. Ağ derin olabilmesi için de en az iki katmanlardan içermesi gerekmektedir. Katman sayısı belirleme aşamasında iki katman ile başlayarak beş katmanlara kadar çıkarak modeller oluşturulmuştur. Her katmanda

da varsayılan nöron sayıları seçerek model test edilmiştir. Tablo 3.1 görüldüğü gibi, birinci model için {64,128} ve {0.1, 0.1}, ikinci model için {64, 128, 256 }ve {0.1, 0.1, 0.2}, üçüncü model için {64,128, 256, 512} ve {0.1, 0.1, 0.2, 0.1}, dördüncü model için {64,128, 256, 512,1024} ve {0.1, 0.1, 0.2, 0.1, 0.2} nöron sayılar ve seyreltme değerler sırasıyla varsayılan tanımlanmıştır.

Tablo 3.1: DNN seçiminde oluşturulan modeller

	Katman	**Nöron Sayısı**	**Seyreltme Katmanı**
1. Model	1	64	0.1
	2	128	0.1
2. Model	1	64	0.1
	2	128	0.1
	3	256	0.2
3. Model	1	64	0.1
	2	128	0.1
	3	256	0.2
	4	512	0.1
4. Model	1	64	0.1
	2	128	0.1
	3	256	0.2
	4	512	0.1
	5	1024	0.2

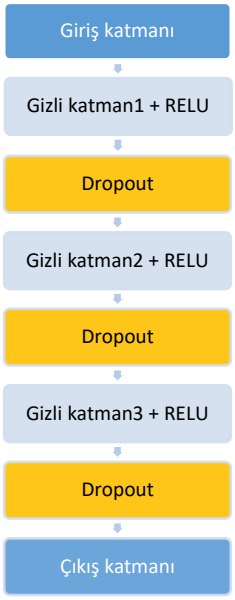
Tablo 3.2'de görüldüğü gibi, 1. model iki katman sayısı ve 0.9502 geçerleme doğruluk oranı olarak, 2.model üç katman sayısı ve 0.9644 geçerleme doğruluk oranı olarak, 3.model dört katman sayısı ve 0.9561 geçerleme doğruluk oranı olarak, 4.model beş katman sayısı ve 0.9601 geçerleme doğruluk oranı olarak elde etmiştir.

Tablo 3.2: DNN model seçimi

Model	**Katman Sayısı**	**Geçerleme Doğruluk Oranı**
1. MODEL	2	0.9502
2. MODEL	3	0.9644
3. MODEL	4	0.9561
4. MODEL	5	0.9601

Ek olarak, modellerimize seyreltmeler (Dropouts) ve aktivasyon fonksiyonlar eklenmiştir. Bu modellerden en **yüksek** geçerleme doğruluk (validation accuracy) oranı olanı seçilmiştir.

Tablo 3.2'de görüldüğü gibi en yüksek kabullenme veya geçerleme doğruluk oranı 0.9644 olan 2. modeldir ve bu model 3 katmandan oluşmaktadır. Modellimizin basitleştirilmiş hali Şekil 3.7'de gösterilmektedir. Bu şekilde, Bir giriş, birinci gizli katman ve seyreltme, ikinci gizli katman ve seyreltme, üçüncü gizli katman ve seyreltme ve çıkış veya çıktı katmanlardan oluşmuştur.



Şekil 3.7: Derin Sinir Ağının Yapısı

Derin Sinir Ağı modelinin belli olması ardından hiper-parametreleri ayarlanması gerekmektedir. Her gizli katmanda optimum nöron sayısı, seyreltme değeri, eniyileme yöntemi (optimizer), toptan boyu (batch-size) ve dönem (epoch) değerleri hiper-parametreleri, Python programlama dili ile kodlanarak belirlenmiştir.

Derin Sinir Ağı hiper-parametreler seçiminde birinci, ikinci ve üçüncü gizli katmanları için {32, 64, 128, 512, 1024} nöron sayıları, her bir gizli katmandan sonra seyreltme (dropout) için [0,1] aralarındaki değerler, eniyileme yöntemi olarak {adam, rmsprop, sgd}, toptan boyu {32, 64, 128}, dönem için {20, 40, 60, 80, 100} değerler belirlemiştir. Gizli katmanlarda aktivasyon fonksiyon olarak ReLU, çıktı katmanında aktivasyon fonksiyon olarak Softmax, yanlılık başlatıcı olarak bias_initializer='zeros',

ağırlıklı başlatıcı olarak kernel_initializer='glorot_uniform'varsayılan hiper-parametreler belirlenmiştir.

Jupyter notebook ortamında hazırlanan Python kodunun hiper-parametreler belirlemek üzere Mnist ve Arapça veri setleri ile çalıştırıldığında Tablo 3.3'de görülebileceği üzere, optimum hiper-parametre değerleri elde edilmiştir. Bu tabloda, modelimiz için Gizli katman1, Gizli katman2 ve Gizli katman3 denk gelen ünite veya nöron sayıları sırasıyla 1024, 512 ve 1024 olarak; Seyreltme1, Seyreltme2 ve Seyreltme3 denk gelen değerleri 0.1, 0.9 ve 0.01 olarak, toptan boyu 128, dönem 60 ve eniyileme algoritması sgd olarak hiper-parametreleri en uygun bulunmuştur.

Tablo 3.3: DNN hiper-parametreleri Mnist ve Arapça veri setleri

Hiper-parametreler	**Hiper-parametre Değerleri**
Gizli katman1 nöron sayısı	1024
Gizli katman2 nöron sayısı	512
Gizli katman3 nöron sayısı	1024
Seyreltme1 (Dropout1)	0.1
Seyreltme2	0.9
Seyreltme3	0.01
Toptan boyu (Batch-size)	128
Dönem (Epoch)	60
Eniyileme algoritması (Optimizer)	Sgd

Program, Derin Sinir Ağının modeli Cifar-10 ve Fashion-Mnist veri setleri ile hiper-parametreler belirlemek üzere çalıştırıldığında Tablo 3.4'te görüldüğü üzere, optimum hiper-parametre değerleri elde edilmiştir. Bu tabloda, modelimiz için Gizli katman1, Gizli katman2 ve Gizli katman3 denk gelen ünite veya nöron sayıları sırasıyla 32, 32 ve 64 olarak; Seyretme1, Seyretme2 ve Seyretme3 denk gelen değerleri 0.04, 0.1 ve 0.3 olarak, toptan boyu 64, dönemi 40, eniyileme algoritması ise 'rmsprop' olarak hiper-parametreleri en uygun bulunmuştur.

3.4.2 Evrişimsel Sinir Ağı (Convolutional Neural Network - CNN)

Evrişimsel Sinir Ağı için, Lenet, Alexnet, VGG standart olarak bilinen CNN modellerden dayanarak kendi ağımız oluşturulmuştur. Lenet, Alexnet, VGG benzer modelleri üç adet oluşturulduktan sonra Mnist veri seti ile en yüksek geçerleme doğruluk oranına sahip olan model olarak seçilmiştir.

Tablo 3.4: DNN hiper-parametreleri Cifar-10 ve Fashion-Mnist veri setleri

Hiper-parametreler	**Hiper-parametre Değerleri**
Gizli katman1 nöron sayısı	32
Gizli katman2 nöron sayısı	32
Gizli katman3 nöron sayısı	64
Seyreltme1 (Dropout1)	0.04
Seyreltme2	0.1
Seyreltme3	0.3
Toptan boyutu (batch-size)	64
Dönem (epoch)	40
Eniyileme algoritması (optimizer)	Rmsprop

Modellerimiz evrişim (convolution), seyreltme (dropout [34]), havuzlama (pooling) , tam bağlı (fully-connected) katmanları içererek, ReLU ve softmax aktivasyon fonksiyonları eklenmiştir. Modelimiz, evrişim katmalarında varsayılan farklı nöron veya ünite değerler ve çekirdek boyutu (kernel size) = 3, seyreltme değerleri varsayılan, havuzlama katmanlarında ise havuz boyutu (window size) = 2, girdi ve çıktı aynı boyutta olacaklarından dolgu (pagging) = 1 veya ‘valid’; ve kaydırma adımı (stride) = (1,1) uygulanmıştır. Tablo 3.5’te görüldüğü gibi, oluşturulan Lenet tipindeki modeli, iki adet evrişim, üç adet seyreltme ve bir tam bağlı katmanlarına sırasıyla {128, 256} nöron sayıları, {0.1, 0.1, 0.2} seyreltme ve {128}tam bağlı nöron sayısı varsayılan değerleri vererek her evrişimsel katmandan sonra bir havuzlama katmanı bulunmaktadır. AlexNet tipinde üç adet evrişim, beş adet seyreltme ve iki tam bağlı katmanlarına sırasıyla {64, 128, 256} nöron sayıları, {0.1, 0.1, 0.2, 0.2, 0.1} seyreltme ve {128, 64} nöron sayıları varsayılan değerleri

tanımlanarak her evrişim katmandan sonra bir havuzlama katmanı eklenmiştir. VGG tipindeki modeli ise dört adet evrişim, dört adet seyreltme ve iki tam bağlı katmanlarına sırasıyla {32, 64, 128, 256} nöron sayıları, {0.1, 0.1, 0.2, 0.2}seyreltme ve {128, 64} nöron sayıları varsayılan değerleri belirlenerek her iki evrişim katmandan sonra bir havuzlama katmanı eklenmiştir.

Tablo 3.5: CNN seçiminde oluşturulan modeller

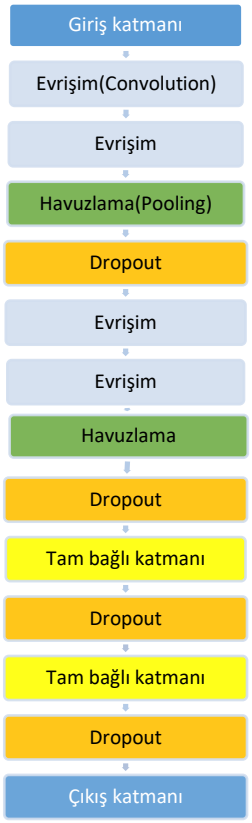
Model(CNN)	Katman	Çekirdek veya havuz boyutu	Nöron Sayısı	Seyreltme Katmanı
Lenet	Evrişim	3	128	-
	Havuzlama	2	-	0.1
	Evrişim	3	256	-
	Havuzlama	2	-	0.1
	Tam bağlı	-	128	0.2
AlexNet	Evrişim	3	64	-
	Havuzlama	2	-	0.1
	Evrişim	3	128	-
	Havuzlama	2	-	0.1
	Evrişim	3	256	-
	Havuzlama	2	-	0.2
	Tam bağlı	-	128	0.2
	Tam bağlı	-	64	0.1
VGG	Evrişim	3	32	-
	Evrişim	3	64	-
	Havuzlama	2	-	0.1
	Evrişim	3	128	-
	Evrişim	3	256	-
	Havuzlama	2	-	0.1
	Tam bağlı	-	128	0.1
	Tam bağlı	-	64	0.1

Modellerimizin oluşumu ardından ve çalıştırıldıktan sonra Tablo 3.6'da görüldüğü üzere, geçerleme doğruluk (validation accuracy) oranı değerleri verilmektedir. Birinci modeli CNN Lenet model tipinden 0.9766; CNN Alexnet model tipinden 0.9796 ve CNN VGG model tipinden 0.9844 geçerleme doğruluk oranı elde edilmiştir.

Tablo 3.6: CNN model seçimi

Model	Geçerleme Doğruluk Oranı
1. CNN (LENET Benzeri)	0.9766
2. CNN(ALEXNET Benzeri)	0.9796
3. CNN(VGG Benzeri)	0.9844

Oluşturulan modellerden, CNN VGG tipinden benzeyen modelimiz ile yüksek geçerleme doğruluk oranı elde ederek model olarak seçilmiştir. Şekil 3.8'de görüldüğü gibi, modelimiz bir giriş, iki adet evrişim, bir adet havuzlama, bir adet seyreltme katmanlar art arda konularak tekrardan iki adet evrişim, bir adet havuzlama, bir adet seyreltme katmanları eklenerek iki adet tam bağlı ve bir çıkış veya çıktı katmanları konularak modelin oluşumu tamamlanmıştır.



Şekil 3.8: Evrişimsel Sinir Ağının yapısı

Evrişimsel Sinir Ağının modeli oluşturulduktan sonra hiper-parametreleri belirlenmiştir. Bu parametre belirlemesinde dört adet Conv1_1, Conv1_2, Conv2_1 ve

Conv2_2 isimlendirilen evrişim katmanları için her biri { 16, 32, 64, 128} nöron sayıları, her bir havuzlama ve tam bağlı katmandan sonra gelen seyreltme katmanı için [0, 1], eniyileme algoritmaları {adam, rmsprop, sgd}, toptan boyu { 32, 64, 128}, dönem için { 20, 40, 60, 80, 100} değerler belirlemiştir. Tüm evrişim katmanlarında aktivasyon fonksiyon olarak ReLU, çıktı katmanında aktivasyon fonksiyon olarak Softmax, yanlılıklık başlatıcı olarak bias_initializer='zeros', ağırlıklı başlatıcı olarak kernel_initializer='glorot_uniform', evrişim kernel_size = 3 ve havuzlama pool_size = 2 varsayılan hiper-parametreler belirlenmiştir.

Tablo 3.7: CNN hiper-parametreleri

Hiper-parametreler	**Hiper-parametre Değerleri**
Conv1_1 nöron sayısı	32
Conv1_2 nöron sayısı	64
Conv2_1 nöron sayısı	32
Conv2_2 nöron sayısı	32
Tam bağlı katman1	256
Tam bağlı katman2	128
Seyreltme1	0.9
Seyreltme2	0.1
Seyreltme3	0.6
Seyreltme4	0.4
Toptan boyutu	128
Dönem	80
Eniyileme algoritması (optimizer)	Adam

Hiper-parametreler belirlemek üzere, CNN modeli için hazırlanan kodu Cifar-10, Fashion-Mnist, Mnist ve Arapça ayrı ayrı dört veri setlerimiz ile çalıştırıldığında Tablo 3.7'de görüldüğü gibi, optimum hiper-parametre değerleri elde edilmiştir. Bu tabloda, modelimiz için Conv1_1, Conv1_2, Conv2_1 ve Conv2_2 evrisim katmalara denk gelen ünite veya nöron sayıları sırasıyla 32, 64, 32 ve 32 olarak; tam bağlı katman1 ve tam bağlı katman2 nöron sayıları 256 ve 128 olarak; Seyreltme1, Seyreltme2 ve

Seyreltme3 denk gelen değerleri 0.9, 0.1 ve 0.6 olarak, toptan 128, dönem 60 ve iyileştirici ‘adam’ olarak hiper-parametreleri en uygun değerleri hesaplanmıştır. Bu CNN modelinde elde edilen hiper-parametreler sonuçları bütün kullanmış olduğumuz veri setleri aynıdır.

3.4.3 Yinelemeli Sinir Ağı (Recurrent Neural Network - RNN)

Yinelemeli Sinir Ağı modelimiz, DNN yapımıza benzetmektedir. Ara katman sayıları belirlemek üzere, iki katman ile başlayarak beş katmanlara kadar çıkarak modeller oluşturulmuştur. Her Uzun Kısa Vadeli Hafıza (Long Short Term Memory - LSTM) katmanında da rasgele nöron sayıları seçerek modeli test edilmiştir.

Tablo 3.8’de görüldüğü üzere, birinci model için {64,128} ve {0.1, 0.1}, ikinci model için {64, 128, 128, }ve {0.1, 0.2, 0.2}, üçüncü model için {64, 64, 128, 128} ve {0.1, 0.1, 0.2, 0.1}, dördüncü model için {32, 64, 64, 128, 128} ve {0.1, 0.1, 0.2, 0.2, 0.1} nöron sayılar ve seyreltme değerler sırasıyla varsayılan tanımlanmıştır.

Tablo 3.8: RNN seçiminde oluşturulan modeller

	Katman	Nöron Sayısı	Seyreltme Katmanı
1. Model	1	64	0.1
	2	128	0.1
2. Model	1	64	0.1
	2	128	0.2
	3	128	0.2
3. Model	1	64	0.1
	2	64	0.1
	3	128	0.2
	4	128	0.1
4. Model	1	32	0.1
	2	64	0.1
	3	64	0.2
	4	128	0.2
	5	128	0.1

Tablo 3.9’da görülebileceği gibi, 1.model iki adet LSTM katman sayısı ve 0.7659 geçerleme doğruluk oranı olarak, 2.model üç adet LSTM katman sayısı ve 0.7811 geçerleme doğruluk oranı olarak, 3.model dört adet LSTM katman sayısı ve 0.7765 geçerleme doğruluk oranı olarak, 4.model beş adet LSTM katman sayısı ve

0.7350 geçerleme doğruluk oranı olarak elde etmiştir. Ek olarak, modellerimize seyreltmeler ve aktivasyon fonksiyonlar eklenmiştir. Modellerimiz Mnist veri seti uygulanarak en yüksek geçerleme doğruluk oranına sahip olan seçilmiştir.

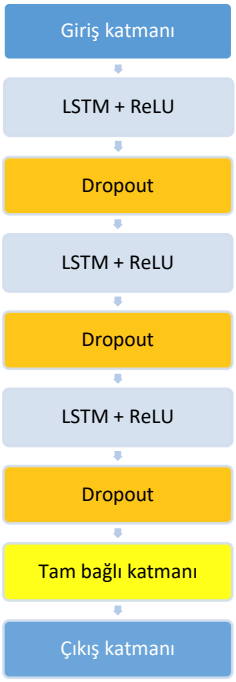
Tablo 3.9: RNN model Seçimi

Model	Katman Sayısı	Geçerleme Doğruluk Oranı
1. MODEL	2	0.7659
2. MODEL	3	0.7811
3. MODEL	4	0.7765
4. MODEL	5	0.7350

Tablo 3.9'dan 3. model en yüksek geçerleme doğruluk oranına sahip olmasından RNN modelimiz olmuştur. Şekil 3.9'da görüldüğü üzere, RNN modelimiz bir giriş, peş peşe birinci LSTM ve Seyreltme1 (Dropout), ikinci LSTM ve Seyreltme2 ve üçüncü LSTM ve Seyreltme3, tam bağlı ve çıktı katmanları oluşturulmuştur.

RNN modelimiz oluşumundan sonra hiper-parametre değerleri belirlenmiştir. Hiper-parametreler seçiminde, diğer yöntemleri olduğu gibi birinci, ikinci ve üçüncü LSTM katmanları için { 32, 64, 128} nöron sayıları, her bir LSTM katmandan sonra seyreltme (dropout) için [0, 1] aralığında, eniyileme algoritması için {adam, rmsprop, sgd}, toptan boyu {32, 64, 128} ve dönem için {20, 40, 60, 80, 100} değerleri kullanılmıştır. Tüm LSTM katmanlarda Aktivasyon fonksiyonları tanh ve Hard-Sigmoid olarak, çıktı katmanında aktivasyon fonksiyon olarak Softmax, yanlılıklık başlatıcı olarak bias_initializer='zeros', ağırlıklı başlatıcı olarak kernel_initializer='glorot_uniform', varsayılan hiper-parametreler belirlenmiştir.

Hiper-parametreler belirlemek üzere, DNN ve CNN modellerimiz olduğu gibi, hazırlanan kodu Cifar-10, Fashion-Mnist, Mnist ve Arapça ayrı ayrı dört veri setlerimiz ile çalıştırıldığında Tablo 3.10'da görüldüğü gibi, optimum hiper-parametre değerleri elde edilmiştir. Bu tabloda, modelimiz için LSTM1, LSTM2 ve LSTM3 adlandırılan RNN katmalara denk gelen ünite veya nöron sayıları sırasıyla 64, 128 ve 128 olarak; tam bağlı katmanı 64 olarak, Seyreltme1, Seyreltme2 ve Seyreltme3 denk gelen değerleri 0.2, 0.1 ve 0.1 olarak, toptan boyu Arapça veri seti için 128, Cifar-10, Fashion-Mnist ve Mnist için 32 olarak; dönem Mnist veri seti için 60 iken Cifar-10,



Şekil 3.9: Yinelemeli Sinir Ağının Yapısı

Tablo 3.10: RNN hiper-parametreleri

Hiper-parametreler	**Hiper-parametre Değerleri**
LSTM1 nöron sayı	64
LSTM2 nöron sayı	128
LSTM3 nöron sayı	128
Tam bağlı katmanı nöron sayı	64
Seyreltme1	0.2
Seyreltme2	0.1
Seyreltme3	0.1
Toptan boyutu, Arapça veri seti için	128
Toptan boyutu, diğer veri setleri.	32
Dönem, Mnist veri seti için	60
Dönem, diğer veri setleri	25
Eniyileme algoritması (Cifar-10)	rmsprop
Eniyileme algoritması (diğer veri setleri)	Adam

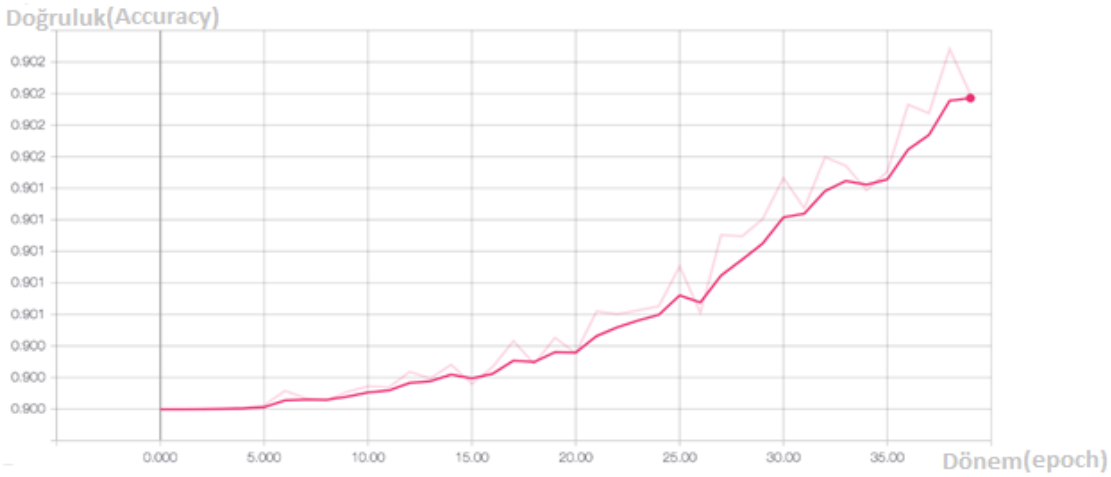
Fashion-Mnist ve Arapça veri setleri için 32 olarak, iyileştirici Cifar-10 veri seti için 'rmsprop' iken Fashion-Mnist, Mnist ve Arapça 'adam' olarak hiper-parametreleri en uygun değerleri belirlenmiştir.

3.5 Algoritmaların Karşılaştırılması

Bu bölümde 4 farklı veri kümesi üzerinde algoritmaların karşılaştırılması verilmiştir. Her bir algoritma için Bölüm 3.4'te yapılan deneysel çalışmalar sonucunda elde edilen en iyi parametre ve modeller kullanılmıştır.

3.5.1 Cifar-10 Veri Kümesi Üzerinde Karşılaştırma

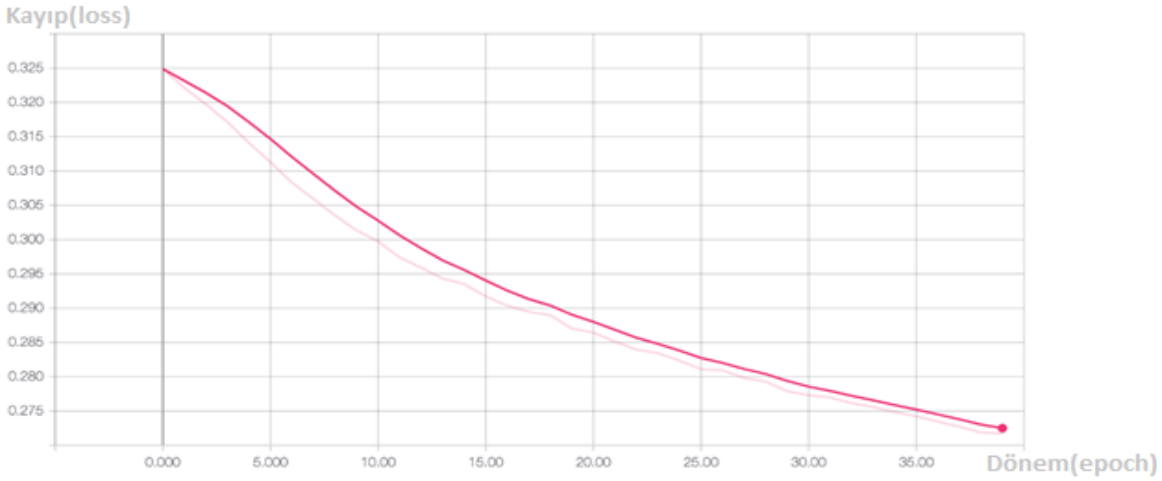
Oluşturulan Derin, Evrişimsel ve Yinelemeli Sinir Ağ modellerimiz, Cifar-10 test verileri tensorboard ortamında kullanarak modellerin optimum dönem (epoch) değerlerine göre doğrulama (accuracy) ve kayıp (loss) grafik sonuçları verilmiştir.



Şekil 3.10: DNN, Cifar-10 test verileri dönem bazında doğruluk grafik sonucu

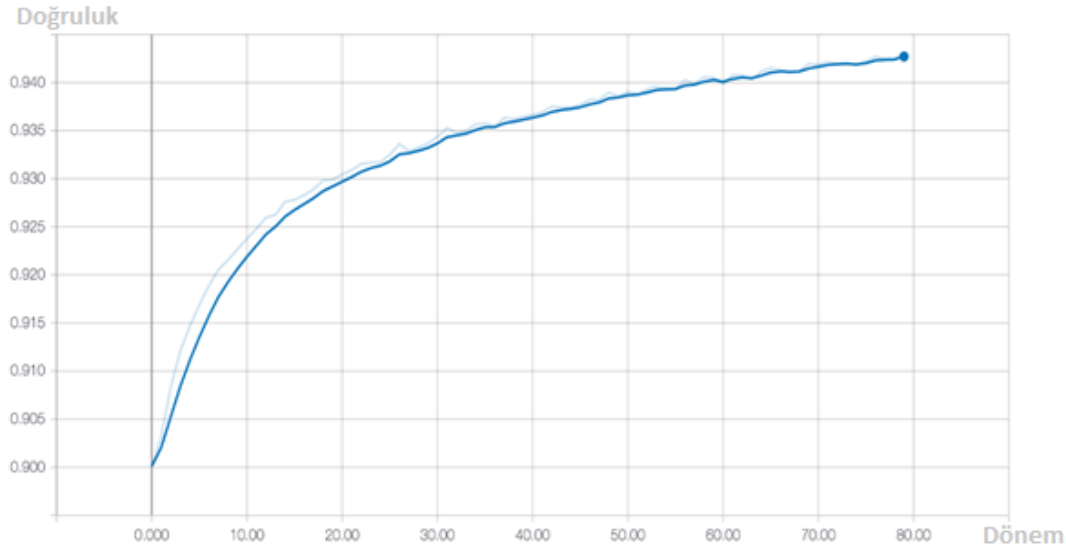
Şekil 3.10 ve Şekil 3.11'de, DNN modelin Cifar-10 veri seti üzerinde doğruluk ve kayıp metrik performansların grafik davranışları verilmektedir. Şekil 3.10'dan anlaşılacağı üzere, birinci dönemden (epoch) başlayarak beşinci döneme kadar doğruluk grafiği 0.900 değeri ile monoton bir şekilde devam ederek beşinci dönemden sonra artırarak optimum dönem sayı olan 40'a varak 0.902 test tamamlanıp 0.9035 (% 90.35) ortalama doğruluk değeri verilmiştir. Şekil 3.11'e bakıldığında, kayıp değeri

birinci dönemden 0.325 değeri ile azaltmaya başlayarak kırkıncı döneme kadar 0.261 ortalama kayıp değeri ile test tamamlanmıştır.

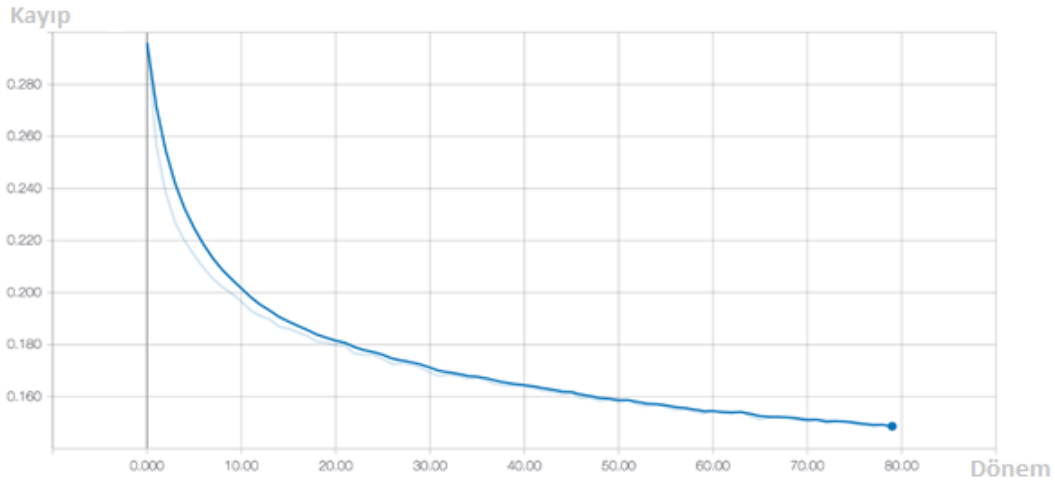


Şekil 3.11: DNN, Cifar-10 test verileri dönem bazında kayıp grafik sonucu

DNN modelinde, doğrulama grafiği gittikçe arttığını ve kayıp grafiğini gittikçe azalttığını izlenmiştir. Bu da, model iyi eğitilmesi anlamına gelmektedir.



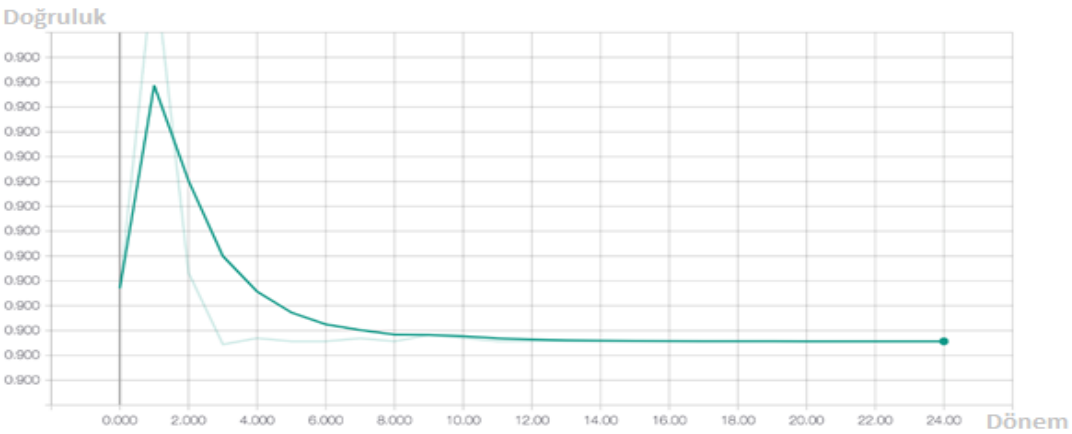
Şekil 3.12: CNN, Cifar-10 test verileri dönem bazında doğruluk grafik sonucu



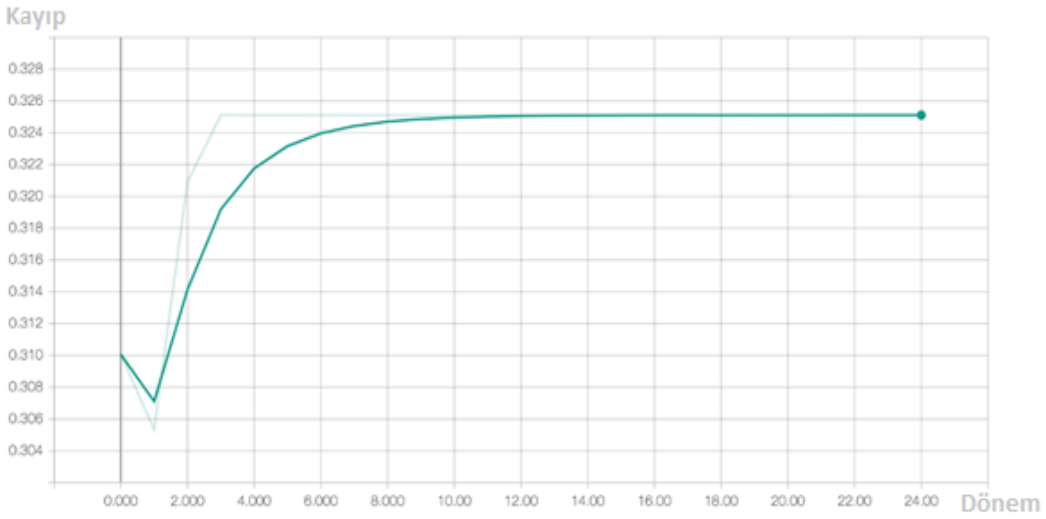
Şekil 3.13: CNN, Cifar-10 test verileri dönem bazında kayıp grafik sonucu

Şekil 3.12 ve Şekil 3.13, CNN modelin 80 dönemlik (epoch) optimum değeri doğruluk ve kayıp performans metriklerinin davranışlarını gösterilmektedir. Şekil 3.12'de, doğruluk değeri 0.900 başlayarak 0.943 test tamamlayıp 0.9372 (%93.72) ortalama doğruluk değerleri kaydedilmiştir. Şekil 3.12'de ise kayıp grafiğini 0.298 civarında bir değer ile azalmaya başlayarak optimum döneme varınca 0.150 test bitirerek ortalama olarak 0.156 ortalama kayıp değeri verilmiştir.

Cifar-10 üzerinde CNN'in doğruluk ve kayıp grafikleri, DNN ile Cifar-10 verileri benzer bir davranışı izlenmektedir. Doğruluk artarken kayıp değeri azalmaktadır. Bu açıdan, modelin iyi eğitildiği anlamı taşınmaktadır.



Şekil 3.14: RNN, Cifar-10 test verileri dönem bazında doğruluk grafik sonucu

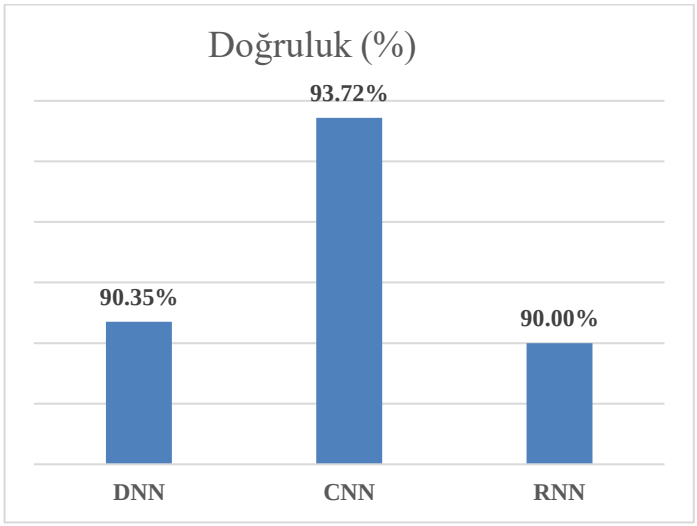


Şekil 3.15: RNN, Cifar-10 test verileri dönem bazında kayıp grafik sonucu

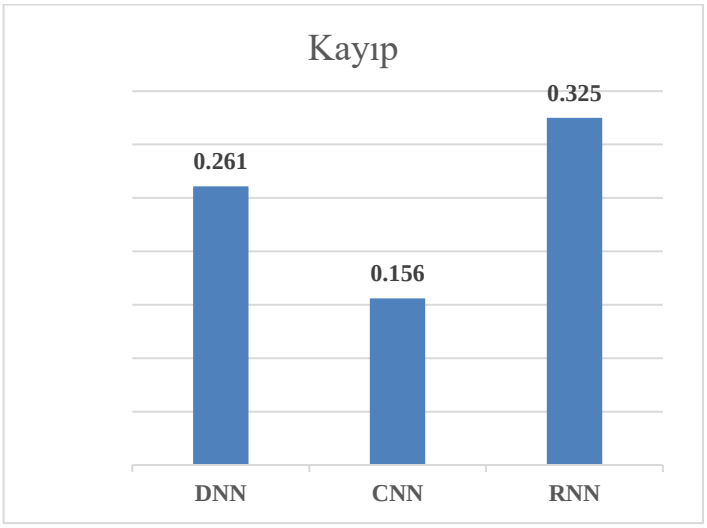
Şekil 3.14 ve şŞkil 3.15'te RNN modelinin Cifar-10 test verileri ile 25 dönem (epoch) optimum değeri olan doğruluk ve kayıp performans metriklerinin hareketlerini sunmaktadır. Şekil 3.14'de 0.9000 doğruluk değeri ile artışı göstererek 0.9002 doğrulama değerine vararak bir düşüş başlayıp on ikinci dönem itibaren, optimum döneme kadar 0.9000 (%90) bir ortalama doğruluk ile test bitmiştir. Şekil 2.15 incelendiğinde, kayıp değeri 0.310'den bir azalma başlayarak ikinci dönemde 0.307 değerinden artmaya başladığı görülmektedir. Onuncu dönemden optimum döneme kadar monoton devam ederek 0.325 bir ortalamam kayıp değeri ile test tamamlanmıştır.

RNN modeli Cifar-10 test veri üzerinde, DNN ve CNN modellere göre daha farklı bir davranış göstermiştir. Doğruluk grafiği artması beklerken belli bir noktaya kadar artı ardından bir düşüş göstererek monoton olarak test bitirmiştir. Kayıp grafiği ise azalması gerekirken belli bir noktada azalıp sonra artmaya başlayarak monoton bir şekilde o da test tamamlanmıştır.

Cifar-10 veri test ile modellerimiz karşılaştırmak için modellere göre Şekil 3.16 ve Şekil 3.17 doğruluk ve kayıp sütun grafikleri çizilmiştir.



Şekil 3.16: Cifar-10 veri seti üzerinde doğruluk değerlerin açısından modellerin karşılaştırılması

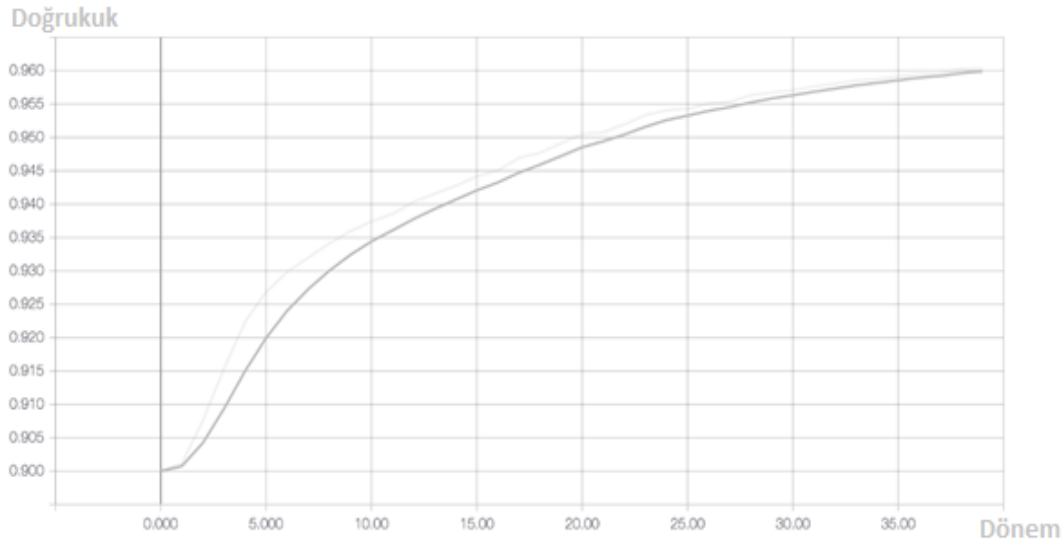


Şekil 3.17: Cifar-10 veri seti üzerinde kayıp değerlerin açısından modellerin karşılaştırılması

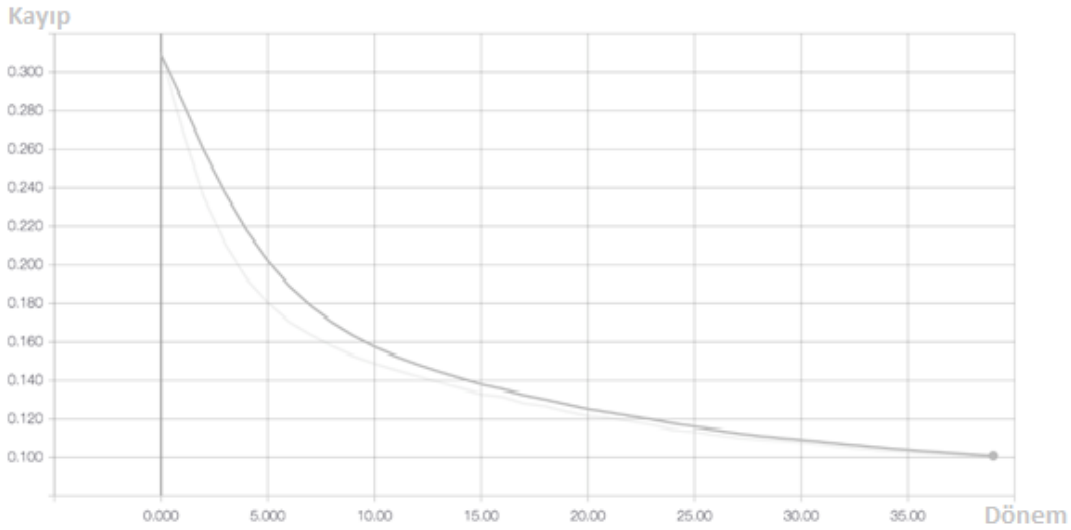
Bir modelin en iyi performans olabilmesi için doğruluk değeri eğitim ve test esnasında yüksek, kayıp değerinin ise düşük olması beklemektedir. Şekil 3.16 ve Şekil 3.17'e baktığımızda Cifar-10 test verileri üzerinde %93.72 doğruluk ve 0.156 kayıp değerleri olan CNN modelini en iyi model olarak kaydedilmiştir. Karşılaşılan modeller, en az performanslı %90 doğruluk ve 0.325 kayıp değerleri olan RNN model olmuştur.

3.5.2 Fashion-Mnist Veri Kümesi Üzerinde Karşılaştırma

Modellerimiz, Fashion-Mnist test verileri ile tensorboard ortamında optimum dönem (epoch) sayıları bazında doğrulama ve kayıp sonuç grafikleri çizilmiştir.



Şekil 3.18: DNN, Fashion-Mnist test verileri dönem bazında doğruluk grafik sonucu

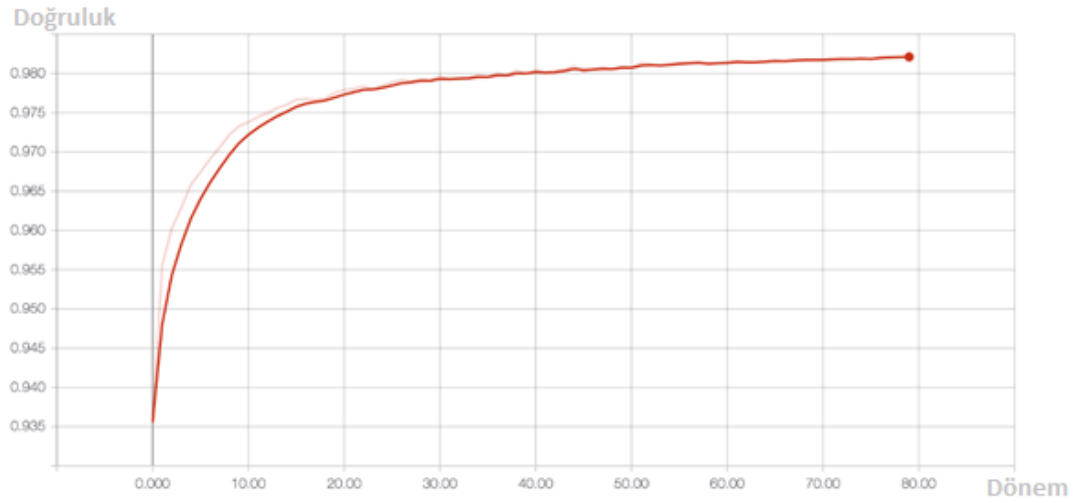


Şekil 3.19: DNN, Fashion-Mnist test verileri dönem bazında kayıp grafik sonucu

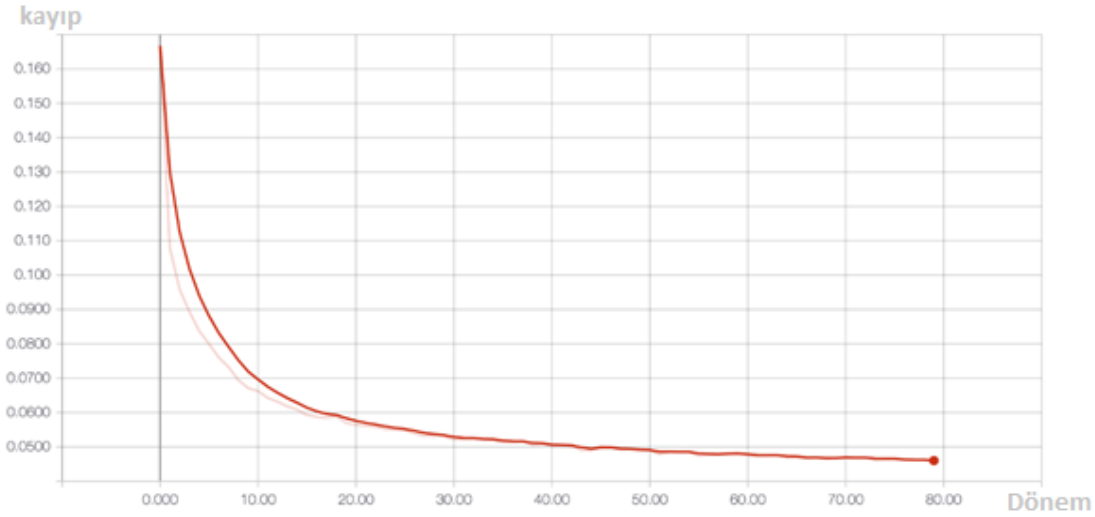
Şekil 3.18 ve Şekil 3.19, DNN modelin Fashion-Mnist veri seti üzerinde doğruluk ve kayıp metrik performansların grafik davranışları gösterilmektedir. Şekil

3.18’de görüldüğü gibi, birinci dönemden (epoch) 0.900 doğruluk değeri ile başlayıp artmaya devam ederek optimum dönem sayı olan 40’a vararak test tamamlanıp 0.9638 (% 96.38) ortalama doğruluk değeri verilmiştir. Şekil 3.19’e bakıldığında, kayıp değeri birinci dönemden 0.309 değeri ile azaltmaya başlayarak kırkıncı döneme kadar 0.088 ortalama kayıp değeri ile test tamamlanmıştır.

DNN modeli Fashion-Mnist veri seti ile doğruluk grafiğinin gittikçe arttığı ve kayıp grafiğinin gittikçe azaldığı izlenmiştir. Model iyi eğitilmesi ve test gerçekleştirildiği anlamına gelmektedir.



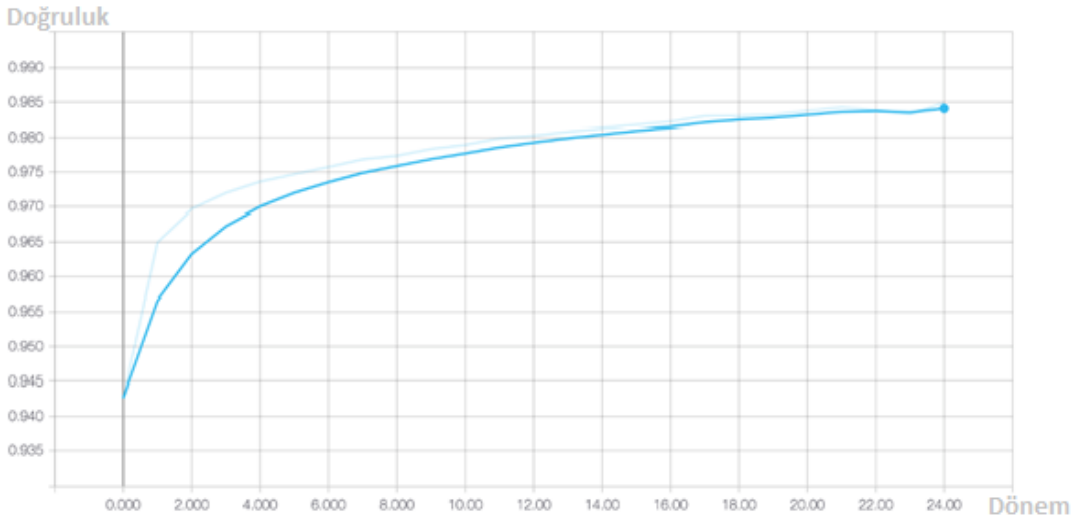
Şekil 3.20: CNN, Fashion-Mnist test verileri dönem bazında doğruluk grafik sonucu



Şekil 3.21: CNN, Fashion-Mnist test verileri dönem bazında kayıp grafik sonucu

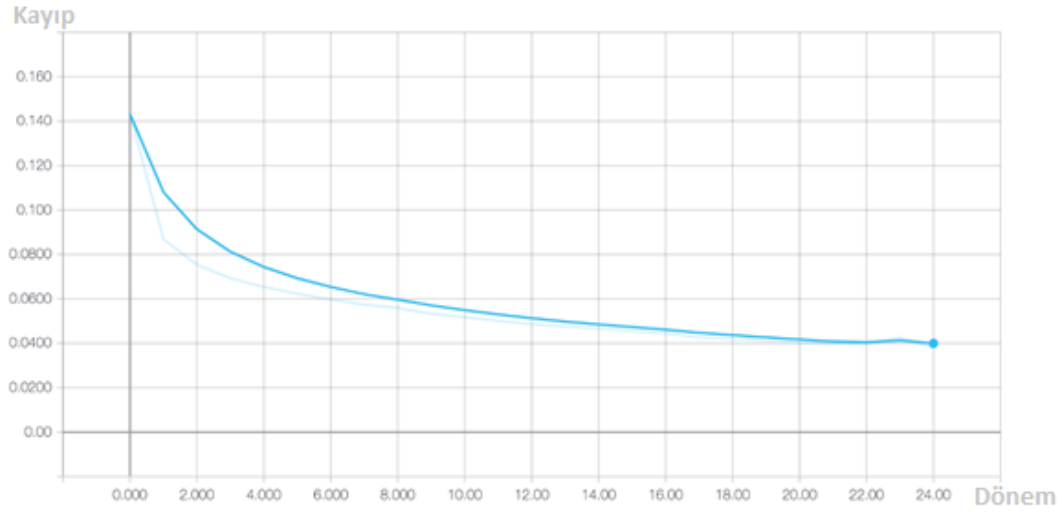
CNN modeli Fashion-Mnist test veri setinden Şekil 3.20 ve Şekil 3.21'de yakınsama davranışları görülmektedir. Şekil 3.20'de birinci dönemden 0.936 doğruluk değeri artmaya başlayarak 80 olan uygun dönem diğerine vararak 0.9838 (%98.38) ortalama doğruluk değeri kaydedilmiştir. Şekil 3.21'de ise kayıp grafiği giderek azaldığını görünmektedir. Test tamamlandığında 0.045 ortalama kayıp değerine kaydedilmiştir.

CNN modeli Fashion-Mnist veri seti ile Şekil 3.20'den doğruluk değeri gittikçe arttığını, Şekil 3.21'de ise kayıp değeri azaldığını izlenmiştir. Bu açıdan, modeli iyi eğitildiği ve test edildiği sonucuna varılmaktadır.



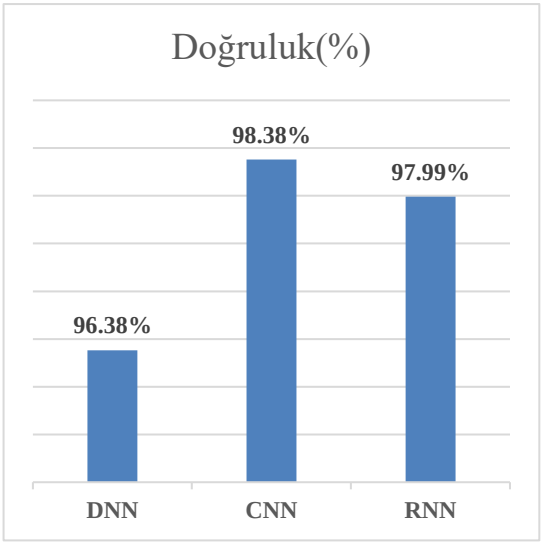
Şekil 3.22: RNN, Fashion-Mnist test verileri dönem bazında doğruluk grafik sonucu

RNN modeli, DNN ve CNN modelleri benzer bir durumu izlenmiştir. Şekil 3.22 ve Şekil 3.23'den doğruluk değeri gittikçe arttığını ve kayıp değeri gittikçe azaldığını görülmektedir. Şekil 3.22'de birinci dönemden 0.943 doğruluk değeri başlayarak optimum dönem değerine varmasıyla 0.9799 (%97.77) ortalama doğruluk değeri verilmiştir. Şekil 3.23'de ise 0.143 kayıp değeri ile başlayarak optimum döneme varıncaya kadar 0.055 ortalama kayıp değeri ile test tamamladığı görünmektedir. RNN modelinin doğruluk ve kayıp şekillerinden iyi eğitildiği ve test edildiği gösterilmektedir.

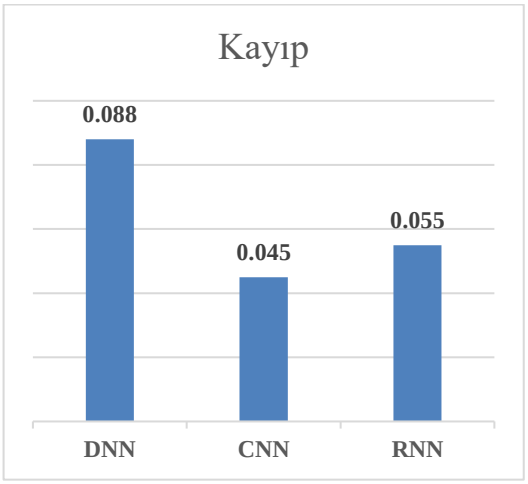


Şekil 3.23: RNN, Fashion-Mnist test verileri dönem bazında kayıp grafik sonucu

DNN, CNN ve RNN modeller, Fashion-Mnist veri üzerinde gerçekleştirilen test ve alınan sonuçları karşılaştırması için Şekil 3.24 ve Şekil 3.25'te doğruluk ve kayıp sütun grafikleri çizilmiştir.



Şekil 3.24: Fashion-Mnist veri seti üzerinde doğruluk değerlerin açısından modellerin karşılaştırılması

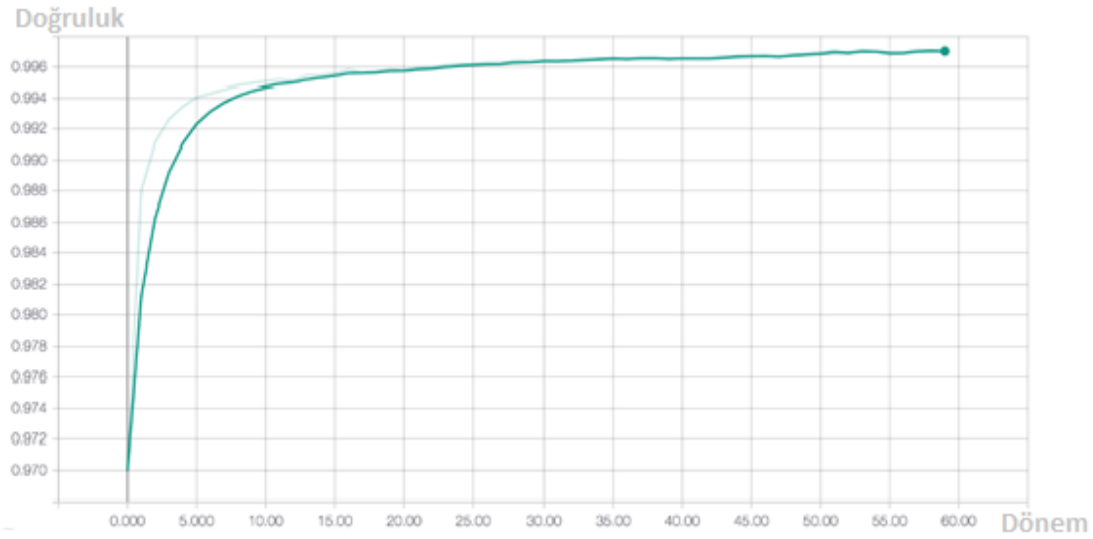


Şekil 3.25: Fashion-Mnist veri seti üzerinde kayıp değerlerin açısından modellerin karşılaştırılması

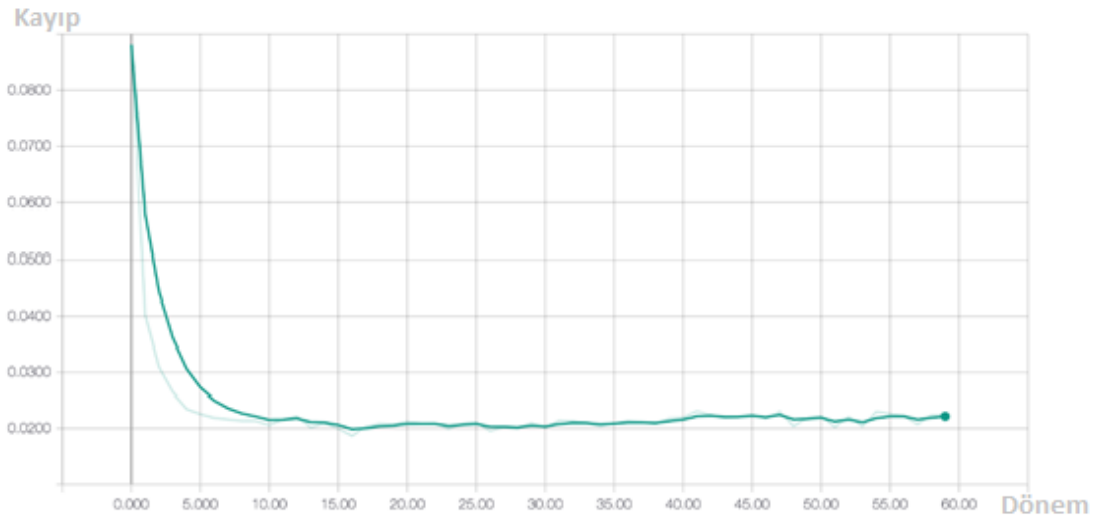
Fashion-Mnist test veri üzerinde Şekil 3.24 ve Şekil 3.25'te verilen sütun grafiklerine baktığımızda %98.38 doğruluk ve 0.045 kayıp değerleri olan CNN modelini en iyi model olarak görünmektedir. Karşılaşılan modeler, en az performanslı ise %96.38 doğruluk ve 0.088 kayıp değerleri olan DNN modeli olmaktadır.

3.5.3 Mnist Veri Kümesi Üzerinde Karşılaştırma

Bu kısmında, Mnist test verileri ile tensorboard ortamında optimum dönem sayıları bazında doğruluk ve kayıp sonuçları şekilleri verilmiştir.



Şekil 3.26: DNN, Cifar-10 test verileri dönem bazında doğruluk grafik sonucu

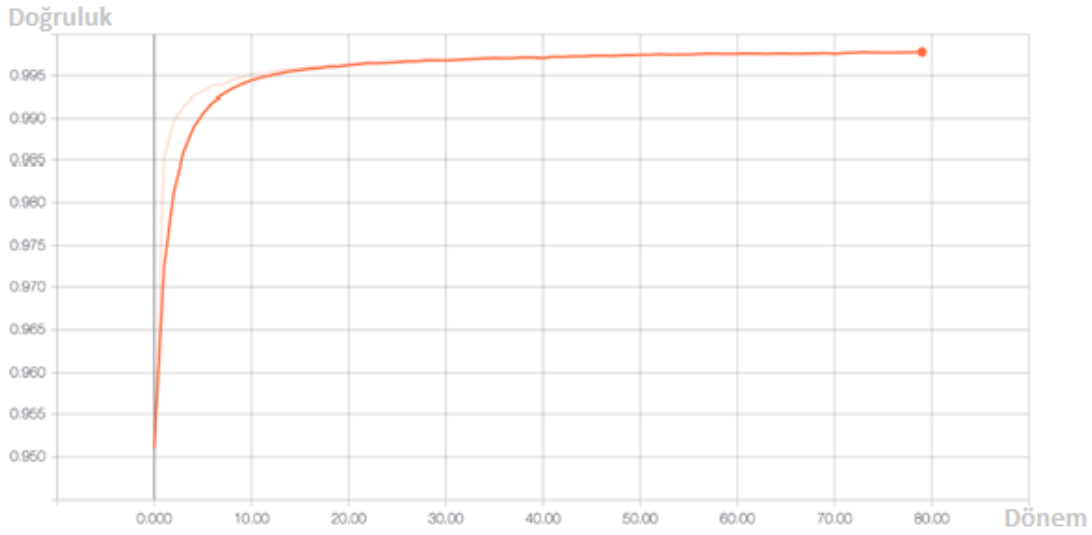


Şekil 3.27: DNN, Cifar-10 test verileri dönem bazında kayıp grafik sonucu

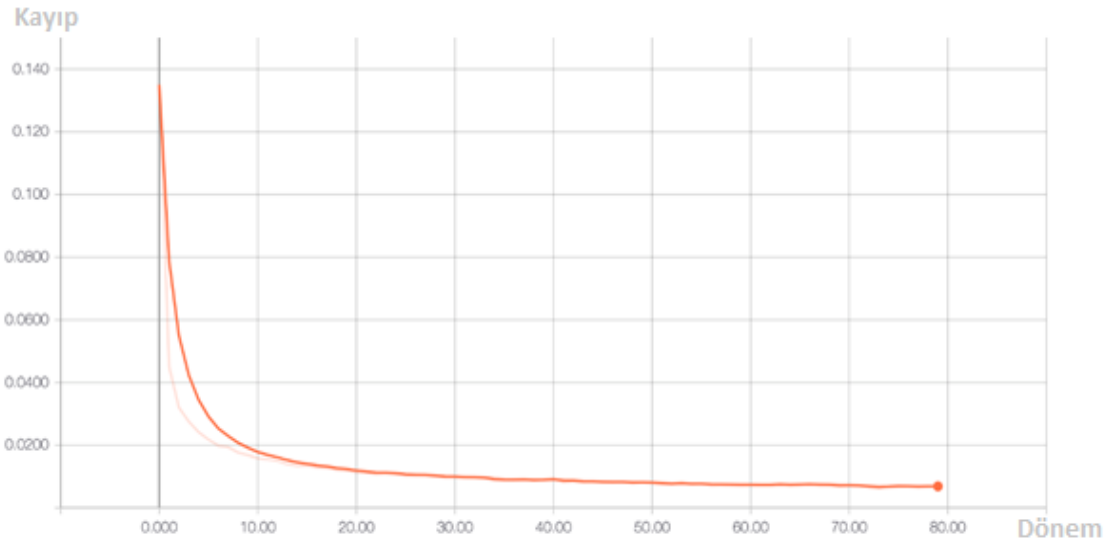
Şekil 3.26 ve Şekil 3.27'de DNN modelin Mnist veri seti üzerinde doğrulama ve kayıp metrik performansların grafik davranışları verilmektedir. Şekil 3.26'den anlaşılacağı üzere, birinci dönemden başlayarak 0.970 doğruluk değeri artışı göstererek optimum dönem sayısı olan 60'a vararak 0.9953 (% 99.53) ortalama doğruluk değeri verilmiştir. Şekil 3.27'ye bakıldığında, kayıp değeri 0.088 değeri ile azaltmaya başlayarak altmışıncı döneme kadar 0.039 ortalama kayıp değeri ile test tamamlanmıştır.

DNN modelinde, Mnist test veri ile doğrulama grafiği gittikçe arttığını ve kayıp grafiğini gittikçe azaldığı izlenmiştir. Bu da, modelin iyi eğitildiği anlamına gelmektedir.

CNN modeli Mnist test veri setinden Şekil 3.28 ve Şekil 3.29'lerde davranışları görülmektedir. Şekil 3.28'de birinci dönemden 0.952 doğruluk değeri artmaya başlayarak 80 olan uygun dönem diğerine vararak 0.9988 (%99.88) ortalama doğruluk değeri kaydedilmiştir. Şekil 3.29'de ise kayıp grafiği 0.135 değerinden dönem optimum değerine varıncaya kadar giderek azaldığını görünmektedir. Test tamamlandığında 0.042 ortalama kayıp değerine kaydedilmiştir.

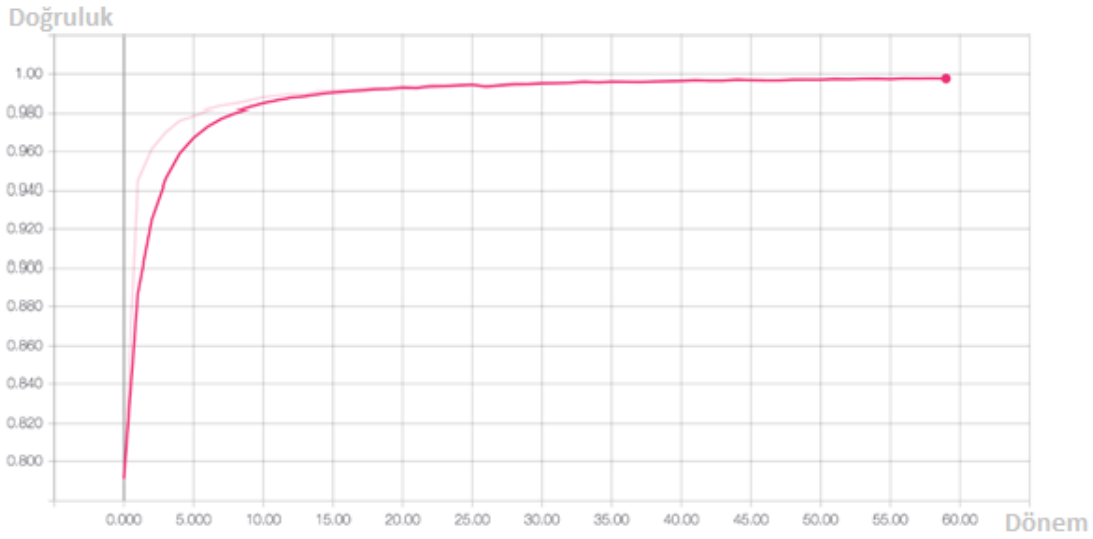


Şekil 3.28: CNN, Mnist test verileri dönem bazında doğruluk grafik sonucu

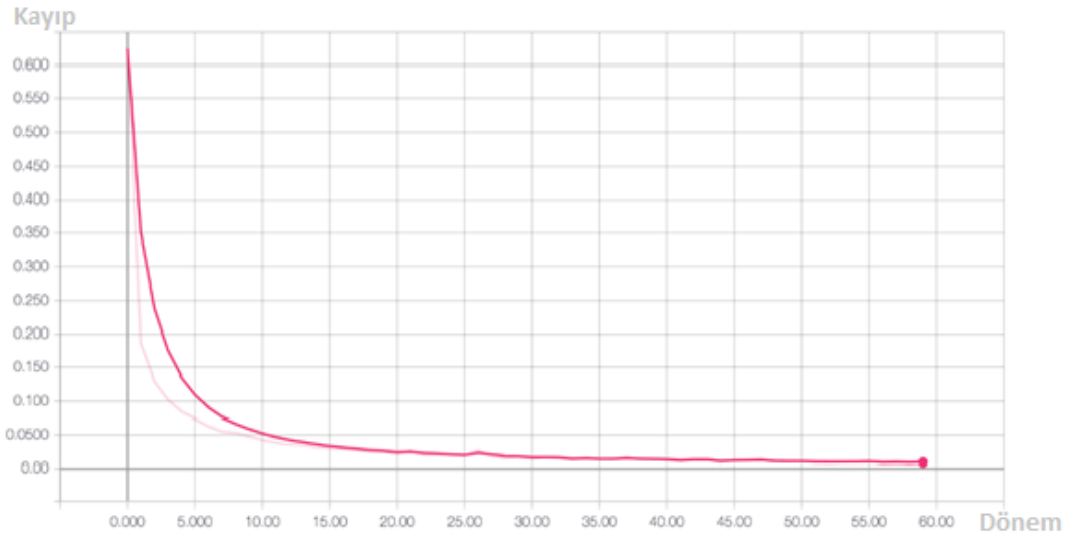


Şekil 3.29: CNN, Mnist test verileri dönem bazında kayıp grafik sonucu

CNN modeli Mnist veri seti ile Şekil 3.28'den doğruluk değeri gittikçe arttığını, şekil 3.29'den ise kayıp değeri azaldığını izlenmiştir. Bu açıdan, modelin iyi eğitildiği ve test edildiği sonucuna varılmaktadır.



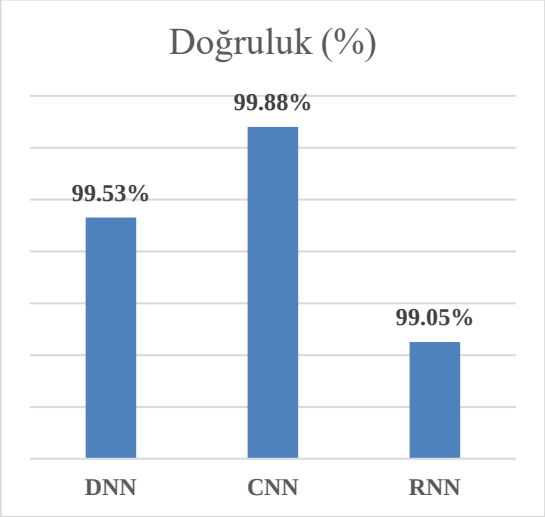
Şekil 3.30: RNN, Mnist test verileri dönem bazında doğruluk grafik sonucu



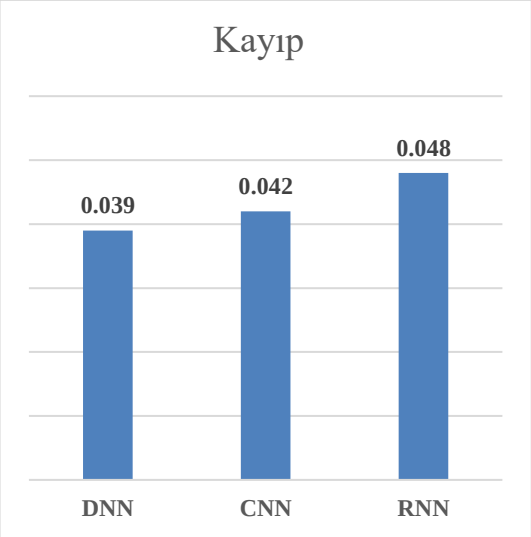
Şekil 3.31: RNN, Mnist test verileri dönem bazında kayıp grafik sonucu

RNN modeli Mnist veri seti test edildiğinde şekil 3.30 ve şekil 3.31'ler elde edilmiştir. Şekil 3.30 görüldüğü üzere, 0.810 doğruluk değeri ile arttırarak 60 olan optimum değerine gelerek 0.9905 (%99.05) ortalama doğruluk değeri elde edilmiştir. Şekil 3.31 ise kayıp değeri 0.624 azalarak 0.048 ortalama kayıp değeri verilmiştir.

Mnist veri test ile modellerimiz karşılaştırmak için modellere göre Şekil 3.32 ve Şekil 3.33'te görüldüğü gibi doğruluk ve sütun grafikleri çizilmiştir.



Şekil 3.32: Mnist veri seti üzerinde doğruluk değerlerin açısından modellerin karşılaştırılması

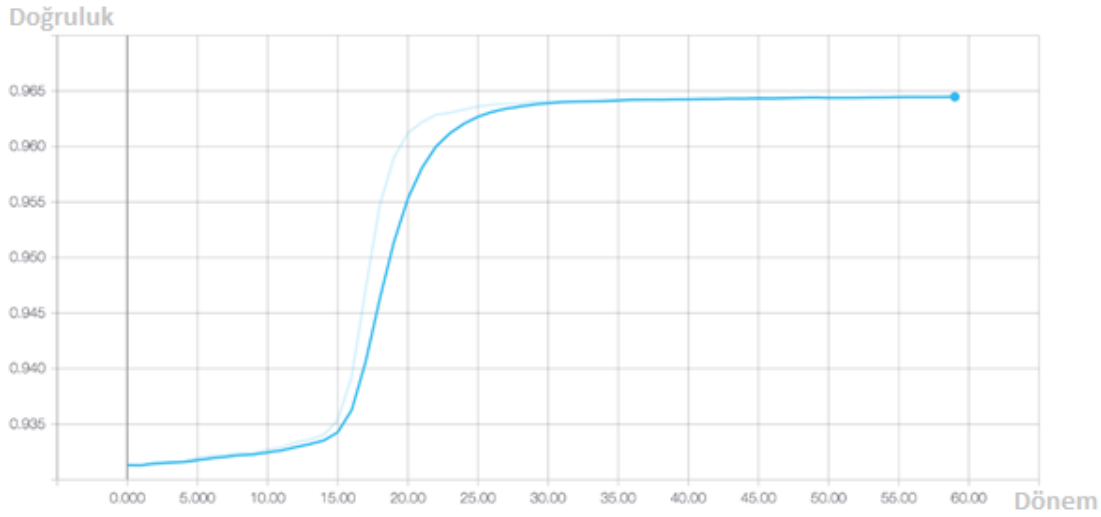


Şekil 3.33: Mnist veri seti üzerinde kayıp değerlerin açısından modellerin karşılaştırılması

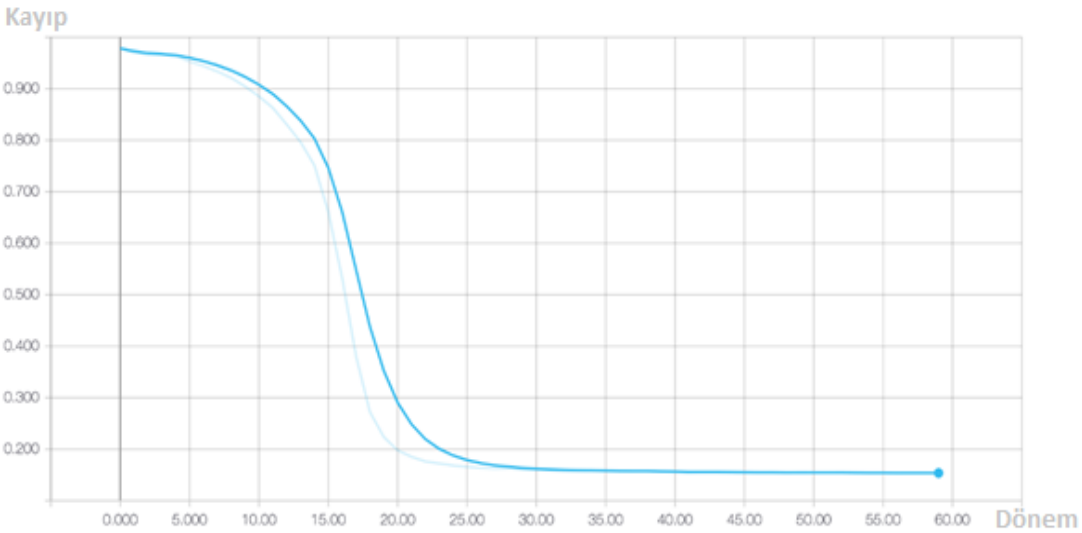
Mnist test veri üzerinde Şekil 3.32 ve Şekil 3.33'de verilen sütun grafiklerine baktığımızda %99.88 doğruluk ve 0.042 kayıp değerleri olan CNN modelini en iyi model olarak görünmektedir. CNN modeli ardından DNN %99.53 doğruluk ve 0.039 kayıp değerleri ikinci iyi performanslı modeli olmuştur. Performansı %99.05 doğruluk ve 0.048 kayıp değerleri olan DNN üçüncü modeli olmuştur.

3.5.4 Arapça Veri Kümesi Üzerinde Karşılaştırma

DNN, CNN ve RNN Modellerimiz, Arapça test verileri ile tensorboard ortamında optimum dönem (epoch) sayıları bazında doğruluk ve kayıp grafikleri çizilmiştir.



Şekil 3.34: DNN, Arapça test verileri dönem bazında doğruluk grafik sonucu

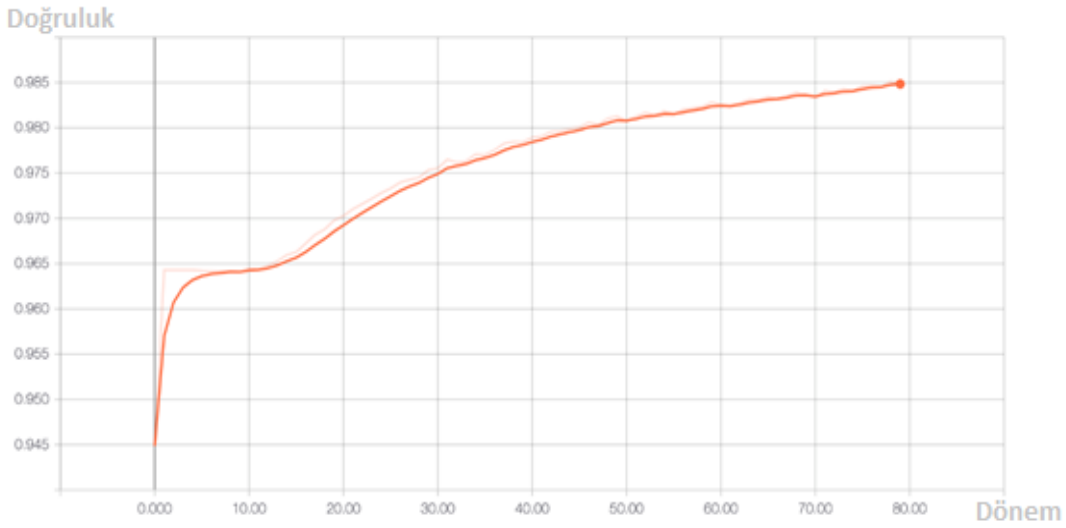


Şekil 3.35: DNN, Arapça test verileri dönem bazında kayıp grafik sonucu

Şekil 3.34 ve Şekil 3.35'de, DNN modelin Arapça veri seti üzerinde doğruluk ve kayıp metrik performansların grafik davranışları gösterilmektedir. Şekil 3.34'te

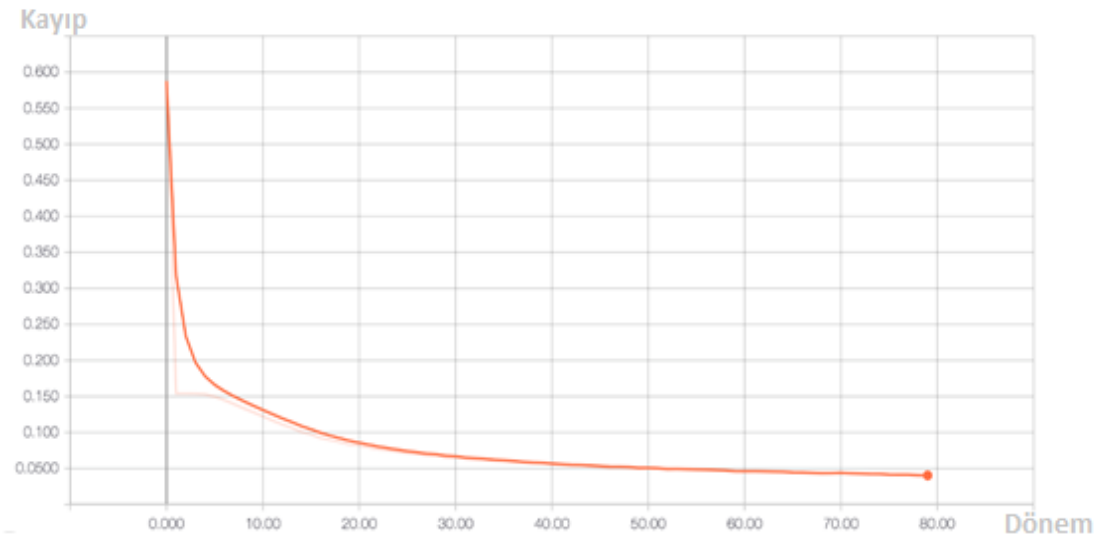
görüldüğü gibi, birinci dönemden (epoch) 0.932 doğruluk değeri ile başlayıp yavaş yavaş artmaya devam ederek on beşinci dönemden sonra daha hızlı bir artışı görünmüştür. Optimum dönem sayı olan 40'a vararak test tamamlanıp 0.9648 (% 96.48) ortalama doğruluk değeri kaydedilmiştir. Şekil 3.35'e bakıldığında, kayıp değeri birinci dönemden 0.978 değeri ile azalmaya başlayarak otuzuncu döneme varmasıyla monoton hareketle test tamamlanıp 0.150 ortalama kayıp değeri verilmiştir.

DNN modeli Arapça veri seti ile doğrulama grafiğinin gittikçe arttığı ve kayıp grafiğinin gittikçe azaldığını izlenmiştir. Modeli iyi eğitilerek test gerçekleştirildiği anlamına gelmektedir.



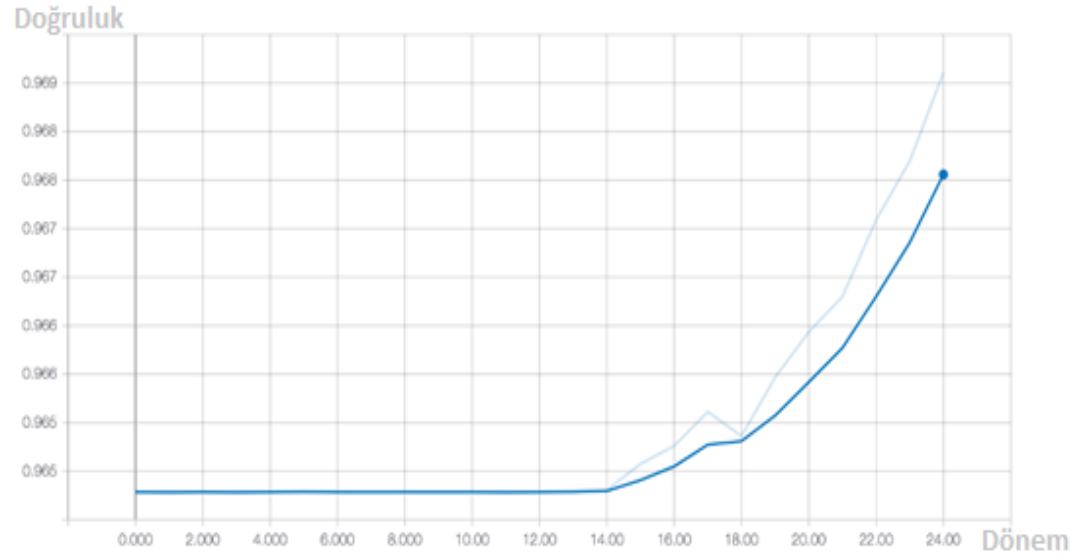
Şekil 3.36: CNN, Arapça test verileri dönem bazında doğruluk grafik sonucu

CNN, Arapça test veri setinden şekil 3.36 ve şekil 3.37'lerde davranışları görülmektedir. Şekil 3.36'de birinci dönemden 0.945 doğruluk değeri artmaya başlayarak 80 olan uygun dönem diğerine vararak 0.990 (%99) ortalama doğruluk değeri kaydedilmiştir. Şekil 3.37'de ise kayıp grafiği 0.588 ile azalmaya başlayarak 0.028 ortalama kayıp değeri test tamamlanmıştır.

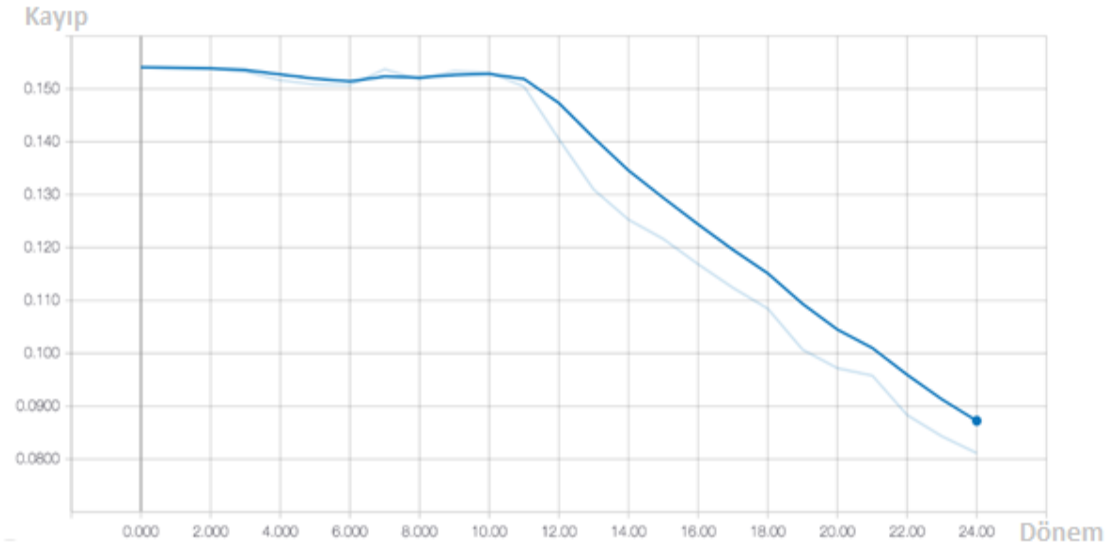


Şekil 3.37: CNN, Arapça test verileri dönem bazında kayıp grafik sonucu

CNN modelinde Arapça veri seti ile Şekil 3.36'de doğruluk değerinin gittikçe arttığı, Şekil 3.37'de ise kayıp değeri azaldığı izlenmiştir. Bu açıdan, modeli iyi eğitildiği ve test edildiği sonucuna varılmaktadır.



Şekil 3.38: RNN, Arapça test verileri dönem bazında doğruluk grafik sonucu

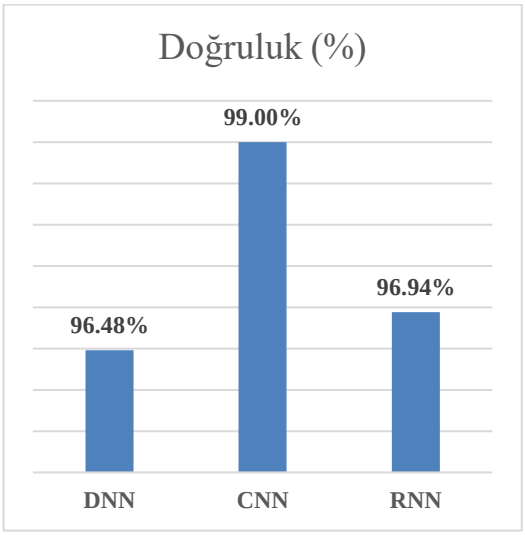


Şekil 3.39: RNN, Arapça test verileri dönem bazında kayıp grafik sonucu

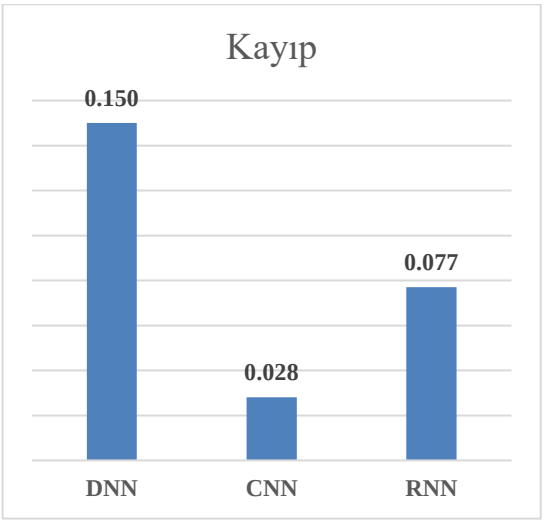
RNN modeli, DNN ve CNN modelleri benzer bir durumu izlenmiştir. Şekil 3.38 ve şekil 3.39'lardan doğruluk değeri gittikçe arttığını ve kayıp değeri gittikçe azaldığını görülmektedir. Şekil 3.38'de birinci dönemden 0.964 doğruluk değeri başlayarak on dördüncü döneme kadar monoton bir şekilde davrandığını izlenmiştir. On dördüncü dönemden sonra artmaya başlayarak optimum dönem değerine varmasıyla 0.9694 (%96.94) ortalama doğruluk değeri verilmiştir. Şekil 3.39'de ise 0.154 kayıp değeri ile başlayıp on birinci döneme varıncaya kadar yavaştan hızlı bir şekilde azaldığını görünmektedir. Optimum değerine vardığında 0.077 ortalama kayıp değeri kaydedilmiştir. RNN modelinin doğruluk ve kayıp şekillerinden iyi eğitildiği ve test edildiği gösterilmektedir.

Arapça veri test ile modellerimiz karşılaştırmak için modellere göre Şekil 3.40 ve Şekil 3.41'de görüldüğü üzere doğruluk ve kayıp sütun grafikleri çizilmiştir.

Arapça test veri üzerinde Şekil 3.40 ve Şekil 3.41'de verilen sütun grafiklerine baktığımızda %99.00 doğruluk ve 0.028 kayıp değerleri olan CNN modelini en iyi model olarak görünmektedir. CNN modeli ardından RNN %96.94 doğruluk ve 0.077 kayıp değerleri ikinci iyi performanslı modeli olmuştur. Performansı %96.48 doğruluk ve 0.150 kayıp değerleri olan DNN üçüncü modeli olmuştur.



Şekil 3.40: Arapça seti üzerinde doğruluk değerlerin açısından modellerin karşılaştırılması



Şekil 3.41: Arapça veri seti üzerinde kayıp değerlerin açısından modellerin karşılaştırılması

Deneysel çalışmalar ardından modellerimizin veri setleri bazında elde edilen doğruluk sonuçları Tablo 3.11'de özetlenmiştir.

Tablo 3.11: Modellerin kıyaslaması

	Cifar-10	**Fashion-Mnist**	**Mnist**	**Arapça**
DNN	90.35	96.38	99.53	96.48
CNN	93.72	98.38	99.88	99.00
RNN	90.00	97.99	99.05	96.94

3.6 Benzer Çalışmalar ile Sonuçlarımızı Kıyaslama

Bu bölümde yapmış olduğumuz çalışma elde edilen iyi modeli, benzer model ve kullanılan aynı veri seti ile karşılaştırılacaktır. Ahmet El-Sawy vd. çalışmalarında CNN modeli oluşturarak Arapça veri seti üzerinde eğitim ve test gerçekleştirilmişlerdir. Diğer taraftan, Ufuk Can Bicici vd. CNN modeli oluşturarak Fashion-Mnist veri seti üzerinde eğiterek test edilmiştir. CNN modelinde Ahmet El-Sawy v.d.'in kullanmış oldukları veri setini bizde kullanmıştık. CNN modelimiz için de Fashion-Mnist üzerinde eğitim ve test gerçekleştirilmiştir. Tablo 3.12 ve Tablo 3.13'de ifade edilen iki çalışmalar kıyaslanması gösterilmektedir.

Tablo 3.12: CNN benzeri modelle karşılaştırılması (Arapça veri seti)

Çalışma yapan	Veri seti	Eğitim, Doğrulama ve test verileri	Doğruluk (%)
E. Ahmet vd [14]	Arapça	16800 Resim 113440 Eğitim resim 3360 Test resim	94.9
Biz	Arapça	16800 Resim 12096 Eğitim resim 1344 Doğrulama resim 3360 Test resim	99.00

Tablo 3.13: CNN benzeri modelle karşılaştırılması (Fashion-Mnist veri seti)

Çalışma yapan	Veri seti	Eğitim, Doğrulama ve test verileri	Doğruluk (%)
U. C. Bicici[35] CNN Baseline	Fashion-Mnist	70000 Resim 60000 Eğitim resim 10000 Test resim	92.27
U. C. Bicici[35] CNN Baseline(*)	Fashion-Mnist	70000 Resim 60000 Eğitim resim 10000 Test resim	91.96
Biz	Fashion_Mnist	70000 Resim 48000 Eğitim resim 12000 Doğrulama resim 10000 Test resim	97.95

Tablo 3.12'den anlaşılacağı üzere, Ahmet El-Sawy v.d. oluşturdukları 16440 resimli Arapça el yazısı veri setinin 13440'i eğitim ve 3360'i ise test olarak ayrılmıştır.

Oluşturdukların CNN modeline o veri seti eğiterek test ettiklerinden %94,6 oranında bir doğruluk değeri elde etmişlerdir. Bizim çalışmamızda ise aynı veri kullanarak 12096 eğitim, 1344 doğrulama ve 3360 test veri olarak ayrılmıştır. CNN modelimiz eğiterek test edildiğimizde %99.0 bir doğruluk değeri elde edilmiştir.

Tablo 3.14'dan ise Ufuk Can BİCİCİ vd. tarafından oluşturulan CNN modelleri Fashion-Mnist veri seti resimlerinin 60000'i eğitim ve 10000'ini test olarak kullanarak modellerin eğitmesi ve test edilmesi ardından % 92.27 ve % 91.96 sonuçları elde etmişlerdir. Oluşturduğumuz modeli ise Fashion-Mnist veri setinin 48000 eğitim, 12000 doğrulama ve 10000 test resim kullanarak eğitim ve testi gerçekleştirmesiyle % 97.95 bir sonucu elde edilmiştir.

3.7 Bulgular

Derin Sinir Ağı, Evrişimsel Sinir Ağı ve Yinelemeli Sinir Ağı (LSTM) modelleri oluşturularak algoritmaların performansları 4 ayrı veri kümeleri kullanılarak karşılaştırılmıştır. Deneysel çalışmalarda kaydedilen bulgular şu şekilde ifade edilmiştir.

- CNN modeli, Mnist veri seti ile en iyi performanslı modeli kaydedilmiştir.
- Tüm modellerimizden Mnist veri seti üzerinde daha iyi performans elde edilmiştir.
- Mnist veri seti üzerinde DNN modeli en düşük kayıp değeri sahip olmasına rağmen ikinci en iyi modeli olmuştur.
- Tüm modellerimiz RNN(LTSM) hariç, doğruluk değeri gittikçe arttıklarını ve kayıp değerleri gittikçe azaldıklarını kaydedilmiştir.
- RNN (LSTM) modeli, Cifar-10 veri seti ile doğruluğunu önce ufak artış ardından ufak düşüş ile monoton bir şekilde eğitim tamamlanarak en uzun eğitim ve test süresi olmuştur. Bu da iyileştirici sebep olabilmektedir.
- Modellerimiz benzer çalışmalarla aynı model ve aynı kullandığımız veri seti ile kıyasladığında modellerimizin iyi sonuçlar verdikleri gözlemlenmiştir.
- Karakter tanıma ve resim sınıflandırmada problemler için RNN yöntemi iyi bir sonuç vermesine rağmen pek önerilmemektedir.

4. ÖNERİLEN YÖNTEM

Bu önerilen yöntemde, Parçacık Sürü Optimizasyon (Partical Swarm Optimization -PSO) algoritmasının bir çeşidi olan İkili Parçacık Sürü Optimizasyonu (Binary Particle Swarm Optimization) algoritması zıt konumluluk kavramı (Opposition-Based Learning - OBL) ile özgün bir kullanımı sağlanarak Arapça veri kümesindeki resimlerin boyutları azaltılması amaçlanmıştır. İkili Parçacık Sürü Optimizasyon ve zıt konumluluk kavramı ile arama uzayının farklı bölgelerinde arama sağlanarak çeşitliliğin artırılması öngörülmüştür. OBL ile birlikte oluşturulan ikili PSO algoritmasıyla ile seçilen özniteliklerin DNN modeli ile resim tanıma ve sınıflandırması işlemi yapılarak sonuca ulaşması sağlanmıştır. Sürü içerisindeki her parçacık her nesilde sınıflandırma işlemi problemini uygulayarak en iyi uygunluk değerine sahip alt kümeye ulaşmayı amaçlamaktadır.

Bu bölümde, Parçacık Sürü Optimizasyon, İkili Parçacık Sürü Optimizasyon, zıt konumluluk kavramı ve önerilen yöntemi olan zıt konumluluk kavramı ile İkili Parçacık sürü Optimizasyonu metotlarından bahsederek deneysel çalışması bilgiler verilmiştir.

4.1 Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)

Parçacık Sürü Optimizasyonu (PSO) popülasyona dayalı bir algoritma olarak, 1995 yılında Kennedy ve Eberhart tarafından geliştirilen stokastik bir optimizasyon tekniğidir. PSO, bir kuş sürüsünün uçuş modelini modellemeye çalışmaktadır. PSO'da, her bir parçacık n-boyutlu bir arama alanı içinde bir aday çözümü temsil etmektedir[36]. PSO'da optimum bir çözüm bulabilmek için sürü (swarm) olarak adlandırılan parçacıklardan her biri çözüm adayı oluşturmaktadır.

Çözüm arama işleminin başlaması için sürüdeki bütün parçacıların çözüm uzayında rastgele değerler almaktadır. Her parçacık belirli bir çözümü temsil eden bir konum bileşenine (position) ve çözüm uzayında bir parçacığın hareketinin yönünü temsil eden bir hız bileşenine (velocity) sahiptir. PSO, üç adımda çalışmaktadır. İlk adımda, popülasyon başlatılması için her parçacığın hız ve pozisyon bileşenleri oluşturmaktadır. İkinci adımda, popülasyondaki parçacıkların pozisyonlarıyla temsil

edilen çözümler değerlendirilmektedir. Üçüncü ve son adımda ise Denklem 4.1 ve 4.2 görüldüğü gibi parçacıkların hızları ve konumlarını güncellenmektedir.

$$V_i^{t+1} = w*V_i^t + c_1 *rand* (pbest - X_i^t) + c_2 *rand* (gbest - X_i^t) \quad 4.1$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad 4.2$$

Denklem 4.1 ve 4.2'lerde V_i^{t+1} ve X_i^{t+1} parçacık *i*'nin (*t+1*) anda ki hız ve konum bileşenleri, V_i^t ve X_i^t parçacık *i*'nin (*t*) anda ki hız ve konum bileşenleri, *t* iterasyon sayısı, c_1 ve c_2 hızlandırma katsayıları, *rand* [0 1] aralığındaki rastgele üretilen sayılar, *w* atalet ağırlığı, *pbest* parçacığın en iyi yerel değeri, *gbest* sürünün en iyi değeri temsil etmektedir [37].

4.1.1 İkili Parçacık Sürü Optimizasyonu (Binary Partical Swarm Optimization - BPSO)

Parçacık sürü optimizasyon algoritmasının çeşitlerinden biri olan ikili parçacık sürü optimizasyon algoritması ayrık (discrete) problemlerin çözümünde kullanılan bir yöntemdir. Öznitelik Seçimi işlemini gerçekleştirilmesinde 0 ve 1 değerlerine sahip olması şarttır. Bunu da PSO algoritmasını ikili PSO algoritması haline getirerek yapılabilmektedir. İkili PSO algoritmasında aktivasyon fonksiyonu olarak Sigmoid fonksiyon kullanılmıştır [37].

4.2 Zıt Konumluluk Kavramı (Opposition-Based learning - OBL)

Zıt konumluluk kavramı, birbirinden farklı durumlarda Diferansiyel Evrim Algoritmasının performansını artırmak için uygulanan bir tekniktir. OBL konsepti ilk defa Tizhoosh tarafından önerilmiştir ve çeşitli optimizasyon algoritmalarına başarıyla uygulanmıştır [38]. Herhangi rastgele bir sayının zıt konumlu durumu çözüme rastgele sayıdan büyük ihtimalle daha yakındır [39]. Bundan dolayı, bir sayının zıt konumlu değeri ile birlikte oluşturulan başlangıç popülasyonunun, en iyi çözüme yakınsamak için daha küçük bir arama uzayına ihtiyaç duyacağı söylenebilmektedir. Bu işlem değerlendirmeyi hızlandırabilmektedir.

$$x' = a + b - x \quad 4.3$$

[a, b] aralığında tanımlanan bir gerçek x sayı olsun. *x*'in Zıt konumlu sayısı (*x*') ile gösterilmektedir. Zıt konum matematik ifadesi denklem 4.3'te tanımlanmıştır [39]. Zıt konumluluk kavramı, sürüdeki parçacıkların deneyimlerini ve ortak bilginin güncellenirken küresel arama kabiliyetini arttırması ile vaktinden önce yakınsamayı önlemek için sunulan bir çekirdek olarak kullanılmıştır [37].

4.3 Zıt Konumluluk Kavramı ile İkili Parçacık Sürü Optimizasyonu

Zıt konumluluk kavramının ikili parçacık sürü optimizasyon algoritması ile birlikte kullanımı çözüm uzayının daha küçük olmasını sağlamaktadır. Bu işlem sürünün oluşumunda her parçacığın bir zıt parçacığı oluşturularak gerçekleştirilmektedir. Zıt parçacıklar, parçacıkların hız bileşenlerinin zıt noktaları alınarak oluşturulan parçacıklardır. Oluşturulan her bir zıt parçacığın hız bileşenlerine göre konum değerleri hesaplanarak uygunluk fonksiyonuyla uygunluk değerleri hesaplanmaktadır. Parçacık sayısının N olduğunu kabul ederek; sürüdeki parçacıklar ve zıt parçacıkları birleştirilmektedir, içlerinden en iyi uygunluk değerine sahip N parçacık seçilerek optimizasyon işlemine devam edilmektedir [37].

Bu çalışma kapsamında, Algoritma 4.1'de sözde kod üzerinde görüldüğü gibi uygunluk değeri sınıflandırma gerçekleştirmek üzere DNN modeli kullanılarak elde edilmiştir. Her bir parçacık her iterasyonda DNN modeli ile seçili öznitelikleri kullanarak bir resim tanıma ve sınıflandırma işlemleri gerçekleştirilmiştir. Seçili öznitelikler ise parçacığın konum bileşeni aracılığıyla elde edilmiştir.

Her iterasyonda sürünün en son geçerleme doğruluk değeri en iyi uygunluk değeri olarak kabul ederek her parçacığın kendi en iyi uygunluk değeri güncellenmiştir. Bu sayede iterasyon sonunda en yüksek uygunluk değerine sahip parçacık elde edilmiştir [37].

4.4 Deneysel Çalışma

Deneysel çalışmalar, Python programlama dili kullanılarak Jupyter Notebook ortamından hazırlanan kodu koşturulmuştur. JupyterNotebook üzerinden BPSO ve OBL kavramıyla yapılan öznitelik seçimi ile alınan sonuçlara göre DNN Arapça veri seti ile sınıflandırma işlemi gerçekleştirilmektedir.

Algoritma 4.1. Zıt Konumlu Parçacık Sürü Optimizasyonu Algoritması ile Öznitelik Seçiminde Sözde Kodu [37]

Zıt Konumlu Parçacık Sürü Optimizasyonu Algoritması ile Öznitelik Seçimi Sözde Kodu

```
1:  Girdi: Parçacık_sayısı (N), Problem_boyutu (d), iterasyon_sayısı(max_iter)
2:  Çıktı: Global en iyi parçacık
3:  Parçacıkların (P) random hız ve konumlarını d boyutunda oluştur
4:  Başlangıçta parçacıkların DNN ile uygunluk değerini hesapla
5:  Zıt parçacıkları oluştur ve DNN ile uygunluk değerlerini hesapla
6:  Parçacıklar ve zıt parçacıkları birleştirip (2N), en iyi N tanesi ile devam et
7:  iter := 0
8:  while (iter <max_iter) do
9:      for i to N
10:         Pi parçacığının hızını güncelle
11:         Pi parçacığının konumunu güncelle
12:         Pi parçacığının uygunluk değerini hesapla
13:     endfor
14:     Sürünün global en iyi parçacığını güncelle
15:     iter :=iter + 1
16: endwhile
```

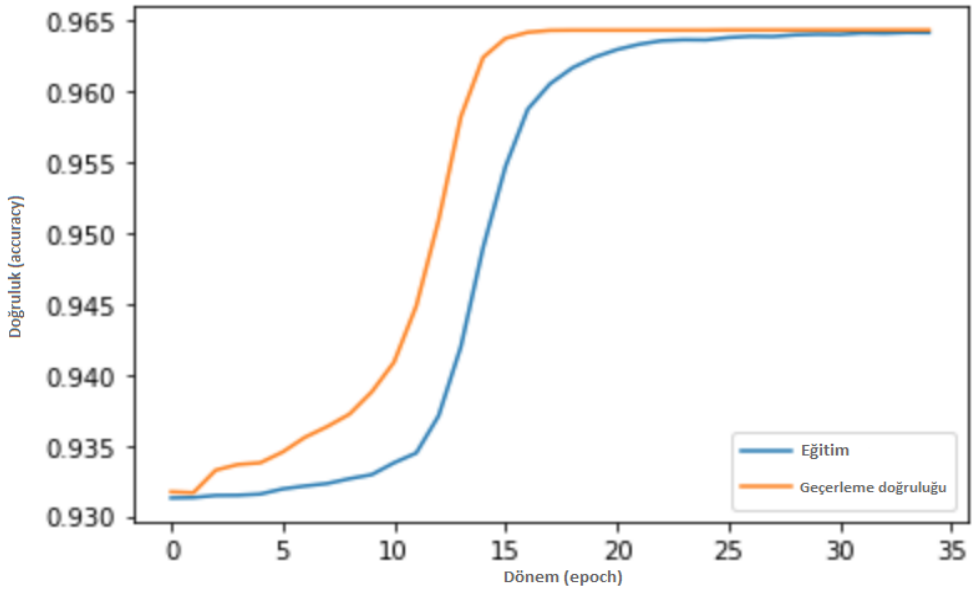
Zıt konumluluk kavramının ikili parçacık sürü optimizasyon algoritması ile birlikte kullanımıyla parçacıkların zıt durumları kullanılarak parçacıklar oluşturulmuştur. Her iterasyonda parçacıkların konum ve hız değerleri güncellenerek en son geçerleme doğruluk değeri seçilmiştir. Problem çözümünde fonksiyonun veya problem boyutu d = 10, iterasyon sayısı = 50, parçacıkların maksimum ve minimum konum (x) değerleri [-20, 20] aralığında, hız (v) değerleri ise [-4,4] aralığında başlangıçta rasgele olarak belirlenmiştir. Öğrenme faktörleri $c1 = c2 = 1.42694$ ve atalet ağırlığı $w = 0.689343$ parametreler olarak tanımlanmıştır. Hız, öğrenme faktörleri ve atalet ağırlığı [40] numaralı kaynaktan referans alınarak belirlenmiştir.

Arapça veri kümesi için DNN modelini gizli katmanlar için art arda 1024, 512 ve 1024 nöron sayıları; 0.1, 0.9, 0.01 seyretme değerleri; 128 toptan boyu, 35 dönem sayısı ve sgd eniyileme algoritması olarak hiper-parametreleri seçilmiştir.

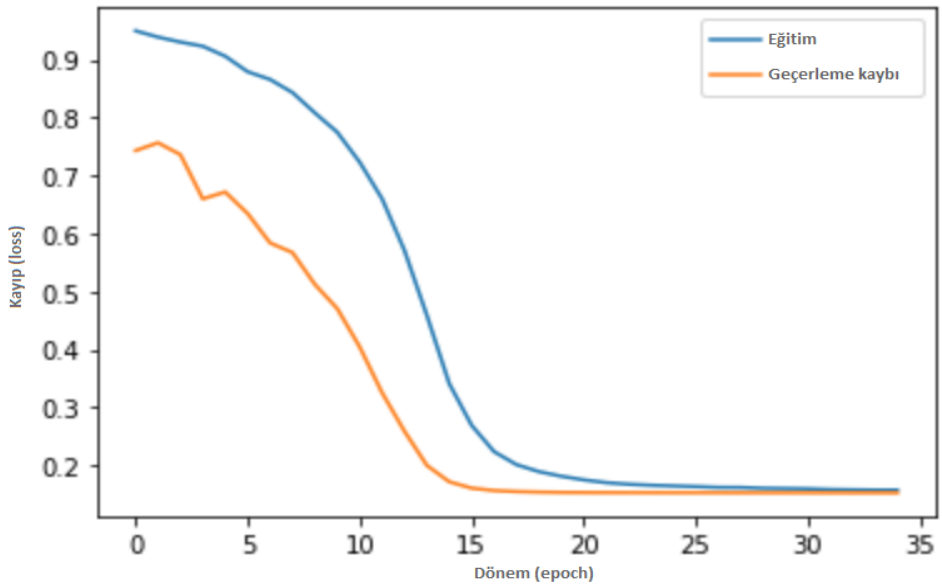
4.5 Sonuçlar

Kodu, toplam 1750 defa ve 2 gün 18 saat 53 saniye bir sürede çalıştırılmasıyla eğitim boyunca Şekil 4.1'de görüldüğü gibi doğruluk ve geçerleme doğruluğu

grafikleri gittikçe arttığı kaydedilmiştir. Diğer taraftan da, Şekil 4.2'de görüldüğü üzere eğitim esnasında, kayıp ve geçerleme kaybı grafikleri gittikçe azalttıkları kaydedilmiştir. Grafiklerden izlenen bu davranışları, BPSO-OBL-DNN modeli iyi eğitildiği ve test edildiği anlamına gelmektedir.



Şekil 4.42: Önerilen yöntemin doğruluk ve geçerleme doğruluğu grafikleri



Şekil 4.2: Önerilen yöntemin kayıp ve geçerleme kaybı grafikleri

Algoritmayı test verileri üzerinde çalıştığında 0.9643 (%96.43) doğruluk ve 0.1531 kayıp değerleri elde edilmiştir. Tablo 3.14'te görüldüğü üzere, BPSO-OBL-DNN modeli ile alınan sonuçları ve bölüm 3.5.4'teki DNN modeli Arapça veri kümesi ile alınan sonuçları (%96.48 doğruluk ve 0.150 kayıp) kıyasladığında birbirleri yakın oldukları kaydedilmiştir. DNN modeline göre, BPSO-OBL-DNN modeli doğruluk açısından 0.005 düşük bir fark kaydederken kayıp değeri bakımından 0.031 bir artışı kaydedilmiştir.

Tablo 3.14: BPSO-OBL-DNN ve DNN modellerin karşılaştırılması

BPSO-OBL-DNN		**DNN**	
Doğruluk (%)	**Kayıp**	**Doğruluk (%)**	**Kayıp**
96,43	0.1531	96.48	0.150

5. SONUÇ

Bu tez kapsamında, Derin Sinir Ağı (DNN), Evrişimsel Sinir Ağı (CNN) ve Yinelemeli Sinir Ağı (RNN) Derin Öğrenme Algoritmaları, el yazı karakter tanıma ve resim sınıflandırma problemlerinin performansları arttırmak için ileri düzeydeki iyileştirici fonksiyon ve optimum hiper-parametrelerini belirleyerek elde edilecek sonuçları önce modellerin, sonrada benzer yapılan çalışmada aynı veri seti ve aynı model göz önünde bulundurularak karşılaştırma yapılmıştır. DNN, CNN ve LSTM olan RNN'in spesifik algoritması modelleri oluşturarak algoritmaların performanslarını ölçtürülmüştür. DNN modeli; bir giriş, üç gizli ve bir çıkış katmanlardan oluşturmaktadır. Her bir gizli katmanda bir ReLU aktivasyon eklenmiş durumda ve ardından bir seyreltme (Dropout) katmanı ilave edilmiştir. LSTM, DNN benzer bir yapı sahiptir, bir giriş, üç adet seyretmeli LSTM ve çıkış katmanlardan şekillendirilmiştir. CNN ise VGG evrişimsel modelinden dayanarak dört adet evrişim, 2 adet havuzlama, 2 adet tam bağlı katmanlardan biçimlendirilmiştir. Her bir havuzlama ve tam bağlı katmandan sonra bir seyreltme katmanı eklenmiştir. Her modelin Çıkış katmanı, softmax aktivasyonlu 10 veya 28 nörondan oluşan veri kümesine göre, bireysel çıktı olasılığını tahmin etmesi için eklenmiştir. Modeller çalıştırmadan önce, optimum bir sonuçları elde etmek için nöron sayıları, seyreltme değerleri, toptan boyu (batch-size), dönem (epoch) sayıları, iyileştirici (SGD, Adagrad, Adadelta, Adam, RMSprop) gibi hiper-parametreleri tensorflow, keras scikit learn kütüphaneleri kullanılmıştır.

Modeller eğitmesi ve test gerçekleştirmek üzere, Cifar-10, Fashion-Mnist, Mnist ve Arapça veri setleri kullanılmıştır. Elde edilen bulgulara göre, CNN modeli en iyi olduğunu gösterilmiştir. Literatürde, yapılmış benzer çalışma modellerin sonuçları da karşıladığında modellerimiz iyi performansları sergiledikleri kaydedilmiştir. DNN, CNN veya RNN modellerinin eğitim ve test grafiklerini incelerken (RNN Cifar-10 haricinde) doğruluk oranı artarken, kayıp değerleri giderek azalıklarını görünmüştür. Bu da modellerin çok iyi eğitildikleri anlamına gelmektedir. RNN modelin Cifar-10 veri setiyle eğitimi esnasında, doğruluk belli bir noktaya kadar arttıktan sonra azalıp monoton bir şekilde eğitimini tamamladığı kaydedilmiştir. Kayıp değerleri de belli bir noktaya kadar azaldıktan sonra artarak monoton bir şekilde

eğitimini tamamlanmıştır. Ayrıca, modellerin GPU üzerinde çalıştırmalarına rağmen, RNN modelinin Cifar-10 veri seti üzerindeki eğitim ve test süresi iki gün 9 saat 22 dakika 42 saniye süre ile gözlemlenen en uzun eğitim modeli olmuştur. Modellerin aralarında karşılaştırması en iyi performanslı CNN modeli Mnist veri seti ile %99.88 doğruluk ve 0.042 kayıp değeri olmuştur.

Bu çalışmada, elde edilen sonuçlar ile gerçek hayattaki el yazı karakter tanıma ve resim sınıflandırma problemlerinin çözümünde derin öğrenme yaklaşımları istifade edilebileceğini gösterilmiştir.

Bu çalışmanın yanında zıt konumluluk kavramı ile İkili Parçacık Sürü Optimizasyonu DNN modeline uygulanarak bir yöntem önerilmiştir. Önerilen yöntemin amacı çözüm uzayının daha küçük olmasını sağlanarak ardından DNN modeli Arapça veri kümesi ile resim tanıma ve sınıflandırmaktır. Önerilen yöntemin olan BPSO-OBL-DNN sonuçları ve DNN Arapça veri seti ile alınan sonuçları birbirleri yakın oldukları kaydedilmiştir.

Bu tezde elde edilen sonuçların ışığında, gelecekte tıp alanında yapılacak çalışmalarda gerçek veri setleri ile oluşturulan modellerin eğitilerek test edilmesi planlamaktadır. Eğitim ve test için örnek olarak diyabet tanısı, elektronik sinyal analizi, tıbbi görüntü analizi ve radyoloji alanlarında yapılacak uygulamalar gösterilebilir. Tüm bunların yanı sıra DNN-CNN, DNN-RNN ve RNN-CNN-RNN gibi hibrid modelleri oluşturularak eğitim ve test gerçekleştirilmesi düşünülmektedir.

KAYNAKÇA

[1] **H. Wehle**, "Machine learning, deep learning, and ai: What's the difference?", *In Internationan Conference on Data scientist innovation day, Bruxelles, Belgium,* July 2017.

[2] **I. Goodfellow, Y. Bengio, ve A. Courville**, "Deep Learning". *MIT Press*, 2016.

[3] **Mathworks**, "What Is Deep Learning? 3 things you need to know", *www.mathworks.com*. [Çevrimiçi]. Available at: https://www.mathworks.com/discovery/deep-learning.html. [Erişim: 01-Ağu-2019].

[4] **Jay Alammar**, "A Visual and Interactive Guide to the Basics of Neural Networks – Jay Alammar – Visualizing machine learning one concept at a time", *jalammar.github.io*, 2018. [Çevrimiçi]. Available at: http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/. [Erişim: 01-Ağu-2019].

[5] **E. Yazan ve M. F. Talu**, "Comparison of the stochastic gradient descent based optimization techniques", *In Proceedings of the 2017 International Symposium od Artificial Interlligence and Data Processing (IDAP), Malatya,Turkey, 16-17 September 2017,* ss. 1–5, 2017.

[6] **M. D. Zeiler**, "ADADELTA: an adaptive learning rate method", *arXiv preprint arXiv: 1212.5701*, 2012.

[7] **D. P. Kingma ve J. Ba**, "Adam: a method for stochastic optimization", CoRR *abs/1412.6980*, ss. 1–15, 2014.

[8] **Andrej Karpathy**, "Convolutional Neural Networks for Visual Recognition", *cs231n.github.io*, 2018. [Çevrimiçi]. Available at: http://cs231n.github.io/neural-networks-1/. [Erişim: 20-Ağu-2018].

[9] **Anish Singh Walia**, "Activation functions and it's types-Which is better?", *towardsdatascience.com*, 2017. [Çevrimiçi]. Available at: https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f. [Erişim: 20-Ağu-2018].

[10] **Avinash Sharma V**, "Understanding Activation Functions in Neural Networks", *medium.com*, 2018. [Çevrimiçi]. Available at: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. [Erişim: 20-Ağu-2018].

[11] **D. Li ve Y. Dong**, "Deep Learning Methods and Application", *Foundations and Trends in Signal Processing*, c.7, sayı 3-4, ss.197, 2013.

[12] **M. A. Nielsen**, "Neural Networks and Deep Learning", *neuralnetworksanddeeplearning.com*, 2015. [Çevrimiçi]. Available at: http://neuralnetworksanddeeplearning.com/index.html. [Erişim: 18-Ağu-2018].

[13] **N. Audebert, B. Le Saux, ve S. Lefèvre**, "Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images", *Remote Sens.*, c. 9, sayı 4, ss. 368, 2017.

[14] **A. El-Sawy, M. Loey ve E. B. Hazem**, "Arabic handwritten characters recognition using convolutional neural network", *WSEAS Transactions on. Computer Research*, c.5, sayı 1, ss. 11-19, 2017.

[15] **S. S. Talathi ve A. Vartak**, "Improving performance of recurrent neural network with relu nonlinearity", *arXiv preprint arXiv: 1511.03771*, 2015.

[16] **Skymind**, "A Beginner's Guide to LSTMs and Recurrent Neural Networks | Skymind", *skymind.ai*, 2018. [Çevrimiçi]. Available at: https://skymind.ai/wiki/lstm. [Erişim: 17-Ağu-2018].

[17] **Christopher Olah**, "Understanding LSTM Networks", *colah.github.io*, 2015. [Çevrimiçi]. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Erişim: 17-Ağu-2018].

[18] **Andrej Karpathy**, "The Unreasonable Effectiveness of Recurrent Neural Networks", *karpathy.github.io*, 2015. [Çevrimiçi]. Available at: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Erişim: 17-Ağu-2018].

[19] **R. Tolosana, R. Vera-Rodriguez, J. Fierrez ve J. Ortega-Garcia**, "Exploring Recurrent Neural Networks for On-Line Handwritten Signature Biometrics", *IEEE Access*, c. 6, ss. 5128–5138, 2018.

[20] **D. Sen Maitra, U. Bhattacharya ve S. K. Parui**, "CNN based common approach to handwritten character recognition of multiple scripts", *In Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2015.

[21] **D. Cireşan, U. Meier, J. Masci ve J. Schmidhuber**, "Multi-column deep neural network for traffic sign classification", *Neural Networks*, c. 32, ss. 333–338, Ağu. 2012.

[22] **X. Y. Zhang, F. Yin, Y. M. Zhang, C. L. Liu ve Y. Bengio**, "Drawing and Recognizing Chinese Characters with Recurrent Neural Network", *IEEE Trans. Pattern Anal. Mach. Intell.*, c. 40, sayı 4, ss. 849–862, 2018.

[23] **A. Graves**, "Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks", *Guid. to OCR Arab. Scripts*, ss. 297–313, 2012.

[24] **A. Krizhevsky, Vinod Nair ve H. Hinton**, "CIFAR-10 and CIFAR-100 datasets", *www.cs.utoronto.ca*, 2009. [Çevrimiçi]. Available at: http://www.cs.utoronto.ca/~kriz/cifar.html. [Erişim: 01-Eyl-2018].

[25] **Zalando**, "Fashion-MNIST database", *research.zalando.com,* 2017. [Çevrimiçi]. Available at: https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/. [Erişim: 05-Eyl-2018].

[26] **C. Cortes, C. J. Burges ve Y. LeCun**, "MNIST handwritten digit database", *yann.lecun.com*. [Çevrimiçi]. Available at: http://yann.lecun.com/exdb/mnist/. [Erişim: 15-Ağu-2018].

[27] **Hazem M El-Bakry**, "Arabic alphabet characters", *www.researchgate.net,* 2017. [Çevrimiçi]. Available at: https://www.researchgate.net/figure/Arabic-alphabet-characters_fig1_313891953. [Erişim: 26-Şub-2019].

[28] **Jason Brownlee**, "Loss and Loss Functions for Training Deep Learning Neural Networks", *machinelearningmastery.com,* 2019. [Çevrimiçi]. Available at: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. [Erişim: 20-Haz-2018].

[29] **Tensorflow Team**, "Get Started with Tensorflow", *www.tensorflow.org*, 2018. [Çevrimiçi]. Available at: https://www.tensorflow.org/tutorials. [Erişim: 20-Ağu-2018].

[30] **E. B. Martín Abadi, Ashish Agarwal, Paul Barham *vd.***, "TensorFlow: Large-scale machine learning on heterogeneous systems", *www.tensorflow.org*, 2015. [Çevrimiçi]. Available at: https://www.tensorflow.org/about/bib. [Erişim: 20-Haz-2018].

[31] **Databricks**, "Visualisation with TensorBoard", *databricks.com,* 2018. [Çevrimiçi]. Available at: https://databricks.com/tensorflow/visualisation. [Erişim: 21-Ara-2018].

[32] **Thushan Ganegedara**, "TensorBoard Tutorial For Beginners (article) - DataCamp", *www.datacamp.com,* 2018. [Çevrimiçi]. Available at: https://www.datacamp.com/community/tutorials/tensorboard-tutorial. [Erişim: 21-Ara-2018].

[33] **Keras Google group**, "Keras: The Python Deep Learning library", *keras.io*. [Çevrimiçi]. Available at: https://keras.io/. [Erişim: 27-Tem-2018].

[34] **N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, ve R. Salakhutdinov**, "Dropout: a simple way to prevent neural networks from overfitting", *Journal machine learning*, c.15, sayı 1, ss.1929-1958, 2014.

[35] **U. C. Bicici, C. Keskin, ve L. Akarun**, "Conditional Information Gain Networks", *Proc. - Int. Conf. Pattern Recognit.*, ss. 1390–1395, 2018.

[36] **T. B. Ludermir ve W. R. De Oliveira**, "Particle Swarm Optimization of MLP for the identification of factors related to Common Mental Disorders", *Expert Syst. Appl.*, c. 40, sayı 11, ss. 4648–4652, 2013.

[37] **M. Özdemir ve Ü. Can**, "Öznitelik Seçme için Parçacık Sürü Optimizasyonu ve Karşıt Konumluluk Öğrenmesinin Birlikte Kullanılması", *Bitirme tezi*, ss. 1–35, 2019.

[38] **B. AI, M.G. Dong, ve C.X. Jang**, "Simple PSO Algorithm with Opposition-based Learning Average Elite Strategy", *Int. J. Hybrid Inf. Technol.*, c. 9, sayı 6, ss. 187–196, 2016.

[39] **H. R. Tizhoosh**, "Opposition-based learning: A new scheme for machine intelligence", *Proc. - Int. Conf. Comput. Intell. Model. Control Autom. CIMCA 2005 Int. Conf. Intell. Agents, Web Technol. Internet*, c. 1, ss. 695–701, 2005.

[40] **L. Shang, Z. Zhou, X. Liu**, "Particle Swarm Optimization-Based Feature Selection in Sentiment Classification", *Soft Computing*, c. 20, ss. 3821-3834, 2016.