**MİMAR SİNAN GÜZEL SANATLAR ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# DERİN ÖĞRENME İLE KARAR VERME SÜREÇLERİ

**DOKTORA TEZİ**

**Olgun AYDIN**

**Anabilim Dalı: İSTATİSTİK**

**Programı: İstatistik**

**Tez Danışmanı: Doç. Dr. Semra Erpolat TAŞABAT**

**MART 2019**

**MİMAR SİNAN GÜZEL SANATLAR ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# DERİN ÖĞRENME İLE KARAR VERME SÜREÇLERİ

**DOKTORA TEZİ**

**Olgun AYDIN**

**İstatistik Anabilim Dalı**

**İstatistik Programı**

**Tez Danışmanı: Doç. Dr. Semra Erpolat TAŞABAT**

**MART 2019**

Olgun AYDIN tarafından hazırlanan DERİN ÖĞRENME İLE KARAR VERME SÜREÇLERİ adlı bu tezin doktora tezi olarak uygun olduğunu onaylarım.

Doç. Dr. Semra ERPOLAT TAŞABAT

Tez Yöneticisi

Bu çalışma, jürimiz tarafından İSTATİSTİK Anabilim Dalında DOKTORA tezi olarak kabul edilmiştir.

Başkan : Doç. Dr. Semra ERPOLAT TAŞABAT

Üye : Prof. Dr. Mehmet Aydın ERAR

Üye : Prof. Dr. Nalan CİNEMRE

Üye : Prof. Dr. Ahmet Mete ÇİLİNGİRTÜRK

Üye : Doç. Dr. Ersoy ÖZ

Bu tez, Mimar Sinan Güzel Sanatlar Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

Mimar Sinan Güzel Sanatlar Üniversitesi Fen Bilimleri Enstitüsü tez yazım klavuzuna uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel etik kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- ücret karşılığı başka kişilere yazdırmadığımı (dikte etme dışında), uygulamalarımı yaptırmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

Olgun AYDIN

***Aileme,***

## TEŞEKKÜR

Tez çalışmam ve tüm doktora serüvenim süresince desteklerini hiç esirgemeyen, her zaman anlayışla ve sabırla yol gösteren, tecrübelerini paylaşan danışmanım sayın Doç. Dr. Semra Erpolat TAŞABAT'a, beni her daim destekleyen, yanımda olan, bir an olsun beni motivasyonsuz bırakmayan çok sevgili partnerim Joanna Wrobel'e, her zaman olduğu gibi tez çalışmam süresince de manevi desteklerini hiç esirgemeyen anne ve babama teşekkürü bir borç bilirim.

**İÇİNDEKİLER**

**Sayfa**

## KISALTMALAR

**ANN** : Artificial Neural Network (Yapay Sinir Ağı)
**AI** : Artificial Intelligence (Yapay Zeka)
**DNN** : Deep Neural Networks (Derin Yapay Sinir Ağları)
**DL** : Deep Learning (Derin Öğrenme)
**LSTM** : Long Short-Term Memories (Uzun Kısa Süreli Hafızalar)
**RNN** : Recursive Neural Networks (Özyinelemeli Sinir Ağları)
**CNN** : Convolutional Neural Networks (Evrişimsel Sinir Ağları)
**ML** : Machine Learning (Makine Öğrenmesi)
**GA** : Genetic Algorithm (Genetik Algoritma)
**GS** : Grid Search (Izgara Araması)
**PdM** : Predictive Maintenance (Tahminleyici Bakım)
**DO** : Dropout Layer (Bırakma Katmanı)
**IIoT** : Internet of Industrial Objects (Endüstriyel Nesnelerin İnterneti)
**CPU** : Central Process Unit (Mikro İşlemci)
**GPU** : Graphical Process Unit (Grafik İşlemci Birimi)

## ÇİZELGE LİSTESİ

**Sayfa**

## ŞEKİL LİSTESİ

**Sayfa**

# DERİN ÖĞRENME İLE KARAR VERME SÜREÇLERİ

## ÖZET

Endüstri 4.0, dördüncü endüstri devrimi veya Endüstriyel Nesnelerin İnterneti (IIoT) olarak adlandırılan sanayi akımı, işletmelere, daha verimli, daha büyük bir esneklikle, daha güvenli ve daha çevre dostu bir şekilde üretim yapma imkanı sunmaktadır. Nesnelerin İnterneti ile bağlantılı yeni teknoloji ve hizmetler birçok endüstriyel uygulamada devrim niteliği taşımaktadır. Fabrikalardaki otomasyon, tahminleyici bakım (PdM – Predictive Maintenance) modellerigibi gelişmeler işletmelere, iş modellerini değiştiren yenilikçi çözümler sunmaktadır. Günümüz endüstriyel uygulamalarında ihtiyaç duyulan yüksek otomasyon seviyesini sağlamak için, daha verimli ekipman kullanılmalı, daha akıllı sistemler oluşturulmalıdır. Bu sayede, hem üretimdeki verim artacak hem de daha güvenli bir çalışma ortamı da sağlanmış olacaktır.

İşletmeler, üretimlerine minimum aksama süresiyle, optimum hızda devam etmek istemektedir. Bu sebeple, üretimde kullanılan, hareketli parçaları olan makinelerin daha verimli kullanılmasını sağlamak için yapılacak bakım planlamaları kritik önem taşımaktadır. Bu konu ile ilgili yaklaşımlardan bir tanesi, ekipmanların durumuna bakılmaksızın bakım süreçlerini sabit aralıklarla gerçekleştirmektir. Bu yöntem planlanması basit bir yöntemdir ancak, zaman zaman ekipmanların arızası gerçekleştikten sonra bakım işleminin gerçekleştirilmesine ya da ekipmanlarda hiç bir problem yokken bakım işleminin gerçekleştirilmesine neden olabilmektedir. Bu da sistemde uzun süreli aksamalar, gereksiz bakım maliyetleri gibi sonuçları doğurmaktadır. Bakım süreçlerine farklı bir yaklaşım olan PdM, makinenin gözlemlenen durumuna bağlı olarak bakım süreçlerinin yönetilmesine olanak kılmaktadır. Bu nedenle bakım işlemleri, arızalar meydana gelmeden önce gerçekleştirilebilmektedir.

PdM yaklaşımına yeni bir akış açısı getirmek amacıyla yapılan bu çalışmada yeni bir derin yapay sinir ağı mimarisi önerilmiştir. Bu mimaride bir girdi katmanı bir LSTM katmanı, bırakma (DO) ve ardından yine bir LSTM katmanı, bir gizli katman ve çıktı katmanı bulunmaktadır. Mimaride kullanılan iterasyon sayısı, parti büyüklüğü Genetik Algoritma (GA) kullanılarak, kayıp fonksiyonunu optimize eden optimizasyon algoritması, çıktı katmanında sonra kullanılan aktivasyon fonksiyonu ve DO oranı ızgara araması (GS) kullanılarak belirlenmiştir. Çalışma kapsamında önerilen mimari, GA ve GS kullanılarak geliştirilmesi, sonuçların hiç bir makine öğrenmesi ve derin öğrenme bilgisine sahip olmayan kişiler tarafından da anlaşılması kolay olması sayesinde karar verme süreçlerine doğrudan katkı sağlaması ile literatürdeki ilk çalışma olma özelliğini taşımaktadır.

## DECISION MAKING PROCESS USING DEEP LEARNING


### ABSTRACT

Industry 4.0, the fourth industrial revolution, or the industrial flow of industrial objects (IIoT), provides the opportunity to perform tasks in a timely manner, more efficient, greater flexibility, safer and more environmentally friendly. New technologies and services associated with the Internet of Things are revolutionary in many industrial applications. Developments such as automation in the factories and predictive maintenance models offer enterprises the opportunity to propose innovative solutions that change business models. In order to achieve the high level of automation, more efficient equipment should be used and smarter systems should be created. In this way, a more secure working environment would be ensured although the productivity in production would increase.

Companies generally would like to continue to work with minimal downtime. In addition, maintenance planning to ensure more efficient use of any machine with moving parts is critical. One of the approaches is to perform maintenance processes at fixed intervals regardless of the condition of the equipment. This method is a simple method of planning, but it may cause maintenance of the equipment after the malfunction of the equipment, or it may cause maintenance to be carried out when there is no problem with the equipment. This may cause long-term disruptions or unnecessary maintenance costs. Predictive maintenance, which is a different approach to maintenance processes, enables maintenance actions to be managed depending on the machine's observed condition. Maintenance is therefore carried out before failures occur.

A new deep neural network architecture has been proposed in this study, which is intended to bring a approach to the predictive maintenance approach. In this architecture there is an input layer, an LSTM layer, a dropout layer (DO) followed by an LSTM layer, a hidden layer, and an output layer. The number of iterations used in the architecture and the batch size were determined by using the Genetic Algorithm (GA), optimization algorithm that optimizes the loss function, the activation function used after the output layer, the DO ratio were determined by using grid search (GS). The proposed architecture is the first study in the literature, developed by using GA and GS and contributing directly to the decision-making processes.

# 1. GİRİŞ

Günümüzde bilgisayar sistemleri hayatın vazgeçilmez bir unsurudur ve günlük yaşamın önemli bir parçası haline gelmiştir. Cep telefonları, buzdolapları, televizyonlar, otomobiller ve günlük hayatta sıklıkla kullanılan birçok cihaz, artık bilgisayar sistemlerinin yardımıyla bireyleri yönlendirmekte, hatta çoğu bireyler yerine kararlar verebilmektedir.

Teknolojik gelişmeler ile birlikte, görülmektedir ki bilgisayar sistemleri sadece karmaşık hesaplamalar için kullanılmamaktadır. Bu sistemler, büyük miktarlardaki verilerden anlamlar çıkararak, gerçekleşen ya da gerçekleşebilecek olan durumlar ile ilgili yorum yapabilme kabiliyetine de sahiptir.

Bu noktada araştırmalar, insan beyninin öğrenme becerisini nasıl edindiğinin merak edilmesi ile başlamıştır. Ardından, araştırmacılar çalışmalarını bu becerilerin bilgisayarlara kazandırılması üzerine yoğunlaştırmıştır.

Büyük miktarlarda veri setleri kullanarak çıkarsamalar yapan, akıllı ve kendi kendine öğrenen sistemler oluşturmak için çok katmanlı “derin” sinir ağların kullanılması konseptinden oluşan "Derin Öğrenme(DL – Deep Learning)", son yıllarda oldukça önemli bir noktaya gelmiştir. DL ile sayısız akıllı sistemler oluşturulmuş ve oluşturulmaya devam etmektedir. Günümüzde, uluslararası sahada bir çok teknoloji devi bu yöntemleri; doğal dil işleme, konuşma tanıma, bilgisayarlı görü, online öneri sistemleri, biyoinformatik, video oyunları, arama motorları, çevrimiçi reklamlar ve finans uygulamalarında kullanmaktadır (Kim, 2016).

Bu çalışmada, uçakların turbo motorlarındaki hataları tahmin etmek amacıyla DL yöntemleri kullanılarak yeni bir mimari önerilmiştir. Bu mimari, büyük veri platformlarından Apache Spark kullanılarak eğitilmiştir. Önerilen bu sistem ile motorların üzerindeki sensörlerden gelen verilerin, operatörler tarafından, hiçbir makine ve DL bilgisine gerek duymadan, yorumlanabilmesini mümkün kılan karar verme süreçleri oluşturulmuştur.

Çalışmanın izleyen bölümlerinde öncelikle DL’nin temeli olan makine öğrenmesinden bahsedilmekte, ardından DL’nin temellerinden, DL’nin kullanım alanlarından, DL’de kullanılan ağlardan, mimarilerden, yazılım kütüphanelerinden bahsedilmektedir. Bunu takiben PdM kavramından bahsedilmekte, ardından önerilen yöntem detaylarıyla

tartışılmaktadır. Son olarak da bulgular detaylı olarak sunulmakta ve çıkarsamalar yapılmaktadır.

## 2. MAKİNE ÖĞRENMESİ

Bilgisayar yardımıyla problemleri çözmek için için algoritmalara ihtiyaç duyulmaktadır. Algoritmalar, girdileri çıktılara dönüştürmek için yapılması gereken bir takım talimatlar dizisidir. Örneğin, sıralama problemi için bir algoritma geliştirebilir. Burada girdi, bir sayı kümesidir ve çıktı bu sayı kümesinin sıralanmış halidir. Bu problemi çözmek için bir çok algoritma bulunmaktadır ve bunlardan en verimli olanı bulmak kritik önem taşımaktadır.

Fakat bazı problemlerin çözümü için ne yazık ki hazır algoritmaları bulmak sıralama algoritmasını bulmak kadar kolay olmamaktadır. Örneğin, e-postalar arasındaki spam olan e-postaları, spam olmayan e-postalardan ayırt etmek için hazır bir algoritma yoktur. Bu problem için girdi, metin içeren bir e-posta, çıktı ise, mesajın spam olup olmadığını belirten bir ifadedir. Örneğin çıktı, “Spam”, “Spam değil” şeklinde olabilir. Bu problemde, girdileri çıktılara dönüştürmek bir önceki problemdeki kadar kolay değildir. Spam olarak kabul edilebilecek e-postalar zaman içinde değişkenlik gösterebilmektedir. Çözüm için temel prensip, bazılarının spam olduğu bilinen binlerce örnek e-posta iletilerinin derlenip spam olan e-postaların ne tür içerikler içerdiğini “öğrenmek” tir.

Bilgisayar teknolojisindeki ilerlemelerle birlikte, şu anda büyük miktarda veriyi depolamak, işlemek ve aynı zamanda bu veriye bir bilgisayar ağı üzerinden fiziksel olarak uzak konumlardan erişmek mümkündür. Çoğu veri toplama cihazı artık dijital olarak veri toplama işlemi gerçekleştirmektedir. Örneğin, milyonlarca müşteriye, binlerce mağaza üzerinden ürün satan bir süpermarket zincirinde, satış noktası terminalleri her bir işlemin ayrıntılarını (tarih, müşteri kimlik kodu, satın alınan ürünler ve miktarları, harcanan toplam para vb.) kaydetmektedir. Genellikle, bu tip bir kuruluş her gün gigabaytlarca veri toplayabilmektedir. Süpermarket zincirinin istediği, hangi tip ürünlerin, hangi tip müşteriler tarafından tercih edildiğini tespit edebilmektir. Yine, bu problem için de mevcut bir algoritma bulunmamaktadır.

Hangi tipteki müşterilerin, hangi tipte dondurma lezzetini tercih ettiğini, bir yazarın bir sonraki kitabını satın alma ihtimalini, yeni bir filmi izleme ihtimalini, x şehrini ya da y web sitesini ziyaret etme ihtimali bilinmemektedir. Bu ihtimaller bilinmediği ve bu problemi çözecek mevcut algoritmalar bulunmadığı için veri toplanıp analiz edilip anlamlı sonuçlar çıkarılması gerekmektedir. Tüketici davranışları ile ilgili verilerin üretilmesinin altında yatan süreç ile ilgili ayrıntılar bilinmiyor olsa da, bu verideki patternlerin tamamen rastgele olmadığı bilinmektedir. Tüketiciler, çoğunlukta süpermarketlere rasgele gitmemekte ve ürünleri rasgele satın almamaktadırlar. Örneğin tüketicilerin genellikle, gazlı içecek alırken cips aldıkları; yazın dondurma aldıkları şeklinde çıkarsamalar yapmak mümkündür. Verinin içerdeği patternleri anlamak için gerekli süreç tamamen ve eksiksiz olarak tanımlanamayabilir, ancak iyi ve faydalı bir yaklaşım oluşturulabilmektedir. Bütün süreci tanımlamak mümkün olmamakla birlikte, belirli kalıplar, patternler veya düzenler tespit edilebilir. Bu tür kalıplar sürecin anlaşılmasına yardımcı olmakta ve öngörülerde bulunmak için kullanılabilmektedir.

Makine öğrenmesi yöntemlerinin veri tabanlarındaki verilere uygulanması, veri madenciliği olarak da adlandırılmaktadır. Analoji, işlendiğinde az miktarda çok değerli materyal ortaya çıkan büyük miktarda toprak ve ham maddenin madenlerden çıkarılmasıdır. Benzer şekilde, veri madenciliğinde, yüksek doğruluğa sahip kullanımı basit bir model oluşturmak için büyük miktarda veri işlenmektedir.

Uygulama alanlarını çeşitlendirmek mümkündür. Perakendeye ek olarak, finans alanında, bankalardaki kredi uygulamalarında, sahtekarlık tespitinde ve borsada kullanılacak modeller oluşturmak için geçmişteki veriler analiz edilmektedir. İmalatta, makine öğrenmesi modelleri optimizasyon, kontrol gibi amaçlar için kullanılmaktadır. Tıpta, tıbbi teşhis için kullanılmaktadır. Telekomünikasyonda, hizmet kalitesini en üst düzeye çıkarmak için geçmişteki çağrılar analiz edilmektedir.

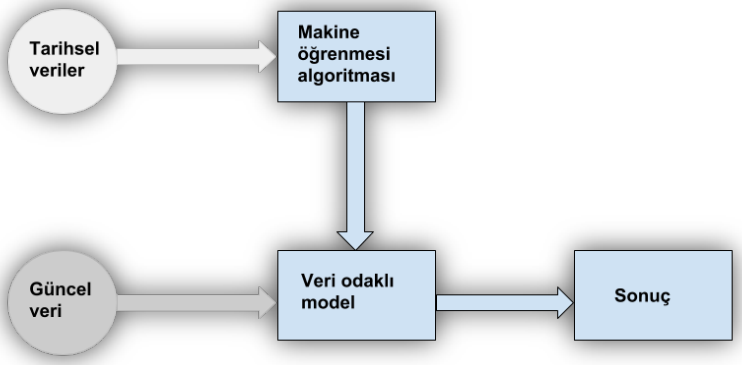
Makine öğrenmesi(ML – Machine Learning) aynı zamanda, yapay zeka (AI – Artificial Intelligence) yöntemlerinin de önemli bir parçasıdır. Bir sistemin akıllı olması için, öğrenme yeteneğine sahip olması gerekir. Sistem, bu değişiklikleri öğrenip kendini adapte edebiliyorsa, sistem tasarımcısının olası tüm durumlar için öngörme ve çözüm sağlaması gerekmemektedir. ML ayrıca görme, konuşma tanıma ve robotik alanlarındaki birçok probleme çözüm önerilmesi konusunda sıklıkla başvurulan yöntemdir..

ML matematiksel modeller oluşturmada istatistik teorisini kullanmaktadır. Çünkü ML'deki temel amaç bir örneklemden çıkarımlar yapmaktır. ML üzerinde bilgisayar biliminin rolü iki yönlüdür: Birincisi, modellerin eğitiminde, optimizasyon problemlerini çözmek, genel olarak sahip olunan büyük miktarda veriyi depolamak ve işlemek için etkili algoritmalara ihtiyaç duyulmaktadır. İkinci olarak, bir model eğitildiğinde, doğru çıkarımlar yapılabilmesi için modelin görselleştirilmesidir.

ML'de temel olarak üç farklı öğrenme tipi bulunmaktadır. Bunlar, denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeli öğrenmedir (Alpaydın, 2010).

## 2.1. Denetimsiz Öğrenme

Denetimli öğrenmede amaç, girdilerden doğru değerleri bir süpervizör tarafından sağlanan bir çıktıya eşlemeyi öğrenmektir. Denetimsiz öğrenmede böyle bir süpervizör yoktur ve yalnızca girdiler mevcuttur. Amaç, girdideki patternleri ortaya çıkarmaktır. Girdi uzayında, belirli kalıpların diğerlerinden daha sık oluşabileceği yapılar mevcut olabilir ve bu yapıların genel olarak ne olduğunu ya da ne olmadığını öğrenmek oldukça önemlidir. İstatistik teorisinde bu süreç, yoğunluk tahmini olarak da adlandırılır. Yoğunluk tahmini içeren bir yöntem, amacı kümeleri veya girdi gruplarını bulmak olan kümelemedir.

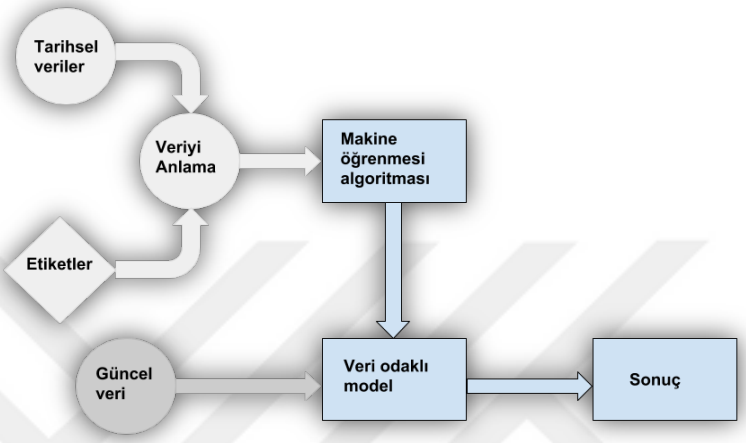



Şekil 1 Denetimsiz Öğrenme

Örneğin, müşterilerin geçmişteki işlemlerini içeren veritabanına sahip olan bir şirket, ne tür müşterilerin sıkça ziyaret ettiğini görmek için müşterilerin profil dağılımını görmek isteyebilir. Böyle bir problemin çözümü için kullanılabilecek kümeleme algoritması, nitelikleri bakımından benzer müşterileri aynı gruplara atayacaktır.

## 2.2. Denetimli Öğrenme

Denetimli öğrenme, bir girdiyi, örnek girdi-çıktı çiftlerine dayanarak bir çıktıya eşleyen bir fonksiyonu öğrenme görevine sahip makine öğrenme yötemidir.

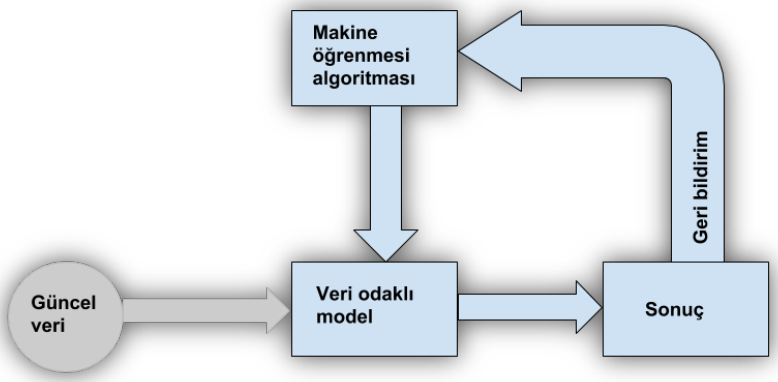



Şekil 2 Denetimli Öğrenme

Başka bir deyişle, etiketlenmiş verilerdeki girdi ile çıktı arasındaki ilişkiyi temsil eden fonksiyonu elde etmeye çalışan makine öğrenme yöntemidir. Denetimli öğrenmede her örnek, bir girdi ve çıktı değerinden oluşan bir çifttir. Bir denetimli öğrenme algoritması, eğitim verilerini analiz eder ve yeni örnekleri haritalamak için kullanılabilecek bir çıkarım fonksiyonu üretir (Russell&Norvig, 2010).

## 2.3. Pekiştirmeli Öğrenme

Bazı uygulamalarda, sistemin çıktısı bir dizi eylemdir. Böyle bir durumda, işlemler önemli değildir. Önemli olan, hedefe ulaşmak için gereken eylemlerin sırasıdır. Bir eylem, iyi bir stratejinin parçası ise, iyi bir eylem olarak tanımlanabilmektedir. Böyle bir durumda, ML algoritması, stratejilerin kalitesini iyi bir şekilde değerlendirebilmeli ve bir strateji oluşturabilmek için geçmişteki eylem dizilerini iyi bir şekilde öğrenebilmelidir. Bu tür öğrenme yöntemlerine, pekiştirici öğrenme denir.




Şekil 3 Pekiştirmeli Öğrenme

Pekiştirmeli öğrenmeye verilebilecek en güzel örneklerden biri tek bir hareketin tek başına önemli olmadığı durumları içeren oyunlardır. Örneğin, satranç oyununda önemli olan doğru hamle dizilerine sahip olmaktır. Bir hamle, iyi bir oyun oynama stratejisinin bir parçası ise, iyi ve etkili bir hamledir. Oyunlar, hem AI hem ML ölçütünde önemli bir araştırma alanıdır. Bunun nedeni, oyunları tanımlamak oldukça kolay ve aynı zamanda oyunları iyi bir şekilde oynamak oldukça zordur. (Alpaydın, 2010).

## 3.DERİN ÖĞRENME

DL, sinir ağları oluşturmak ve eğitmek için kullanılan bir yaklaşımdır. DL algoritmaları, karar verme kara kutuları olarak düşünülebilir. Bir kedi resmini işleyip bu resmin kedi içeren bir resim olup olmadığını anlayan bir algoritma, DL algoritmasına örnek olarak verilebilir. Sinir ağının hafızası, girdilerin nasıl bir araya getirilip birleştirileceğini tanımlayan ağırlıklar olarak bilinen sayı dizileri yardımıyla sonuçları kara kutu içinde çalışacak şekilde kontrol eder. Resimdeki objenin kedi olduğunu algılamak gibi gerçek dünyadaki problemleri çözmek için, karmaşık fonksiyonlar içeren ağlara ihtiyaç duyulmaktadır. 2012 yılında, sinir ağlarındaki mevcut gelişmelerin paylaşıldığı Imagenet yarışması kapsamında yayınlanan makale ve sonuçları, DL çalışmaları için gerçek bir dönüm noktası olmuştur.

Bunun yanında, doğal dil işleme ve konuşma tanımada da benzer gelişmeler elde edilmiştir. Bu yeni teknikler ışığında, daha önce çözülmemiş sorunların üstesinden

gelmek için sinir ağlarını inşa etmek ve eğitmek mümkün hale gelmiştir. Ham girdi verilerindeki özelliklerin belirlenmesi, çoğu ML yöntemi için zorlu bir süreçtir. DL, bu manuel adımı kullanmaktan ziyade, girdilerdeki en kullanışlı kalıpları bulmaya dayalı bir eğitim sürecine sahiptir. (Warden, 2014).

## 3.1. DERİN ÖĞRENME UYGULAMA ALANLARI

Bu bölümde, DL yöntemlerinin uygulama alanları hakkında daha iyi bilgiye sahip olmak için , makine ve DL yötemleri ile çözülen gerçek hayat problemlerine yer verilmiştir.

Görüntü işlemeyi otomatikleştirmek için kullanılan DL yöntemleri ile "Bu bir bisiklet mi?", “Bisiklete binen bir kişi var mı yoksa robot mu?” şeklinde sorulara otomatik bir şekilde yanıt bulmak mümkün hale gelmiştir.

Apple’ın Siri uygulaması, konuşma tanıma için verilebilecek en iyi örneklerden biridir. Siri’nin ses tanıma özelliği sayesinde, takvimde randevu oluşturmak ya da internet üzerinden bir konu hakkında arama yapmak mümkündür.

Farklı diller arasında çeviri yapabilme, derin sinir ağlarının kullanıldığı, son dönemde oldukça yaygınlaşan bir alandır. DL sayesinde, bir metin, el yazısı ya da ses, anlık olarak farklı dillere çevrilebilmektedir.

Son dönemde oldukça popülerleşen uygulamalardan bir diğeri olan sohbet botu, herhangi bir web sitesinde "nasıl yardımcı olabilirim?" şeklinde bir destek bağlantısını tıkladığınızda, sizin sorularınızı DL sayesinde yanıtlamaya yardımcı olmaktadır. DL ile geliştirilen sohbet botları, ifadelerinizi yorumlayıp gerçek kişi gibi davranarak daha anlamlı ve uzun sohbetler yapabileceğiniz komplike hizmetler sunabilmektedir.

Gerçek zamanlı güvenlik kamera görüntüsü ile tehlikeli durum tespit eden, ev sahibi ve misafire ait parmak izlerini ayırt edebilen sistemler vb. nesnelerin interneti(IoT) şemsiyesi altında toplanmaktadır. IoT cihazlarından anlık ve büyük miktarlarda gelen verilerden çıkarsamalar yapmak, sonuçlar üretmek de DL’nin bir başka uygulama alanıdır.

Herhangi bir bireyin sağlık durumunu tespit ederek bireyin karantinaya ihtiyaç duyup duymadığını ya da sadece yüksek ateşi olup olmadığını tanımlamak, termal

görüntülerin işlenmesi sayesinde mümkün kılınmaktadır. Radyoloji veya onkoloji gibi çeşitli disiplinlerde görüntüleri analiz ederek farklı doku tiplerin tespit edilmesi, tıbbi görüntülemede yaygın olarak kullanılan görüntü işleme teknikleri yardımıyla mümkün kılınmaktadır. Büyük veri kümelerine sahip bu kayıtlar sayesinde, içgörü ve korelasyonları tanımlamak için DL algoritmalarını kullanmak, daha doğru ve efektif çözümlere ulaşmaya önemli katkı sağlamaktadır.

Derin yapay sinir ağları (DNN- Deep Neural Networks), büyük veri kümeleri, görüntü işleme, matematiksel işlemler, otomasyonun kolaylaştırılması açısından büyük bir öneme sahiptir. DL'nin kullanımında en güçlü örneklerden biri; Google'dır. 2011 yılında, Google Beyin Takımı DL konusunda araştırmalara başlamıştır. Üç yıllık bir keşif sürecinden sonra, Google, İngiltere merkezli girişimi satın almıştır: Derin Düşünce(DeepMind). Ayrıca, Alpha Go oyunlarında ve masaüstü oyunlarında bu şirket kapsamında geliştirilen derin sinir ağlarını kullanılmıştır. Google Cloud Video Intelligence ürünü ile video analizi mümkün kılınmıştır. Sunucularda toplanan veriler yardımıyla eğitilen derin sinir ağları sayesinde otomatik sonuçların üretilmesine olanak kılan başka bir servis ise Google Asistan Konuşma Tanımadır. Bu projededen de, yine Google Beyin Takımı sorumludur. Ayrıca, bu sistem şimdi yeni sürümü ile Google Neural Machine Translation olarak adlandırılmaktadır. Google'ın sunduğu diğer hizmetlere, Google Beyin Takımının Youtube kullanıcıları için daha efektif öneriler sunmak için DL kullanarak yaptığı çalışmalar örnek olarak verilebilmektedir (Marr, 2017).

## 3.2. DERİN YAPAY SİNİR AĞLARI

DNN, bilgisayarların deneyimlerden öğrenmelerini ve dünyayı kavramlar hiyerarşisi bağlamında anlamalarını sağlayan bir makine öğrenimi yöntemidir. Bilgisayar, bilgiyi deneyimlerden topladığı için, bilgisayarın ihtiyaç duyduğu tüm bilgilerin insan tarafından belirtilmesine gerek kalmamaktadır.

İlk algılayıcılar (perceptrons) gibi sinir ağlarının önceki versiyonları, bir girdi ve bir çıktı katmanından oluşmaktaydı ve en fazla bu iki katmana ek olarak bir gizli katman eklenmekteydi. Günümüzde, üç katmandan fazlasını (girdi ve çıktı dahil) içeren ağlar "derin" ağlar olarak nitelendirilmektedir. Bu yüzden derin, birden fazla gizli katman

anlamına gelen bir teknik terimdir. DNN'lerde, her bir düğüm katmanı önceki katmanın çıktısına dayanan farklı özellikler dizisi üzerinde çalışmaktadır. Sinir ağı ne kadar derinleştirilirse, katmanlar, kendilerinden önceki katmandaki özellikleri bir araya getirip birleştirdiklerinden, düğümlerin tanıyabileceği özellikler bir o kadar karmaşık olmaktadır.

Bu bölümde derin ağların üç ana mimarisine ve bunları oluşturmak için daha küçük ağların nasıl kullanıldığına ve mimarilerine ayrıntılı olarak değinilmektedir. Üç ana mimariye odaklanılmıştır. Bunlar, genellikle görüntü modelleme için kullanılan Konvolüsyonel Sinir Ağları (CNN-Convolutional Neural Networks), genellikle zaman serilerini modellemek için kullanılan Uzun Kısa Süreli Bellek (LSTM – Long Short Term Memories) Ağları ve Tekrarlamalı Sinir Ağlarıdır (RNN – Recurrent Neural Networks). Çalışma kapsamında LSTM ağları kullanılarak yeni bir mimari önerildiği için, LSTM hakkında daha detaylı bilgiye yer verilmiştir.

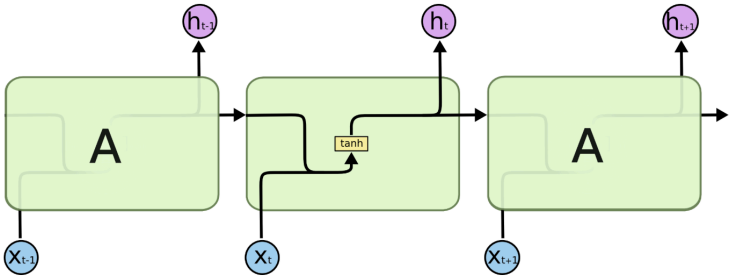
### 3.2.1 Tekrarlamalı Sinir Ağları

RNN, ara katmanlardaki çıktılardan giriş birimlerine veya önceki ara katmanlara geri beslenmenin yapıldığı bir ağ yapısıdır. Bu çeşit sinir ağlarının dinamik hafızaları vardır. Nöronların çıktısı sadece o anki giriş değerlerine bağlı değildir ayrıca önceki girdi değerlerine de bağlıdır. Bu ağlar özellikle çeşitli tipteki zaman serilerinin tahmininde oldukça etkilidir. (Goodfellow et. al. 2016).

#### 3.2.1.1.Uzun Kısa Süreli Bellek Ağları

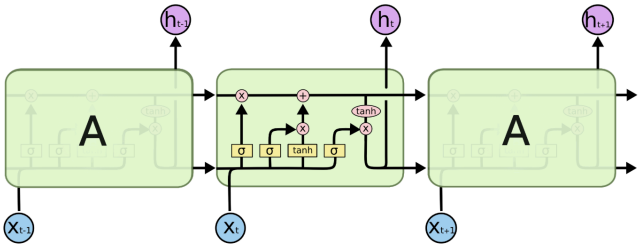
LSTM, RNN'nin bir çeşididir. LSTM, veri setindeki uzun vadeli ilişkileri öğrenen bir ağ geliştirme amacı ile 90'lı yılların sonlarında Sepp Hochreiter ve Bergen Schmidhuber tarafından önerilmiştir. LSTM'ler, uzun vadeli bağımlılık probleminden kaçınmak için tasarlanmıştır.

Tüm tekrarlayan sinir ağları, bir sinir ağının yinelenen modülleri formuna sahiptir. Standart RNN'lerde, bu tekrarlayan modül, Şekil 4'te de gösterildiği gibi tek bir *tanh* katmanı gibi basit bir yapıdadır.



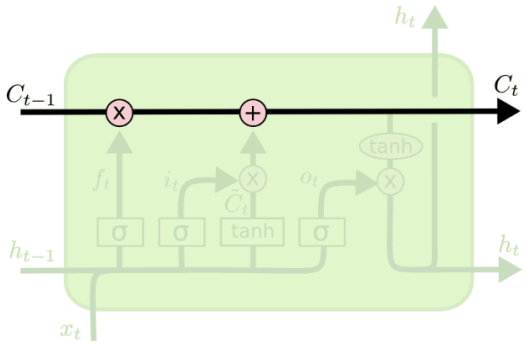

Şekil 4. Standart RNN ağındaki tekrarlayan modüller
(İmaj Kaynak: Understanding LSTM Networks,
http://colah.github.io/posts/2015-08-Understanding-LSTMs/ )

LSTM ağları da RNN'lerdeki gibi zincire benzeyen tekrarlayan modüllere sahiptir. fakat tekrar eden modüller RNN'deki yapıdan farklı bir yapıya sahiptir. Tek bir sinir ağı katmanına yerine, Şekil 5'te de gösterildiği gibi çok özel bir şekilde etkileşime giren dört tane katman bulunmaktadır.




Şekil 5. LSTM ağındaki tekrarlayan modüller
(İmaj Kaynak: Understanding LSTM Networks,
http://colah.github.io/posts/2015-08-Understanding-LSTMs/ )

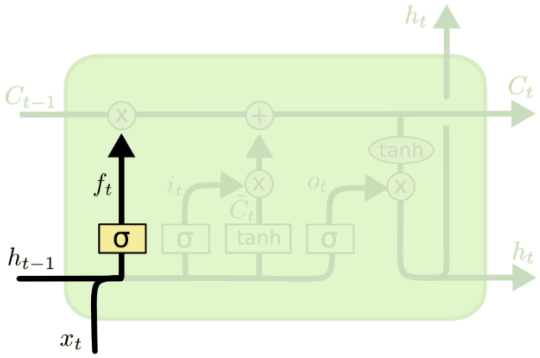
LSTM ağlarında kilit göreve sahip birim LSTM ağlarındaki hücrelerin durumunu bildiren birimdir. Bu birim Şekil 6'da görülen yatay çizgi şeklinde diyagramın üstünden geçmektedir. Hücre durumu bir anlamda taşıma bandı gibidir. Sadece bazı küçük lineer etkileşimlerle, tüm ağ boyunca uzanır.




Şekil 6. LSTM yapısında bilgilerin hücreler arasında taşınması
(İmaj Kaynak: Understanding LSTM Networks,
http://colah.github.io/posts/2015-08-Understanding-LSTMs/ )

LSTM, kapı adı verilen yapılar tarafından dikkatlice düzenlenen hücre durumuna bilgi çıkarma veya ekleme kabiliyetine sahiptir. Kapılar, bilginin bir sonraki hücre durumuna taşınmasına olanak sağlamaktadır. Kapı adı verilen yapılar, sigmoid bir sinir ağı katmanından ve noktasal çarpma işleminden oluşurlar. Sigmoid katmanı, her bir bileşenden ne kadarının bir sonraki aşamaya geçmesi gerektiğini tanımlayan sıfır ile bir arasında değerler üretmektedir. Bu katmanın sıfır değerini alması "hiçbir bilginin bir sonraki aşamaya geçmesine izin verme" anlamına gelirken, bir değerini alması ise "bir sonraki aşamaya tüm bilgilerin geçmesine izin ver" anlamına gelmektedir. Bir LSTM ağı, hücre durumunu korumak ve kontrol etmek için üç farklı kapıya sahiptir.

LSTM'deki ilk adım, hangi bilgilerin hücre durumundan atılacağına karar vermektir. Bu karar "unutma kapısı katmanı" adlı bir sigmoid katman tarafından verilir. Bu kapı Şekil 7'de de göstedildiği gibi $h_{t-1}$ ve $x_t$ değerlerine bakar ve $C_{t-1}$ hücre durumunda her bilgi için 0 ile 1 arasında bir değer atar. Değerin bir olması "bu bilgiyi tamamen koru" yu temsil ederken, bu bilginin sıfır olması "bu bilgiyi tamamen unut" anlamına gelmektedir. Ardından 1 numaralı denklemde gösterilen $f_t$ fonksiyonu elde edilmektedir. Önceki kelimelere dayanarak bir sonraki kelimeyi tahmin etmeye çalışan bir dil modeli örneği düşünüldüğünde böyle bir problemde, hücre durumu mevcut öznenin cinsiyetini içerebilir, bir sonraki adımda yeni bir özneye sahip olunduğunda doğru zamirler kullanılamayabilir, eski öznenin cinsiyetini unutmak bir sonraki kelimenin tahmin edilmesini daha etkili bir hale getirecektir.




Şekil 7. Bir sonraki adıma hangi bilgilerin geçeceğine karar verilmesi

(İmaj Kaynak: Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/ )

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \qquad (1)$$

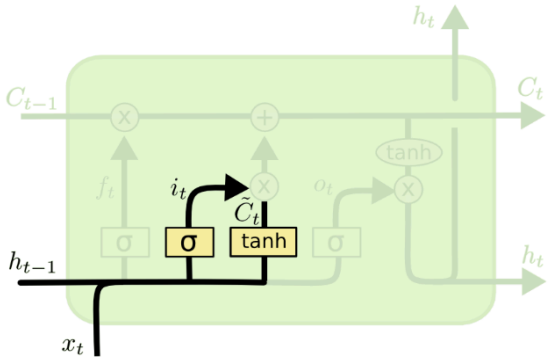
Denklem 1'de gösterilen $W_f$ $f$ fonksiyonu için ağırlık matrisini, $b_f$ ise $f$ fonksiyonu için yanlılık miktarını içeren matrisi ifade etmektedir.

Bir sonraki adım, hücre durumunda hangi yeni bilgilerin depolanacağına karar vermektir. Şekil 8'de gösterilen bu adım iki aşamada gerçekleşmektedir. İlk olarak, "giriş kapısı katmanı" adı verilen bir sigmoid katman, hangi değerlerin güncelleneceğine karar vermektedir. Ardından, bir *tanh* katmanı, duruma eklenebilecek yeni aday değerleri içeren $\tilde{C}_t$ vektörünü oluşturur. Şekil 8'de gösterilen $i_t$ denklem 2'de, $\tilde{C}_t$ vektörü denklem 3'te detaylarıyla gösterilmiştir.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \qquad (2)$$

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c \qquad (3)$$

Dil modeli örneğinde olduğu gibi, unutmakta olunan eski öznenin cinsiyeti yerine yeni öznenin cinsiyeti hücre durumuna eklenmek istenmektedir.




Şekil 8. Bir sonraki adımda hangi bilgilerin depolanacağına karar verilmesi

(İmaj Kaynak: Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/ )

Takip eden katmanda, eski hücre durumunu yani $C_{t-1}$ 'i, yeni hücre durumu $C_t$ olarak güncellenme işlemi gerçekleşmektedir. Yani dil modeli örneğinde, eski öznenin cinsiyeti hakkındaki bilgilerin gerçekten unutulduğu katmandır. Önceki adımlarda, bir sonraki aşamalara aktarılmamasına karar verilen bilgiler bu adımda tamamen unutulur ve yeni hücre durumu $C_t$ 'yi elde etmek için denklem 4'te de gösterildiği şekilde, bir

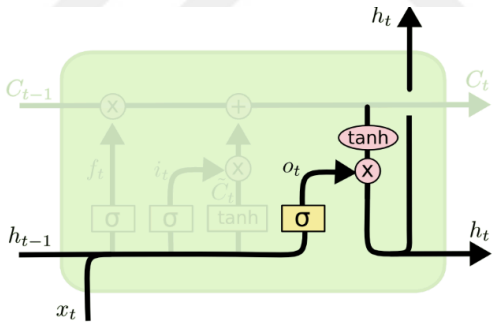
önceki hücre durumunu belirten $C_{t-1}$ bir önceki adımda elde edilen $f_t$ ile çarpılmaktadır ve ardından bu çarpma işlemine $i_t * \tilde{C}_t$ eklenmektedir.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \qquad (4)$$

Son olarak, hangi bilgilerin çıktı olarak değerlendirileceğine karar verilmesi gerekmektedir. Bu çıktı, yeni hücre durumuna dayalı olmakla birlikte bu bilginin filtrelenmiş bir versiyonu olacaktır. İlk olarak, hücre durumunun hangi kısımlarını çıkartacağımıza karar veren bir sigmoid katman çalışmaktadır. Daha sonra, hücre durumu elde edilen değerlerin −1 ile 1 arasında olması istendiğinden, Şekil 9'da da gösterildiği şekilde bu değerlerler *tanh*'dan geçirilip sigmoid kapısının çıktısı ile çarpılmaktadır. Şekil 9'da gösterilen $o_t$ ve $h_t$ aşağıdaki denklemlerde detaylandırılmıştır.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \qquad (5)$$

$$h_t = o_t * \tanh(C_t) \qquad (6)$$




Şekil 9. Bilgilerden hangilerinin çıktı olarak değerlendirileceğine karar verilmesi

(İmaj Kaynak: Understanding LSTM Networks,
http://colah.github.io/posts/2015-08-Understanding-LSTMs/ )

### 3.2.2. Konvolüsyonel Sinir Ağları

CNN, İleri beslemeli yapay sinir ağlarının en gelişmiş halidir ve görüntü tanımada oldukça başarılı sonuçlar vermektedir. Bu yapıda resim parçalara ayrılır ve parçalara filtre uygulanır, uygulanan filtrenin boyutuna göre resimde küçülmeler olmaktadır. Örneğin; 8×8 boyutundaki bir filtre resmin 8×8 alanındaki bir bölümüne

uygulandıktan sonra çıkan sonuç bir pikselde tutulmakta ve bunun sonucu olarak çıktı 1×1 boyutunda olmaktadır. Ortaya çıkan bu pikseller birbiriyle ilişkilendirilerek tanımlama yapılmaya çalışılmaktadır (Goodfellow et. al. 2016).

## 3.3. DERİN ÖĞRENMEDE KULLANILAN YAZILIM KÜTÜPHANELERİ

ML yöntemleri sürekli olarak gelişmektedir. Kilit nokta, uygulamaları daha akıllı hale getirmek için mobil cihazlarda çalışan makine öğrenme modelleri geliştirmektir. DL, karmaşık problemleri çözmeyi mümkün kılmaktadır. Günümüzde, programlama zorluklarını sadeleştiren, hızlı ve efektif DL algoritmalarının oluşturulmasına olanak sağlayan bir çok DL yazılım kütüphanesi bulunmaktadır. Her kütüphane, farklı amaçlar için farklı bir şekilde inşa edilmiştir.

Bu bölümde, sıklıkla tercih edilen Tensorflow, Theano, Torch, Caffe, Keras yazılım kütüphanelerinden bahsedilmektedir. Çalışma kapsamında önerilen mimari Keras kullanılarak geliştirildiği için bu bölümde Keras daha detaylı ele alınmıştır.

Bir açık kaynak kodlu yazılım kütüphanesi olan Tensorflow veri akış grafikleri kullanarak nümerik hesaplamalar yapmak için tasarlanmıştır. Grafiklerin sınırları çok değişkenli veri dizilerini tanımlamaktadır. Tensorflow, çoklu CPU ya da GPU desteğiyle güçlü ve efektif hesaplama kabiliyeti sunmaktadır. Google Beyin Takımı tarafından geliştirilen bu kütüphane, C++, Python gibi birden fazla programlama dili ile kullanılabilmektedir. (Tensorflow Architecture).

Theano, Montreal Üniversitesi'ndeki MILA adındaki araştırma grubu tarafından geliştirilmiştir. Bu kütüphane ile Python ve C++ programlama dilleri kullanarak oldukça etkili modeller üretebilmek mümkündür. Ayrıca Theano, kullanıcının donanımını optimum kullanarak verimliliği en yukarı çekmektedir. (Brownlee J., 2016)

Geniş yelpazeli ML desteği sunan bilimsel hesaplama kütüphanesi olan Torch, çoğunlukla GPU'lar üzerinde efektif çalışmak üzere dizayn edilmiştir. Kütüphane, LuaJIY ve CUDA implementasyonunu kullandığından kullanışlılığı ve efektifliği oldukça kaydadeğerdir. Torch, araştırmacılara basit ve aynı zamanda maksimum esneklik sunmak temel amacıyla tasarlanmıştır. Ayrıca, topluluk destekli ML, bilgisayarlı görü, sinyal işleme, paralel işleme, ses ve görüntü işleme kütüphaneleri ile mükemmel bir uyum içerisinde çalışmaktadır.(Torch).

Caffe, Berkeley AI Araştırma Grubu tarafından geliştirilmiş, ilk yılında yaklaşık 1000 araştırmacı tarafından kullanılmaya başlanmış ve zaman içerisinde geliştirmeler ile efektif hale gelmiş bir DL kütüphanesidir. Kütüphane, CPU'dan GPU'ya oldukça etkili geçiş yapılmasına olanak sunmaktadır. Caffe topluluğu akademik araştırma projeleri tarafından fonlanmakta ve geliştirilmeye devam edilmektedir. (Y. Jia et al., 2014).

Keras, Python ile yazılmış, TensorFlow, CNTK veya Theano'nun üzerinde çalışabilen üst düzey sinir ağları geliştirme ortamı sunmaktadır. Keras ile çok hızlı bir şekilde yapay öğrenme algoritmaları eğitilebilmektedir. Keras, kullanıcı dostu olması, modüler ve genişletilebilirlik özelliği sayesinde kolay ve hızlı prototip oluşturmayı mümkün kılmaktadır. Evrişimli ağları, yinelenen ağları ve bu iki ağın kombinasyonlarını da desteklemektedir. CPU ve GPU üzerinde hiçbir sorun olmadan performans gösterebilmektedir. Keras, makineler için değil, insanlar için tasarlanmış bir yazılım ortamıdır. Kullanıcı deneyimini en yüksek seviyeye çıkartma felsefesi ile geliştirilmiştir. Yaygın kullanım durumları için gereken kullanıcı eylemleri sayısını en aza indirmekte ve kullanıcı hatası üzerine net, işlem yapılabilir geri bildirim sağlamaktadır.

Keras'ın çekirdek veri yapısı modeldir. En basit model doğrusal bir katman yığınından oluşan sıralı modeldir.

Daha karmaşık mimarileri eğitmek için, rastgele katman grafikleri oluşturmaya izin veren Keras işlevsel arayüzü kullanılmaktadır. Keras'ta sıralı model (sequential) aşağıdaki şekilde oluşturulmaktadır.

```
from keras.models import Sequential

model = Sequential()
```

Katmanları eklemek oldukça basittir. "**.add()**" kullanılarak, katmanlar kolaylıkla eklenebilir.

```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

Modelin yapısından, katmanlarından emin olunduktan sonra, öğrenme süreci "**.compile()**" komutu ile konfigüre edilebilmektedir. Örneğin, aşağıdaki örnekte kayıp fonksiyonu olarak "categorical_crossentropy", optimizasyon algoritması olarak "sgd" seçilmiştir. Ayrıca, performans metriği olarak da doğruluk(accuracy) seçilmiştir.

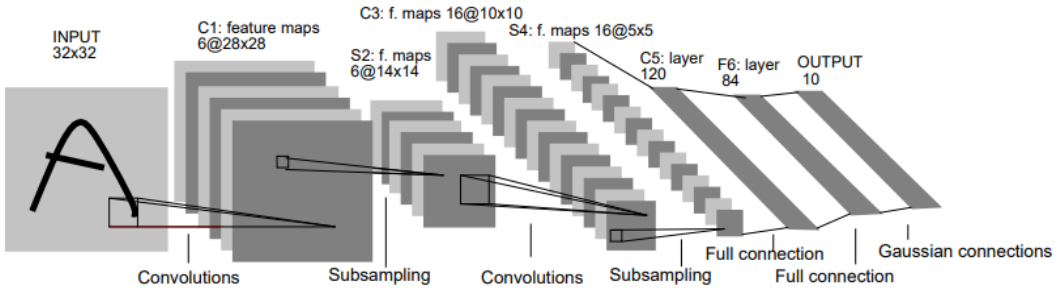
*model.**compile**(loss='categorical_crossentropy',optimizer='sgd',metrics=['accuracy])*

### 3.4. POPÜLER DERİN YAPAY SİNİR AĞI MİMARİLERİ

Teknoloji harikası DNN'ler, Dünyadaki saygın araştırma kuruluşları tarafından uzun yıllar süren çalışmaların sonucunda elde edilmiştir. Bu ağlar, hali hazırda bir çok problemi başarıyla çözmüş ve bir çok araştırmacı tarafından sıklıkla kullanılmaktadır.

Bu bölümde kronolojik sıraya göre, popüler olan teknoloji harikası DNN'lere değinilecek ve mimari yapıları ile ilgili bilgiler verilecektir.

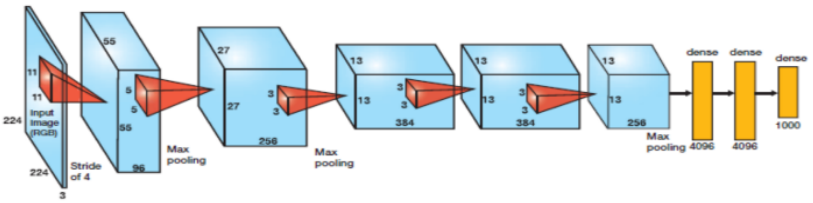
LeNet mimarisi, öncelikli olarak dökümanlardaki karakterleri tanıma problemlerini çözmek amacıyla LeCun vd. tarafından 1998 yılında geliştirilmiştir. Şekil 10'da LeNet detaylarıyla gösterilmektedir. Mimari, 32x32 boyutlu imajlar için oluşturulan girdi katmanı hariç, yedi katman içermektedir. (LeCun et al., 1998)




Şekil 10. LeNet mimarisi
İmaj kaynak: (LeCun vd., 1998)

Dünyaca ünlü ImageNet yarışmasında yüksek doğruluk oranı elde etmeyi başaran ilk ve öncü mimarilerden olan AlexNet, Krizhevsky vd. tarafından 2012 yılında önerilmiştir. Mimari, birbirine tamamen bağlı üç katmanın takip ettiği beş adet konvolüsyonel tabakadan oluşmaktadır. Aktivasyon fonksiyonu olarak tanh ya da

sigmoid fonksiyonu yerine ReLu (Rectified Linear Unit) fonksiyonu kullanılmıştır. (Krizhevsky vd., 2012).

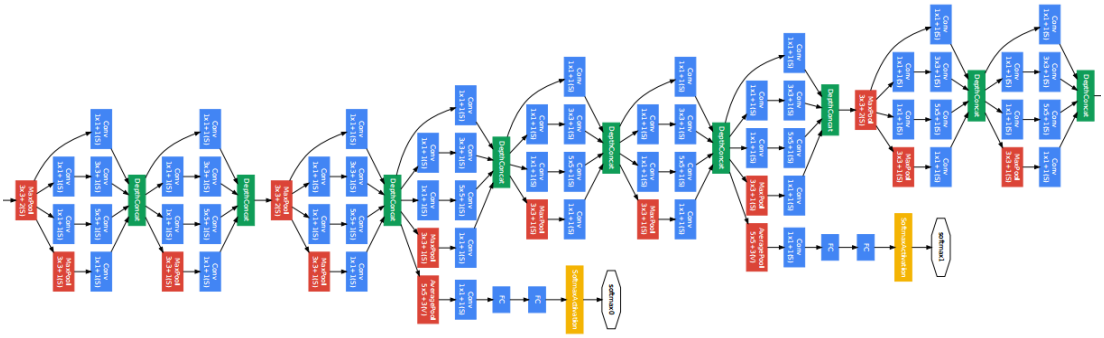



Şekil 11. AlexNet mimarisi

İmaj Kaynak: Krizhevsky vd., 2012.

İmaj verileri ile DNN oluştururken ezberlemeyi azaltmada yaygın olarak kullanılan yöntem, veri setini yapay bir şekilde genişletmektir. Bu yöntemden yola çıkılarak Şekil 12'de görüldüğü gibi mimari, veri arttırma aşaması için iki ayrık bölüm içermektedir.

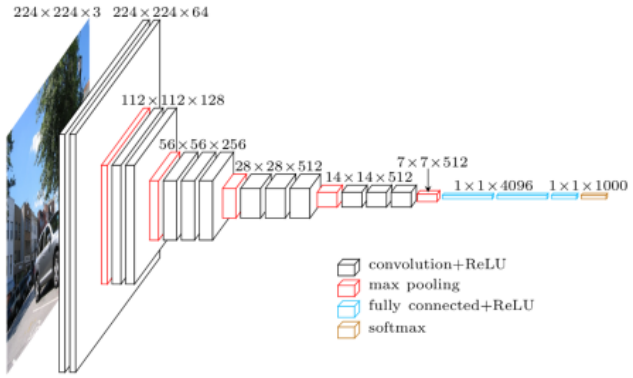
GoogleNet'in 22 tabakalı yapısı ve daha az parametreye sahip olması onu, AlexNet'ten daha hızlı ve daha doğru sonuçlar üreten bir mimari yapmaktadır.



Şekil 12. GoogleNet mimarisi

İmaj Kaynak: Szegedy vd., 2015

Öncelikli olarak akıllı cihazlarda çalışabilecek bir mimari kullanma fikri ile geliştirilmeye başlanan bu mimariyi limitli hesaplama gücüne sahip cihazlarla kullanmak mümkündür. Şekil 12'de gösterilen mimarinin derinliği ve ağ tabakası hesaplama maliyeti göz önünde bulundurulmak kaydı ile arttırılabilmektedir. (Szegedy vd., 2015). 2015 yılında Simonyan ve Zisserman tarafından geliştirilen VGG'nin genel fikri 3x3'lük çok küçük konvolüsyonel filtreler kullanarak ağlar üretmektir.

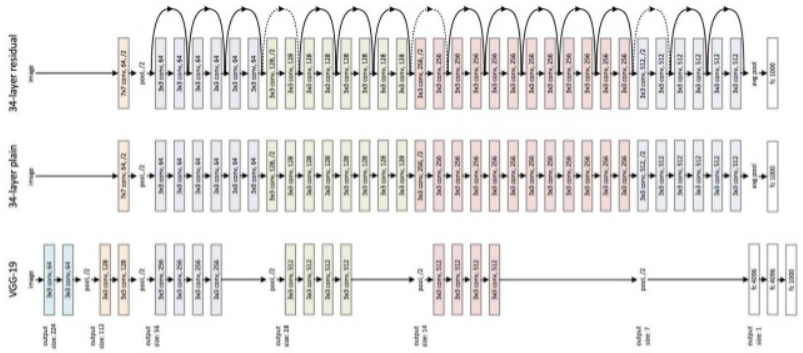



Şekil 13. VGG16 Mimarisi

İmaj Kaynak: Cord, 2016

VGG, on dört milyondan daha fazla görüntü ve bin adet farklı etiket içeren veri setiyle yüksek performanslı bir şekilde çalışmıştır. (Simonyan vd., 2015)

He vd. tarafından 2015 tarafında geliştirilen ResNet'teki ana fikir, daha önce ortaya atılan mimarilere göre aynı zamanda hem daha karmaşık ve derin bir yapı kullanıp hem de eğitim süresini kısaltmaktır. He vd., katmanların girdilerine referans olan artık (residual) öğrenme fonksiyonlarını kullanarak bir mimari geliştirmeye çalışmıştır ve artık(residual) ağların, optimize edilmesi kolay ve daha doğru olduğunu kanıtlamaya çalışmıştır.




Şekil 14. ResNet mimarisi

İmaj Kaynak : https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

ImageNet veri seti ile yapılan denemelerde VGG ağlarından 8 kat daha fazla katman sayısına (152) sahip olmasına rağmen, ResNet mimarisi, VGG ağlarından daha az kompleks bir yapıya sahip olduğu görülmüştür. (He , vd. 2015)

# 4.DERİN ÖĞRENMEDE UYGUN PARAMETRE SEÇİMİ

ML ve DL araştırmalarının temelinde, belirli bir veri setinden ilgilenilen bazı unsurları yakalayabilen yöntemlerin geliştirilmesi yatmaktadır. Bu unsurlar, veri içindeki kümelenmiş yapıları (kümeleme) veya belirli özelliklere dayalı olarak hedef değişkenlerinin değerlerini tahmin etme (sınıflandırma) veya sürekli (regresyon) olabilen belirli hedef değerlerini tahmin etme yeteneğini içermektedir. Biyolojik olarak esinlenen sinir ağlarından çekirdek yöntemlerine ve topluluk modellerine kadar geniş bir yelpazede öğrenme yöntemleri vardır (Bishop vd.,1995) Bu yöntemlerdeki ortak özellik, öğrenme yaklaşımının faydasını en üst seviyeye çıkarmak için kullanıcı tarafından uygun şekilde ayarlanması gereken bir dizi hiper parametreler içermeleridir. Hiper parametreler, öğrenme algoritmasının çeşitli yönlerini yapılandırmak için kullanılmaktadır ve ortaya çıkan model performansını etkilemektedir.

Hiper parametre araması; manuel olarak, belirli kurallar kullanılarak veya önceden tanımlanmış hiper parametre setlerini test ederek yapılmaktadır (Pedregosa vd., 2011). Bu yaklaşımlar tekrarlanabilirlik açısından isteneni verememektedir. Bu sebepten dolayı, hiper parametre araştırmasını otomatikleştirme fikri, son yıllarda bu alanda çalışmalar yapan araştırmacıların ilgisini çekmiştir ve araştırmacılar otomatikleştirilmiş yaklaşımların, uzmanlar tarafından çeşitli problemlere ilişkin manuel aramadan daha iyi performans gösterdiğini kanıtlamıştır (Bergstra vd., 2011, 2012).

yöntemlerinin temel amacı, önceden tanımlanmış, test seti $\boldsymbol{X}^{(te)}$ üzerinde $\mathcal{L}(\boldsymbol{X}^{(te)}; M)$ şeklinde belirtilen kayıp fonksiyonlarının aldığı değeri en aza indiren bir $M$ modelinin eğitimi olarak özetlenebilmektedir. Bu $M$ modeli $\boldsymbol{X}^{(eğ)}$ eğitim veri seti ile, A algoritması kullanılarak oluşturulduğu varsayımı altında, öğrenme algoritması olan A, hiper parametre olarak adlandırılan bir grup parametreye sahiptir. Bu parametreler $\lambda$ olarak belirtildiğinde, $M$ modeli A($\boldsymbol{X}^{(eğ)}$; $\lambda$) şeklinde ifade edilebilmektedir. $M$ modelinin, destek vektör makinesi sınıflayıcısı varsayıldığında, A algoritması C ve $\sigma$ parametrelerine sahip olacaktır. Bu durumda hiper parametreler $\lambda$ = [C, $\sigma$] olacaktır. Temel amaç, $\mathcal{L}(\boldsymbol{X}^{(te)}; M)$ kayıp fonksiyonunu minimize eden, $\lambda$ parametrelerini tahmin ederek yani $\lambda^*$ 'ı bularak $M$ modelini oluşturmaktır (Claesen vd., 2014).

ML ve DL'de kullanılan hiper parametre optimizasyonu yöntemlerinden en popüler olanları, Izgara araması(GS), Rasgele arama, Bayesgil Optimizasyon, Gradyan Tabanlı Optimizasyon, Evrimsel Optimizasyon, Popülasyon Tabanlı şeklinde sıralanmaktadır (Claesen, 2015). Çalışma kapsamında hızlı ve pratik çözüm sunan GS yöntemi ve kapsamlı, daha komplike ve Evrimsel Optimizasyon ailesinden olan Genetik Algoritma kullanılmıştır. Bu sebeple bu bölümde ve izleyen bölümde bu iki yöntem hakkında detaylı bilgi verilmektedir.

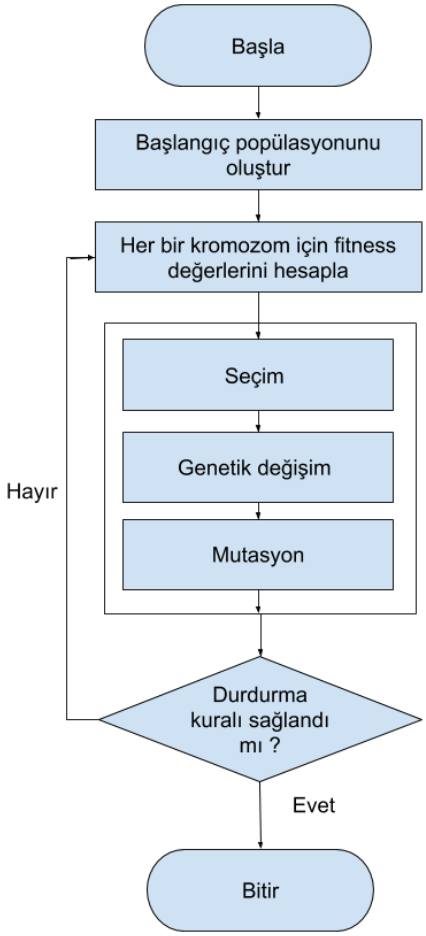
GS, hiper parametre optimizasyonu için en yaygın kullanılan yöntemlerdir. Son yıllarda gerçekleşen bir çok önemli global optimizasyon araştırmalarına rağmen, manuel arama ve GS yöntemleri hala sıklıkla tercih edilmeye devam edilmektedir. Çünkü, manuel optimizasyon, araştırmacılara bir dereceye kadar içgörü kazandırmaktadır ve manuel optimizasyon için teknik ek yük veya engel bulunmamaktadır. Izgara araştırmasının uygulanması oldukça basittir ayrıca, tamamen manuel sıralı optimizasyondan daha iyi $\lambda$ bulmaktadır.

Hiper parametre optimizasyonunu gerçekleştirmenin geleneksel yolu olan GS, hiper parametre alanının manuel olarak belirlenmiş bir alt kümesinde basitçe kapsamlı bir arama veya parametre taraması yapmaktadır. GS algoritması, tipik olarak eğitim setindeki çapraz doğrulama setinin değerlendirilmesi ile ölçülen bazı performans ölçütleri tarafından yönlendirilmektedir (Chicco, 2017).

## 4.1.GENETİK ALGORİTMA İLE UYGUN PARAMETRE SEÇİMİ

Darwin'in evrim teorisi, çeşitli disiplinlerdeki birçok araştırmacı için ilham kaynağı olmuştur. Darwin'in teorisinde öne sürdüğü gen, doğal seleksiyon, geçit ve mutasyon gibi temel terimler kullanılarak birçok evrimsel algoritma geliştirilmiştir. Bu evrimsel algoritmaların en önemlilerinden biri genetik algoritmalardır. İlk olarak, Goldberg ve Holland tarafından, evrim süreci bir bilgisayar ortamında yorumlanmıştır (Goldberg vd., 1998). Bu, GA'lar için çok önemli bir adım olmuştur. Goldberg, bu çalışmalar sayesinde GA'ların gerçek hayatta 80'den fazla örneğinin olduğunu kanıtlamıştır (Goldberg,1989). Daha sonra, tüm bu gelişmeleri takiben, Koza genetik programlamayı geliştirmiştir. GA'nın temel amacı, güçlü ve zayıf bireyi belirleyen güçlü bireyin doğal seleksiyon, çaprazlama ve mutasyona uğramasıdır. GA'da,

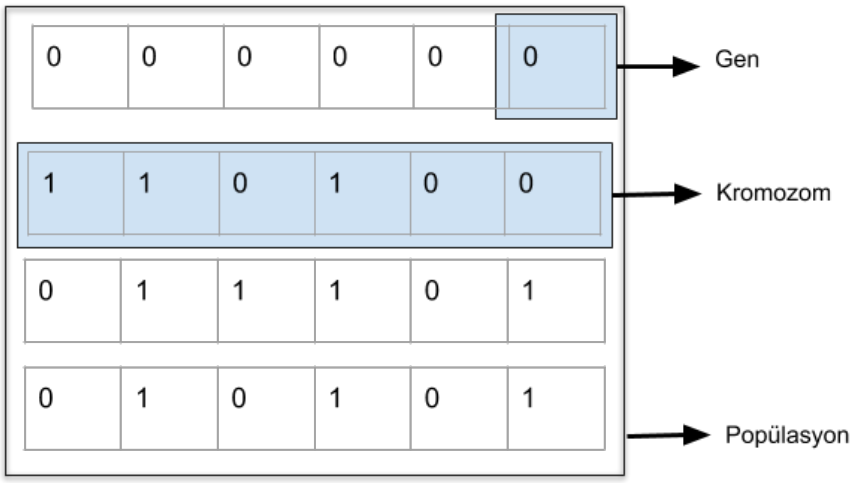
bireylerin bu aşamalardan geçen en iyi bireyi bulması amaçlanmaktadır (Koza, 1992) GA'nın akış diyagramı Şekil 15'teki gibi ifade edilebilmektedir.




Şekil 15. Genetik Algoritma akış diyagramı

GA'da aşağıdaki temel adımlar bulunmaktadır:

- Başlangıç popülasyonu oluşturma
- Seçim
- Genetik değişim
- Mutasyon




Şekil 16. GA'da popülasyon, gen, kromozom

GA'da ilk adım olarak başlangıç popülasyonu oluşturulmaktadır. Bu adım, popülasyon adı verilen Şekil 16'da da gösterilen bir grup bireyle başlar. Her birey çözülmek istenilen soruna bir çözüm niteliğindedir. Birey, gen olarak bilinen bir dizi parametre ile karakterize edilmaktedir. Bir kromozom oluşturmak için genler bir dizede birleştirilmektedir. GA'da, bir bireyin gen seti, harflar ile temsil edilir. Gen setlerinde ikili değerler kullanılmaktadır (1 ve 0).

İkinci adım olan seçim adımı, güçlü olanın hayatta kalma ilkesinin uygulanmaya başlandığı ilk adımdır. Bu aşamada, bireyler gelecekte birbirleriyle eşleşecek şekilde yaratılmaktadır. En güçlü adaylar açıklık değerlerine göre belirlenir. GA ile çözülmek istenilen algoritmanın amacına göre, adaylar birbiriyle eşleşmekte ve yüksek kaliteye sahip nesli üretmektedir. Bir başka deyişle, örneğin çözülmek istenilen problem eğer maksimizasyon problemi ise, en büyük fitness değerine sahip bireyler alınmaktadır. Eğer, minimizasyon problemi ise, bu sefer en düşük fitness değerine sahip bireyler alınmaktadır.

Genetik değişim olarak adlandırılan adımda ise yeni bir nesil üretilmektedir. Doğal seçilimden seçilen yüksek kaliteli bireyler ebeveyn olarak kabul edilir ve bu bireyler yeni bireyler oluşturmak için eşleştirilmektedir. Bu haritalama, her bir kromozomdaki her bir bireysel gen sekansını birbiriyle değiştirerek gerçekleştirilmektedir. Bu işlem çaprazlama olarak adlandırılır.

GA'daki son adım mutasyon olarak adlandırılmaktadır. Bazen bazı eşler, eşleştirilecek bireylerde tekrar tekrar değiştirilmek istense bile aynı kalabilmektedir. Bu durum

farklı bireylerin oluşumunu engelleyebilmektedir. Bu nedenle, en iyi çözüm sunulamayabilir. Bu durumun ortaya çıkma olasılığı çok düşük olmakla birlikte, bu durumdan kaynaklanan sorunları önlemek için, yaratılan bireylerin geninde çok küçük bir değişiklik yapılması daha iyi sonuç alınmasına katkı sağlayacaktır. Böylece, farklı bireylerin ortaya çıkması ile gelecek nesiller de farklılaşma gösterecektir.

Son yıllardaki ML ve DL çalışmalarına bakıldığında GA'nın hiper paremetre optimizasyonu için başvurulan bir yöntem olduğu görülmektedir. Ananto vd. 2018 yılında internet üzerindeki haberlerin popüler olup olmadığını tespit etmek için ML yöntemleri kullanmıştır. ML yöntemlerinin parametrelerini tahmin ederken ise GA'yı kullanmışlardır. Çalışmalarda belirtildiği üzere, GA ile, GS'ye kıyasla daha kısa hesaplama süresiyle optimum hiper parametreleri elde etmişlerdir (Ananto vd., 2018). Salah vd. tarafından 2018 yılında yapılan çalışmada optimal zaman gecikmelerini ve kaç katmanlı LSTM ağının kullanılması gerektiğini tespit etmek için GA kullanılmıştır (Salah, 2018). Chung vd. tarafından 2018 yılında yapılan çalışmada, hisse senetlerine yönelik tahmin modeli geliştirmek için GA algoritma kullanılarak LSTM ağlarındaki optimum gizli nöron sayıları tespit edilmiştir (Chung, 2018).

Bu çalışmalarda genel olarak GA, problemin çözümünde kullanılan kayıp fonksiyonunu minimize eden belli hiper parametreleri bulmak için kullanılmıştır. Özet olarak aşağıdaki adımlar takip edilmiştir.

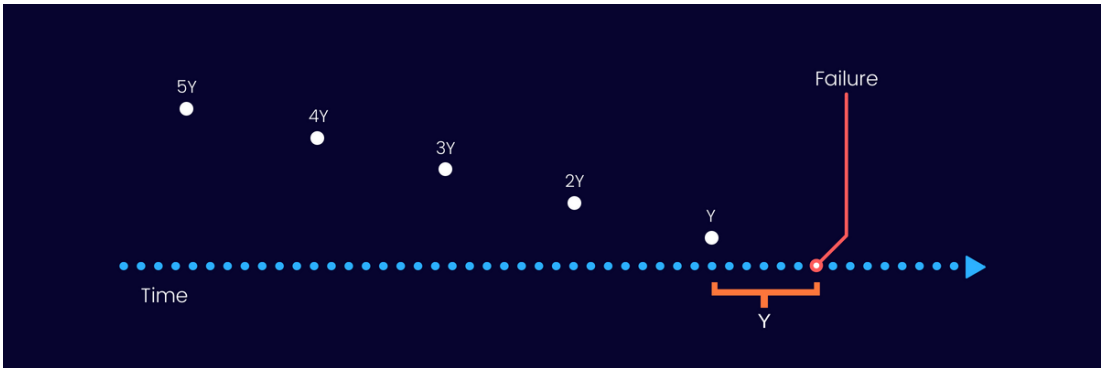
- Tahmin edilmek istenilen parametreler için başlangıç popülasyonunu oluşturulması,
- Popülasyondaki değerler ile modelin oluşturulması ve kayıp fonksiyonunun değerinin takip edilmesi,
- Yeni değerler üretilmesi ve kayıp fonsiyonunun aldığı değerlerin kontrol edilmesi,
- Eğer gerekli ilerleme, yani kayıp fonksiyonun aldığı değerler daha fazla minimize olmuyorsa işlemin sonlandırılması.

## 5. TAHMİNLEYİCİ BAKIM MODELLEMESİ

PdM, bir sistemdeki sorunlar gerçekleşmeden önce sorunları tahmin etmek için sistemden gelen verileri analiz ederek bu arızaların önceden tespit edilmesine olanak sunan bir yöntemdir.

Genellikle üretim yapan kuruluşlarda, çalışan sistemdeki arızaları, sistem çalışırken önlemek için planlı bakımlar yapılmaktadır. Bu durum, beklenmedik bir sebepten dolayı meydana gelebilecek arızaların önüne geçemeyebilmektedir. Sistem sağlıklı durumda olduğunda, gereksiz yere sistemin parçaları değiştirilebilmekte ya da sistemin tüm ekipmanlarının detaylı bir şekilde incelenmesi yapılmaktadır. Bu da gereksiz kaynak tüketimine, bu sebeple verimliliğin düşmesine neden olmakdır.

Endüstri 4.0 tanımı ile yaygınlığı ve kullanımı git gide artış gösteren öngörücü bakım, endüstriyel IoT teknolojilerinin kullanılması birlikte üreticiler ve varlık yöneticileri için önemli bir yatırım alanı haline gelmiştir. Bu yöntem, varlık sağlığını izlemeye, bakım planlarını optimize etmek ve operasyonel risklere karşı gerçek zamanlı uyarılar üretmek için analitik yöntemler geliştirmeye, üreticilerin servis maliyetlerini düşürmelerine ve sistemlerinin çalışma süresini en üst düzeye çıkarmaya, aynı zamanda üretim verimliliğinin de artmasına yardımcı olmaktadır..




Şekil 17. PdM Sisteminde Hatanın Gerçekleşmeden Tahmin Edilmesi

İmaj Kaynak (https://www.seebo.com/predictive-maintenance/)

Bir endüstriyel varlık üzerinde yapılacak PdM için, aşağıdaki temel bileşenler gereklidir:

Sensörler – Herhangi bir fiziksel ürüne veya makineye takılı veri toplamaya yarayan elektronik birimler.

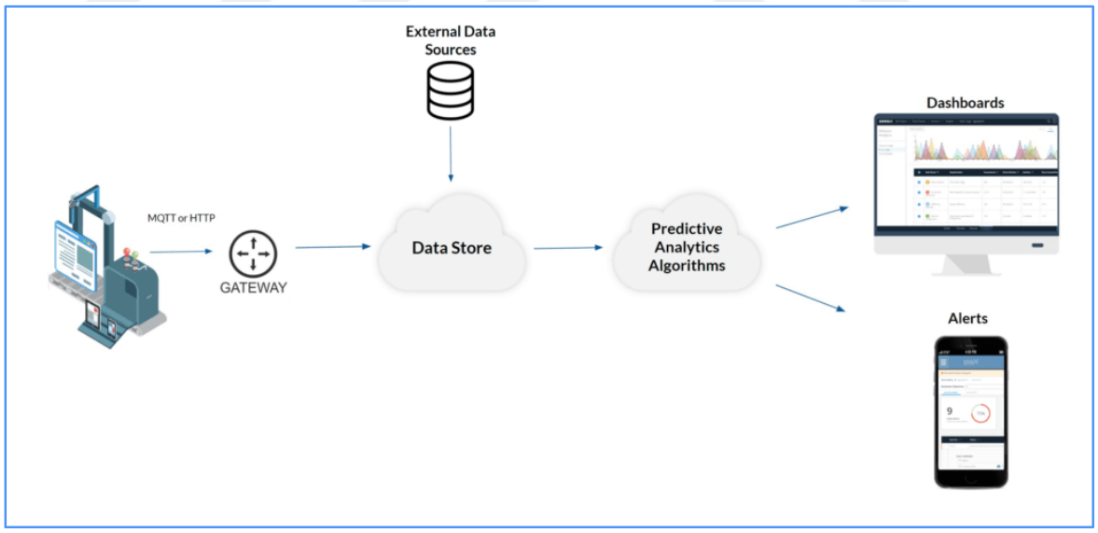
Veri iletişimi – Gözlemlenen makine ya da sistem ile veri depolama merkezi arasında, verilerin güvenli bir şekilde akmasını sağlayan iletişim sistemi.

Veri deposu – Sensörlerden toplanan verilerin depolandığı, işlendiği ve analiz edildiği veri merkezi.

Analitik Yöntemler – Sensör verilerindeki desenleri tanımak için bu verilere uygulanan tahmine dayalı analitik algoritmalar.

Kök neden analizi - bakım ve süreç mühendisleri tarafından öngörüleri araştırmak ve yapılacak düzeltici eylemi belirlemek için kullanılan veri analiz yöntemi.

Makine verileri, iletişim protokolleri ve ağ geçitleri kullanılarak sensörlerden merkezi bir veri deposuna aktarılmaktadır. Burada, kök neden analiz yazılımı kullanılarak incelenen aksama sürelerini azaltmak ve sistemin durumu ile ilgili tahminler üretmek için, tahminleyici analitik algoritmalar uygulanmaktadır.



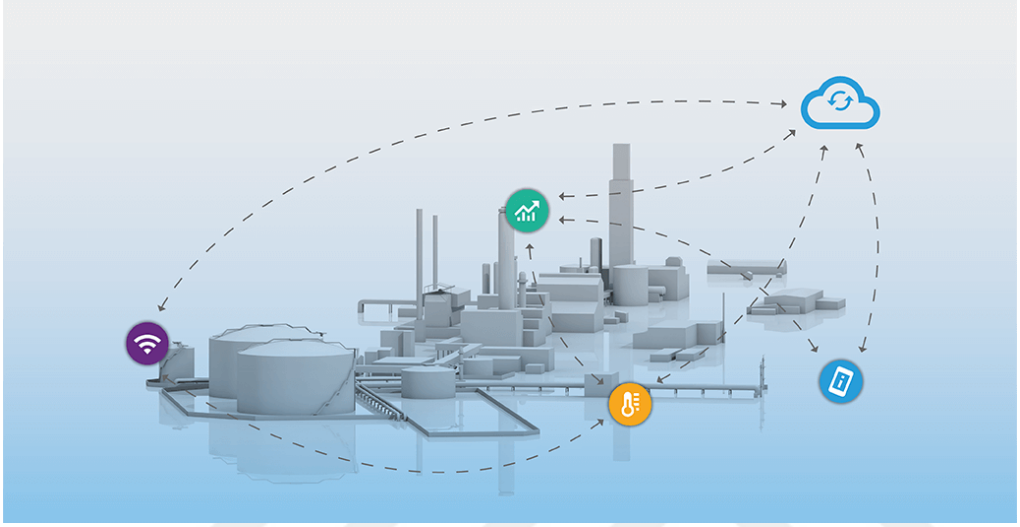

Şekil 18. Örnek PdM sistem mimarisi

İmaj Kaynak (https://www.seebo.com/predictive-maintenance/)

PdM sistemini etkin bir şekilde uygulamak için, arıza parametrelerinin haritalandırılması ve PdM sistemindeki tüm elemanlar (sensörler, iletişim protokolleri, ağ geçidi, bulut ve tahmine dayalı analitik) için detaylı dökümantasyon oluşturulması

gerekmektedir. Görsel bir IoT modelleyicisi kullanarak, mühendislik ekipleri, veri akışlarını, gösterge tablolarını ve sistemin mantığını içeren üretim sorunlarını grafiksel olarak tespit edebilirler.

Üreticiler, öngörücü bakım ile bir dizi avantaj elde edebilmektedir. Bunlar, azaltılmış bakım süresi, yeni gelir akışları, arttırılmış müşteri memnuniyeti ve rekabet avantajı şeklinde sıralanabilmektedir.



Şekil 19. PdM Modellemesi Mimarisine Genel Bakış

İmaj Kaynak (https://www.seebo.com/predictive-maintenance/)

Stratejik bakım planlaması ve proaktif onarımlar için otomatik raporlar, bakım süresini %20 ile %50 arasında azaltmaktadır ve genel bakım maliyetlerini % 5 ile %10 arasında azaltmaktadır. Bu öngörüler, üreticiye ve müşterilerine zamandan ve paradan tasarruf etmelerini sağlamaktadır. Artırılmış verimlilik ve analitik odaklı bilgiler, gereksiz bakımları azaltarak, sistemlerin/makinelerin ömrünün uzamasına katkı sağlamakta ve sistemin başarısızlığından önce meydana gelen sorunların tespit edilmesine yardımcı olarak genel ekipman etkinliğini arttırmaktadır. Üreticiler, müşterileri için PdM gösterge panoları, optimize edilmiş bakım programları veya parçaların değiştirilmesi gerekmeden önce teknisyen sevkiyatı hizmeti dahil olmak üzere, analitik odaklı hizmetler sunarak endüstriyel PdM'den ayrıca bir gelir elde edebilirler. Verilere dayalı olarak müşterilere dijital hizmetler sağlama yeteneği, sürekli gelir akışları ve şirketler için yeni bir büyüme fırsatı sunabilir. Örneğin, parçaların değiştirilmesi gerektiğinde müşterilere otomatik olarak uyarılar gönderilerek müşteri memnuniyetinin artması sağlanabilir (Predictive maintinence).

## 5.1. TAHMİNLEYİCİ BAKIMDA KULLANILAN YAKLAŞIMLAR

Tahmine dayalı bakım için en yaygın iki yaklaşım, kural tabanlı ve ML tabanlı yaklaşımdır. Durum izleme olarak da adlandırılan kural tabanlı yaklaşımda, sistemden, sensörler aracılığı ile sürekli veri toplanmaktadır. Sistem, önceden belirlenen kural tabanlı eşik değerlerine ulaşıldığında uyarılar üretmektedir. Bu yaklaşımda, ürün ekipleri, makinelerin arızalanmasına neden olan faktörleri belirlemek için mühendislik ve müşteri hizmetleri birimleri birlikte çalışmaktadır. Ürün veya parça arızalarının ortak nedenleri belirlendiğinde, sistemlerin sanal bir modeli oluşturulabilir. Burada, çeşitli IoT sistem bileşenleri arasındaki davranışları ve bağımlılıkları tanımlayan "eğer öyleyse, bu akisyonu al" şeklinde belirlenen kurallar oluşturulur. Örneğin, sıcaklık ve dönüş hızı önceden tanımlanmış belirli bir seviyenin üzerindeyse sistem, arızaya neden olabilecek durum ile ilgili aksiyon alınması için sistemlerin bakımından sorumlu operatörlere, durum izleme panoları vasıtasıyla uyarılar göndermektedir. Bu kurallar, bir düzeyde otomatik ve PdM imkanı sağlamaktadır. Ancak yine de bir ürün ekibinin hangi parçalar ile veya hangi çevresel öğeler ile ilgili veri toplanması gerektiğine titizlikle karar vermesi kritik önem taşımaktadır.

AI, PdM ve imalat endüstrisindeki birçok durum için uygulanabilmektedir. Dünyada, endüstriyel AI'dan faydalanan birçok tesis bulunmaktadır. ML ve AI tabanlı sistemler ile üretim sürecinden toplanan büyük miktarda verileri kullanarak anlamlı analizler yapmak mümkündür. ML ile PdM, farklı senaryolar yürütmek ve neyin yanlış gideceğini önceden tahmin etmek için tarihsel veriler üzerinde oluşturulan ML algoritmaları sayesinde mümkün kılınmaktadır.

Bu sayede, gelişmiş algoritmalar bir makinenin davranışlarını öğrenir ve ne tür durumlarda arızanın ortaya çıkabileceği hakkında uyarılar üretebilir hale gelmektedir. ML için gerekli olan algoritmalar girdi (sensör verileri, çevresel faktörler vb.) ve çıktı verilerini (sistemin sağlık durumu) analiz etmektedir. Girdi değişkenleri, sıcaklıktan basınca ve motor hızına kadar bir çok farklı bilgi içermektedir. Çıktı ise gelecekte meydana gelebilecek arıza hakkında bilgiyi içermektedir.

## 5.2. TAHMİNLEYİCİ BAKIM MODELLEMESİ İÇİN ÖNERİLEN MAKİNE ÖĞRENMESİ YÖNTEMLERİ

Sistem güvenilirliği mühendislik uygulamaları için en kritik noktalardan biridir. Sistemin bazı kısımlarının başarısız olması sistemin tamamını etkileyebilmektedir. Örneğin, türbin motorları, güç kaynakları, batarya vb. birimlerde meydana gelen aksaklıklar tüm sistemde aksaklık meydana gelmesine neden olabilmektedir. Sistemin tamamının bozulma durumundan kaçınmak için sistemin belirli kısımlarının ya da tamamının iyi bir şekilde bakım süreçlerinden geçirilmesi oldukça önemlidir. Yaygın olarak bakım stratejilerinde, başarısızlık gözlendiğinde sistemin bir kısmı onarılmaktadır. (Zhou, 2016).

Herhangi bir sistem biriminin mevcut durumunu tahmin etmek için, Jardine ve ark. duruma dayalı bakımı (CBM) önermiştir. CBM'nin temel amacı, gereksiz bakım işlemlerinden kaçınmak ve sistemin herhangi biriminde bir anomali tespit edildiğinde bakım süreçlerinin yürütülmesidir (Jardine, 2006). Etkili CBM stratejisi geliştirmek için sistemin kalan kullanılabilir ömrünün (RUL) yüksek doğrulukla tahmini oldukça önem taşımaktadır. RUL'nin, ilgili sistemin birimlerinde bulunan sensörlerden gelen verileri işleyerek tahmin edilmesi mümkündür. 2008 yılında Felix vd., RUL'yi tahmin etmek için zaman-frekans özellikli bir metot önermiştir. Metot, zaman-frekans yüzeylerinin karmaşıklığını ölçümlemektedir. RUL'yi tahmin etmek için veri odaklı algoritma geliştirilmiştir. Çalışma kapsamında NASA tarafından sağlanan C-MAPSS veri seti kullanılmıştır (Felix, 2008). Porotsky, IEEE PHM 2012 Konferansı kapsamında gerçekleşen yarışmada çapraz doğrulama prosedürüne dayanan parametre optimizasyonunu kontrol etmek için yeni bir model geliştirmiş ve birincilik ödülüne layık görülmüştür, çalışma kapsamında yine NASA tarafından sağlanan C-MAPSS veri seti kullanılmıştır (Porotsky, 2012).

2014 yılında Jain ve ark. RUL'yi tahmin etmek için yapay sinir ağları (ANN) geliştirmiştir. Bu çalışmada, NASA tarafından sağlanan C-MAPPS veri seti üzerinden daha efektif RUL tahmini yapabilmek üzere, YSA temelli bir yaklaşım önerilmiştir. Önerilen model zamana dayalı istatistiksel özellikler üzerine inşa edilmiştir (Jain, 2014). Sateesh ve arkadaşları, RUL tahmini için, Meta-bilişsel Regresyon Sinir Ağı (McRNN) adlı yeni bir yaklaşım önermiştir. McRNN, optimum ağ parametrelerini elde etmek için Genişletilmiş Kalman Filtresi (EKF) kullanılmaktadır. Çalışma

kapsamında C-MAPSS veri seti kullanılmış ve bir önceki çalışmalardan daha etkili bir RUL tahmin metodolojisi elde edilmiştir. (Sateesh vd., 2016).

Santos vd. LSTM ağlarının yeteneklerinden yararlanarak hem uzun hem de kısa süreli aralıklarla arızaları önceden tahmin edebilen SMART parametrelerine dayanan sabit disk sürücüleri için PdM yaklaşımını sunmuştur (Santos, 2017).

Tarafımızdan 2017 yılında yapılan çalışmada, bir motorun mevcut durumunu tahmin etmek için LSTM ağları kullanılarak karar verme süreci geliştirilmiştir. LSTM ağlarının eğitim süreci, yüksek performanslı büyük ölçekli veri işleme ortamında NASA tarafından sağlanan C-MAPSS veri seti kullanılarak gerçekleştirilmiştir (Aydin vd., 2017).

Chen vd. tarafından yapılan çalışmada, LSTM kullanılarak tahmini bakım modeli geliştirilmiştir, LSTM performansı, destek vektör regresyon makinesi (SVRM) ile karşılaştırılmış ve LSTM'in, daha iyi sonuç verdiği gösterilmiştir (Chen, 2017). Das vd. yaptıkları çalışmada, DL yöntemleri kullanarak PdM yöntemleri için Desh isimli bir çözüm önermiştir (Das, 2017). Cachada vd, 2018 yılında olası makine arızalarının ortaya çıkmasının daha erken tespiti için toplanan verilerin gelişmiş ve çevrimiçi analizini göz önünde bulunduran Endüstri 4.0 ilkelerine uyumlu, akıllı ve kestirimci bir bakım sistem mimarisi ile akıllı karar verme sistemlerinin önemini belirten çalışma yayınlamıştır (Cachada, 2018). Öztanır tarafından 2018 yılında yapılan bir diğer çalışmada, son yıllarda bakım planlamada ve pek çok alanda başarılı sonuçlar veren DL tekniğinin kullanımı ve diğer makine öğrenme yöntemlerine göre performansı incelenmiştir. Deneyler NASA tarafından sağlanan C-MAPSS veri seti kullanılarak gerçekleştirilmiştir. (Öztanır, 2018). Zhang vd., 2018 yılında, toplanan verilere dayanarak ekipmanın çalışma durumunu analiz etmek, çalışma durumunu tahmin etmek için LSTM mimarisi tasarlamıştır (Zhang, 2018).

## 6. ÖNERİLEN YÖNTEM

Nesnelerin İnterneti (IoT)'nin hızla geliştiği günümüzde, bir çok firma, sensörler aracılığı ile veri toplamakta ve bu verileri anlık olarak analiz edip gerekli aksiyonları alabilecek konuma gelmiştir. Özellikle motor üreticileri, motor bileşenlerine yerleştirdikleri sensörler ile topladıkları sinyalleri işleyip motor bileşenlerinin sağlığını gözlemleme kabiliyetine erişmiştir. IoT sensörlerinin ürettikleri verilere göre

üreticiler, motor bileşenlerinin koşullarını tahmin etmek için de akıllı sistemler inşa etmektedir.

Pratik olarak her hangi bir sistemin bileşenlerinin, yaşam ömürlerini yitirmeden önce değiştirilmesi gerekmektedir. Bileşen arızalarının tahmini sayesinde, bileşenler önceden onarılarak bakım maliyetleri azaltılırken, operasyonların verimli bir şekilde sürdürülmesi de mümkün kılınmaktadır. Bakım süreci, üretim kapasitesini ve hizmet kalitesini doğrudan etkilediği için, optimize edilmiş bakım, kuruluşların daha fazla gelire sahip olmaları ve sanayileşmiş dünyada rekabet ortamında ayakta kalabilmeleri için kilit faktör niteliği taşımaktadır.

Bir motorun mevcut durumunu anlamak için iyi tasarlanmış tahmin sisteminin yardımı ile, arıza meydana gelmeden önce bileşenler aktif hizmetten çıkarılabilmekte, etkili bakım süreçleri sayesinde bileşenlerin ömrü uzamakta, ekipmanın kullanılabilirliği artmakta ve aynı zamanda bakım maliyetleri de düşmektedir.

Sensörlerden toplanan gerçek zamanlı veriler, sistem bileşenlerinin bozulma paternlerini modellemek için oldukça önemlidir. Bu konu ile ilgili son yıllarda yapılan çalışmalara bakıldığında, Bölüm 5.2'de de belirtildiği üzere, motor bileşenlerinin kalan ömrünü ve güvenilirliğini tahmin etmek için araştırmacılar bu konu ile ilgili çozüm önermeye, arıza oranlarının olasılığını bulmaya dayalı Gizli Yarı Markov modelleri kullanarak baslamıştır. Tahmini bakım modelleri inşaa etmek için araştırmacıların ilgi alanı yıllar içerisinde Yapay Sinir Ağları (ANN) üzerine yogunlaşmıştır. Genellikle, ileri besleme sinir ağı topolojisi kullanılarak tahminleyici sistemler tasarlanmaktadır. Bu sistemlerde, minimum tahminleme hatasını veren ağ, optimum çıktıyı temsil edecek şekilde seçilmektedir. Özellikle son iki yılda yapılan çalışmalarda Tekrarlayan Sinir Ağları (RNN) kullanılmıştır. Bu çalışmalarda, RNN tabanlı yöntem olan LSTM ağları kullanılarak PdM modellemesi geliştirilebileceği ve LSTM ağlarının diğer ağlara ve algoritmalara göre üstünlük sağladığı gösterilmiştir.

Bu çalışma kapsamında önerilen yöntem de RNN'nin özel bir uygulaması olan LSTM ağlarına dayanmaktadır ve bir motorun mevcut durumunu tahmin etmek için karar verme süreçlerini içeren akıllı bir sistem önerisi sunmaktadır. LSTM ağlarının eğitim süreci, yüksek performanslı büyük veri platformu olan ve dağıtık hesaplama imkanı sunan Apache Spark ortamında gerçekleştirilmiştir. Bölüm 5'te de görüldüğü gibi NASA Ames'deki Prognostics CoE tarafından sağlanan bir motor bozulma simülasyonu olan veri seti PdM modellemesi konusunda çalışma yapan araştırmacılar tarafından sıklıkla kullanılmaktadır ve literatüre geçmiş açık kaynaklı veri setidir.

Çalışma kapsamında da bu veri seti kullanılarak ağ eğitilmiş ve yine bu veri seti üzerinde ağ test edilmiştir. Veri seti, çok değişkenli zaman serisi formatındaki motorun farklı çalışma koşulları ve farklı hata modlarının kombinasyonlarını içermektedir. Çizelge 1'de detaylı olarak belirtildiği üzere veri setinde, üç operasyonel ayar ve 21 sensöre ait ölçümler yer almaktadır. Sensörler; sıcaklık, motor basıncı, yakıt, soğutma sıvısı tahliyesi ile ilgili ölçümleri içermektedir (Saxena vd., 2008).

Çizelge 1. Veri Setindeki Değişkenlerin Tanımı

| **Operasyonel Ayarlar** | |
|---|---|
| **Ayar No** | **Tanım** |
| 1 | Yükseklik |
| 2 | Mak sayısı |
| 3 | Gaz çözme açısı |
| **Sensör Ölçümleri** | |
| **Sensör No** | **Tanım** |
| 1 | Fan girişindeki toplam sıcaklık (° R) |
| 2 | LPC çıkışındaki toplam sıcaklık |
| 3 | HPC çıkışındaki toplam sıcaklık |
| 4 | LPT çıkışındaki toplam sıcaklık (° R) |
| 5 | Fan girişindeki basınç (psia) |
| 6 | Bypass kanalındaki toplam basınç (psia) |
| 7 | HPC çıkışındaki toplam basınç (psia) |
| 8 | Fiziksel fan hızı (rpm) |
| 9 | Fiziksel çekirdek hızı (rpm) |
| 10 | Motor basıncı oranı (P50 / P2) |
| 11 | Motor basıncı oranı (P50 / P2) |
| 12 | Yakıt akışının Ps30'a oranı (pps / psi) |
| 13 | Düzeltilmiş fan hızı (rpm) |
| 14 | Düzeltilmiş çekirdek hızı (rpm) |
| 15 | Bypass Oranı |
| 16 | Brülör yakıt-hava oranı |
| 17 | Taşma Entalpi |
| 18 | Talep edilen fan hızı (rpm) |
| 19 | Talep edilen düzeltilmiş fan hızı (rpm) |
| 20 | HPT soğutma sıvısı sızması (lbm / s) |
| 21 | LPT soğutma sıvısı sızması (lbm / s) |

Model inşa edilme aşamasına geçilmeden önce ilk olarak, motorun kalan kapasitesini gösteren $R_i$ yüzdesi, aşağıdaki denklem kullanılarak, tarafımızdan 2017 yılında yapılan çalışmada (Aydin vd., 2017) kullanıldığı şekilde hesaplanmıştır.

$R_i$ = (Arızaya kadar geçen zaman – Motorun yaşı) / Arızaya kadar geçen zaman (7)

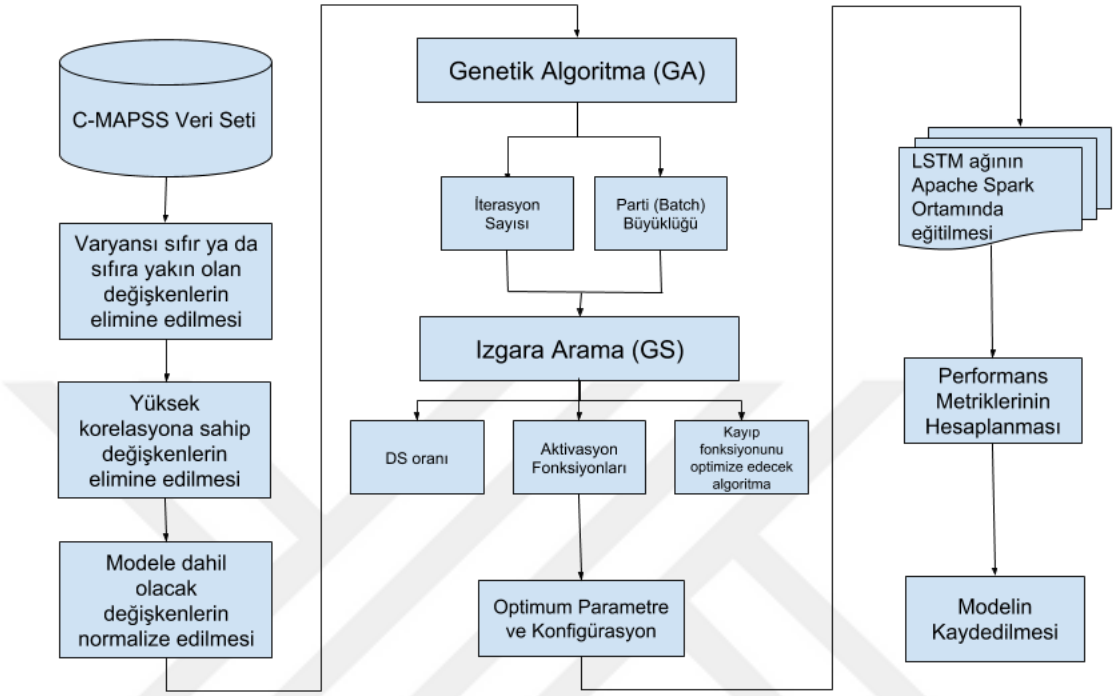
Denklem 7 ile elde edilen $R_i$, bire eşit olduğunda, motorun kalan kapasitesi %100 olduğu anlamına gelmektedir. $R_i$ sıfıra eşit olduğunda, motorun kalan kapasitesi %0'dır yani motor arızalıdır ve çalışmamaktadır. Yine 2017 yılında tarafımızca gerçekleştirilen çalışmada literatürde ilk olarak $R_i$ değerlerinin motor bileşenlerin sağlık durumunu belirten dört sınıfa ayrılması önerilmiştir. Bu dört sınıf, Çizelge 2'de gösterildiği gibi "motor sağlıklı", "kritik durum", "motoru bakıma al", "motor arızalı", şeklindedir. "motor sağlıklı", motorun kullanım ömrünün başında olduğunu gösterir. "kritik durum", motorun kısa zaman içerisinde bakıma gitmesi gerektiğini gösterir. "motoru bakıma al", motorun kısa zaman içerisinde bozulacağını ve acilen bakıma alınması gerektiğini, "motor arızalı", motorun hizmet dışı olduğunu, kullanılamaz hale geldiğini belirtmektedir. Bu aralıklar, uçak motoru konusunda uzman mühendislerin görüşleri alınarak belirlenmiştir.

Çizelge 2. Motorun sağlık durumunu gösteren kategoriler

| **R Aralığı** | **Motor durumu** |
|---|---|
| (0.85-1] | Motor Sağlıklı |
| (0.7-0.85] | Kritik Durum |
| (0.5-0.7] | Motoru Bakıma Al |
| [0-0.5] | Motor Arızalı |

Çıktı olarak, Çizelge 2'de belirtilen, "motor sağlıklı", "kritik durum", "motoru bakıma al", "motor arızalı" şeklinde belirtilen dört adet farklı durumdan bir tanesi tahmin olarak üretilmektedir. Motor üzerindeki bir çok sensörden gelen verilerin analiz edilmesi, bakım süreçlerini yönetmek için, yararlı olan doğru ve net sonuçların ortaya çıkmasına yardımcı olmaktadır. ML, ANN modelleri inşa edilirken mümkün mertebe yanıt değişkenini en iyi şekilde temsil eden açıklayıcı değişkenlerin seçilmesi önem

taşımaktadır. Elbette bazı sensör verilerini göz ardı etmek ve modelleme dışı bırakmak önyargı seviyesini arttırabilir, bu nedenle, bu iki durum arasında kusursuz bir denge sağlanmalıdır.

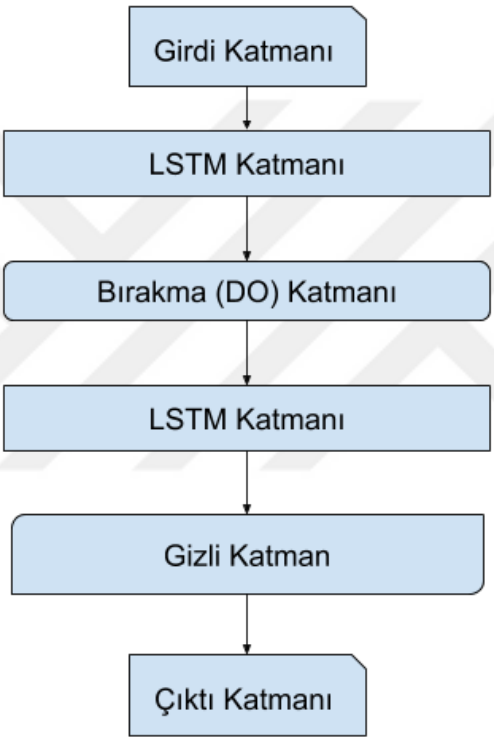



Şekil 20. Önerilen Metodoloji (GAGS-LSTM)

Önerilen metodoloji Şekil 20'de gösterildiği gibidir. Motordan gelen verilerdeki desenleri öğrenmek için etkili bir tahmin modeli oluşturmak amaçlandığından, modellemeye geçmeden önce değişken mühendisliği (feature engineering) yapılmıştır. Bu kapsamda, varyansı sıfır olan değişkenler, aralarında yüksek korelasyona sahip değişkenler elimine edilmiştir. Ardından modele dahil edilecek olan nihai değişkenler normalize edilerek aynı düzleme taşınmıştır. İkinci aşamada, mimaride kullanılacak parametrelere karar verilen GA ve GS işlemlerinin gerçekleştirildiği aşamalar bulunmaktadır. Bu aşamalarda, ağın eğitilme aşamasına geçilmeden, DO katmanındaki DO oranına, LSTM katmanlarından sonra hangi aktivasyon fonksiyonlarının kullanılacağına, ağın kaç iterasyonda eğitileceğine, her iterasyonda kullanılacak olan parti büyüklüğüne, kayıp fonksiyonunu optimize etmekte kullanılacak optimizasyon yöntemine karar verilmektedir.

Ağda kullanılacak olan parametrelere karar verilen aşamanın ardından, Sekil 21'de gösterilen; giriş katmanı, bir LSTM katmanı, bir dropout katmanı yine bir LSTM

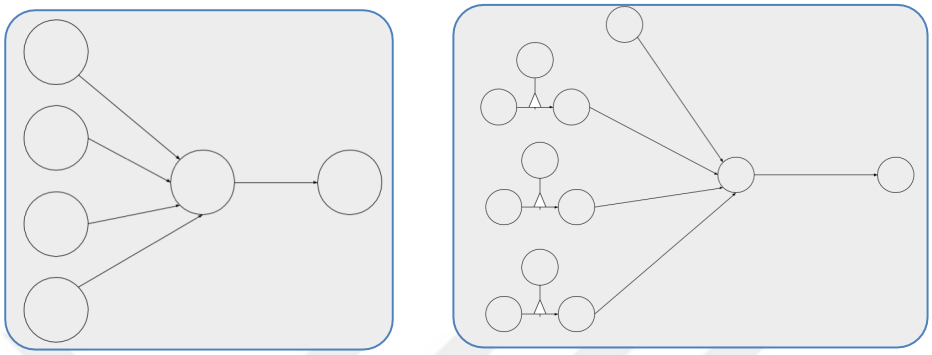
katmanı, çıkış katmanından önce bir gizli katman, ve çıkış katmanı içeren DNN ile modelin eğitilme aşaması gelmektedir. Giriş katmanı, farklı sensörlerden gelen verileri karşılamaktadır. Bu veriler zaman serisi formatındadır ve motor çalıştığı süre içerisinde motorun durumu ile ilgili bilgileri içermektedir. LSTM ağları, Bölüm 3.2.1.1'de belirtildiği gibi içerisinde girdi, unutma ve çıktı birimleri yardımıyla hangi zaman birimindeki verilerin unutulması gerektiğine, hangi zaman birimindeki verilerin unutulmayıp bir sonraki zaman birimine taşınacağına karar verme özelliğine sahip ağlardır.




Şekill 21. Önerilen LSTM Mimarisi

Önerilen mimaride, ilk LSTM katmanından sonra ağın öğrenme sürecinin regulerize edilmesine yardımcı olan DO katmanı bulunmaktadır. Bu katman, her iterasyonda bilgileri maskeleyerek ağın eğitim veri setindeki kalıpları ezberlemesini önlemektedir. Çok sayıda parametreye sahip DNN'ler çok etkili makine öğrenme yöntemleridir. Ancak, ağın ezberlemesi problemine bu tür ağlar eğitilirken sıklıkla karşılaşılmaktadır. Bu ağlar oldukça fazla parametre içerdiği için zaman zaman ağı eğitme süreci oldukça uzun sürebilmektedir.

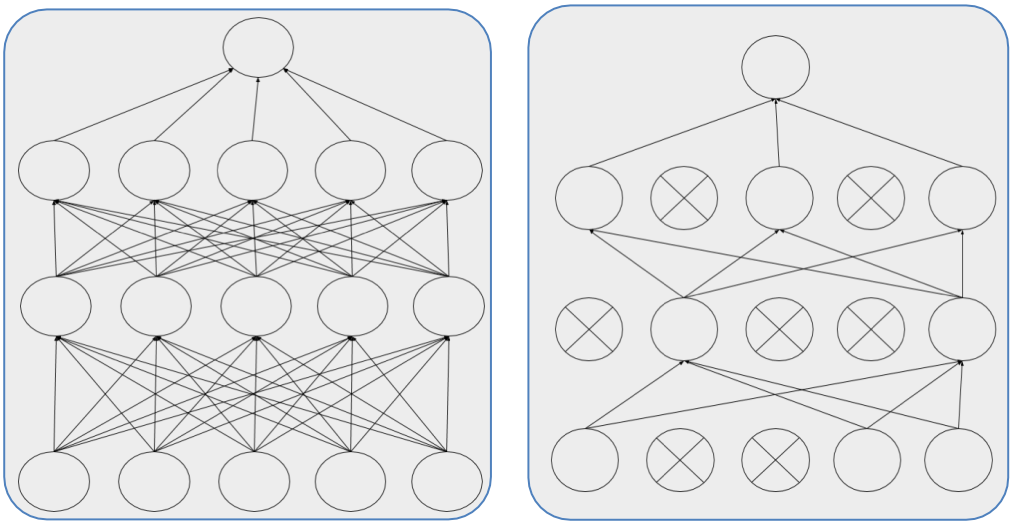
DO, bu sorunu gidermek için kullanılan bir tekniktir. Bırakma yönteminin arkasında yatan temel fikir, ağın eğitimi sırasında ağ üzerindeki nöronların rasgele deaktive edilmesidir.



Standart Sinir Ağı DO Adapte Edilmiş Sinir Ağı

Şekil 22. DO'lu ve DO'suz Ağın Kıyaslaması

Bırakma tekniğinin, sinir ağlarının görüntü işleme, konuşma tanıma, sınıflandırma problemlerindeki performansını arttırdığı kanıtlanmıştır. Ağın, eğitim veri setini öğrenme aşamasında ezberlemesi her zaman yapay sinir ağının genelleştirilmesini zorlaştırmaktadır.



Şekil 23. DO uygulanmamış (Soldaki) ve DO Uygulanmış (Sağdaki) Sinir Ağı

Bazı düzenlileştirme (regularization) teknikleri, parametrelere bir sınırlama getirerek ve buna bağlı olarak kayıp fonksiyonunu değiştirerek bu sorunun çözülmesine yardımcı olmaktadır. DO, diğer tekniklerden farklı olarak, ağın kendisini değiştirmesi ve ağı düzenli hale getirme üzerine kuruludur. 2014 yılında Srivistava, Hinton, Krizhevsky, Sutskever, Salakhutdinov tarafından geliştirilmiş bir tekniktir. DO tekniği, derin sinir ağlarının eğitimi sırasında *p* olasılığı ile nöronları geçici olarak rasgele bir şekilde deaktive etmektedir ve ağın eğitilme sürecini, heuristik bir şekilde, her adımda ağın içerisinde bulunan nöronlardan seçilen farklı nöron setlerini kullanarak yapılmasına olanak kılmaktadır. Nöronların hali hazırdaki öğrenilmiş bilgileri öğrenmeye çalışmayıp öğrenilmemiş bilgileri öğrenmelerine olanak tanımaktadır. Bu sayede ağın, eğitim setini ezberleme olasılığı düşmektedir (Srivastava vd., 2014).

Önerilen ağda, DO katmanının ardından gelen bir diğer LSTM katmanı yine veri setindeki ilgili ve ilgili olmayan bilgileri filtreleyerek veri setindeki kalıpları öğrenmektedir. Her iki LSTM katmanı da aynı prensipte çalışmaktadır, yalnızca degerli bilgilerin seçilmesine izin veren filtreler olarak işlev görmektedirler. İkinci LSTM katmanının çıktısı olarak yedi değişken bulunmaktadır. İkinci LSTM katmanının ardından gelen ek gizli katmanın amacı, yedi değişkeni girdi olarak alıp bu değişkenlerin sayısını dörde düşürerek dört değişkeni çıktı katmanına iletmektir. Böylece çıktı katmanı, nihai sınıf atamalarını yapmak için yedi girdi degişkeni yerine dört girdi değişkeni üzerinde çalışmaktadır. Çıktı katmanına giren değişken sayısı dörde indirgenerek sistemin daha verimli ve daha doğru hale getirilmesi amaçlanmıştır. Gizli katman, yedi girdi değişkeninin her birinde yer alan bilgileri, son çıktı katmanına gönderilecek olan dört yeni meta girdi değişkenine rasgele bir şekilde dağıtılmasını sağlamaktadır.

LSTM ağlarının parametre optimizasyonu için literatürde 2018 yıllında yapilan iki çalışma tespit edilmiştir. Salah vd. tarafından 2018 yılında yapılan çalışmada optimal zaman gecikmelerini ve kaç katmanlı LSTM ağının kullanılması gerektiğini tespit etmek için GA kullanılmıştır (Salah, 2018). Yine 2018 yılında Chung vd. tarafından yapılan çalışmada, hisse senetlerine yönelik tahmin modeli geliştirmek için GA kullanılarak LSTM ağlarındaki optimum gizli nöron sayıları tespit edilmiştir (Chung, 2018).

Bu bilgiler ışığında, çalışma kapsamında geliştirilen ağın eğitim sürecindeki iterasyon sayısı ve parti büyüklükleri GA kullanılarak belirlenmiştir. Genel prensip, GA yardımıyla kayıp fonksiyonunu minimize eden iterasyon sayısını ve parti büyüklüğünü tespit etmektir. Ağın eğitim sürecindeki iterasyon sayısı ve her iterasyonda kullanılacak olan parti büyüklüklerine GA ile karar verilirken aşağıdaki adımlar izlenmiştir.

1. İterasyon sayısı ve parti büyüklüğü için başlangıç iterasyonu oluştur (*İlk beş birey iterasyon sayısı için, sonraki beş birey parti büyüklüğü için olmak üzere başlangıç popülasyonu 10 birey olarak belirlenmiştir*).
2. Her kromozom için kayıp fonksiyonu değerini hesapla.
3. Seçim, genetik değişim ve mutasyon işlemlerini gerçekleştir.
4. Her kromozom için kayıp fonksiyonu değerini hesapla.
5. Minimum kayıp fonksiyonu değerine ulaşıldıysa işlemi durdur, ulaşılmadıysa adım 3'e geri dön.

GA ile elde edilen sonuçlar detaylarıyla birlikte Bölüm 7'de belirtilecektir. Literatürde LSTM ağlarındaki iterasyon sayılarına ve her iterasyondaki parti büyüklüklerine karar vermek için GA kullanılan bir çalışmaya rastlanılmamıştır bu kapsamda da bu çalışma bir ilk olma niteliği taşımaktadır.

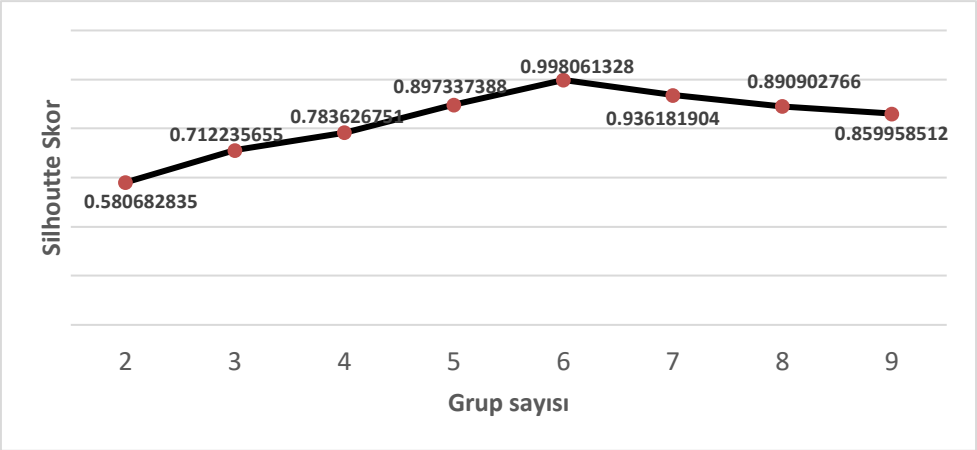
Efektif bir model elde etmek için tüm parametrelerin optimum olarak elde edildiğinden yani kayıp fonksiyonunu minimize eden parametrelerin bulunduğundan emin olunması gerekmektedir. Önerilen ağda kullanılan tüm parametreleri optimum olarak elde etmek icin diğer parametreler olan DO oranına, kayıp fonksiyonunu optimize eden optimizasyon fonksiyonuna, çıktı katmanından sonra kullanılacak olan aktivasyon fonksiyonuna GS kullanılarak karar verilmiştir. GS, bir ızgarada belirtilen her algoritma parametresi kombinasyonu için metodik olarak model oluşturmakta ve değerlendirecek olan hiper parametre ayarı için kullanılan bir yöntemdir (Chin-Wei et. al., 2010). GS kullanılarak elde edilen deney sonuçlarına Bölüm 7'de detayları ile değinilmektedir.

Önerilen çözüm, motorun arıza durumuna geçmeden önce bir uyarı üretmesini hedeflemekte ve LSTM mimarilerinin eğitimi sırasında gerekli iterasyon sayısını ve

parti büyüklüğünü GA ile aynı zamanda DO oranını, kayıp fonksiyonunu optimize etmek için kullanılacak optimizasyon algoritmasını, çıktı katmanından sonra kullanılacak aktivasyon fonksiyonunu GS ile otomatik ve optimum olarak tespit eden ilk çalışma olma niteliği de taşımaktadır. Geliştirilen sistem farklı veri setlerine adapte olabilme ve her veri setinin özelliğine uygun optimum parametreleri tespit etme kabiliyetine sahiptir. Bu sayede, kurumlar, araştırmacılar motorların/sistemlerin sağlık durumları hakkında uyarılar oluşturma imkanına sahip olabilecek, sistem bozulmadan önce problemli parça onarılarak ya da değiştirilerek tüm sisteminin durmasının önüne geçilebilecektir.

## 7. BULGULAR

Çalışma kapsamında yapılan deneylerden elde edilen sonuçlar, mevcut motor durumunun "motoru bakıma al" olarak tahmin edilmesi durumunda gerekli önlemlerin alınabilmesi için önemli olduğunu göstermektedir. NASA Ames'deki Prognostics CoE tarafından sağlanan bir motor bozulma simülasyonu olan açık kaynaklı veri seti incelendiğinde, Bölüm 6'da detaylı bir şekilde belirtildiği üzere, veri setinde




Şekil 24. Grup sayıları ve silhoutte skorlar

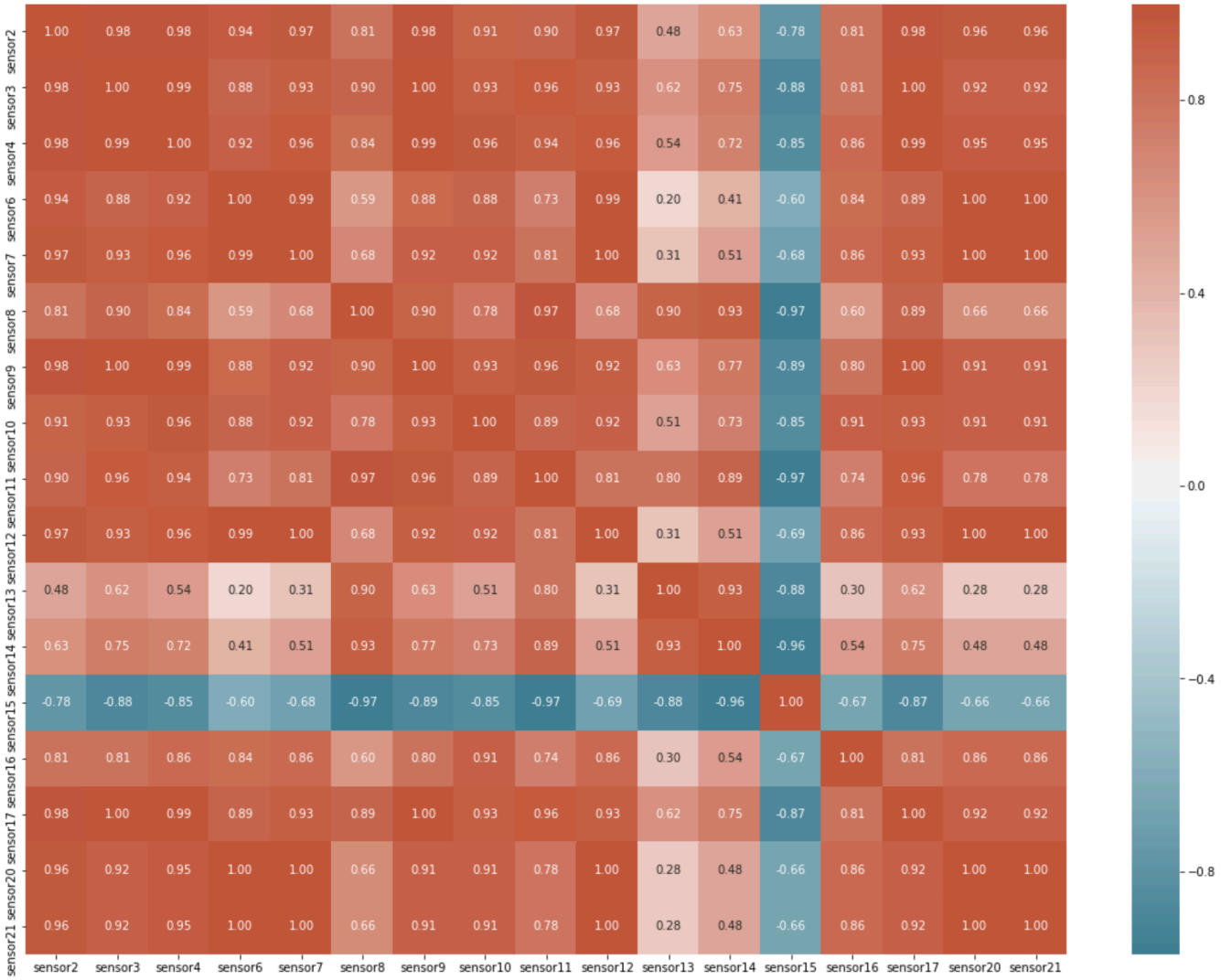
motorun üç farklı durumunu temsil eden üç farklı ayara ait değişkenler, motorun hayat döngüsünü gösteren döngü numarası değişkeni ve sensörlerden gelen verileri gösteren değişkenlerin bulunduğu görülmüştür. Etkili bir model kurabilmek için değişkenler modele özenle dahil edilmesi gerektiğinden, özellik seçme (feature engineering) kapsamında, üç farklı motor ayarı bilgisine ait değişkenler bir değişkende birleştirilmiş ve kümeleme algoritması k-ortalamalar uygulanarak altı alt gruba ayrılmıştır.

K-ortalamalar algoritmasındaki küme sayısına karar vermek için grup adedi 2'den 9'a kadar silhoutte skorları hesaplanmıştır ve Şekil 24'te de görüldüğü gibi silhoutte skorunun en yüksek değere, grup sayısı 6 iken ulaştığı görülmektedir. Ardından bu altı grup içinde varyansı tüm gruplar için sıfır olan değişkenler tespit edilip modele dahil edilmemiştir. Çizelge 3'te de görüldüğü gibi; sensör1, sensör5, sensör18, sensör19 sıfır varyansa sahip olduğu için modelin dışında tutulmuştur. Çünkü, bu sensörler zamana ve farklı sistem ayarlarına bağlı olarak hiç değişkenlik göstermemiştir.

Çizelge 3. Sensörlerin Ayar Gruplarina Gore Varyanslari

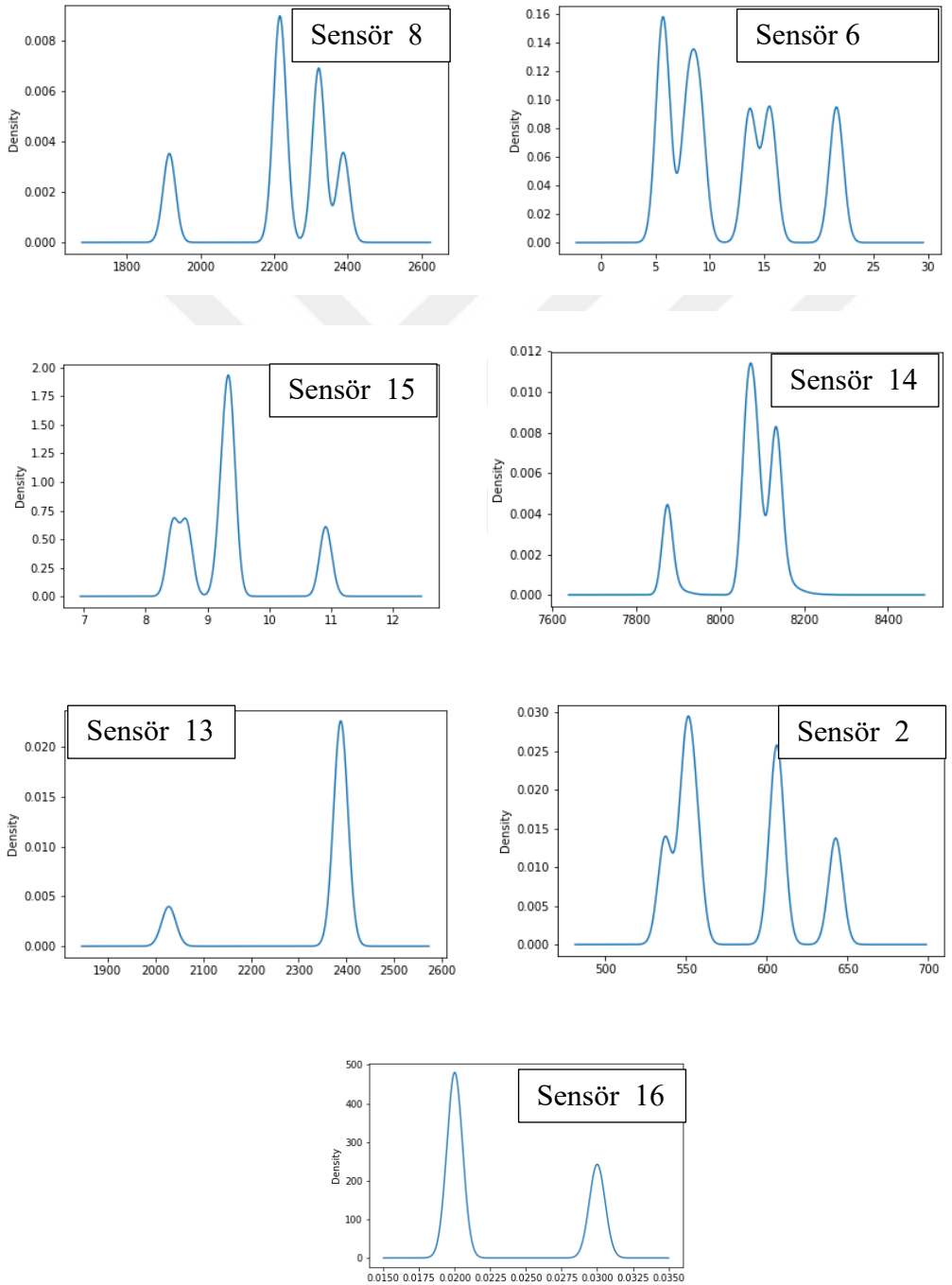
| Yeni Ayar Grup Numarası | sensor1 | sensor2 | sensor3 | sensor4 | sensor5 | sensor6 | sensor7 | sensor8 | sensor9 | sensor10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 0.19278 | 32.08744 | 57.06070 | 0.00000 | 0.00002 | 0.19596 | 0.09108 | 326.92376 | 0.00000 |
| **2** | 0 | 0.22358 | 33.78096 | 68.88127 | 0.00000 | 0.00002 | 0.42472 | 0.00873 | 311.78145 | 0.00000 |
| **3** | 0 | 0.12440 | 27.52906 | 48.07277 | 0.00000 | 0.00002 | 0.20060 | 0.06903 | 200.65908 | 0.00000 |
| **4** | 0 | 0.23371 | 34.73644 | 76.29816 | 0.00000 | 0.00000 | 0.73169 | 0.00465 | 386.50923 | 0.00000 |
| **5** | 0 | 0.21927 | 33.01496 | 60.84646 | 0.00000 | 0.00002 | 0.35979 | 0.01656 | 316.45155 | 0.00002 |
| **6** | 0 | 0.19301 | 31.97951 | 54.44673 | 0.00000 | 0.00000 | 0.22566 | 0.07995 | 296.84603 | 0.00000 |

| Yeni Ayar Grup Numarası | sensor11 | sensor12 | sensor13 | sensor14 | sensor15 | sensor16 | sensor17 | sensor18 | sensor19 | sensor20 | sensor21 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0.05493 | 0.11744 | 0.10602 | 276.92093 | 0.00136 | 0.00000 | 1.98661 | 0.00000 | 0.00000 | 0.01161 | 0.00406 |
| **2** | 0.06234 | 0.27514 | 0.00908 | 244.89687 | 0.00139 | 0.00000 | 2.13909 | 0.00000 | 0.00000 | 0.02083 | 0.00744 |
| **3** | 0.04596 | 0.11654 | 0.07698 | 170.94166 | 0.00192 | 0.00000 | 1.71610 | 0.00000 | 0.00000 | 0.01244 | 0.00450 |
| **4** | 0.06596 | 0.50366 | 0.00473 | 284.51418 | 0.00132 | 0.00000 | 2.21332 | 0.00000 | 0.00000 | 0.03013 | 0.01091 |
| **5** | 0.05555 | 0.23894 | 0.01722 | 243.97113 | 0.00145 | 0.00002 | 2.05644 | 0.00000 | 0.00000 | 0.01776 | 0.00640 |
| **6** | 0.05275 | 0.14330 | 0.09197 | 242.63281 | 0.00145 | 0.00000 | 1.94845 | 0.00000 | 0.00000 | 0.01266 | 0.00477 |



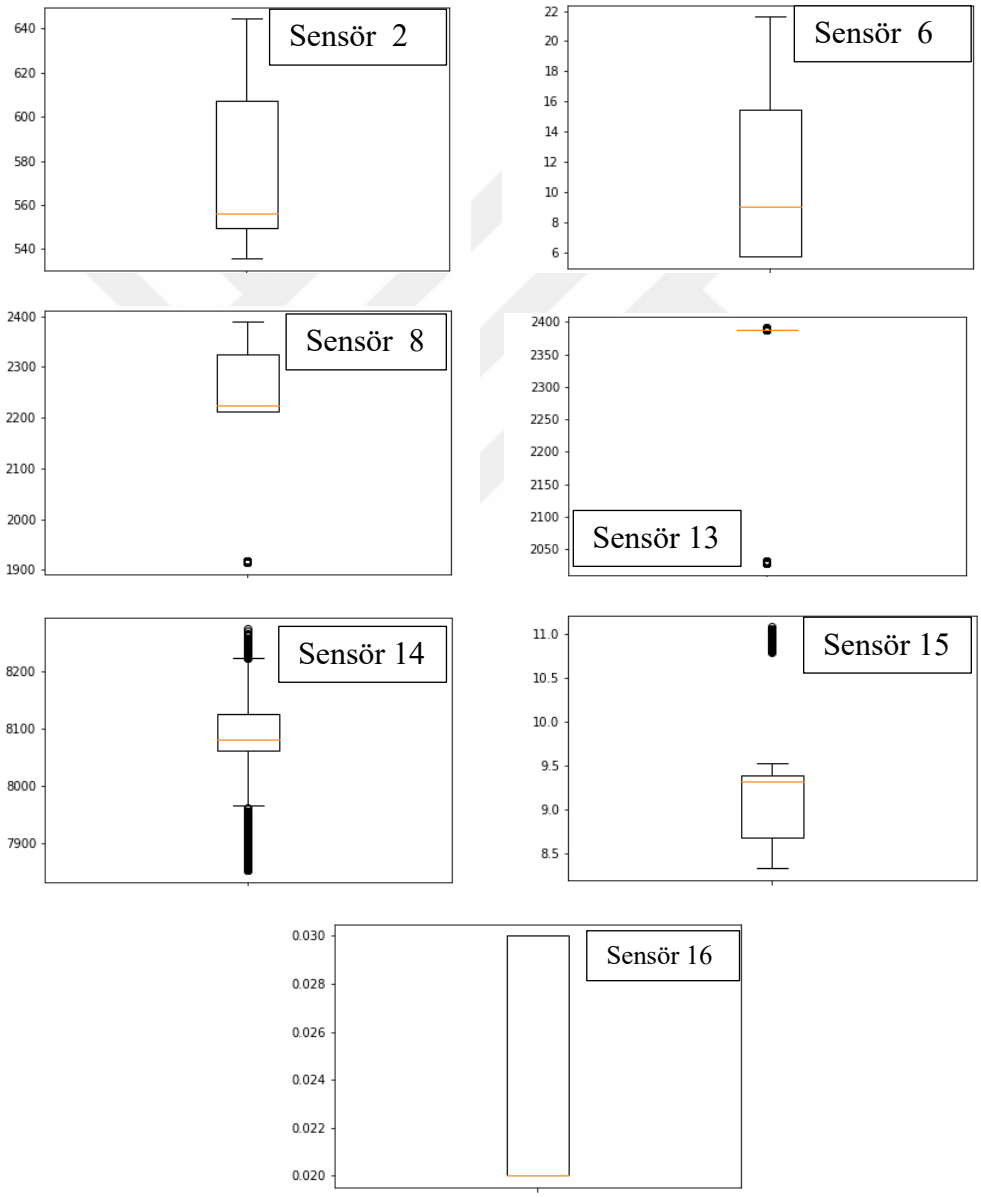

Şekil 25. Sensörler için Korelasyon Matrisi

Zamana bağlı değişim göstermeyen değişkenleri elimine ettikten sonra bir sonraki adımda, yüksek korelasyona sahip değişkenler elimine edilmiştir. Şekil 25'te sensörlerin birbirleri ile korelasyonları görülebilmektedir. Bu durumda sensör2, sensör6, sensör8, sensör13, sensör14, sensör15 modele dahil edilecek olan sensörlere ait değişkenler olarak belirlenmiştir. Şeki 26'da sensör2, sensör6, sensör8, sensör13, sensör14, sensör15, sensör16'dan gelen verilerin dağılımları gösterilmiştir.




Şeki 26. Sensörlerden gelen verilerin dağılımları

Sensörlerden gelen verilerin dağılımına bakıldığında, verilerin tek bir noktada toplanmadığı açıkça görülmektedir. Dağılımların birden çok tepeli olması motorların çalışma anında, sensörlerin motorların davranışlarına ait bilgileri faydalı bir şekilde toplayabildiğini göstermektedir. Şekil 27'de sensörlere ait kutu grafikleri gösterilmiştir. Kutu grafiklerine de bakıldığında yine aynı şekilde, sensörlerden gelen verilerin simetrik dağılıma sahip olmadığı ve motorun çalışma sürecinde üretilen verinin tahmin edilmesi zor bir yapıya sahip olduğu da görülmektedir.




Şekil 27. Sensörlerden gelen verilere ilişkin boxplotlar

Modele dahil edilecek olan girdi değişkenlerine karar verildikten sonra, önerilen metodoloji kapsamında, Bölüm 6'da da belirtildiği üzere, öncelikle ağın kaç iterasyonda eğitileceği ve her iterasyondaki parti büyüklüğünün ne kadar olacağı GA yardımıyla belirlenmiştir. Çizelge 4'te GA'dan elde edilen iterasyon sayısı, parti büyüklüğü ile eğitilen modellerin eğitim seti ve doğrulama seti üzerindeki doğruluk oranları gösterilmektedir.

Çizelge 4. GA ile Elde Edilen İterasyon Sayıları ve Parti Büyüklükleri

| **İterasyon Sayısı** | **Parti Büyüklüğü** | **Eğitim Seti Doğruluk Oranı** | **Doğrulama Seti Doğruluk Oranı** |
|---|---|---|---|
| 27 | 24 | 0.8397 | 0.8309 |
| 6 | 20 | 0.8358 | 0.8304 |
| 18 | 22 | 0.8403 | 0.8097 |
| **31** | **20** | **0.842** | **0.8341** |
| 12 | 22 | 0.8389 | 0.8289 |
| 11 | 3 | 0.8352 | 0.8189 |
| 20 | 13 | 0.8401 | 0.831 |
| 27 | 30 | 0.8419 | 0.8271 |
| 12 | 16 | 0.8404 | 0.824 |
| 4 | 24 | 0.8357 | 0.8292 |

Çizelge 4'ten de görüldüğü üzere en yüksek doğruluk oranı parti büyüklüğü 20 ve iterasyon sayısı 31 iken elde edilmiştir. İterasyon sayısı ve parti büyüklüğüne karar verildiğinden, bir sonraki adım, GS kullanılarak DO oranına, kayıp fonksiyonunu optimize edecek olan optimizasyon algoritmasına, çıktı katmanın ardından kullanılacak aktivasyon fonksyonuna karar verilme aşamasıdır.

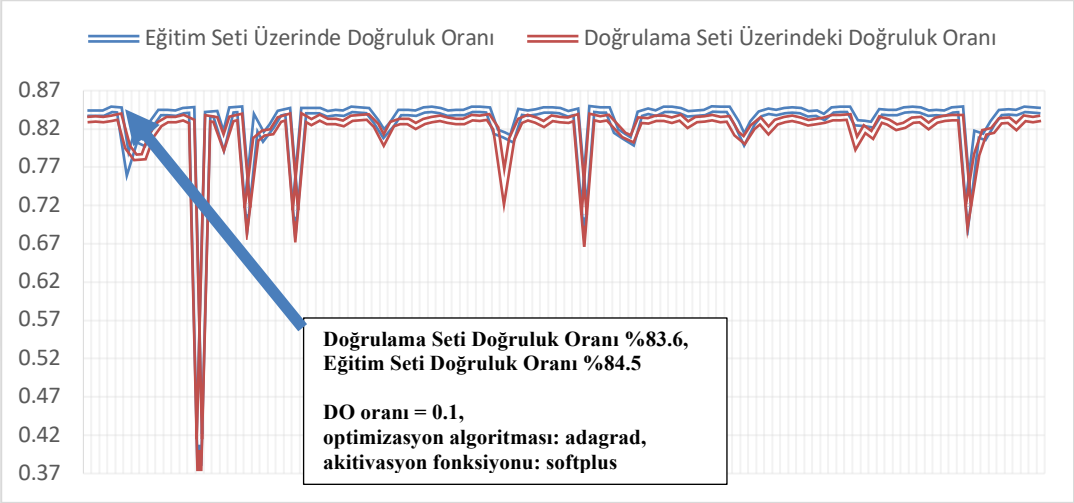
**DO oranı için:** 0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9

**Optimizasyon algoritması için:** adadelta, adagrad, sgd, adam, adamax

**Aktivasyon fonksiyonu için:** softmax, softplus, sigmoid

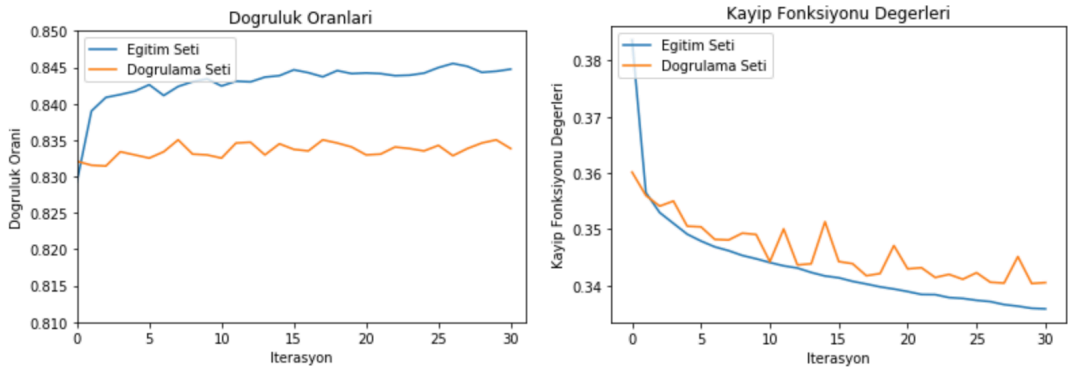
Değerlerinden oluşan toplam 120 farklı kombinasyon denenmiştir. Bu sürece ait çıktılar oldukça büyük olduğundan, sonuçlara ulaşılmasında kullanılan kodlar EK1'de paylaşılmıştır. Bu kodlar çalıştırılarak ilgili sonuçlara detaylı bir şekilde ulaşılabilir. Şekil 28'de farklı kombinasyonlar sonucu elde edilen eğitim seti üzerindeki doğruluk

oranı ve doğrulama veri seti üzerindeki doğruluk oranları gösterilmiştir. Grafikte de dikkat çekildiği üzere, %83.6 ile en yüksek doğrulama seti doğruluk oranı veren kombinasyonda DO oranı "0.1", optimizasyon algoritması "adagrad", akitivasyon fonksiyonu "softplus" olduğu görülmektedir. Tüm bu bilgiler ışığında nihai modelin eğitilmesi aşamasına geçilmiştirs.




Şekil 28. GS Sonucunda Elde Edilen Farklı Kombinasyonların Doğruluk Oranları

Yukarıda bahsedilen deney sonuçları ışığında, DO oranı "0.1", optimizasyon algoritması "adagrad", akitivasyon fonksiyonu "softplus", iterasyon sayısı "31", parti büyüklüğü "20" olarak, veri setinin %20'si test seti, %80'i eğitim seti olarak belirlenmiş ve GAGS-LSTM ağı eğitilmiştir. Eğitim sırasında modelin ezberleme ihtimalini gözlemlemek için eğitim setinin %10'u doğrulama seti olarak kullanılmıştır. Şekil 29'da modelin her iterasyondaki eğitim seti üzerindeki ve doğrulama seti üzerindeki doğruluk oranları, her iterasyondaki eğitim seti ve doğrulama seti üzerindeki kayıp fonksiyonu değerleri gösterilmiştir.



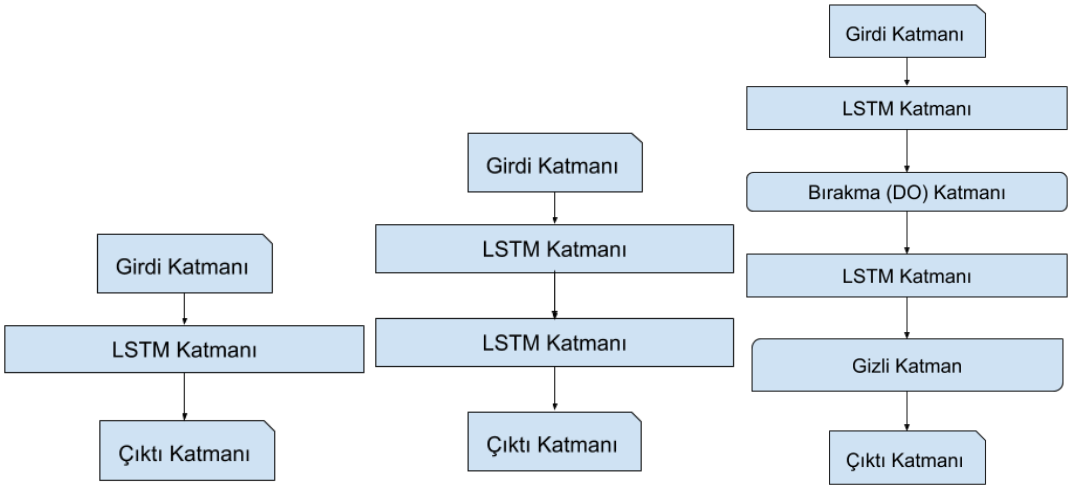

Şekil 29. GAGS-LSTM ağı için Doğruluk Oranları ve Kayıp Fonksiyonu değerleri

Test seti kullanılarak, modelin eğitilmesi sırasında kullanılmayan veri kullanılarak sınıfları (“motor sağlıklı”, “kritik durum”, “motoru bakıma al”, “motor arızalı”) tahmin etme performansını gözlemlemek için hata matrisi (confusion matrix) hesaplanmış ve Çizelge 5’te gösterilmiştir. Hata matrisine göre, “motor sağlıklı” %90, “kritik durum” %74, “motoru bakıma al” %65, “motor arızalı” %91 başarı ile tahmin edilmiştir. Buna göre, model hiç görmediği veri seti (test seti) üzerinde %80.3 doğruluk oranı ile çalışmıştır.

Çizelge 5. GAGS-LSTM Mimarisi İçin Hata Matrisi

| | | **Gerçek Sınıflar** | | | |
|---|---|---|---|---|---|
| | | **motor arızalı** | **motoru bakıma al** | **kritik durum** | **motor sağlıklı** |
| **Tahmin Edilen Sınıflar** | **motor arızalı** | **0.911859** | 0.220044 | 0.003663 | 0.001515 |
| | **motoru bakıma al** | 0.087068 | **0.653595** | 0.168498 | 0 |
| | **kritik durum** | 0 | 0.136166 | **0.742857** | 0.076515 |
| | **motor sağlıklı** | 0 | 0 | 0.093773 | **0.90303** |

Önerilen ağın performansını; girdi, tek bir LSTM katmanı ve çıktı katmanı içeren geleneksel LSTM ağı, girdi, iki adet LSTM katmanı ve çıktı katmanı içeren ağın performansı ile kıyaslamak icin bu üç ağ aynı veri seti uzerinde eğitilmiş, eğitim seti doğruluk oranı ve doğrulama seti doğruluk oranları arasındaki fark değerlendirilmiştir. Performans karşılaştırılmasına kolaylık sağlanması tek LSTM katmanından oluşan ağ LSTM1, iki LSTM katmanından oluşan ağ LSTM2 olarak adlandırılmıştır. Çalışma kapsamında önerilen ağ da Bölüm 6’da da belirtildiği şekilde GAGS-LSTM olarak adlandırılmıştır. Şekil 30’da LSTM1, LSTM2 ve GAGS-LSTM yapısında kullanılan mimariler karşılaştırılmalı olarak gösterilmiştir.




LSTM1 LSTM2 GAGS-LSTM

Şekil 30. LSTM1, LSTM2 ve GAGS-LSTM Mimarileri

Çizelge 6'da görüldüğü gibi ikinci LSTM katmanın eklenmesi eğitim seti doğruluk oranını %71'den %81'e, doğrulama seti doğruluk oranını %72'den %81'e çıkartmış ve gizli katman yardımıyla da eğitim seti doğruluk oranı %81'den %84'e, doğrulama seti doğruluk oranı %81'den %83'e yükselmiştir. Karşılaştırılmalı sonuçlara göre çalışma kapsamında önerilen mimarinin (GAGS-LSTM) en iyi sonuçları verdiği tespit edilmiştir.

Çizelge 6. Önerilen mimarinin diğer ağlarla karşılaştırması

| **Ağlar** | **Eğitim Seti Doğruluk Oranı** | **Doğrulama Seti Doğruluk Oranı** |
|---|---|---|
| LSTM | 0.7128 | 0.7242 |
| LSTM2 | 0.8051 | 0.8137 |
| GAGS-LSTM | **0.8449** | **0.8338** |

Mimarinin oluşturulması için tüm veri işleme aşamaları, ağın eğitilmesi, hiper parametre optimizasyonu aşamaları Google Cloud üzerinde oluşturulmuş olan 16 GB ram ve 1 adet NVIDIA K80 GPU'ya sahip sanal makine üzerinde gerçekleştirilmiştir. Yüksek hesaplama gücü gereksinimi olduğundan GPU kullanımı tercih edilmiştir. Pogramlama dili olarak, son yıllarda en çok kullanılan programlama dillerinden olan Python tercih edilmiştir. Python, aynı zamanda kullanıcılara bir çok farklı yazılım kütüphanesi sunmaktadır. Keras da bunlardan biridir. Keras, kapsamlı bir DL

kütüphanesidir. Bu sebeple ağların eğitilmesi Keras kütüphanesi kullanılarak gerçekleştirilmiştir.

Performansı daha da yukarı çekmek ve esnekliği sağlayabilmek için büyük veri ortamları da bu sanal makineler üzerinde çalışır hale getirilmiştir. Bu kapsamda, büyük veri ortamlarından son yıllarda adından sıkça bahsedilen Apache Spark kullanılmıştır. Keras kütüphanesini Apache Spark ile birlikte kullanmak için, bu amaç doğrultusunda geliştirilmiş olan bir diğer Python kütüphanesi elephas kullanılmıştır.

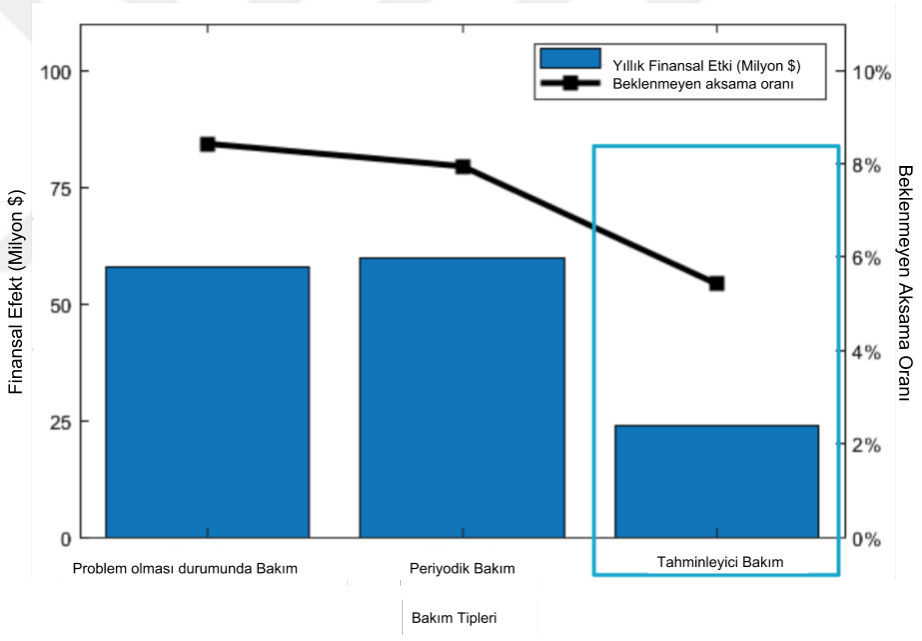
## 8. SONUÇ

Endüstri 4.0 en son endüstriyel devrim olarak adlandırılmaktadır. Ancak buharlı tren ve tekstil yerine bu kez, veri toplama ve yorumlama şeklini tamamen değiştiren AI teknikleri kullanılmaktadır. Makineler, artık daha makinenin durumu hakkında daha etkili kararlar alabilmek için sensörlerle donatılmaktadır. AI teknolojisinin bu alanda kullanılması, iş verimliliğinin ve iş güvenliğinin artmasına önemli katkı sağlamaktadır. Üretim alanında, IoT teknolojisi, PdM için çok önemli bir unsurdur. IoT teknolojisi sayesinde sensörler aracılığıyla ile sistemlerden ve makinelerden anlık veriler toplanabilmektedir. Bu verilerin analiz edilmesi ile oluşturulan tahmin modelleri sayesinde sistem üzerindeki, insan gözüyle tespit edilemeyen değişiklikler ve hatalar önceden tespit edilebilmektedir. Bir problemi gerçekleştikten sonra çözmek yerine, PdM sistemi vaktinden önce uyarılarak üreterek, bu problem ortaya çıkmadan gerekli önlemlerin alınmasına yardımcı olmaktadır.

Endüstriyel makinelerde ve sistemlerde bulunan parçalar zamanla bozulmakta ve bu sebeple bakım sürecinin düzenli bir şekilde yürütülmesi kritik önem taşımaktadır. Çünkü bakımların zamanında yapılmaması sistemdeki parçaların, hatta tüm sistemin çalışmasını engelleyebilir, yüksek onarım maliyetleri doğurabilir ve hatta üretim hattında ciddi anlamda gecikmelere sebebiyet verebilmektedir. Bu nedenle, hasarın ne zaman meydana geleceğini tahmin edebilmek ve gerekli önlemleri önceden almak maliyetleri büyük ölçüde azaltabilmektedir.

İyi bir şekilde planlanmamış bakım senaryoları, üretim tesisin verimliliğini ciddi anlamda düşürebilmektedir. Ayrıca, sistemlerde yaşanan teknik arızalardan dolayı ortaya çıkan üretim bandındaki aksama süreleri üreticilerin kaynaklarını efektif

kullanamamasına sebep olabilmektedir. Tahmini bakım, bir makinenin ne zaman kontrol edilmesi gerektiğini veya arızalanabileceğini öngörerek bu gibi durumların ortadan kaldırılmasına yardım edebilmektedir. Bir makinenin ya da bir sistemin ne zaman arızalanabileceğini önceden bilmek, bir makinenin ya da sistemin aksama süresini ortadan kaldırmaya yardımcı olmakla birlikte aynı zamanda cihazı kullanan çalışanların da güvenliğini arttırmaya yardımcı olmaktadır. Bir makine yalan söylemez veya içgüdüsel bir his duymaz, bunun yerine tahmine dayalı bakım kararları yalnızca verilere dayanmakta ve merkezi bir karar vericiye gereksinimi azaltmaktadır. General Electric tarafından yayınlanan raporda, Şekil 31'de de gösterildiği gibi PdM'nin, diğer bakım yöntemlerine göre sistemin beklenmeyen aksama süresini azalttığı ve buna bağlı olarak oluşabilecek finansal kaybı engellemeye önemli ölçüde katkı sağladığı belirtilmiştir.




Şekil 31. Farklı Bakım Yöntemlerinin Beklenen etkileri

Imaj Kaynak: GE, The Impact of Digital on Unplanned Downtime

Raporda ayrıca, PdM yönteminin, sistemin beklenmeyen sebeplerden ötürü aksama oranını %2'nin üzerinde azalttığı ve bu gibi durumlarda oluşabilecek 40 milyon dolarlık finansal kaybın önüne geçmeye yardımcı olduğu belirtilmektedir (GE, 2016).

General Electrics önderliğinde 2018 yılında yapılan bir başka çalışma, Avrupa ülkelerindeki üreticilerin PdM metodolojisine ne kadar hazır oldukları, bu amaç için firmaların ne gibi önlemler aldığını, PdM konusunda ne gibi yatırımlar yaptığı ile ilgili

çarpıcı sonuçları göz önüne sermektedir. Çalışma kapsamında Avrupa ülkelerinde üretim yapan sanayi devi firmaların üst düzey temsilcileri ile görüşülmüş ve PdM konusunda firmaların yaklaşımları incelenmiştir. Çarpıcı sonuçlardan ilki, çalışmaya katılan firmaların %93'ü hali hazırda sahip oldukları bakım süreçlerinin yetersiz ve çok etkili olmadığını ve bakım süreçlerinin geliştirilmesi gerektiğini belirtmiştir. Firmaların %55'i PdM süreçlerini uygulamaya başladığını ve bu firmaların %23'ü ise PdM süreçlerinin iş süreçlerine olumlu katkı sağladığını belirtmiştir (GE, 2018).

Bu bilgiler yardımıyla görülmüştür ki, endüstriyel makinelerin ya da sistemlerin mevcut durumu hakkında anlık bilgi veren karmaşık sensör verileri kullanılarak, ML ve DL bilgisine sahip olmayan birinin dahi anlayabileceği bir karar verme sürecinin DL ile büyük veri platformlarında geliştirilebilmesi oldukça öneme sahiptir. Çalışma kapsamında GAGS-LSTM isimli yeni bir DL mimarisi önerilmiştir. GAGS-LSTM, uçak motorunun üzerinde bulunan sensörlerden gelen verileri kullaranarak önce özellik mühendisliği yapmakta ardından her sensörden gelen veri kümesini kendi içinde normalize etmekte, GA ve GS ile mimaride kullanılan iterasyon sayısı, her iterasyonda kullanılacak olan parti büyüklüğü, DO oranı, kayıp fonksiyonunu optimize etmekte kullanılacak olan optimizasyon yöntemi, çıkış katmanın ardından kullanılacak olan aktivasyon fonksiyonu ile ilgili en iyi sonuçları veren kombinasyonları bulmaktadır. Bu kombinasyonlar optimum hiper parametreleri vermektedir. Bu optimmum hiper parametreler ile bir adet giriş, bir adet LSTM, bir adet DO, ardından bir adet daha LSTM, gizli katman ve çıkış katmanından oluşan DL ağı yardımıyla uçak motorunun anlık olarak durumu tahmin edilmektedir. Önerilen mimari, Python kullanılarak Keras, Elephas ile Apache Spark ortamında geliştirilmiş ve eğitim seti doğruluk oranı % 84.5, doğrulama veri seti doğruluk oranı %83.4 olarak elde edilmiştir. Önerilen modelin doğruluğu, uçak motorunun parçalarının bozulmadan önce bir uyarı üretmesini sağlayacak nitelikte olduğunu göstermektedir. Bu sistem yardımıyla, şirketler gelir ve hizmet kalitesini arttırırken bakım maliyetlerini düşürme şansına sahip olabilirler.

Bu çalışma, aynı zamanda GA ve GS yardımıyla kompleks mimarilerin kendi kendine öğrenen akıllı yapılara dönüştürülebileceğini de göstermektedir. GAGS-LSTM mimarisi bir çok farklı endüstriyel kurumda makinelerin, motorların üzerinde yer alan sensör verilerine, ML ya da DL'ye yönelik bilgi birikimi gerektirmeden adapte edilebilme özelliği de taşımaktadır.

## KAYNAKLAR


**Alpaydın, E.** (2010). *Introduction to Machine Learning. Second Edition.* Retrieved December 27, 2018, from https://kkpatel7.files.wordpress.com/2015/04/alppaydin_machinelearning_2010.pdf

**Aydin O., Guldamlasioglu S.** (2017) Using LSTM networks to predict engine condition on large scale data processing framework. *Electrical and Electronic Engineering (ICEEE), 2017 4th International Conference on. IEEE*

**Brownlee J.** (2016) Introduction to the Python Deep Learning LibraryTheano. Retrieved December 26, 2018, from https://machinelearningmastery.com/introduction-python-deep-learning-library-theano/.

**James Bergstra and Yoshua Bengio** (2012) Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(1):281–305.

**James S Bergstra, R´emi Bardenet, Yoshua Bengio, and Bal´azs K´egl.** ( 2011). Algorithms for hyperparameter optimization. In Advances in Neural Information Processing Systems, pages 2546–2554,

**Cachada, A., Barbosa, J., Leitño, P., Gcraldcs, C. A., Deusdado, L., Costa, J., Romero, L**. (2018). Maintenance 4.0: Intelligent and Predictive Maintenance System Architecture. In 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 139-146). IEEE.

**Chicco D** (2017). "Ten quick tips for machine learning in computational biology". BioData Mining. 10 (35): 35. doi:10.1186/s13040-017-0155-3. PMC 5721660. PMID 29234465.

**Christopher M Bishop et al** (1995) Neural networks for pattern recognition.

**Chandradevan R.** (2017) AutoEncoders are Essenital in Deep NeuralNet. Retrieved December 26, 2018, from https://towardsdatascience.com/autoencoders-are-essential-in-deep-neural-nets-f0365b2d1d7c.

**Chen, Zaifa, Yancheng Liu, Siyuan Liu.** (2017) Mechanical state prediction based on LSTM neural netwok. *Control Conference (CCC), 2017 36th Chinese. IEEE.*

**Chin-Wei H., Chih-Chung C., Chih-Jen L.** (2010). A practical guide to support vector classification. *Technical Report*, National Taiwan University.

**Claesen, M., & De Moor, B.** (2015). Hyperparameter search in machine learning. arXiv preprint arXiv:1502.02127.

**Cord M.** (2016) Deep CNN and Weak Supervision Learning for Visual Recognition. Retrieved December 26, 2018, from https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/.

**Das, A., Vishnu, A., Siegel, C., & Mueller, F.** (2017). Desh: Deep learning for hpc system health resilience. SC Poster Session.

**Goodfellow I., Bengio Y., Courville A.** (2016). *Deep Learning* (pp.326-327 & 367-415), Massachusetts Institute of Technology.

**Greff K., Srivastava R., Koutnik J., Steunebrink B. R., Schmidhuber J.** (2015) LSTM: A search space odyssey. Retrieved December 26, 2018, from arXiv preprint arXiv:1503.04069.

**General Electrics,** (2016) Oil & Gas, The Impact of Digital on Unplanned Downtime: AN OFFSHORE OIL AND GAS PERSPECTIVE

**General Electrics** (2018) Digital Industrial Revolution with Predictive Maintenance: Are European businesses ready to streamline their operations and reach higher levels of efficiency?

**Chung, Hyejung, and Kyung-shik Shin** (2018) . "Genetic algorithm-optimized long short-term memory network for stock market prediction." Sustainability 10.10 (2018): 3765.

**He K., Zhang X., Ren S., Sun J.** (2015) Deep Residual Learning for Image Recognition. *Proceedings of Computer Vision and Pattern Recognition*.

**Heimes, Felix O.** (2008) Recurrent neural networks for remaining useful life estimation. *Prognostics and Health Management, PHM 2008. International Conference on. IEEE*, 1-6, doi: 10.1109/PHM.2008.4711422.

**D. E. Goldberg and J. H. Holland** (1998) *Genetic algorithms and machine learning*, Machine Learning, vol. 3, no. 2-3, pp. 95–99.

**D. E. Goldberg** (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, NY, USA.

**Jahnke, P.** (2015) *Machine Learning Approaches for Failure Type Detection and Predictive Maintenance* (Yüksek lisans Tezi).

**Jain A., Kundu P., Lad B. K.** (2014) Prediction of remaining useful life of an aircraft engine under unknown initial wear minutes. *5th International & 26th All India Manufacturing Technology, Design and Research Conference (AIMTDR 2014.* 494:1- 494:5, doi: 10.1007/978-81-322-2352-8.

**Jardine A., Lin D., Banjevic D.** (2006) A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing* (20), 1483-1510, doi: 10.1016/j.ymssp.2005.09.012.

**Jia Y., Shelhamer E., Donahue J., Karayev S., Long J., Girshick R., Guadarrama S., Darrell T.** (2014) Caffe: Convolutional Architecture for Fast Feature Embedding. Retrieved December 26, 2018, from https://arxiv.org/abs/1408.5093.

**J. R. Koza** (1992) Genetic Programming: on the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, Mass, USA.

**Chollet, François.** (2018), "Keras: The python deep learning library." Astrophysics Source Code Library (2018).

**Kim, K.G.** (2016). "Deep Learning", Book Review, *Healthcare InformaticsResearch*, *22*(4), 351-354. Retrieved December 26, 2018, form https://doi.org/10.4258/hir.2016.22.4.351.

**Krizhevsky A., Sutskever I., Hinton G. E.** (2012) Imagenet Classification With Deep Convolutional Neural Networks, *NIPS'12 Proc. of the 25th Inter. Conf. on Neural Infor. Proc. Sys.* vol 1,1097-1105.

**LeCun Y., Boser B., Denker J.S., Henderson D., Howard R. E., Hubbard W., Jackel L. D.** (1989) Backpropagation Applied to Handwritten Zip Code Recognition, *AT&T Bell Lab, Neural Computation* (1) 541-551.

**LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P.** (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

**Mahamad A. K. , Saon S., Hiyama T.** (2015) Predicting remaining useful life of rotating machinery based artificial neural network. *Computers & Mathematics with Applications*(60), 1078-1087.

**Marr B.** (2017). *The Amazing Ways Google Uses DeepLearningAI*, Retrieved December 26, 2018, from https://www.forbes.com/sites/bernardmarr/2017/08/08/the-amazing-ways-how-google-uses-deep-learning-ai/#6c13240b3204

**Marc Claesen, Frank De Smet, Johan A.K. Suykens, and Bart De Moor** (2014), EnsembleSVM: A library for ensemble learning using support vector machines. Journal of Machine Learning Research, 15:141–145.

**Öztemel, E.** (2012). *Yapay sinir ağlari.* Retrieved December 26, 2018, from http://papatyabilim.com.tr/PDF/yapay_sinir_aglari.pdf

**Öztanır, O.** (2018). Makine Öğrenmesi Kullanılarak Kestirimci Bakım (Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü).

**Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al** .(2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

**Porotsky S.** (2012) Remaining useful life estimation for systems with non-trendability behaviour. *Prognostics and Health Management (PHM).*

**Predictive maintenance** (t. y.) https://www.seebo.com/predictive-maintenance/

**Radford Metz, L., Chintala S.** (2016) Unsupervised representation learning with deep convolutional generative adversarial networks, *In ICLR*.

**Roweis S., Saul L., Hinton G.** (2002) Global coordination of local linear models, *In NIPS.*

**Russell S. J. & Norvig P.** (2010) *Artificial Intelligence: A Modern Approach, Third Edition*, Prentice Hall.

**Bouktif, Salah, et al**.(2018) "Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches." Energies 11.71636.

**Santos L., Fernando D.** (2017) et al. Predicting failures in hard drives with lstm networks. *2017 Brazilian Conference on Intelligent Systems (BRACIS). IEEE.*

**Sateesh Babu G., Xiao-Li Li , Suresh S.** (2016) Meta-cognitive Regression Neural Network for function approximation: Application to Remaining Useful Life estimation. *Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE*, 4803 – 4810, doi: 10.1109/IJCNN.2016.772783.1

**Saxena Abhinav, Simon Don** (2008) Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation, 2008 INTERNATIONAL CONFERENCE ON PROGNOSTICS AND HEALTH MANAGEMENT

**Simonyan K. & Zisserman A.** (2015) Very deep convolutional networks for large-scale image recognition. Retrieved December 26, 2018, from https://arxiv.org/pdf/1409.1556.pdf.

**Singleton, R. K., Strangas, E. G., & Aviyente, S.** (2013). Time-frequency complexity based remaining useful life (RUL) estimation for bearing faults. In 2013 9th IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives (SDEMPED) (pp. 600-606). IEEE.

**Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.** (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929-1958.

**Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D.,. Vanhoucke V, Rabinovich A.** (2015) Going Deeper with Convolutions, *CVPR2015*.

**TensorflowArchitecture.** (t. y.) https://www.tensorflow.org/extend/architecture

**Torch** (t. y.), web tutorial, http://torch.ch/

**Waibel, Hanazawa T., Hinton G., Shikano K., Lang K.J.** (1989) Phoneme Recognition Using Time-Delay Neural Networks. *IEE Transactions on Acoustics, Speech and Processing* (37-3).

**Wicaksono, A. S., & Supianto, A. A.** (2018). Hyper Parameter Optimization using Genetic Algorithm on Machine Learning Methods for Online News Popularity Prediction. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 9(12), 263-267.

**Warden P.** (2014). *What is deep learning, and why should you care?*, Retrieved December 27, 2018, from https://www.oreilly.com/ideas/what-is-deep-learning.

**Zaharia M., Chowdhury M., Das T., Dave A., Ma J., Mccauley M., Franklin M., Shenker S., Stoica I.** (2012) Fast and interactive analytics over Hadoop data with Spark, *login,* (37), 45-51.

**Zhang, W., Guo, W., Liu, X., Liu, Y., Zhou, J., Li, B., Yang, S.** (2018). Lstm-based analysis of industrial iot equipment. IEEE Access, 6, 23551-23560.

**Zhou Q., Son J., Zhou S., Mao X., Salman M.** (2016) Remaining "Useful life prediction of individual units subject to hard failure", *IIE Transactions* (46), 1017-1030. doi: 10.1080/0740817X.2013.876126.

# EKLER

## EK A. GA İLE HİPERPARAMETRE OPTİMİZASYONUNDA KULLANILAN PYTHON KODU

```
def train_evaluate(ga_individual_solution):
    epoch = BitArray(ga_individual_solution[0:5])
    batch = BitArray(ga_individual_solution[5:])
    epoch_s = epoch.uint
    batch_s = batch.uint
    print('\nEpoch Size: ', epoch_s, ', Batch Size: ', batch_s)
    if epoch_s == 0 or epoch_s == 0:
        return 10,
    model = Sequential()
    model.add(LSTM(218, input_dim=inputs, init='normal',return_sequences=True))
    Dropout(0.1)
    model.add(LSTM(218, input_dim=inputs, init='normal'))
    model.add(Dense(4,init='normal'))
    model.add(Dense(4,activation='softplus'))

model.compile(loss='categorical_crossentropy',optimizer='adadelta',metrics=['accuracy'])
history = model.fit(trainX, categorical_labels_train, nb_epoch=epoch_s, batch_size=batch_s,validation_data=(testX_s, categorical_labels_test))
loss_last = history.history[loss][-1]
return loss_last,
population_size = 5
num_generations = 5
```

```
gene_length = 10
creator.create('FitnessMax', base.Fitness, weights = (-1.0,))
creator.create('Individual', list , fitness = creator.FitnessMax)
toolbox = base.Toolbox()
toolbox.register('binary', bernoulli.rvs, 0.5)
toolbox.register('individual', tools.initRepeat, creator.Individual, toolbox.binary, n = gene_length)
toolbox.register('population', tools.initRepeat, list , toolbox.individual)
toolbox.register('mate', tools.cxOrdered)
toolbox.register('mutate', tools.mutShuffleIndexes, indpb = 0.6)
toolbox.register('select', tools.selRoulette)
toolbox.register('evaluate', train_evaluate)
population = toolbox.population(n = population_size)
r = algorithms.eaSimple(population, toolbox, cxpb = 0.4, mutpb = 0.1, ngen = num_generations, verbose = False)
```

## EK B. GS İLE HİPERPARAMETRE OPTİMİZASYONUNDA KULLANILAN PYTHON KODU

```
a_last = list()
do_r = [0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9]
optimizers = ['adadelta','adagrad','sgd','adam','adamax']
activations = ['softmax','softplus','sigmoid']
for dor in do_r:
    for opt in optimizers:
        for act in activations:
            print('\nDo rate: ', dor, ', optimizer: ', opt, ', activation: ',act)
            model = Sequential()

model.add(LSTM(218,input_dim=inputs,init='normal',return_sequences=True))
            Dropout(dor)
            model.add(LSTM(218, input_dim=inputs, init='normal'))
            model.add(Dense(4,init='normal'))
             model.add(Dense(4,activation=act))
            model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accur
            acy'])
            history = model.fit(trainX, categorical_labels_train, nb_epoch=31,
            batch_size=20,validation_data=(testX_s, categorical_labels_test))#,
            verbose=1)
            accuracy_last = history.history['acc'][-1]
            print('\nAccuracy Last: ', accuracy_last)
            a_last.append(accuracy_last)
```

## EK C. GAGS-LSTM MODEL EĞİTİMİNİN SONUÇLARININ GÖRSELLEŞTİRİLMESİNDE KULLANILAN PYTHON KODU

```
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Dogruluk Oranlari')
plt.axis([0, 31,0.81, 0.85])
plt.ylabel('Dogruluk Orani')
plt.xlabel('Iterasyon')
plt.legend(['Egitim Seti', 'Dogrulama Seti'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Kayip Fonksiyonu Degerleri')
plt.ylabel('Kayip Fonksiyonu Degerleri')
plt.xlabel('Iterasyon')
plt.legend(['Egitim Seti', 'Dogrulama Seti'], loc='upper left')
plt.show()
```

**ÖZGEÇMİŞ**

OLGUN AYDIN – Gdansk,Polonya

Telefon: +90 (507) 500 95 20 - +48 (796) 121 565

e-mail: olgunaydinn@gmail.com, info@olgunaydin.com

web: olgunaydin.com

github: github.com/olgnaydn

linkedin: linkedin.com/in/olgun-aydin

**Eğitim:**

Doktora, İstatistik Bölümü, Mimar Sinan Universitesi (2014-2019)

Yüksek Lisans, İstatistik Bölümü, Ege Universitesi (2010-2013)

Lisans, İstatistik Bölümü, Ege Universitesi (2006-2010)

**Bilgisayar becerileri:**

MATLAB, C++, SPSS, R, Tableau, Minitab, LISREL, VBA, PostgreSQL, PL/R, Hive, Impala, SparkR, Penthao Report Designer, Facebook API, Instagram API, Twitter API, Foursquare API, Python, Google Analytics, Google BigQuery. Shiny.

**İş Deneyimi:**

• PwC, Poland (2019 Ocak- ) Team Lead, Data Science

• Epam Systems, Poland (2017 Ekim-2018 Aralık) Senior Data Scientist

• STM Savunma Teknolojileri Mühendislik AŞ.(2016 Ekim- 2017 Ağustos) Data Scientist

• Zingat.com- (2016 Mayıs – 2016 Ekim) Lead Analyst

• REIDIN – (2014 Şubat- 2016 Ocak) Senior Data Analyst

• Freelancer (2010 Haziran-2014 Ocak) Data Analyst

**Yayınlar (2016-2018):**

• XVth Spanish Biometric Conference and Vth Ibero American Biometric Meeting, 24 Sept. 2015, Bilbao, SPAIN

The R GUI: Best Body Size Estimator for Optimum Total Cholestrol Level and Total Cholestrol Level Estimator by Age Groups, Aydin Olgun – Oral Presentation

• International 9th Statistics Congress, 31 Oct. 2015, Antalya, TURKEY

Modelling Of Residential Sales Price With Kriging Using Different Distance Metrics In Different Correlation Functions, Tasabat Semra, Aydin Olgun– Oral Presentation

• International Conference on Information Complexity and Statistical Modeling in High Dimensions with Applications (IC-SMHD-2016), 20 May 2016, Capadocia, TURKEY

Missing Value Imputation System For Property Listings Attributes, Tasabat Semra, Aydin Olgun– Oral Presentation

• International Conference on Information Complexity and Statistical Modeling in High Dimensions with Applications (IC-SMHD-2016), 19 May 2016, Capadocia, TURKEY

Comparison of different approaches on estimation of total cholesterol level, Aydin Olgun, Tasabat Semra, Caamaño Guler Ipek , Suarez Cadarso Carmen. – Oral Presentation

• The 17th Annual Conference of Faculty of Finance and Accounting (ACFA), 27 May 2016, Prague, CZECH REPUBLIC

Count Data Modelling about Relationship between Dubai Housing Sales Transactions and Financial Indicators, Aydin Olgun, Hayat Elvan, Hepsen Ali– Oral Presentation

• Design of dimensionally stable composites using efficient global optimization method, Levent Aydin Levent, Aydin Olgun, Artem H. Secil, Mert Ali Journal of Materials: Design and Applications, doi:10.1177/1464420716664921 – SCI Expanded.

• Modelling Of Residential Sales Price With Kriging Using Different Distance Metrics In Different Correlation Functions, Tasabat Semra, Aydin Olgun, Gazi University Journal of Science – SCOPUS.

• EUREFE'16, 15 July 2016, Aydin, TURKEY

Measuring the Effects of Real Estate Market on Unemployment in Turkey, Aydin Olgun, Hayat Elvan, Hepsen Ali– Oral Presentation

• Xth International Statistics Days Conference,Giresun, TURKEY

Image Processing for Label Detection With R, 2016, Aydin Olgun, Tasabat Erpoat Semra– Oral Presentation

• Xth International Statistics Days Conference,Giresun, TURKEY

Estimation Dubai Property Prices by Kriging, 2016, Aydin Olgun, Hayat Akturk Elvan, Isikdag Umit – Oral Presentation.

• European R Users Meeting 2016, 13 October 2016, Poznan, POLAND,

Dynamic Inflation Rate Calculation of Fast-Moving Consumer Goods: Shiny-SparkR App, Aydin Olgun– Oral Presentation.

• Count Data Modeling About Relationship Between Housing Sales Transactions and Financial Indicators, Aydin Olgun, Hayat Akturk Elvan, Hepsen Ali, New Trends in Finance and Accounting, Springer, 2017, Hardcover ISBN 978-3-319-49558-3.

• Aydin, Olgun, and Seren Guldamlasioglu. "Using LSTM networks to predict engine condition on large scale data processing framework." In 2017 4th International Conference on Electrical and Electronic Engineering (ICEEE). IEEE, 2017.

• Working with PostgreSQL in R, Olgun Aydın, Software Developer's Journal, 2017 Vol.1, R Programming.

• Clustering in R, Olgun Aydın, Software Developer's Journal, 2017 Vol.3, R Programming.

• Efficient Use of Capital: Paradox of Real Estate and Industry in Turkey, Ali Hepsen, Mehmet Asici, Olgun Aydin, International Journal of Economics and Finance, 2017, DOI: https://doi.org/10.5539/ijef.v9n8p221

• Real Estate Investment Trusts in Turkey: Structure, Analysis, and Strategy, Ali Hepsen, Murat Goksin Berberoglu, Olgun Aydın, Journal of Business, Economics and Finance, Vol.6, No.2, 2017, pp.191-199

• Estimation of Housing Demand with Adaptive Neuro-Fuzzy Inference Systems (ANFIS), Olgun Aydin, Elvan Aktürk Hayat, The Impact of Globalization on International Finance and Accounting, 2018, Springer, Hardcover ISBN 978-3-319-68761-2