T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARININ ÇİZGE VERİTABANLARI ÜZERİNDE GERÇEKLEŞTİRİMİ

Doğa Barış ÖZDEMİR

Bilgisayar Mühendisliği Anabilim Dalı

ÇANAKKALE

T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARININ ÇİZGE VERİTABANLARI ÜZERİNDE GERÇEKLEŞTİRİMİ

Doğa Barış ÖZDEMİR

Bilgisayar Mühendisliği Anabilim Dalı

Tezin Sunulduğu Tarih: **3/3/2021**

Tez Danışmanı:

Dr. Öğr. Üyesi Ahmet Cumhur KINACI

ÇANAKKALE

Doğa Barış ÖZDEMİR tarafından Dr. Öğr. Üyesi Ahmet Cumhur KINACI yönetiminde hazırlanan ve **3/3/2021** tarihinde aşağıdaki juri karşısında sunulan **" Yapay Sinir Ağlarının Çizge Veritabanları Üzerinde Gerçekleştirimi "** başlıklı çalışma, Çanakkale Onsekiz Mart Üniversitesi Lisansüstü Eğitim Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak oy birliği ile kabul edilmiştir.

JÜRİ

Dr. Öğr. Üyesi Ahmet Cumhur KINACI .....
Başkan

Doç. Dr. Suhap ŞAHİN .....
Üye

Dr. Öğr. Üyesi Ali Murat TİRYAKİ .....
Üye

Prof. Dr. Pelin KANTEN
Müdür
Lisansüstü Eğitim Enstitüsü

Sıra No:.....

İNTİHAL (AŞIRMA) BEYAN SAYFASI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belirttiğimi beyan ederim.

Doğa Barış ÖZDEMİR

TEŞEKKÜR

Bu tezin gerçekleştirilmesinde, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen saygı değer danışman hocam Dr. Öğr. Üyesi Ahmet Cumhur KINACI, çalışma süresince tüm zorlukları benimle göğüsleyen Selin HANEY'e ve hayatımın her evresinde bana destek olan değerli aileme sonsuz teşekkürlerimi sunarım.

Doğa Barış ÖZDEMİR

Çanakkale, Mart 2021

SİMGELER VE KISALTMALAR

YSA	Yapay sinir ağı (Artificial Neural Network)
ESA	Evrişimli sinir ağı (Convolutional Neural Network)
TSA	Tekrarlayan sinir ağı (Recurrent Neural Network)
ZGY	Zamanla geri yayılım (Backpropagation Through Time)
SVG	Ölçeklenebilir vektör grafikleri (Scalable Vector Graphics)
GTBSA	Geçitli tekrarlayan birimli sinir ağlı (Gated Recurrent Neural Network)
%	Yüzde oranı
RDBMS	İlişkisel veri tabanı sistemi (Relational Database Management System)
SQL	Yapısal sorgu dili (Structured Query Language)
NoSQL	Yapısal olmayan sorgu dili (Not Structrured Query Language)
Cypher	Neo4j sorgu dili
JSON	Javascript nesne gösterimi (Javascript Object Notation)
XOR	Dışlamalı ya da (Exclusive or)
AND	Ve mantık kapısı
H5	Hiyerarşik veri formatı
ReLu	Düzeltilmiş doğrusal birim (Rectified Linear Unti)
OMH	Ortalama mutlak hata
OHK	Ortalama hata karesi
XaaS	Hizmet olarak her şey (Everything As A Service)

ÖZET

YAPAY SİNİR AĞLARININ ÇİZGE VERİTABANLARI ÜZERİNDE GERÇEKLEŞTİRİMİ

Doğa Barış ÖZDEMİR
Çanakkale Onsekiz Mart Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi
Danışman: Dr. Öğr. Üyesi Ahmet Cumhur KINACI
3/3/2021, 70

Yapay sinir ağlarının gelişimi ve yaygınlaşan kullanımı nedeniyle kullanıcıların daha kolay yönetilebilir süreçlere ihtiyacı ortaya çıkmıştır. Bu ihtiyaçlar eğitilmiş modellerin paylaşılması, aktarılması ve tekrar kullanılması olarak özetlenebilir. Ayrıca modellerin görsel olarak düzenlenebilmesi ve izlenebilmesi gereksinimi de ortaya çıkmıştır. Çalışmada bu gereksinimlerin karşılanabilmesi için bir yazılım sistemi oluşturulmuştur. Önerilen sistemin temelinde çizge veritabanlarının kullanılması tercih edilmiştir. Yapay sinir ağları teknik olarak çizgeler ile ifade edilebilmektedir. Yapay sinir ağ mimarilerinin bu tarz bir veritabanına aktarılması ve işletilmesi daha kolay olmaktadır. Çalışmada yapay sinir ağ modelleri üzerinde birden çok araştırmacının iş birlikçi çalışabileceği bir yazılım geliştirilmiştir. Modellerin saklanması ile eğitim ve test aşamalarının görselleştirilmesi sağlanmıştır. Önerilen sistemde modellere versiyonlanma yeteneği kazandırılmıştır. Yaygın kullanılan YSA kütüphanelerinin ortak olarak kullandığı model saklama biçimi olan H5 dosyalarının çizge veri tabanına aktarılması için yöntem oluşturulmuştur. Modeldeki girdiler ve çıktılar dahil tüm veriler çizgede tutulabilmektedir. Bu sayede model ile verinin aynı ortamda tutulması sağlanmıştır. Yapay sinir ağlarının hesaplama ihtiyaçları çoğunlukla çizge veri tabanının sorgulama dili kullanılarak gerçekleştirilmiştir. Bu sayede veritabanı dışında bağımlılıkların azaltılması hedeflenmiştir. Sorgu dilinin yetersiz olduğu noktalarda ise arka yüz sunucularında bu işlemler tamamlanmıştır.

Anahtar sözcükler: Çizge veri tabanları, yapay sinir ağları, görselleştirme, veri temsili

ABSTRACT

IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORKS ON GRAPH DATABASES

Doğa Barış ÖZDEMİR
Çanakkale Onsekiz Mart University
School of Graduate Studies
Master of Science Thesis in Computer Engineering
Advisor: Asst. Prof. Dr. Ahmet Cumhur KINACI
3/3/2021, 70

Due to the development and widespread use of artificial neural networks, users need more manageable processes. These needs can be summarized as sharing, transferring and reusing trained models. In addition, the need to be able to visually organize and monitor models has emerged. In the study, a software system was created to meet these requirements. It was preferred to use graph databases on the basis of the proposed system. Artificial neural networks can be technically expressed with graphs. It is easier to transfer and operate artificial neural network architectures to such a database. In the study, a software has been developed for more than one researcher to work collaboratively on artificial neural network models. By storing the models, training and testing phases were visualized. In the proposed system, the models are equipped with the ability to be versioned. A method was developed to transfer H5 files, which is the model storage format commonly used by the widely used ANN libraries, to the graph database. All data, including inputs and outputs in the model, can be kept on the graph. In this way, it is ensured that the model and data are kept in the same environment. The computational needs of artificial neural networks are mostly realized by using the query language of the graph database. In this way, it is aimed to reduce dependencies outside the database. In cases where the query language is insufficient, these operations have been completed on the backend servers.

Keywords: Graph databases, artificial neural networks, visualization, data representation

İÇİNDEKİLER

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

BÖLÜM 1
GİRİŞ

Yapay sinir ağları (YSA) yaygın olarak bir çok problemin çözümünde başarıyla kullanılan bir makine öğrenmesi yöntemidir. Son on yılda derin öğrenme yaklaşımlarıyla çok daha fazla katmana sahip ve çok daha fazla veri ile eğitilen ağ yapıları ortaya çıkmıştır. Bu ağlar ne kadar fazla katmana sahipse ve ne kadar fazla veriyle eğitilecekse o oranda hesaplama gereksinimine ihtiyaç duyarlar. Bir YSA modeli oluşturulduktan sonra bu modelin paylaşılması ve saklanması amacıyla dosyalara kaydedilmesi genel kullanım senaryosudur. Özellikle yüksek hesaplama maliyetiyle oluşturulmuş modellerin saklanması ve tekrar kullanılabilmesi oldukça önemlidir. Bir YSA modelinin paylaşılması ve tekrar kullanılması senaryolarında kabaca şu adımların tekrarlanması gerekir; dosyanın aktarılması, dosyanın kod seviyesinde okunması, okunan dosyadan modelin çıktı üretilebilecek yapıda belleğe aktarılması, yeni girdilerin modele verilip çıktıların alınması ve üretilen çıktıların saklanması. Bu adımların gerçekleştirilmesi için teknik yoğun bir çalışma yapılması ihtiyacı vardır. Ayrıca oluşturulmuş YSA modelinin yeni versiyonlarının oluşturulması durumunda versiyonların yönetilmesi de ayrıca bir problem olarak ortaya çıkabilmektedir.

Yukarıda bahsedilen durumların oluşmasının temel olarak nedeninin model ve verinin farklı ortamlarda saklanması ve bu ikisinin buluşturulmasının ise çoğunlukla yazılım geliştirme faliyetleri ile gerçekleştirilmesine bağlayabiliriz. Bu tez çalışmasında YSA modelleri ile verilerin aynı ortamda olması sağlanarak modellerin veri ile doğrudan beslenebilmesi ve çıktıların oluşturulmasını sağlayacak bir çözüm önerilmiştir. Bu çözüm ayrıca modellerin versiyonlanması problemine de çözüm getirmektedir.

Literatürde yapay sinir ağlarının saklanması için ilk olarak nesne yönelimli yaklaşımlar ile veritabanlarının kullanılması ön plana çıkmaktadır. İlk yapılan çalışmalar ile yapay sinir ağları ilişkisel veritabanlarında tutularak veritabanı sorgularıyla işletilmesi önerilmiştir (Schikuta ve diğerleri, 1996). İlişkisel veritabanlarında yapı değiştirildiğinde saklanan tüm YSA modellerininin yapısının etkilenmesi, farklı YSA mimarilerinin veritabanında tutulmasında zorluk yaratabilmektedir. İlişkisel olmayan veritabanlarında YSA modelleri saklanarak ölçeklenebilir ve genelleştirilebilir bir saklama yöntemi oluşturulabileceğinin üzerinde durulmaktadır (Schikuta ve Mann, 2013). Ayrıca model oluşturabilmesi için görsel arayüz gerekliliği ortaya çıkmıştır.

Çözüm olarak sunulan sistemde veriler ile YSA modellerinin aynı veri tabanında

saklanması hedeflenmiştir. Aynı veritabanı içerisinde modeller çıktı üretmek için yine veritabanındaki verileri kullanabilecektir. Modellerin çıktı üretmesi için gerekli olan hesaplamalar veritabanı sisteminin desteklediği temel matematik operasyonlarının kullanılmasıyla gerçekleştirilmeye çalışılmıştır.

Tez kapsamında önerilen çözüm yaklaşımının uygulanması için veritabanı sistemi olarak bir çizge veritabanı olan Neo4j[1] kullanılmıştır. Çizge veritabanlarının tercih edilmesindeki önemli nedenlerden biri YSA modellerinin de çizge yapısında olmasıdır. Bu sayede aktarım ve kullanım için çizge mantığının dışına çıkmaya gerek kalmamaktadır. Aktarılan modellerin veritabanında eğitilmesi konusu bu tez kapsamında ele alınmamıştır. Ayrıca YSA model süreçlerinde yüksek performanslı sonuçlar elde etmek bu çalışma kapsamındaki hedeflerden biri olmamıştır.

Önerilen çözüm yaklaşımının gerçekleştirimi için model aktarımı ve işletimini sağlayan bir uygulama arayüzü geliştirilmiştir. Önerilen uygulama ile YSA modelleri kodlama yapmaya ihtiyaç duymadan görsel olarak gözlemlenebilmektedir. Araştırmacıların model gözlem ve tahminleme süreçlerini beraber yürütebilmesi sağlanmıştır. Çizge veritabanlarının verileri yapı bağımsız tutması sayesinde genelleştirilebilir bir YSA modeli saklama ortamı sağlanmıştır. Öğrenim aktarımı yapılan modellerin görsel olarak düzenlemesi yapılabilmektedir. Tahminleme sürecinin performansı hedeflerden biri olmasa da yeterli hızda tahminleme sonucu elde edildiği gözlemlenmiştir.

İkinci bölümde yapay sinir ağları ve hesaplamaları, çizge teorisi ve çizge veritabanları, yaygın kullanılan yapay sinir ağı yazılımları irdelenmiştir. Çizgelerin model temsiline katkıları ve veritabanında yapay sinir ağlarının saklanmasıyla ilgili çalışmalar incelenmiştir. Üçüncü bölümde çalışmada geliştirilen yazılım ile YSA süreçlerinin çizge veritabanlarında gerçeklenmesi için izlenen yöntemlerden bahsedilmiştir. Yaygın kullanılan modeller önerilen yöntem ile gerçeklenerek öne sürülen çizge veritabanlı yapay sinir ağları kavramı desteklenmiştir. Dördüncü bölümde çizge veritabanlarında sinir ağlarının gerçeklenirken dikkat edilmesi gereken durumlar ve sınırlar, yaygın kütüphaneler ile karşılaştırmalar, önceki literatür çalışmalarında önerilen veritabanında model saklama yöntemleri karşısında avantajları tartışılmıştır. Beşinci bölümde çalışma ile elde edilen sonuçlar ve öneriler sunulmuştur.

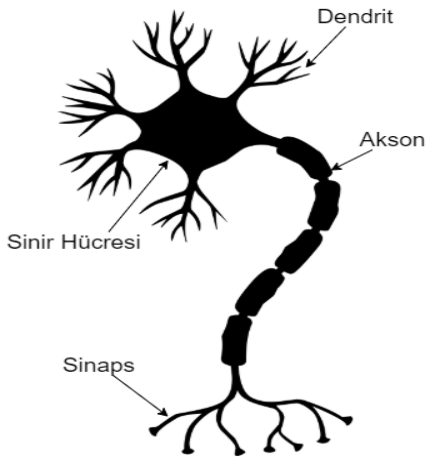
[1] Neo4j Graph Platform, http://www.neo4j.org

BÖLÜM 2
ÖNCEKİ ÇALIŞMALAR

Bu bölümde yapay sinir ağ modelleri ve hesaplama fonksiyonları incelenmiştir. Çizge teorisinin çıkışı, çizge veritabanlarının veritabanları arasındaki yeri anlatılmıştır. Literatürdeki YSA saklama ve işleme üzerine yapılan çalışmalar ve örnek gösterilen problemlere literatürdeki çözüm önerileri irdelenmiştir.

2.1. Yapay Sinir Ağları

İnsan sinir sisteminde veri akışının sağlanmasını ve işlenmesini sağlamak üzere sinir hücreleri yani nöronlar, sinir hücrelerindeki bilgi girişlerinin toplanmasına yarayan dendritler, sinir hücrelerinden çıktıları elde eden ve taşımayı gerçekleştiren aksonlar, sinir hücreleri arasındaki bağları kuran ve ağırlıklarına göre iletim aktivasyonunu gerçekleştiren sinapsların oluşturduğu yapılar yer almaktadır. Şekil 1'de biyolojik sinir hücresinin elemanları gösterilmiştir. Yapay sinir ağları yapay sinir nöronlarından ve birbirleri arasındaki bağlardan oluşmaktadır. Yapay sinir ağları nöronlarındaki ağırlık değerlerini kullanarak ve gerektiğinde değiştirerek öğrenme gerçekleştirmeyi amaçlar.



Şekil 1. Biyolojik sinir hücre elemanları

Günümüzde sınıflandırma, kümeleme ve tahminleme problemlerinin çözümünde etkin olarak yapay sinir ağları kullanılmaktadır. Yapay sinir ağları insan beyninin işleyişini bilgisayar ortamında gerçeklemeye dayanmaktadır. Yapay sinir ağları çok çeşitli çalışma alanlarında başarıyla uygulanan bir teknolojidir. Yapay sinir ağlarının geliştirilmesiyle beynin bilgisayar ortamında temsili sağlanarak insanın yetenekleri olan öğrenme, taklit

etme, problem çözme yetilerini makinelerin de kazanması amaçlanmıştır. Yapay sinir ağları biyolojik sinir yapısının işlevlerinden ve yapısından ilham alan matematiksel modellerdir.

Yapay sinir hücresi girdi, ağırlık, toplam fonksiyonu, aktivasyon fonksiyonu, çıktı olmak üzere beş bölüm içermektedir. Girdi yapay sinir hücresini besleyen veri bölümüdür. Veri girdisi olduğu gibi bir önceki katmanın çıktısı da olabilmektedir.

Ağırlıklar hücrenin girdi değerlerinin etkisini belirleyen değerlerdir. Sinir hücresine birden çok girdi arasında hangi girdilerin daha önemli olduğu, model öğrenimi sırasında atanan ve değiştirilen ağırlık değerlerinin tayiniyle belirlenir. Bir girdi daha büyük ağırlığa sahip olduğunda bir önceki hücreyle bağlantının önemi artmaktadır. Ağırlıkların hangi sırada ve örüntüde değiştirileceği yapay sinir ağı modelinin türüne göre değişiklik göstermektedir.

Toplam fonksiyonu, hücreye olan girdilerin kendi ağırlıklarla çarpılması her bir girdi için elde edilen değerlerinin toplanması ile sapma değeri olan bias değerinin bu toplama eklenmesi işleminin yapıldığı fonksiyondur. Sapma değeri olan bias değeri sinir hücresi çıktılarında daha uygun sonuçlar elde etmek için ayarlanan bir değerdir. Toplam fonksiyonu bir veri dizisi değil, sayı çıktısı oluşturmaktadır. Girdiler $g_1, g_2, .., g_n$, ağırlıklar $a_1, a_2, .., a_n$, çıktılar $c_1, c_2, .., c_n$ olarak isimlendirildiğinde toplam fonksiyonu Denklem 2.1'de gösterilmektedir.

$$\sum_{i=1}^{n} g_i.a_i \tag{2.1}$$

Çıktı sinir hücresinin gerçekleştirdiği hesaplamaların sonucu olarak dışarıya verdiği değer veya değerlerdir. Çıktı bulunduğu katmana göre başka bir hücrenin girdisi ya da genel olarak YSA'nın çıktısı olabilmektedir. Genel olarak YSA girdiden veri alır, ağırlıklarla çarpılır, çarpımlar toplanır, aktivasyon fonksiyonu çalıştırılır, ve çıktının üretimi sağlanabilir. Bias değerinin eklenip eklenmemesi araştırmacıya ve modele bağlı değişkenlik gösterir.

Aktivasyon fonksiyonu toplam fonksiyonundan sonra çalışarak ağın yapısına bağlı olarak modeli geliştirenler tarafından hücrede işlenecek verilere, çözeceği probleme, öğrenme algoritmasına göre seçilmektedir. Aktivasyon fonksiyonları sinir ağlarının doğrusal çalışmasını engellemektedir. Öğrenmenin gerçekleşmesi için zorunlu olarak kullanılması gerekmektedir. Sinir ağlarında katman sayıları arttırılarak doğrusallığın azaltımı sağlanmaktadır. Sinir ağında katman sayısı arttıkça problem çözümünde daha az doğrusallık ve daha doğru öğrenim gerçekleştirilebilmekte buna karşın işlem maliyeti artmaktadır. Aktivasyon fonksiyonunun çıktıları pozitif sayılar olabileceği gibi negatif sayılar da olabilmektedir. Matematiksel olarak, aktivasyon fonksiyonu türevi

hesaplamamızı sağlayan sürekli bir fonksiyondur. Nöronun aktive olup olmayacağını belirler. En temel yapay sinir hücresi aktivasyon fonksiyonu olmadan, 0 ile 1 sonuçları üreterek öğrenme gerçekleştiremez.

Aktivasyon fonksiyonları girdileri ve çıktı değerleri olarak bir eğri ile görselleştirilebilmektedirler. Görselleştirildikleri eğriyi güncellerken eğime bağlı olarak eğrinin hangi yönde ve ne kadar değiştirileceği türevi alınarak bulunur.

Sigmoid fonksiyonu girdileri $[-\infty;\infty]$ aralığından alarak [0; 1] çıktıları üretmektedir. Sigmoid fonksiyonunun matematiksel gösterimi Denklem 2.2'de yapılmıştır.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

Girdilerdeki en ufak değişiklik çıktılarda büyük değişikliklere yol açmaktadır. Genellikle ikili sınıflandırmalarda çıktı katmanında kullanılır. Girdilerin tüm değerlerini, değerler 0 ila 1 arasında olacak şekilde, birbirlerine göre oranlayarak çıktı üretir. Softmax fonksiyonu Sigmoid'e göre girdilerin sınıflarda olma yüzdelerini hesaplar. Softmax'in formülü Denklem 2.3'te gösterilmiştir.

$$F\left(X_i\right) = \frac{\mathrm{Exp}\left(X_i\right)}{\sum_{j=0}^{k} \mathrm{Exp}\left(x_j\right)} \tag{2.3}$$

Softmax birden çok girdiyi 0 ila 1 arasında toplamları 1 edecek şekilde oranlar, bu oranlama ile girdilerin hedef çıktı sınıflarında yüzdesel olarak ne kadar yakın olduğu çıktısını elde ettirir. Çok sınıflı sınıflandırma problemlerinde sıklıkla kullanılmaktadır. Genellikle YSA'larda Softmax için ayrı bir katman oluşturulur. Hem Sigmoid hem de Softmax formülünde üstel çarpanlar kullanıldığı için hesaplaması maliyetlidir. Sigmoid fonksiyonunda işlevi yok olan gradyan problemiyle karşılaşılabilmektedir. İşlevi yok olan gradyan problemi geri yayılım modellerinde gradyan hesaplarının sürekli daha sivrileşerek ve küçülerek geriye yayılmasından dolayı tahminleme oranının düşmesidir.

Tanh fonksiyonu girdileri $[-\infty;\infty]$ aralığından alarak [-1; 1] aralığında çıktılar üretmektedir. Tanh fonksiyonu Denklem 2.4'teki şekilde formülize edilmektedir.

$$F(x) = tanh(x) \tag{2.4}$$

Genellikle gizli katmanlardaki hücrelerde kullanılmaktadır. Çıktılar 0 değeri merkez noktası belirlenerek üretilmektedir. Tanh, formülü gereği doğrusal değildir. Bu yüzden çok katmanlı model mimarilerine uygundur. Gradyan değerlerini sonuçları çok daha Sigmoid'e

göre keskindir. Tanh aktivasyonunu uygulamak daha kolaydır, bu nedenle Sigmoid fonksiyonu yerine genellikle tercih edilir. Tanh yinelenen sinir ağlarında (RNN) sonuçları bir katmandan diğer katmana taşıma konusunda sıklıkla kullanılmaktadır. Tanh fonksiyonunda işlevi yok olan gradyan problemiyle karşılaşılabilmektedir.

ReLu fonksiyonu $[-\infty;\infty]$ girdileri alarak [0; max(x)] aralığında çıktılar üretmektedir. Sigmoid ve Tanh fonksiyonlarına göre hesaplama maliyeti düşüktür. ReLu, Denklem 2.5'te formülize edilmiştir.

$$F(x) = max(0,x) \tag{2.5}$$

Gizli katmanda varsayılan olarak kullanılan aktivasyon fonksiyonudur. YSA'larda nöron ölümü olarak adlandırılan bir probleme yol açabilmektedir. Nöron ölümü bir hücrenin sürekli aynı değeri yani 0 sayısını vermesidir. Ölen nöron hesaplamada hiç bir rol oynayamaz ve saf dışı kalır. Ölen ReLU nöronu tekrardan canlandırılamaz. Ölen nöronun gradyanı 0'dır. 0 olan gradyanın, gradyan inişi de 0 olduğu için ağırlıklarını hiç bir zaman güncellenememektedir bu nedenle ölü nöron olarak adlandırılmaktadır. Leaky ReLu, ölü nöron sorununu çözmek için negatif girdilere pozitif gradyan uygulayarak nöronun tekrardan canlandırılması için bir şans oluşturur. Leaky ReLu Denklem 2.6'daki şekilde gösterilmektedir.

$$F(x) = max(0.1x,x) \tag{2.6}$$

Sigmoid ve Tanh aktivasyonları da aynı sorunlardan etkilenmektedir. Ancak her zaman için küçük bir gradyan değeri uzun süreli model öğrenimlerinde nöronların kurtarılmasında etki gösterebilmektedir.

Maxout aktivasyonu, Leaky ReLu ve ReLu fonksiyonlarının genelleştirilmesiyle elde edilmiştir. Maxout aktivasyonu Denklem 2.7'de gösterilmiştir.

$$F(x) = \max\left(w_1^T x + b_1, w_2^T x + b_2\right) \tag{2.7}$$

Maxout, ölen nöron problemini yaşamaz. ReLu'ya oranla iki katı parametre ihtiyacı olduğundan, daha çok eğitime ihtiyaç duyar.

Yapay sinir ağları öğreticili ve öğreticisiz olmak üzere iki tür öğrenme yaklaşımına ayrılmaktadır. Öğreticili modellerin eğitiminin gerçekleşmesi için öğrenilmesi istenen problemin sonuçlarına hakim, modeli sonuçlarını bildiği veriler ile besleyecek bir

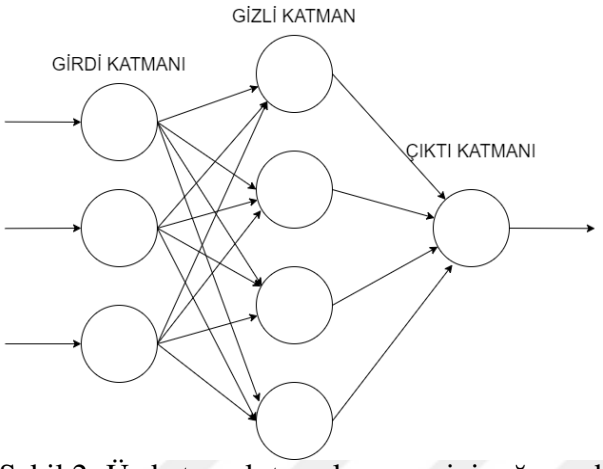
öğreticiye ihtiyaç duyar. Öğreticisiz öğrenme modeli ise bir öğreticiye ve sonuçlara hakimiyete gerek olmadan, öğrenilmesi istenen durumdaki verilerin birbiri arasındaki ilişkilerine ve maliyet hesaplarına dayanan öğrenme modelidir. Yapay sinir ağlarının öğrenme gücü tıp, endüstri, biyoloji, ekonomi, mühendislik ve benzeri bir çok alanda başarılı sonuçlar elde edilmesini sağlamıştır.

Yapay sinir ağları çok yönlülüğü ile insan beyninin öğrenme yapısını taklit ederek teorik olarak her alanda çalıştırılabilmektedir. Yapay sinir ağlarının iki temel yetkinliği vardır bunlardan ilki öğrenme algoritmasını oluşturması, ikincisi nöronlar arası bağlardaki ağırlıkların tutulması ile öğrenilmiş bilgiler olarak kullanılmasıdır. Yapay sinir ağları aynı anda bir çok bilgiyi paralel işleyebilmektedir, hızlı ve hataya toleranslıdır. Nöronlardaki ağırlıkların değişimiyle ve yeniden eğitilmesiyle kazandığı uyum özelliğiyle durağan olmayan alanlarda da işlem gerçekleştirebilmektedir. YSA doğrusal olmayan modelleri de desteklemektedir. YSA'lar bu özellikleriyle karışık mühendislik problemlerinin çözümlerinde sıklıkla kullanılmaktadır.

Yapay sinir ağları insan sinir ağlarını temel alan bir sanallaştırma temsili gerçekleştirmektedir. Mcculloch ve Pitts (1943) yapay sinir ağının temellerini oluşturmuş ve ilk ağın örneğini sunmuşlardır. Mcculloch ve Pitts (1943) ikili değerler olan binary değerler ile öğrenmenin gerçekleştirilebileceğini öne sürmüşlerdir. Öne sürülen hücreye gelen girdiler engelleyici ve uyarıcı olarak ikiye ayrılmaktadır ve girdiler eşik değerini aştığında çıktının üretimi sağlanmaktadır. Hebb (1949), Mcculloch ve Pitts (1943)'in öne sürdüğü bu model üzerinden modern makine öğrenmesinin ilk kurallarından birini ortaya çıkarmıştır. Bu kural aynı model sonuçların elde edildiği girdiler arasındaki bağın kuvvetlendirilmesi gerektiğini, farklı ise aradaki bağın zayıflatılması gerektiğini savunmaktadır. Yeni ağırlık güncellemeleri çıktıların çarpımıyla ve eski ağırlığın toplamıyla elde edilmektedir. Minsky ve Papert (1969) temelleri yeni atılan temel yapay sinir ağlarına algılayıcı denilen yeni yaklaşımı kazandırarak, yapay sinir ağlarının her şeyi öğrenemeyeceğini XOR mantık kapısını algılayıcının çözemeyeceği örneği ile ortaya çıkarmıştır. Bu sebeple yapay sinir ağlarının geliştirilmesi on yıl boyunca askıya alınmıştır. Teknolojinin gelişmesiyle, yeni öğrenme yöntemlerinin araştırılmasına başlanmış ve yeni mimarilerin oluşturulmasıyla yapay sinir ağlarının etkinlikleri tekrardan gündeme alınmıştır. Yapay sinir ağlarında yeni yöntemlerin ve uygulama alanlarının ortaya çıkışıyla öğrenme optimizasyonu, veri temsilinin geliştirilmesi gibi konularda çalışmalar yapılmaktadır. Yapılan çalışma kapsamında yapay sinir ağlarının temsili ve verilerinin işlenmesi konularında halihazırda kullanılan yöntemler dışında bir yöntem öne sürülmüştür.

Sinir ağlarının temsilinde sinir ağı hücrelolteri düğümler, sinir hücreleri arasındaki

ilişkileri temsil eden sinapslar ise ilişkiler olarak nitelendirilmektedir. Yapay sinir ağı, düğüm ve düğümler arası ilişkileri içeren katmanlardan oluşmaktadır. En temel yapay sinir ağı yapısı giriş ve çıktı katmanı olarak iki katmanlı algılayıcı olan perseptrondur. Günümüzde kullanılan yapay sinir ağları temel olarak üç katman içermektedir. Katmanlar işlenecek verilerin oluşturduğu girdiler katmanı, işlemi yapacak hücrelerin oluşturduğu gizli katman ve sonuç olarak çıktıları üretecek çıktı katmanıdır. Yapay sinir hücreleri arasında ilişkiler tek yönlü veya çift yönlü olabilmektedir. Şekil 2'de örnek üç katmanlı bir yapay sinir ağı modeli resmedilmiştir.



Şekil 2. Üç katmanlı temel yapay sinir ağı modeli

Yapay sinir ağları çözmekle ilgili oldukları sınıflandırma, kümeleme ve tahminleme problemlerini istenen başarımla gerçeklemek için test verileriyle eğitilmelidir. Modeller ilgili oldukları alanda girdileri ve çıktı sonuçları bilinen veya bilinmeyen test örnekleriyle eğitilmelidir. Belirli bir mantık çerçevesinde oluşturulmuş verilere veri seti denmektedir. Veri setlerini belirli oranlarda bölerek eğitim ve test veri setleri oluşturulmaktadır. Bu veri setleri ile başarımlar istatistiksel olarak hesaplanarak modelin başarımı hakkında bilgi edinilmektedir.

2.1.1. Yapay Sinir Ağ Modelleri

Yapay sinir hücrelerinin temel yapısı perseptron ya da diğer adıyla algılayıcıdır. Perseptron, sinir ağlarında tek bir katmanın ismi olarak da adlandırılabilir. Perseptronların çok katmanlı olmasıyla sinir ağı ortaya çıkmaktadır. Perseptron doğrusal ikili sınıflandırma yapar. Öğreticili öğrenmede kullanılmaktadır. Perseptron girdi katmanı, ağırlık, sapma diğer adıyla bias, toplam ve aktivasyon fonksiyonlarını içermektedir. Rosenblatt (1958) tarafından öne sürülmüş perseptron, yani algılayıcı modeli, modern sinir ağlarının temelini oluşturmuştur. Girdi katmanından alınan verilerin işlenip sonraki katmanlara aktarılamasından dolayı ileri beslemeli ismini almıştır. Eğitilen sinir hücreleri hata

yayılımının sağlanması için geri yayılıma ihtiyaç duymaktadır. Teoride yeterince sinir ağında yeterince gizli sinir hücresi olduğunda girdi ve çıktı arasındaki ilişkiyi bulabileceği düşünülmektedir, ancak pratikte kullanım alanı çok sınırlıdır. Tek başına kullanılmalarından ziyade yapay sinir ağının yapıtaşı olarak kullanımları yaygındır.

Minsky ve Papert (1969) yaptıkları çalışmada perseptron modelinin basit mantık kapısı olan XOR mantık kapısını çözemediğini göstermişlerdir. Model eğitimi aşamasında tekrar sayısının ve örnek veri kümesinin büyütülmesiyle ağırlık ve sapma değerlerinin istenen sonuç değerlerine yaklaşması sağlanabilmektedir. Model parametrelerinin test verilerinin arttırılmasıyla, genelleştirilebilir model paramaterleri elde edilerek perseptronun, diğer bir adıyla algılayıcının, sınıflandırma başarımı artmaktadır. Girdiler g_1, g_2, ağırlıklar a_1, a_2 çıktılar c_1, c_2 olarak isimlendirilmiştir. Verilen örnekte aktivasyon fonksiyonu Sigmoid'dir. Çıktı toplam fonksiyonu Denklem 2.8'de gösterilmiştir.

$$C1 = A1.G1 + A2.G2 + b \tag{2.8}$$

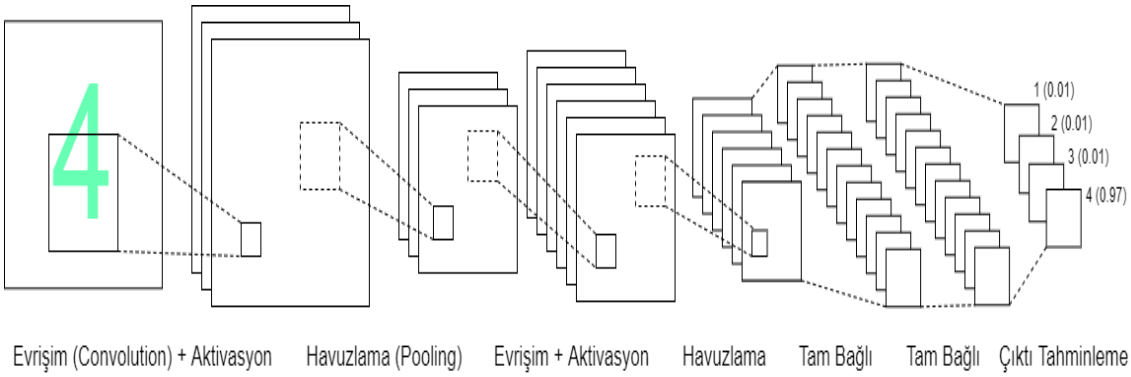
Girdi katmanından algılayıcıya veriler girerken ağırlıklarla çarpılır ve toplanır, sapma değeri toplanır ve sonuç olarak aktivasyon fonksiyonuna aktarılarak çıktı sağlanır. Nöronun öğrendiği değer Ç1 olarak adlandırılmaktadır. Nöronun ürettiği çıktı değeri ile çıktı katmanından direkt sonucun elde edilmesi sağlanabileceği gibi başka bir nöronun girdisi olarak da kullanılabilmektedir.

Evrişimli sinir ağları (ESA), görüntü girdi verileriyle, problemlerin üzerinde bilgisayar görüsü alanında etkili olmasıyla başarı ve ivme kazanmış bir yapay sinir ağı modelidir. Evrişim ya da diğer bir adıyla konvolüsyon matris çarpımında kullanılan matematiksel bir işlemden gelmektedir. Bir ESA görüntüyü işlerken üç farklı yapı kullanmaktadır. Üç rengin karışımıyla edilmiş görüntü verisi nöron seti, renkli görüntünün üç katmanını kırmızı, yeşil ve mavi olarak üç ayrı görüntüyü oluşturarak analiz edilmektedir. Önemli özellikleri tanımlamak için görsel her renk için birer birer analiz edilmektedir. ESA mimarisinde bir katmandaki tüm nöronların bir sonraki katmandaki tüm nöronlarla iletişim kurduğu tamamen bağlı sinir ağı yapısı kullandığı zamanlarda büyük görüntüleri analiz etme konusunda yetersiz kalabilmektedir. Görüntüler büyüdükçe hesaplama maaliyetleri artmaktadır. Bir ESA, bir katmandaki nöronların bir sonraki katmandaki tüm nöronlara bağlanmadığı üç boyutlu bir yapı kullanır. Her katmanda, bir nöron seti görüntünün küçük bir bölgesini veya özelliğini işlemektedir. Bu yapının nihai çıktısı, olasılık çıktılarının olduğu bir vektör veya sayı dizisidir. ESA görüntüyü taramakta ve tararken her adımda küçük bir bölümünün analizini yapmayı ve her özelliğin gerekli

sınıfa ait olasılıklarla bir özellik dizesi oluşturmayı içeren bir evrişim gerçekleştirmektedir.

Görüntülerin anlamlandırılabilmesi ve YSA modelleri ile öğrenilmesi için Cun ve diğerleri (1990) evrişimli sinir ağlarının temelini el yazısı rakamların tanınması problemi için geri beslemeli bir ağ modeli oluşturduklarında atmışlardır. Lecun, Bottou, Bengio ve Haffner (1998) yılında yaptığı çalışma ile el yazısı rakam tanıma problemini artık ESA olarak isimlendirdikleri önceki çalışmalarını temel alarak geliştirmişler ve derin öğrenmenin temellerini oluşturmuşlardır. Yapılan çalışmalar ile derin öğrenme ağlarının görsel veri problemi üzerinde gerçeklenmesiyle derin öğrenme, görsel verileri işleme alanında yetkinlikler kazanmaktadır.

ESA'da temel olarak 5 katman tipi bulunmaktadır. Bu katmanlar evrişim (Convolution) katmanı, doğrusal olmama (Non Linearity) katmanı, havuzlama (Pooling) katmanı, düzleştirme (Flattening) katmanı ve tam bağlı (Fully-Connected) katmanıdır. Aynı görevi üstlenen birden çok katman olabilmektedir. Convolution katmanı, görselin en önemli özelliklerini çıkarmak için uygulanan bir katmandır. Non Linearity, katmanı konvolüsyon işleminin ardından aktivasyon uygulama katmanıdır. Günümüzde ESA'larda en yaygınca kullanılanı aktivasyon fonksiyonu ReLu'dur. Pooling katmanı, ağırlıkların azaltımıyla ilgilenir, aynı görselin önemli özelliklerini daha küçük boyutlarda ifade etmek için kullanılmaktadır. En önemli bilgileri korurken, her özelliğin boyutsallığını azaltma işini havuzlama katmanı yapar. En popüler kullanılanı Max Pooling'tir. Flattening katmanı, işlenen görsellerin yapay sinir ağının işleyeceği formata dönüştürülmesiyle ilgilidir. Fully connected katmanı, ise YSA'nın bulunduğu katmandır. Öğrenim ve tahminleme işlemleri bu katmanda gerçeklenir. En popüler ve kabul gören ESA'lar Simonyan ve Zisserman (2014)'ın geliştirdiği VGGNet, Szegedy ve diğerleri (2015)'nin geliştirdiği GoogLeNet, Krizhevsky, Sutskever ve Hinton (2012)'ın geliştirdiği AlexNet, Lecun ve diğerleri (1998)'nin geliştirdiği LeNet'tir. ESA'da görüldüğü üzere verinin optimizasyonu ve temsili çok büyük bir önem kazanmıştır. ESA'da öğrenim sürecindeki girdi verilerinin temsil işlemleri görsellerin optimizasyonunda, yani daha öğrenme işleminin başlamasından öncesine taşınmıştır. Bu yaklaşım da araştırmacılara veri temsilinin önemini göstermektedir. Yüz tanıma, görüntülerde nesneleri tespit ve sınıflandırması, robot ve otonom araçlarda bilgisayar görüsünü güçlendirmek, sahne tespiti, anlamsal çözümleme, cümle sınıflandırması gibi bir çok alanda başarılı olarak kullanılmaktadır. Günümüzde en çok kullanılan YSA modellerinden biridir. Şekil 3'te evrişimli sinir ağı modeli katmanları ve akışları gösterilmiştir.



Şekil 3. Evrişimli sinir ağı örneği

Tekrarlayan Sinir Ağı (TSA), sinir ağlarının doğasından kaynaklanan ardışık girdi verileriyle çalışmasında etkin rol oynamaktadır. Rumelhart, Hinton ve Wilson (1986) geliştirdikleri model ile öğrenim sırasında elde edilen hataların geri besleme ile tekrar tekrar ağırlık güncellemesinde kullanılmasını sağlamışlardır. TSA, ağ analizi veya yazılım güvenliği gibi konularda, birbiri ardına zaman içinde sisteme girdisi olan metin, görsel medya, ses veya birden farklı girdinin oluşmasında öğrenim gerçekleştirmek için yetkinliğe sahiptir. Bir TSA ağı bir giriş dizisini kabul ederek, önceki girdilerin hatırlanmasını sağlar ve her yeni girişle birlikte yeni bir öğrenme katmanı ekleyerek çalışır.

TSA, zaman içindeki girdi dizisinde çalışmaktadır. Örneğin, videodaki kareler veya bir cümledeki kelimeler bu girdi dizisini oluşturabilmektedir. Sinir ağında her girdi için bir sinir katmanı bulunur. TSA ağı öğrendiğinde, çok katmanlı bir geri dönüşüm biçimi olan zamanla geri yayılım (ZGY) gerçekleştirmektedir. ZGY, en son zaman adımından, her bir önceki adıma aşamalı olarak geri dönmek için her zaman bir nöron için en iyi ağırlıkları bulmak için gradyan inişini kullanmakta ve aynı zamanda bilgi aktarımını yöneten optimal ağırlıkları öğrenmek için de bir adımdan diğerine geçişlerde zincir kuralını kullanmaktadır. Dil modellemesi ve yazı üretimi, konuşma algılama, görsel etiketleme, yazı verilerinde dil çevirisi, zaman serileriyle çalışan problemlerde öğrenim gerçekleştirmek için TSA sıklıkla kullanılmaktadır. Gizli katmanlardan elde edilen çıktılar hafıza sinirlerine iletildikten sonra girdi katmanlarında hesaplamaya katılarak tekrarlama gerçekleştirilir. En çok bilinenleri Elman ve Jordan ağlarıdır. Gizli hücreler $h_1, h_2..., h_t$, çıktı hücreleri $y_1, y_2..., y_t$, aktivasyon fonksiyonları $\sigma_1, \sigma_2..., \sigma_t$, girdi hücreleri $x_1, x_2..., x_t$, sapmalar $b_1, b_2..., b_t$, ağırlıklar $W_1, W_2..., W_t$, hafıza sinirleri $U_1, U_2..., U_t$ olarak ifade edildiğinde TSA matematiksel olarak Denklem 2.9 ve Denklem 2.10 denklemlerindeki şekilde formülize edilmektedir.

$$h_t = \sigma_h \left(W_h x_t + U_h h_{t-1} + b_h \right) \tag{2.9}$$

$$y_t = \sigma_y \left(W_y h_t + b_y \right) \tag{2.10}$$

Elman ağı hafıza hücreleri gizli katmanlar tarafından beslenmekte ve girdi katmanlarını aktive olmasına göre beslemektedir. Jordan ağlarında hafıza hücreleri Elman ağlarından farklı olarak çıktı katmanından beslenmektedir, tek farkı budur. Elman ve Jordan ağlarında girdilerde ileri yayılımlı öğrenme kuralı uygulanmaktadır. Jordan (1989) tarafından geliştirilen Jordan ağları ve Elman (1990) tarafından geliştirilen Elman ağları aynı zamanda basit tekrarlayan ağlar olarak isimlendirilmektedir.

2.1.2. Öğrenme Algoritmaları

Yapay sinir ağlarında modellerin oluşturulmasında ve uygulanmasında, ilgili olacağı veriler üzerinde bir öğrenme gerçekleştirme amacı bulunmaktadır. Verilerin bulunduğu alana ve modeli oluşturan araştırmacının veriler üzerindeki hakimiyetine göre uygulanacak yapay sinir ağına karar verilmektedir. Araştırmacı verileri öğrenim sonunda hangi sınıfa ait olacağını işaretleyebilir bilgi düzeyine sahipse, öğreticili öğrenme yapılan sinir ağı modelini tercih etmesi gerekirken, verilerin kendi arasındaki bağ hariç bilinmiyorsa ve sınıfı işaretlenemiyorsa öğreticisiz öğrenme sağlayan sinir modellerini tercih etmelidir. Öğreticisiz öğrenmede amaç, veriler arasında örüntüleri ve benzerlikleri yakalayan bir model üretmektir. Öğreticisiz öğrenme sonucunda veriler kümelenmektedir. Öğreticili öğrenme, sınıfları belirlenerek işaretlenmiş veriler ile oluşturulmuş bir sinir ağı modeli ortaya çıkararak, bilinmeyen bir veri geldiğinde hangi sınıfa ait bir veri olduğunun çıktısını vermek amacıyla sınıflandırma yapar. Öğreticili öğrenmede etiketlenmiş veriler sisteme girilen bir verinin karşılığında hangi çıkış değerinin üretildiğinin bilindiği verilere denmektedir. Çıktılar, veri dizileri olabilecekleri gibi sayı da olabilirler. Sınıflandırma ayrık verileri tahmin etmek için kullanılmaktadır. Yayılım tabiri ise devamlı verilerin tahmininde kullanılmaktadır. Öğreticili ve öğreticisiz öğrenme yöntemlerinin yanında, yarı öğreticili öğrenme ve destekli öğrenme yöntemleri de bulunmaktadır.

Öğreticili öğrenme yapan sinir ağ modelleri, hem girdiler hem de çıktıların modelde işaretlenmesi ve ağ üzerinde hatanın ve öğrenim ağırlıklarının yayılmasıyla çalışmaktadır. Model eğitimi için işaretlenmiş ve derlenmiş verilere eğitim seti, eğitimin sonucunda modelin başarımını ölçmek için işaretlenmiş verilere test veri seti denilmektedir. Görsel üzerinden duygu durum analizi, obje tanıma, hareket ve mimik tanıma, metin sınıflandırma,

metinlerde duygu durum analizi gibi insan tarafından etiketlenebilen tüm veriler için öğrenme, öğreticili öğrenme ile sınıflandırarak yapılabilmektedir. Öğreticili öğrenmede öğrenmenin yapılabilmesi için etiketli verilerin, tahminleme modeli ve öğrenme yöntemi olması yeterlidir. En çok kullanılan öğreticili öğrenme algoritmaları en yakın komşu, karar ağaçları, doğrusal yayılım, destek vektör makinesi, Naive Bayes ve yapay sinir ağlarıdır.

Öğreticisiz öğrenmede problemi çözecek olan verilerde ilişkiler model tarafından öğrenildikten sonra ilgili olduğu alanda araştırmacıya yeni bilgiler öğretmek üzere çalışmaktadır. Problem uzayındaki verileri anlamlandırmak için sıkça kullanılan öğrenme yöntemidir. Hiçbir etiket ve sınıfın bulunmasına gerek yoktur. Veri madenciliğinde sıklıkla öğreticisiz öğrenmeye başvurulmaktadır. Kümeleme algoritmaları ve ilişki kurma algoritmaları ve problem uzay verileri öğreticisiz öğrenme gerçekleştirilmesi için yeterlidir. K-means sınıflandırma algoritmasına öğreticisiz öğrenmede sıklıkla başvurulmaktadır.

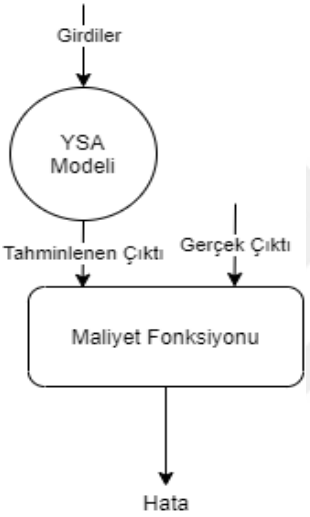
Yarı öğreticili öğrenme, etiketlenmiş veriler ile etiketlenmemiş verilerin bir arada kullanılabileceği problem veri uzaylarında çalışmak amacıyla ortaya çıkartılmıştır. Veri etiketlemek, maliyeti büyük bir işlemdir ve problem uzayına hakim bir uzman gerektirmektedir. Verilerin çoğunluğunda etiketlerin bulunmadığı bir problem uzayında uygulanması uygun olacaktır. Konuşma analizi, internet içerik sınıflandırması, DNA ve protein dizesi sınıflandırması problemleri ihtiyaç duyulduğu alanların başında gelmektedir.

Destekli öğrenme yönteminde eğitim sırasında model değerlendirme sonucunu ajan ya da destekçi olarak adlandırılan insana, verinin sınıf doğruluğunu kontrol etmesi için vermektedir. Çıktısı elde edilen bu model ile öğrenmeye insan faktörü dahil olmaktadır. Model, destekçiden alınan etiket ile model hem öğrenimini günceller hem de problemin çözülmesi için destek olan kişiye olabildiğince tahmin doğruluğu yüksek sonuçlar çıkartır. Destekli öğrenmede hata yapılmaması gereken durumlarda ve problem uzayının büyük olduğu alanlarda kullanılmaktadır. Yinelemeli olarak her an kendini güncellemesi gerekmektedir. Destekli öğrenme yöntemiyle oluşturulmuş modele örnek olarak bir doktorun hastanın üzerindeki bulguları eğitimi istenilen seviyeye geldiği düşünülen modele girmesi ile olası hastalıkların kendisine gösterilmesi ve doktorun tecrübesini kullanarak bu hastalıkları onaylaması veya yanlış olduğunu modele iletmesi ile modelin daha iyi öğrenme yapması gerektiğinin ortaya çıkması durumu verilebilir. Örnekte hata yapılmaması gereken riski yüksek bir alan olan sağlıkta destekli öğrenmenin bir kullanım alanı olduğu gösterilmiştir. Sınıflandırma yapmak üzere en iyileme yaklaşımı ile çalışmaktadır. Destekli öğrenme yönteminde öğrenme adımlarının tekrarında Markov karar zinciri devreye girmektedir. Modelin çıktısı ile destekçinin verdiği karar etiketleri Markov zincirinde tutulmaktadır. Otonom ve yarı araçlar, robotik, belirli problem uzayına sahip oyunlar gibi

alanlarda sıklıkla kullanılmaktadır. Geçici fark öğrenmesi, mücadeleci ağ, pekiştirmeli öğrenme Q-öğrenmesi en çok kullanılan yöntemlerdir.

2.1.3. Maliyet Fonksiyonları

Yapay sinir ağlarında problem uzayındaki verilerde öğrenim gerçekleştirildiğinde elde edilen tahmin çıktılarının gerçekte olması gereken çıktılar arasındaki fark öğrenimin doğruluğunu göstermektedir. Tahminlenen ile gerçekte olması gereken farkı hesaplayan fonksiyona maliyet fonksiyonu ya da kayıp fonksiyonu denmektedir. Ağırlıklar maliyet fonksiyonunu en aza indirmek üzere güncellenmektedir. Şekil 4'te maliyet fonksiyonunun YSA modeli üzerindeki konumu gösterilmektedir.



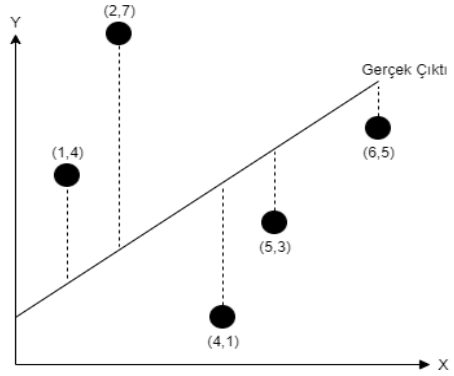
Şekil 4. Maliyet fonksiyonunun YSA modelindeki yeri

Ortalama hata karesi diğer adlarıyla L2 veya Mean Squared Error maliyet fonksiyonu, doğrusal geri yayılmada yaygınca kullanılan bir maliyet fonksiyonudur. Tahminlenen ile gerçek çıktı arasındaki farkın karesinin alınmasıyla tüm veri kümesi için optimum hata bulunmaya çalışılmaktadır. Modelin tahminlediği çıktı x_i, gerçek çıktı $\tilde{x}_i$, eleman sayısı n olarak alındığında matematiksel olarak Denklem 2.11 şeklinde formulize edilmektedir.

$$\text{OHK} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2 \tag{2.11}$$

Çıktıların farklarının karesi hesaplandığından sonuçlar hep pozitif çıkmaktadır. Şekil 5'te siyah noktalar YSA model çıktılarını temsil etmekte, yatay ve dikey eksenlerin arasında çizilen çizgi gerçek çıktıların olması gerektiği değerleri, siyah noktalar ile gerçek çıktı arasındaki kesikli çizgiler ise gerçek ile model çıktısı arasındaki farkı ifade etmektedir. Örnek bir YSA modeli olduğu varsayılarak oluşturulan iki boyutlu koordinat sisteminde

tahminlenen çıktılar ve gerçekte olması gereken çıktılar gösterilerek ortalama hata karesinin görselleştirilmektedir. Ortalama hata karesi ile maliyet hesaplarının dezavantajı, ortalama hata karesiyle maliyet hesaplandığı için yanlış değerleri hesaplarken de hesap katsayısı yüksek olacağından hata oranının yükselmesine sebep olabilmektedir.



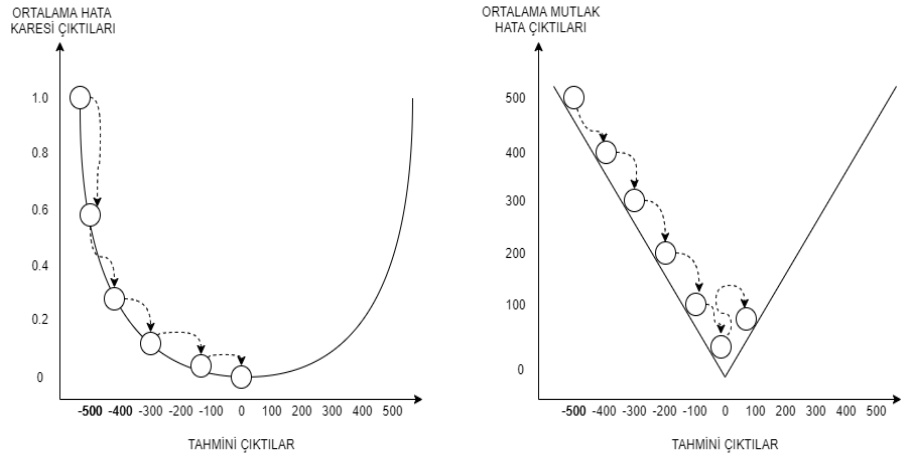
Şekil 5. YSA model çıktılarının ortalama hata karesinde kullanılması

Ortalama mutlak hata diğer adlarıyla L1 veya Mean Absolute Error maliyet fonksiyonu, doğrusal geri yayılmada kullanılan bir maliyet fonksiyonudur. Tahminlenen YSA model çıktısı ile gerçek çıktının arasındaki farkın mutlak değerlerinin toplamıdır. Tahminlenen çıktı ile gerçek çıktı arasındaki farkın hesaplanmasından dolayı ve yönleri bulunmayan hatalar ölçüldüğü için ortalama hata büyüklüğü bulunmaktadır. Maliyet fonksiyonunun çıktı değerleri pozitif değerlerdir. Modelin tahminlediği çıktı x_i, gerçek çıktı $\tilde{x}_i$, eleman sayısı n olarak alındığında matematiksel olarak Denklem 2.12 şeklinde formulize edilmektedir.

$$OMH = \frac{\sum_{i=1}^{n} |x_i - \tilde{x}_i|}{n} \tag{2.12}$$

Hatalı değerlerin maliyetini hesaplarken OMH daha isabetli sonuçlar göstermektedir. OHK'yi YSA modellerinde gerçeklemek daha pratiktir. OMH'nin OHK'ye göre gradyan inişi öğrenme süreci boyunca aynı oranda olacağı için gerçek değeri atlama ihtimali yüksektir, ancak hata değerlerinin maliyet sonucunu etkilemesi noktasında daha kararlı sonuçlar vermektedir. OHK'nin gradyanı büyük maliyet çıktılarında yüksek, küçük maliyet çıktılarında ise düşüktür, bu şekilde maliyeti en doğru yerinde hesaplamaya daha yatkındır. Hatayı yüksek tutacak verilerin model öğrenmesinde rolü yüksek ise OHK'ne, hatayı yükseltecek verilerin gerçekten de hata olması gerektiği bir öğrenme süreci modelleniyorsa OMH seçilmelidir. Şekil 6'da ortalama hata karesi ve ortalama mutlak hata arasındaki fark gösterilmektedir. Tahminlenen çıktılar yuvarlak ile gösterildiğinde, ortalama hata karesinin çıktılarında ilk inişte, çıktılar arasındaki fark çok iken giderek azalmakta ve doğru sonuca

ilerlemekte iken, OMH'de her iniş arasındaki mesafe aynı olmakta ve beklenen gerçek çıktı değeri olan 50 değerinin yakalanması zorlaşmaktadır.



Şekil 6. Tahminlenen çıktıların ortalama hata karesi ve ortalama mutlak hata fonksiyonlarındaki adım temsili

Çapraz düzensizlik hatası, gerçek ile model çıktısında tahminlenen maliyet arasındaki dağılımda olasılıksal farklılığı ölçerek dağılımların birbirine yakınlığı ya da uzaklığının ölçülmesinde kullanılan bir maliyet fonksiyonudur. Modelin tahminlediği çıktı x_i, gerçek çıktı $\tilde{x}_i$, dağılım fonksiyonu $P()$ olarak tanımlandığında matematiksel olarak Denklem 2.13 ile formülize edilmektedir.

$$CDH = -\sum_{x} P(x) \log P(\tilde{x}) \tag{2.13}$$

Çok sınıflı sınıflandırıcılarda maliyet fonksiyonunu oluşturan çıktıların yayılımlarının arasındaki düzensizlik farkına bakıyor olmak avantaj getirmektedir.

2.1.4. Optimizasyon Algoritmaları

Maliyet fonksiyonu ile hatası bulunan YSA modelindeki ağırlıkların ve sapmanın sonucunda oluşan değerleri en optimize ve isabetli hale getirmek için ihtiyaç duyulan algoritmalara optimizasyon algoritmaları ve fonksiyonları denilmektedir. Modelin başarımının arttırılması için hesaplanan hatanın en aza indirilmesi gerekmektedir. Yapay sinir ağında optimizasyon işlemi ileri beslemeyle ya da geri yayılım kullanılarak yapılmaktadır. Maliyet fonksiyonuyla hesaplanan hata, hataları en aza indirecek şekilde hesaplanırken model ağırlıklarını ve sapmayı değiştirmek için modelde bir önceki katmana geri yayılım ile yayılmaktadır. Model ağırlıkları optimizasyon algoritması ile güncellenmektedir. Optimizasyon fonksiyonları genellikle gradyanı, ağırlıklara göre maliyet fonksiyonunun türevini hesaplar ve ağırlıklar hesaplanan gradyanın tersi yönünde

değiştirilir. Optimizasyon fonksiyonu modelde minimum kayıp fonksiyonuna ulaşana kadar tekrar tekrar çalıştırılmaktadır. Optimizasyon algoritmaları aktivasyon fonksiyonu, maliyet fonksiyonu ile beraber yapay sinir ağı modellerinin vazgeçilmez bir parçasıdır, isabetli modellerin üretiminde önemli bir rol oynamaktadır. Optimizasyon algoritmaları, sabit öğrenme hızlı algoritmalar ve uyarlanabilen öğrenme algoritmaları olarak iki başlık altında incelenebilmektedir.

Sabit öğrenme hızlı algoritmalara yaygınca kullanılan rastgele gradyan iniş algoritması örnek gösterilebilir. Bu algoritmalarda öğrenme hızı olarak bir hiper parametre yani araştırmacının verdiği sabit bir parametre verilmesi gerekmektedir. Uygun bir öğrenme hızını seçmek için araştırmacının bir çok deneme yaparak bu parametreyi en optimal değerini bulması ve problem uzayına ve öğrenmesine hakim olması gerekmektedir. Küçük bir öğrenme hızı seçildiğinde doğru çıktılara ulaşmak ve yakınsamak zorlaşacağı gibi büyük bir öğrenme hızı seçildiğinde de doğru çıktılara ulaşamadan değeri teyet geçme durumu söz konusu olabilmektedir. Küçük öğrenme hızı hiper parametresi seçildiğinde öğrenme yavaş gerçekleşmektedir. Momentum kavramı da çıktının yerel minimumuna yaklaştıkça öğrenme hızının belirlendiği bir hiper parametre olarak literatürde yer almaktadır.

Gradyan inişinin başarıya ulaşmasının zor olduğu kısım hiper parametrelerinin önceden tanımlanmasının gerekliliği ve YSA modeline ve problemin veri uzayına bağlı olarak değişken olmasıdır. Gradyan inişinde ağırlık ve sapma değerlerinin güncellemelerine aynı öğrenme hızının uygulanması YSA modelinde önemsiz veya önemsiz olması gereken değerlerde de aynı hız ile öğrenim sağlanması sorun oluşturabilmektedir. Problem uzayındaki veriler az ise her parametrenin isabetli öğrenim için aynı hızda güncellenmemesi istenebilmektedir.

Adadelta, Adagrad, Adam, RMSprop uyarlanabilir gradyan iniş algoritmalarıdır. Rastgele gradyan inişinden farklı olarak uyarlanabilir öğrenme hızı ile çalışmaktadırlar. Öğrenme hızının araştırmacı tarafından hiperparametrelerle belirlenmesi öğrenimi uzatacağından maliyetli bir çalışma yöntemidir. Sezgisel yaklaşım sağlayan parametreler ile uyarlanabilir optimizasyon algoritmalarıyla maliyet azaltılabileceği gibi isabet de arttırılabilmektedir.

Gradyan iniş algoritması, tüm verilerin çıktılarını kullanarak gradyan hesabı yaparak yerel minimum noktası bulunana kadar çıktı değerlerinin tersi yönde güncellenmesini ağırlıkları ve sapmayı değiştirerek sağlamaktadır. Toplu gradyan inişi, rastgele gradyan inişinin tek bir güncelleme yapmasından farklı olarak her eğitim adımında değer güncellemeleri yaparak, daha hızlı sonuç elde etmeye yönelik çalışmaktadır.

Ağırlık, sapma ve aktivasyonlar için θ, öğrenme oranı ya da hızı için η, gradyan

öğrenme hızı için ∇_θ, maliyet fonksiyonu için J, x'i eğitim örneği y'yi çıktı sınıfı olarak temsil ederek ayrıldığında, her θ için hesaplanan parametre alınarak öğrenme hızı ve maliyet ile çarpılarak değişimin bulunmasıyla rastgele gradyan inişi matematiksel olarak Denklem 2.14 ile formulize edilmektedir.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x, y) \tag{2.14}$$

Rastgele gradyan inişi her adımda değerlerden bir tanesini rastgele seçerek güncellediği için daha gürültülü sonuçlar elde etmektedir ve daha çok sayıda tekrar gerekmektedir. Gürültü, yerel minimuma ulaşmak için izlenen akışta yeterli bir sürede gerçekleştiği sürece önemli değildir. Hesaplama maliyeti nedeniyle rastgele gradyan inişi toplu gradyan inişine göre daha yaygın kullanılmaktadır. Toplu gradyan inişinde her bir adımda toplu olarak değerler güncellendiğinden bir sonraki adıma kadar tek bir yönde yerel minimum yönünde iniş gösterilmekte iken, rastgele gradyan inişinde her adımda bir rastgele değer güncellendiğinden daha yüksek sapmalar göstermektedir.

Mini-toplu gradyan inişi, toplu gradyan inişindeki gibi tüm değerleri kullanmak yerine, eğitim setini b ile ifade edilen küçük boyutlu gruplara bölmektedir. Bu nedenle, her bir yinelemede YSA model değerlerini güncellemek için mini-toplu b grupları kullanılmaktadır. Rastgele gradyan inişinde olduğu gibi mini toplu gradyan inişinde de sapmalar görülmektedir. Büyük veri değerleriyle çalışıldığında hesaplama maliyeti açısından rastgele gradyan inişi, direkt olarak yerel minimuma ulaşılmak istenen durumlarda toplu gradyan inişi, ikisinin de ortasında avantajlarını kullanmak adına mini toplu gradyan inişi tercih edilmektedir.

Momentum gradyan inişinde adımlar arasındaki salınımı azaltmakta ve problem uzayındaki tahmin verileri arasında çok büyük farkların olduğu durumlarda daha hızlı ve isabetli sonuçlar almak için ortaya konulmuş bir yaklaşımdır. Momentum ile gradyan inişinde modeldeki ağırlıklar güncellenirken hali hazırdaki optimizasyon adımındaki gradyanını ve bir önceki gradyanı alarak minimuma ulaşma noktasına daha hızlı yaklaşmaya yardımcı olmaktadır. Değer farklarının ani değiştiği problem uzaylarında sonuca yaklaşma daha hızlı gerçekleşmektedir. Gradyan inişi Denklem 2.15 ile ifade edilirse θ, $\theta = \theta - v_t$ şeklinde hesaplanarak bir önceki gradyan inişiyle momentum ile hızlanarak minimuma yakınsamaktadır.

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta; x, y) \tag{2.15}$$

Adagrad (Duchi, Hazan ve Singer, 2011) hiper parametreler ile öğrenim hızının belirlenmesinde yaşanan sorunlara karşı, öğrenim hızının her problem uzayına uygun uyarlanabilir hesaplanmasının maliyetini hafifletmek üzere ortaya konulmuştur. Güncellenecek değer θ, öğrenme hızı η, sıfıra bölünme sorununu gidermek için ε, işlem yapılacak matrisi belirtmek için I, t anındaki gradyan tahmini g_t olarak alındığında matematiksel olarak her bir Adagrad güncellemesinde kullanılan matematiksel formül Denklem 2.16 ile ifade edilmektedir. T anındaki gradyan tahmini matematiksel olarak Denklem 2.17 ile ifade edilmektedir. G_t gradyanların dış çarpımlarının toplamı matrisi ise Denklem 2.18 ile formülize edilmektedir. Optimizasyon işlemi sonrasında, değer güncellemesinde tam matris kullanılabilir olmasına rağmen, derin öğrenme gibi çok boyutlu matrislerde maliyetlidir. Bu yüzden karekök ve matrisin köşegeninin kullanılmasıyla hesaplama kolaylıkla yapılabilmektedir.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\varepsilon I + \operatorname{diag}(G_t)}} \cdot g_t \tag{2.16}$$

$$g_t = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathscr{L}\left(x^{(i)}, y^{(i)}, \theta_t\right) \tag{2.17}$$

$$G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^\top \tag{2.18}$$

Adagrad algoritması, öğrenme hızını çok boyutlu ve çok değerli modellerde hızlı düşürmektedir ve seyrek değerli modellerde yavaş düşürmektedir. Öğrenme hızının çok hızlı düştüğü durumlarda ve Adagrad'a bağlı olarak gradyanların kümülatif birikmesiyle, öğrenme oranı çok hızlı sıfıra yakınsayabilmekte ve modelin öğrenemeyeceği bir durumda kalmasına sebep olarak modelin başarımını etkileyebilmektedir.

RMSprop, ortalama karekök yayılımı anlamına gelmektedir. Adagrad'dan farkı, gradyanların toplamını değil, üssel olarak azalan ortalama gradyanları ile çalışmasıdır. RMSprop, Adagrad algoritmasının momentum ile çalışan yeni bir versiyonudur. RMSprop'ta tüm geçmiş gradyanlarla değil, en son hesaplanan gradyanlarlarla hesaplama gerçekleştirilmektedir. RMSprop'un öğrenme hızı Adagrad'tan yavaş değişmektedir. Adagrad'tan yavaş değişen öğrenme hızına rağmen yerel minimuma yaklaşması daha hızlı olmaktadır. Güncellenecek değer θ, öğrenme hızı η, sıfıra bölünme sorununu gidermek için ε, t anındaki gradyan tahmini g_t olarak alındığında RMSprop matematiksel olarak Denklem 2.19 ile ifade edilmektedir. Yeni parametre olan E, momentum γ şeklinde ifade edildiğinde, Denklem 2.20 ile formülize edilmektedir.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\varepsilon + E\left[g^2\right]t}} \nabla J\left(\theta_{t,i}\right) \tag{2.19}$$

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

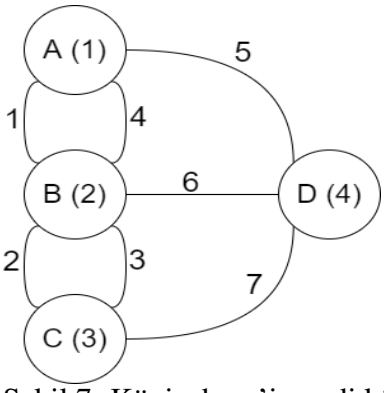
Adagrad'dan farklı olarak RMSprop ile öğrenme hızı η'nın öğrenmeyi gerçekleştiremeyecek kadar azalmasına engel olunmaktadır. RMSprop minimuma erişmişken, moment'in aynı adım sayısında minimuma erişmede geride kalması söz konusudur.

Adam (Kingma ve Ba, 2014), uyarlanabilir moment tahmininin kısaltması olarak, gradyan inişlerinde önceki momentlerin karesinin ortalaması alınarak ve tahminlenerek öğrenme hızlarını hesaplamaktadır. Adam, diğer optimizasyon algoritmalarıyla karşılaştırılacak olursa en iyi performansı göstermektedir. Adam'da hem momentum hem de uyarlanabilir öğrenme hızı ile daha hızlı yakınsama sağlamaktadır. Moment ve öğrenme hızını uyarlayabilir yaparak moment ile RMSprop algoritmalarını içeren bir yaklaşımdır. Adam öğrenme hızı η, sıfıra bölünme sorununun önüne geçmek için ε, önceki eğimlerin üssel kareleri alınarak hesaplanmış değer için v kullanıldığında matematiksel olarak Denklem 2.21 ile ifade edilmektedir.

$$\theta_{t+1} = \theta_t - \frac{\eta \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.21}$$

2.2. Çizge Teoremi

Çizge teoremi Königsberg köprüleri problemini çözmek için ortaya atılmıştır ve daha sonra matematiğin bir dalı haline gelmiştir. Çizge teoreminde durumlar birer düğüm ve birbiri arasındaki ilişki ise çizgilerle temsil edilmektedir. Düğümler köşe, ilişkiler ise kenar olarak adlandırılmaktadır. Königsberg şehrinde iki nehrin arasındaki köprüler yalnızca bir kez geçilmesi şartıyla yürüyüş yapılabilirliğinin sorgulanması meşhur Königsberg köprüleri problemini oluşturmaktadır. Eular problem temsilini nokta ve çizgilerle ifade ederek bir çizgeye dönüştürmüştür. Eular, köprüleri yalnızca bir kez kullanarak böyle bir yürüyüş gerçekleştirilemeyeceğini bulmuş ve çizgenin temel kurallarını oluşturmuştur. Çizgeler ile matematiksel sistem ve modellerin topolojilerinin temsili güçlendirilmiştir. Şekil 7'de Königsberg'in köprüleri sayılar ile ifade edilmiş ve ayak basılan yerler yani durumlar ise harflerle ifade edilmiştir.

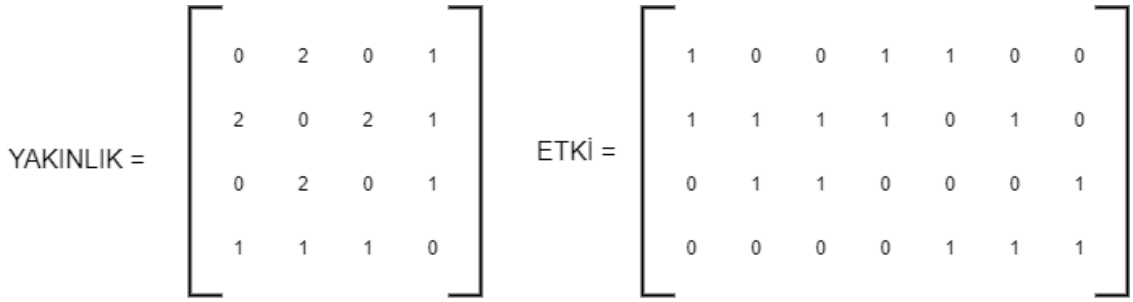


Şekil 7. Königsberg'in yedi köprü problemindeki köprüler ve adımlar

Çizgelerde ilişkiler çift yönlü olabileceği gibi tek yönlü de olabilmektedir. Şekil 2'deki temel yapay sinir ağının temsili, tek yönlü ilişkiler ile ifade edilen çizge ile gerçekleştirilebilmektedir. Çizgelerde her bir köşenin derecesi vardır, dereceler sahip oldukları kenar sayılarını ifade etmektedir. Örneğin Şekil 7'deki çizgede D köşesinin derecesi üçtür. Eğer bir köşe kendine dönen bir kenara sahipse kenar döngü olarak adlandırılır. Çizgelerde bir köşeden diğer köşeye ulaşmaya yürüyüş denmektedir. Örneğin Şekil 7'de B köşesinden D köşesine sırasıyla 3 ve 4 kenarlarını izleyerek bir yürüyüş yapılabilir.

Çizge üzerinde her kenarı içeren bir yürüyüş yapılırak aynı köşeye erişilebiliyorsa bu çizgeye Eular çizgesi denmektedir. Yürüyüşün başladığı köşeden aynı köşeye dönerken her köşenin kullanılmasına gerek yoksa bu çizgeye Hamilton çizgesi denmektedir. Çizgeler her zaman tam bağlı olarak ifade edilmezler, bağlı çizge ve bir çok parçadan oluşan bağlantısız çizgeler olmak üzere bağlarına göre iki çizge türü bulunmaktadır. Ağaç olarak adlandırılan çizgeler döngü içermeyen köşe ve kenarlara sahip çizgelerdir. Çizgeleri matrislerle ifade etmek mümkündür. Çizgeler yakınlık ve etki matrisleri olmak üzere iki matrisle ifade edilebilmektedir. Yakınlık matrisi köşeler arasında direkt olarak kaç kenarın olduğunun tutulduğu matristir. İki boyutlu yakınlık matrisinde, Şekil 7'deki çizgeyi ele alırsak dört köşeden oluştuğu için 4'e 4'lük bir matris gösterimi yakınlığı ifade etmek için yeterlidir. Yakınlık matrisi simetrik bir matristir. Şekil 7'deki çizgede A'yı 1, B'yi 2, C'yi 3, D'yi 4 kabul edersek, 1. köşe ile 2. köşe arasındaki yakınlık matrisi hücresindeki değer 2 olmalıdır. Yakınlık matrisi gibi etki matrisi de iki boyutludur. Etki matrisinde, yatay eksen indeksine i dikey eksen indeksine j dersek, i köşeleri, j ise kenarları temsil etmektedir. Dikey ve yatay kenar temsil isimlendirmesiyle birlikte Şekil 7'deki çizgenin etki matrisi 4'e 7'lik bir matristir. Şekil 7'deki çizgeyi etki matrisinde ifade edersek, i ekseni köşeleri j ekseni ise kenarları temsil eder. Örneğin A köşesini temsilen i1, 1 kenarını temsilen j1 indislerindeki değer 1 olacaktır. Şekil 8'de Königsberg köprü problemindeki çizgenin

yakınlık ve etki matrisleri gösterilmiştir.



Şekil 8. Königsberg çizge temsilinde kullanılan etki ve yakınlık matrisleri

Yapılan çalışmada yapay sinir ağlarının veri temsilinin çizge teoremiyle güçlendirilmesi amaçlanmıştır.

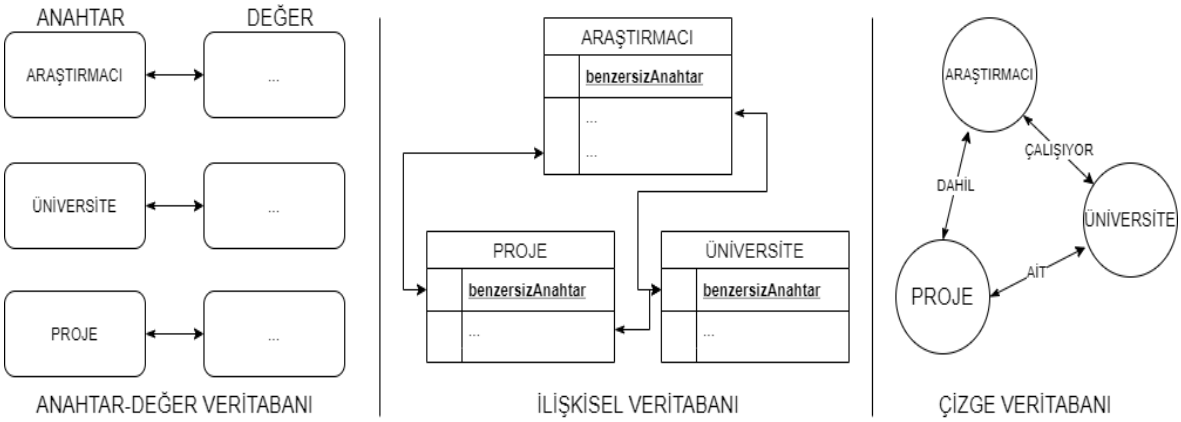
2.3. Çizge Veri Tabanları

Çizge veri tabanları, yer, kişi, nesne veya ilgili verileri köşeler olarak, verilerin arasındaki ilişkileri kenarlar olarak niteleyen, çizge veri modelini kullanan NoSQL tabanlı veri tabanlarıdır. NoSQL, sabit bir şema gerektirmeyen, birleşmeleri önleyen ve ölçeklendirmesi kolay olan ilişkisel olmayan bir veri tabanı yönetim sistemidir. NoSQL veri tabanı kullanmanın amacı, çok yönlü veri depolama ihtiyaçları olan dağıtılmış veri depoları içindir. NoSQL, büyük veri, veri ilişkilerinin çok önemli olduğu ve gerçek zamanlı web uygulamaları için kullanılır. Örneğin, Twitter, Facebook, Google gibi şirketler her gün terabaytlarca kullanıcı verisi toplar ve topladığı bu büyük verilerin tutulması ve temsilinde çizge veri tabanlarını kullanır. NoSQL veri tabanı "sadece SQL değil" anlamına gelecek şekilde kısaltılmıştır. NoSQL konseptini ilk olarak Carlo Strozzi tarafından ortaya koymuştur.

Geleneksel ilişkisel veri tabanları RDBSM olarak adlandırılmaktadır. İlişkisel veri tabanları, verileri depolamak, almak ve değiştirmek üzere yapısal olması zorunlu olan SQL sözdizimini kullanır. Bunun yerine, bir NoSQL veri tabanı sistemi, yapılandırılmış, yarı yapılandırılmış, yapılandırılmamış ve çokbiçimli verileri depolayabilen çok çeşitli veri tabanı teknolojilerini kapsar. NoSQL veri tabanlarında veri şemasının statik olmamasından dolayı dökümanlar kolayca değiştirilebilir ve yeni alanlar kolayca eklenerek çıkartılabilir. NoSQL veri tabanlarında büyük hacimde veriler tutabilir ve hızlı şekilde verilerin işlenmesine olanak sağlar. NoSQL verileri tutma şeklinden dolayı kolayca ölçeklenebilirdir, bu yüzden yüksek performanslı sunuculara ihtiyaç duymaz. NoSQL'in erişilebilirliği bir veride hata yaşandığı taktirde işleme devam edebilmesinden dolayı RDBSM veri tabanlarına oranla daha yüksektir. Çizge teoremiyle hatalı verinin işlenemeden de

aşılabilmesine olanak sağlayabilmektedir. Verileri anahtar değer olarak tuttuğundan ilişkisel veri tabanlarına oranla daha hızlıdır.

NoSQL anahtar veri, döküman tabanlı, sütun tabanlı, çizge tabanlı olarak dörde ayrılmaktadır. Çalışmada kullanılan veri tabanı olan Neo4j veri tabanı çizge tabanlı NoSQL olarak geçmektedir. Neo4j'nin sorgulama dili Cypher olarak adlandırılmaktadır. *Neo4j Graph Platform* (2007), günümüzde en çok kullanılan çizge veri tabanı olmuştur. Şekil 9'da veri tabanları, araştırmacı, proje ve üniversite varlıkları ve arasındaki ilişkiler ele alınarak gösterilmiştir. Şekil 9'da gösterilen anahtar-değer veri tabanları bir şemadan bağımsız olarak temsil edecekleri varlığı bir anahtarla ifade ederek, değerlerine bu anahtar vasıtasıyla erişimi sağlanması yöntemiyle çalışır. Veri ilişkilerini anahtarları aracılığıyla kurar ve dinamik şema özelliğini bu şekilde kazanır. Derin ilişki aramalarında veya örüntü bulmada düşük performans gösterir. İlişkisel veri tabanları varlıkların isimlerini tablolara atfeder ve ilgili varlıkları tablolarda tutar. Tablolar varlık başına dinamik değildir. Tablolarda yapılan bir değişiklik diğer tüm varlıklarda da bir özellikmiş gibi eklenir, bu nedenle dinamik bir şemadan bahsetmek mümkün değildir. Varlık ilişkilerini tablo üzerinde tutulan benzersiz anahtarlar ile diğer tablolara işaret ederek yapar. Derin ilişki ve örüntü aramalarında düşük performans sergiler. Çizge veri tabanlarında varlık başına bir temsil olduğundan varlığın birebir temsili ve ilişkilerinin temsili mümkündür. Varlıklar genel olarak isimlendirilmek ve gruplanmak istediklerinde etiketler ile isimlendirilirler örneğin "ARAŞTIRMACI" bir etikettir, aynı etikete sahip olan tüm varlıklar aynı özelliklere sahip olmak zorunda değildir. Çizge veri tabanlarının bu yaklaşımıyla hayattaki bir çok temsil birebir gerçeklenebilmektedir. Diğer veri tabanlarından farklı olarak çizge veri tabanlarında ilişkiler de etiketlerle gruplanabilmekte ve istenilen varlıklar arasında anlamsal ilişkiler temsil edilebilmektedir. Çizge veri tabanlarında ilişkilerde de veri tutulabilmektedir, bu özelliğiyle YSA'lar gerçeklenirken katmanlar ve nöronlar arasındaki ilişkilerde yapılan uygulanacak ağırlıklar ve katsayılar ilişkilerde de tutulabilir. Çalışmanın ilerleyen kısımlarında ilişkilerde ve nöronlarda model verilerinin tutulmasının yaratacağı avantaj ve dezavantajlar konusu işlenmiştir.



Şekil 9. Veri tabanları ve veri temsil şekilleri

2.4. Önceki Literatür Çalışmaları

Günümüzde çeşitli uygulamalarda çizge yapısı sıklıkla kullanılmaktadır. Sınıflandırma konusunda çizge yapısı temsiliyle başarıya ulaşmış çalışmalar bulunmaktadır. YSA verileri temsilleri gereği çizge yapılarında gösterilebilmektedir. Yapay sinir ağlarında sinir hücreleri çizge veri tabanlarındaki düğümlerle ve yapay sinir hücrelerinin arasındaki bağlar ise çizgi veri tabanlarındaki ilişkilerle ifade edilebilmektedir. Sosyal medya verilerinde, nesneler ile nesneler arasındaki ilişkilerin yoğun önemde olduğu alanlarda, karmaşık ağların, doğal ilişkilerinin tutulmasının önem arz ettiği durumlarda çizge veri tabanlarının kullanımı oldukça yüksektir. Verinin belirli yapılar dışında tutulmasının gerekli olduğu, yapı bağımsız veri tabanı çözümlerinde, çizge veri tabanlarının kullanımı önemli bir yer tutmaktadır. Canlı ve sürekli değişiklik gösteren veri ortamında sınırlandırma yapmak geçmişte olduğu gibi değişmez katı yapılardaki veri sınıflandırmalarından farklı ele alınmalıdır. Günümüzde en popüler ve en büyük geliştirme topluluğuna sahip çizge veri tabanı Neo4j'dir (Guia, Soares ve Bernardino, 2017).

2.4.1. Çizge Sinir Ağları

Literatürde YSA ve çizge konusunu birlikte ele alan çeşitli çalışmalar yapılmıştır. Gori, Monfardini ve Scarselli (2005) kendi oluşturdukları çizge sinir ağları modelini duyurmuş ve sinir ağlarının çizgeler üzerinde temsil edilebileceğini sunmuşlardır. Bu yeni modelin geliştirilmesiyle çizge sinir ağ modelinin testi için örnek problemler oluşturmuşlardır. Bu problemler bağlantı bazlı, sınıf bazlı problemler ve genel problemler olarak üçe ayrılmaktadır. Yeni model çalışma problemlerinin hepsinde başarılı sonuçlar elde etmiştir.

Bianchini, Gori ve Scarselli (2005) yaptıkları çalışmada sayfa değeri hesaplaması probleminde çizge sinir ağlarını kullanarak hesaplamanın çok kolay ve anlaşılır olacağını

göstermişlerdir. İnternet sayfaları ve aralarındaki ilişkilerin doğal olarak çizgelerle ifade edilebilmesinin ve değer hesabı yapılırken temsil gücünü ve işlem gücünü arttırdığını gözlemlemişlerdir. Scarselli, Gori, Tsoi, Hagenbuchner ve Monfardini (2008) yeni bir çizge sinir ağı modeli kavramını ortaya atmışlardır. Öklid uzayında çizgeleri işleyerek sinir ağının gerçeklenebileceği ve öğreticili öğrenmeyi teorik olarak yapabileceği modeli öne sürmüşlerdir. Ayrıca hesaplama maliyetini çıkartarak genelleştirilmesiyle alakalı çalışmalar yapmışlardır. Bu yaklaşımlarıyla gizli katmanlardaki ilişkileri aydınlatmayı amaçlamışlardır.

Scarselli, Gori, Tsoi, Hagenbuchner ve Monfardini (2009) çizge sinir ağı modeli kavramını ortaya atarak ilk kez çizge veri temsilli yapay sinir ağlarını literatüre kazandırmışlardır. Çizgelerin hem düğüm hem de arasında kurulabilecek ikili ya da tekli bağlantıların öklid uzayında sinir ağlarını temsil edebileceğini ifade etmişlerdir. Yapılan çalışma ile çizgeler üzerinde gerçeklenen yapay sinir ağlarının hesaplama maliyetleri ve genelleştirme problemlerinin üstünde durmuşlardır. Yaptıkları çalışmada bilinmeyen ilişkilere sahip çizgeler üzerinde hesaplama gerçekleştirilmesi, sosyal ağ gibi çevrimiçi ve dinamik olan sürekli büyüyen çizgelerde sunulan modeli işlevsiz kılmaktadır. Yapılan çalışmada statik çizge modelleriyle öğrenim gerçekleştirilmiştir. Yaptıkları çizge veri temsili ile yapay sinir ağlarının gerçekleştirimi ile yaptığımız çalışmaya esin kaynağı olmuşlardır. Gizli katmanlardaki belirsizliklere ışık tutma amacıyla sinir ağlarındaki veri temsil ve işlemenin önemini vurgulamışlardır.

Grover ve Leskovec (2016) öğrenme algoritmalarını çizgelerde gerçekleyen, genelleştirilebilir bir algoritma çatısı ortaya atmışlardır. Geliştirilen node2vec yöntemiyle çizge düğümlerinde devamlı özellik öğrenmesini yapacak bir yöntem geliştirerek, yapay sinir ağlarının çizge üzerinde gerçeklenebileceği konusunda katkıda bulunmuşlardır. Derin öğrenmenin temel adımlarından olan devamlı özellik temsilleri ile çizgelerde derin öğrenme gerçekleştirme yapılabilmesi için bir altyapı ortaya koymuşlardır.

Neuhaus ve Bunke (2004) yaptıkları çalışmada çizge tabanlı sınıflandırma algoritmaları için maliyet düşürücü bir hesaplama yöntemi sunmuşlardır. Yaptıkları bu çalışma ile çizge üzerinde sınıflandırma yapılırken düzenleme mesafesi hesaplarını geliştirmişlerdir. Neuhaus ve Bunke (2005) yaptıkları çalışmada çizge üzerindeki sınıflandırma çalışmalarını ele alarak parmak izi görsel özelliklerinin çıkarımını yaptıktan sonra çizgeye aktararak oluşturdukları mesafe algoritmasıyla benzerliklerini ölçmüşler ve sınıflandırma problemine çözüm geliştirmişlerdir. Çalışmada görsel özellik çıkarımlarının ilişkili çizgede tutulmasıyla kazanılan temsil ve işlem gücünü sınıflandırmada kullanmışlardır.

Kipf ve Welling (2016) yaptıkları çalışmada ESA ağlarının çizge üzerinde daha verimli şekilde yarı öğreticili öğrenme gerçekleştirmişler ve gizli katmandaki temsilin arttığına dikkat çekmişlerdir. Yaptıkları çalışma belirli modellere özgü olmakta, model verileri bir veri tabanında veya serviste değil statik olarak tutulmakta ve işlenmektedir.

Yao ve Holder (2015) çizge sınıflandırması konusunda statik çizgelerde yapılan sınıflandırma eksiklerine işaret edip, çizge verilerinin doğası gereği sürekli değişken olduğuna dikkat çekmişlerdir. Sürekli büyüyen ve değişen büyük çizgelerde destek vektör makinesi yöntemini Weisfeiler-Lehman yöntemiyle beraber kullanarak sınıflandırma problemini çözmek için bir yaklaşım geliştirmişlerdir. Ayrıca çizgede yeni eklenen düğümlerin daha yakın ve etkili komşularının tespitiyle daha iyi çalışan bir çizge sınıflandırması yaklaşımı ortaya koymuşlardır. Çizge üzerinde sınıflandırma yapmanın çizge büyüdükçe bellek limitine takılacağını bu yüzden tüm çizgenin her seferde işlenemeyeceğine işaret etmişler ve en son işlenen veri yığınına göre artan sınıflandırma yapmışlardır.

2.4.2. Problem Öğrenmede Çizgelerin Temsil Gücü

Görsellerde insan duruş tahminlemesi, çizgelerin ve sınıflandırmanın birlikte kullanımının arttığı diğer bir konudur. Bu konuda Bergtholdt, Kappes, Schmidt ve Schnörr (2010) bilgisayar görüsünde obje tespiti için çizgeleri kullanarak sınıflandırma yapmışlardır. Görseller üzerinde insan duruşunun tespitinde vücut bölümlerini çizgeler üzerinde temsil ettikten sonra üzerinde tahminleme ve sınıflandırma yapmışlar ve kullanılacağı alana göre özelleşen dört model ortaya sunarak bir çatı geliştirmişlerdir. Bu çalışmada tam bağlı çizge kullanımıyla gürültünün sistemdeki etkilerinin azaltımını göstermişler ve çizgenin görseller üzerindeki temsil gücünün arttırılmasıyla sınıflandırmanın da verimini arttırmışlardır. Görsellerdeki insan duruş tahminlemesinde ESA modelini ayrık çizgelerle oluşturan Tompson, Jain, LeCun ve Bregler (2014) tahminleme probleminde başarım elde etmişlerdir. Geliştirilen modelin tahminleme sonuçlarını gerçek zamana yakın elde etmesinden dolayı çalışmanın farklı uygulama alanlarında kullanıma uygun olduğunu bildirmektedirler.

Ginde ve diğerleri (2018) büyük veri görselleştirmesinde ve tutulmasında Neo4j'yi kullanmışlardır. Neo4jde tuttukları ve görselleştirdikleri bilimsel makalelerdeki yazar, dergi, enstitü, ülke özellikleri çizge düğüm ve ilişkilerine çevirerek aralarındaki gizli örüntüleri aydınlatmışlardır. Büyük verilerin anlamlı tutulmasında ve tahmin edilemeyen ilişki örüntülerinin gözlemlenebilmesinde çizge temsili örnek gösterilebilir.

Nekhaev ve Demin (2017) derin sinir ağlarında makine öğrenimi yaparken gizli katmanlardaki bilgilerin halihazırdaki temsil yöntemleriyle iyi anlaşılamadığını

vurgulamışlardır. Yaptığımız çalışmadan farklı olarak sinir ağlarının görselleştirilmesi ve temsilinde ele aldıkları konu verinin tutulması değil, derin öğrenmedeki gizli katmanlarda görsel özelliklerinin hangi nöronun ilgili olduğunun bulunmasıyla sağlamışlardır. Gizli katmanlardaki nöronların anlaşılamamasını vurgulamışlar ve modellerin daha iyi anlaşılması için bir yaklaşım ortaya koymuşlardır.

Bijari, Zare, Kebriaei ve Veisi (2020) metin madenciliğinde halihazırda kullanılan geleneksel yapıların, ilişkilerin bulunmasında ve kurulmasında yeteri kadar etkili çalışmadığını öne sürmüşlerdir. Cümle seviyesinde metin temsili yapan çizge tabanlı bir model ortaya çıkarmışlardır. Derin sinir ağlarında bu model beslenerek duygu durum sınıflandırması yapılması sağlanmıştır. Geleneksel metotlara göre karşılaştırma yaparak başarılı sonuçlar elde etmişler ve modelin genelleşitirilebilir olduğunu öne sürmüşlerdir. Veri setleri, model üzerinde herhangi ön eğitilmiş kelime kalıplarına ihtiyaç duymadan çalışabilmektedir. Yapılan diğer çalışmalarda olduğu gibi araştırmacılar modeldeki gizli karakteristikleri çizge temsili ile yakalayarak modelleri aydınlatılmasını sağlamışlardır.

Perozzi, Al-Rfou ve Skiena (2014) DeepWalk isimli bir model öne sürerek sosyal ağlardaki ilişkileri çizge girdisiyle çalıştırarak öğrenim gerçekleştirmişlerdir. Çok sınıflı sınıflandırma problemleri üzerinde modellerini gerçeklenerek öğrenme sağlamışlardır. Çizge modelinin sınıflandırma problemlerindeki temsil gücünü ortaya koymuşlardır. Sundukları modelin çevrimiçi veriler üzerinde öğrenme çalışılabileceğini göstermişlerdir. Çizge modelinin öğrenim her tür veri ve alana uyarlanabileceğini öne sürmüşlerdir.

Zednik (2019) yaptığı çalışmada Humphreys (2009) yaptığı çalışmaya referans vererek kara kutu problemini saydamlık üzerinden açıklamıştır. Saydamlık "Bir x sisteminde çalışan ajanın t zamanında bağlı olduğu tüm varlıkların durumunu bilememesi" olarak tanımlayan Humphreys'in bu çıkarımından yola çıkarak yapay sinir ağlarının saydam olmadığı sonucuna varılabilmektedir. Saydamlığı sağlanamayan yapay sinir ağlarının sadece çıktılar ve katmanlara bağlı sayılması ile problemi çözülen konuların tam olarak modeller tarafından öğrenilemeyeceği sonucuna varılabilmektedir.

Frosst ve Hinton (2017) yaptıkları çalışma ile derin öğrenme ağlarını karar ağaçlarına çevirerek derin ağ modellerinin temsillerini arttırmışlardır. Gizli katmanların anlaşılır hale gelmesini sağlamışlardır. İlk gizli katman ile son gizli katmanda aktivasyon fonksiyon çıktıları gözlemlenebilmekte olmasına rağmen, arada olabilecek onlarca gizli katmandaki sonuçların gözlemlenemeyeceğini dile getirmişlerdir. İstatistiksel değerlendirmeler ile yeterli başarımda sonuçlar elde etmenin derin ağların ve de insan beyninin öğrenme mekaniğinin incelenmesini kısıtladığını öne sürmüşlerdir. Ortaya koydukları hafif karar ağaçları dönüşümü ile veri üzerinde eğitilen bir derin sinir ağ modelinin genelleştirilmesine

göre daha genelleştirilebilir bir model elde ederek eğitim yaptıkları ağaç ile farklı alandaki problemlerin çözüm başarımının daha yüksek olacağını ortaya koymuşlardır. Kurdukları modeli eğitilmiş derin sinir ağı modellerin çevrimi ile kara kutu ve temsil problemini çözmek üzere kurduklarından modeli sıfırdan oluşturmamışlardır. Bu tez çalışmasında Frosst ve Hinton (2017) çalışmasıyla benzerlik olarak hali hazırda eğitilmiş ağların önerilen sisteme aktarımı ve işlenebilmesi sağlanmaktadır. Tez çalışmasında farklı olarak sunulan sistem üzerinde sinir ağını oluşturma ve modelin oluşturulmaya başlandığı andan itibaren tutulmasında, işlenmesinde çizge temsilinin avantajlarının kullanılması sağlanmıştır. Karar ağaçları, çizgelerin sonsuz döngü oluşturmayacak biçimdeki hallerine örnek olarak gösterilebilmektedir.

Hamilton, Ying ve Leskovec (2017) GraphSAGE çalışmasını geliştirerek, çizgeler üzerinde öğrenme yaparak genelleştirilebilir çıktılar vermeyi amaçlamışlardır. Çoğu aynı problem uzayında yapılan çalışmanın genelleştirilemeyeceği ve belirli bir çizge üzerinde öğrenme yaptığını öne sürmüşlerdir. GraphSAGE'nin öğrenme yapabilmesi için öğreneceği çizgenin hali hazırda oluşturulmuş olması gerekmektedir. Çizge üzerindeki düğümler ve komşulukları çıkararak özellik çıkarımları yapmakta, topladığı komşuluk bilgileri ile çizgenin bağlamının ve düğümlerin etiketlerinin tahminlemesini yapmaktadır. Hiç görülmemiş çizgelerde öğrendiği ilişkiler ile düğüm ve ilişki etiketlerini tahminleyebilmektedir. Sunulan tezden farklı olarak sadece hali hazırda olan çizgeler ile öğrenim gerçekleştirebilmektedir. Öğrenimini gerçekleştirdiği özelliklerden farklı özelliklerdeki düğümler içeren çizgelerde çalışmamaktadır.

Hamilton (t.y.) çizge temsilli öğrenme çalışması ile çizge sinir ağı hakkında literatürde bulunan tüm kaynakları toplayarak, derleme bir kitap oluşturmuştur. Hamilton çalışmasını yaparken çizge sinir ağı alanında bir çok öncü gelişmenin olmasıyla gelişimi hızlanan bir konu olduğuna dikkat çekerek hızlı gelişen literatürü tam olarak sunamadığını belirtmiştir. Hamilton çizge temsilli öğrenmenin ilerletilebilmesi için ağlardaki derin ilişkilerin anlaşılmasının yeni mimari ve çalışmaların artmasıyla olacağını öngörmektedir. Çalışma alanını kapsamlı şekilde inceleyen Hamilton'ın çalışmasında yeni uygulama alanları açacağını belirttiği çizge temsilli öğrenme ile tez çalışmamızda hedeflenen kazanımlar ve hedefler öngörüleri doğrular niteliktedir.

2.4.3. Veritabanlarında Saklanan Yapay Sinir Ağları

Yapay sinir ağlarının teorideki birebir temsilini saklamak için veritabanları kullanılmaktadır. Nesne yönelimli programlama prensibiyle çalışmasından dolayı ilişkisel veritabanları ilk tercih edilen veritabanları olmuştur. Çalışmaların yapıldığı dönemde çizge

veritabanları bulunmamaktadır. Schikuta ve diğerleri (1996) önerdikleri NeuDb çalışması ilişkisel veritabanında YSA saklayan ve işleyen ilk çalışmalardan biri olmuştur. Postgres ilişkisel veritabanında YSA modelinin tutulması için SQL sorguları çalıştırılmaktadır. Eğitim için veritabanı prosedür fonksiyonları çalıştırılmaktadır. Araştırmacı modelin eğitim ve test sürecini gözlemleyememektedir. Veritabanında tutulan YSA'ların simüle edilerek araştırmacıya gösterilme ihtiyacı oluşmuştur. Yahia ve Elsawi (t.y.) sundukları çalışma ile YSA modellerini ve eğitim prosedürlerini ilişkisel veritabanında tutmuştur. NeuDb çalışmasında olduğu gibi eğitim süreçlerini işleyecek kodlar veritabanı prosedürlerinde tutulmaktadır. Yaptıkları çalışmada mantık kapılarını eğitecek bir önyüz programı geliştirerek araştırmacıların kullanımına sunmuşlardır.

Schikuta, Brunner ve Schultes (1998b) YSA'ları MS Access ilişkisel veritabanında tutmayı önererek araştırmacıların YSA eğitimlerini izleyebileceği NeuroAccess çalışmasını duyurmuştur. Bu çalışmada eğitim prosedürleri statik olarak veritabanında tutulmaktadır. Çalışmada eğitim sürecinin gözlemlenebileceği simülasyon yazılımı geliştirilmiştir. Yazılım ile hata eğrisi, ağ yapısı, eğitim adımlarında elde edilen sonuçlar gösterilerek araştırmacıların gözlemi sağlanmıştır. Schikuta ve Glantschnig (2007) yaptığı NeuroOracle çalışması ile YSA modellerini ve eğitim süreçlerini Oracle veritabanlarında saklamıştır. Önceki çalışmalarda olduğu üzere modelleri oluşturmak için veritabanı sorguları yazılmalıdır. Yapay sinir ağlarını ayrıca bilgi tabanlı veritabanlarında tutan çalışmalar mevcuttur (Schikuta, 2008). Uzman sistemlerde bilgi tabanları (knowledge-based) veritabanları sıklıkla kullanılmaktadır. YSA'larda öğrenilen kurallar bilgi veritabanını beslemektedir.

YSA'ların veritabanlarında tutulmasıyla araştırmacılar dosya dönüşümleri yapmadan YSA'lara çevrimiçi erişebilmektedirler. YSA modeli oluşturmak eğitim ve testlerini gerçekleştirmek için araştırmacılar genellikle kendi yazılımlarını geliştirmektedir. Araştırmacıların kendi geliştirdikleri yazılımlar ile genelleştirilebilir bir YSA ortamı oluşturmaktan uzaklaşılmaktadır. Schikuta (2002) geliştirdikleri NeuroWeb yazılımı ile araştırmacıların yapay sinir ağlarını saklayan ve işleyen bir sunucuya bağlanarak sinir ağlarını oluşturup eğitim yapabilecekleri ortamı oluşturmuşlardır. Bu çalışma ile yapay sinir ağları ile ilgilenen herkesin araştırmacılar kadar konuya hakim olmadan da kendi ağlarını deneyebileceği genelleştirilebilir ve geliştirilebilir yazılımların ilk adımlarını atmıştır.

Huqqani, Li, Beran ve Schikuta (2010) NeuroAccess çalışmasındaki temeller ile YSA modeli oluşturma ve eğitim simulasyonunu bulut ortamında gerçekleyecek N2Cloud yazılımını sunmuşlardır. Büyük endüstriyel uygulamalarda kullanılan ölçeklenebilirlik sunan bulut sunucu ortamında YSA işlenerek eğitim hızları arttırılmaktadır. N2Cloud

kullanmak için araştırmacılar uygulama indirerek bulut ortamında YSA oluşturup eğitebilmektedir. Schikuta ve Mann (2013) yaptıkları çalışma ile N2Cloud'tan farklı olarak YSA'nın hem ilişkisel SQL veritabanlarında hem de NoSql döküman tabanlı veritabanında tutulmasını ve işlenmesini sağlayan N2Sky yazılımını sunmuşlardır. Bu çalışma ile bulut ortamında araştırmacıların YSA saklama ve eğitmede kaynak paylaşımı yaparak büyük verisetleriyle talep üzerine (on-demand) çalışmasını sağamışlardır. N2Sky, YSA'ları saklama ve işlemeyi XaaS servis mimarisinde gerçekleştirmektedir. N2Sky ile araştırmacıların çalışma ortamları ve oluşturdukları YSA'lar kullanıcı üyelikleri ile ayrılmaktadır. Yapay sinir ağlarını nesne yönelimli olmasına rağmen kodlama yapılmadan oluşturulamamaktadır. YSA katman tasarımları ve bağları sorgularla gerçekleştirilebilmektedir. YSA modellerinin dağıtımının yapılabilmesi için genelleştirilebilir bir yapıya ihtiyaç duyulduğunu söylemişlerdir. Ontolojik olarak yapay sinir ağlarının temsiline ihtiyaç duyulduğunu ve YSA'ları birebir ifade edebilecek saklama yöntemlerinin gerektiğini gelecek çalışmalar için önermişlerdir.

2.5. Yapay Sinir Ağı Kütüphaneleri

Araştırmacılar yapay sinir ağlarını gerçekleyebilmesi için üst seviyeli diller ile çalışan yardımcı kütüphane ve yazılım çatılarına ihtiyaç duymaktadırlar. Yapılan çalışmada oluşturulan yazılım ile bu yardımcı kütüphaneler ve yazılım çatılarına alternatif olarak farklı bir yöntem ortaya koymaktadır. Genel olarak kabul edilen ve yaygınca kullanılan Keras, Tensorflow, PyTorch ve Caffe kütüphaneleri bu bölümde incelenmiştir.

2.5.1. Tensorflow

Tensorflow, Theano altyapısından oluşturulan Google'ın Google Brain Team ekibi tarafından (Abadi ve diğerleri, 2015) sembolik matematiksel işlemleri gerçeklemek ve yapay sinir ağlarının eğitim aşamalarında kullanılmak için geliştirdiği, sinir ağı araştırma ve geliştirme aşamalarında kullanılan dünyaca en yaygın yazılım çatılarından biridir. Gmail, Nvidia gibi veri yoğun işler yapan ürün ve şirketlerde kullanılmakta ve yeni araştırmalara bağlı olarak sürekli güncellenmektedir.

Tensorflow çatısında Python, Javascript, Go, CSharp, Java gibi dillerde geliştirme yapılabilmektedir. Python ile model geliştirme yapılması en yaygın kullanım şeklidir. Tensorflow ile paralel ve küme hesaplaması yapılabilmektedir. Tensorflow yapay sinir ağı modelleri statik olarak mobil cihazlarda çalıştırılabilmektedir. Tensorflow model eğitim, test ve dizaynında araştırmacının kodlama diline ve kütüphaneye hakimiyetinin yüksek olmasını gerektirmektedir. İnce ayarlamalar ve tüm model tasarımının iyi

gözlemlenebilmesi gerekmektedir.

Tensorflow ismi adındaki Tensör kelimesinden gelmektedir. Sayısal ifadeleri olması gereken fiziksel veriler tensör olarak adlandırılır. Sıfırıncı dereceden tensör olan skalar tensörler fiziksel ifadesi bir tek sayı ile ifade edilebilen tensör çeşididir. Fiziksel objenin ya da enerjinin skalar tensör ile ifadesine 5 kilogram, 90 km/s, 30 Celcius derece gibi örnekler verilebilir. Vektör tensörleri birinci derece tensörler olarak adlandırılmaktadır. Skalar tensörlerin üstüne bir de yönü temsil etmek için kullanılmaktadır. Örneğin batı yönünde 90 km/s ifadesi vektör olarak temsil edilebilmektedir. Skalar tensörler sadece sayı ile, vektörler sayı dizisi ile ifade edilmekte, üçüncü derece ve üstü tensörler ise sayı dizisi matrisi ile ifade edilmektedir. Üçüncü derece ve üstü tensörlerin diğer tensörler gibi özel bir ismi olmamakla birlikte, derecesi ile kullanımı yaygınca görünmektedir. Örneğin bilgisayar ortamında bir resmi, RGB uzayında 3 farklı ikinci derece tensör matrisi ile veya üçüncü derece bir tensör matrisi ile tanımlamak mümkündür. Tensorflow YSA model eğitimi yapılacak verileri ve modeli tensörler ile tutmakta ve işlemektedir.

Tensorflow çatısı model eğitim metriklerinin takip edilebileceği bir web arayüzünden oluşmaktadır. Tensorflow Cuda platformunu kullanarak Nvidia grafik kartları üzerinde YSA model hesaplamalarını daha hızlı gerçekleştirmeyi sağlayabilmektedir. Grafik kartlarında olduğu gibi işlemci üzerinde de model hesaplamaları yapılabilmektedir. Grafik kartı üzerinde işlemlerin yapılabilmesi için Python, Python'un gelişmiş matematiksel işlemlerde kullanılan kütüphanesi olan Anaconda, Nvidia grafik kartının Cuda'yı destekleyen versiyonunun kurulması, Tensorflow kütüphanesinin grafik işlemleri için olan kütüphanesi gerekmektedir. Nvidia grafik kartlarının hepsi Cuda'yı desteklememekte ve eski Cuda sürümlerini destekleyen kartların yeni özelliklerden mahrum kalmasına yol açmaktadır.

Tensorflow derin öğrenme ile ilgili özelleşmiş şekilde kullanılmakta, farklı sinir ağ modellerinde sınırlı destek vermektedir. Tüm özellikleri Python dili kullanılarak çalıştırılabilmektedir. Tensorflow döngü içeren modellere destek vermemektedir, döngü içeren YSA eğitim modellerinde kullanılamamasına yol açmaktadır.

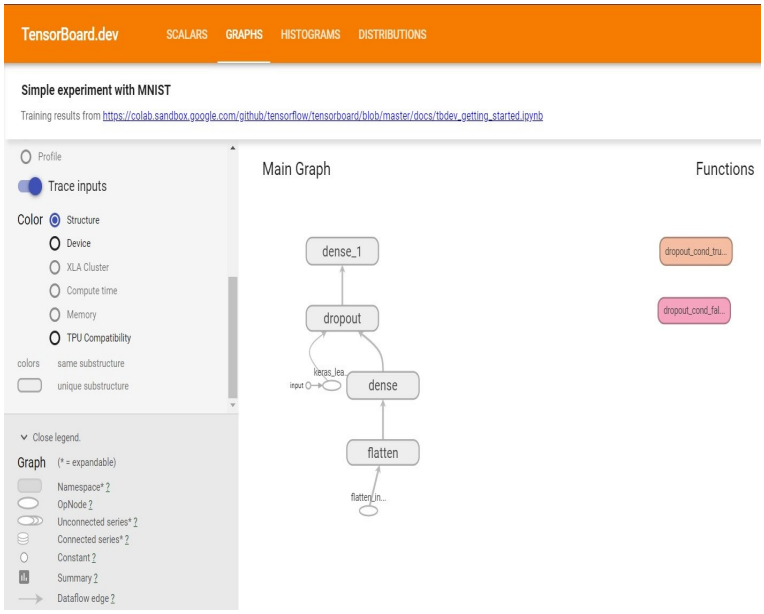
Tensorflow'da modeller eğitilirken değişkenler, veri dizileri gibi üst ve soyut veriler meta, ağırlıklar, sapmalar, gradyanlar ve diğer değer içeren veriler ckpt, modelin son hali ise checkpoint isimli ayrı ayrı dosyalarda tutulmaktadır. Eğitimin herhangi bir anında durması ya da hata alması durumunda, checkpoint adında modelin kontrol noktasını kaydetmek gerekmektedir, yoksa model bozulabilir ya da veri kaybı yaşanabilir.

Tüm model H5 (The HDF Group, 2000) ve Tensorflow'un SavedModel yöntemlerinden birisiyle çıktısı alınabilmektedir. Geliştirilen modelin son kullanım alanında çalıştırabilecek halde dönüştürülebilmesi için donmuş bir modele dönüştürülerek

diskteki bir dosya halinde taşınır olabilmektedir. Eğitim boyunca toplanan performans ve eğitim metrikleri log dosyalarında tutulmaktadır. Bu yaklaşım ile Tensorflow, eğitimin bitmesi için tüm öğrenim boyunca devam etmesi gereken uzun bir süreçte eğitimin herhangi bir zamanda durdurulması yaklaşımı yerine, kontrol noktaları ile modelin son durumu koruma ve eğitim sonundaki çıktıları statik ve donmuş modeller olarak aktarma prensipleriyle çalışmaktadır. YSA modeli yeterli doğruluğu ve isabetliliği gösterdiği düşünülen durumda, SavedModel olarak adlandırılan yaklaşım ile modelin son halini Protobuf binary dosyasında dışa aktarmaktadır.

Protobuf (Varda, 2008) diğer adıyla protokol tamponları yapısal verilerin tutulması için bir yöntemdir. Protobuf, IDL arayüz tanımlama dili olarak tanımlanmakta ve farklı dilde yazılmış programlar ve işletim sistemleri arasında anlaşılabilir ve dönüştürülebilir veri taşımak için kullanılmaktadır. Protobuf uzak prosedür çağrılarında sıklıkla kullanılmaktadır. Modelin tüm verilerini içeren byte akışının modelin kullanılacağı sistemde ve programlama dilinde işlenme şeklini tanımlayarak çalışmaktadır.

Tensorflow'un diğer donmuş ya da eğitimi tamamlanmış model formatı olan H5 formatı ile çıktı alınabilmektedir. H5 büyük verilerde hiyerarşik veri formatı ile verilerin tutulmasını sağlamaktadır. H5'te veriler dataset, group, attribute, type, property alanlarını içererek meta veriler üzerinden herhangi bir işletim sisteminin dosya sistemindeki gibi dosya yollarına bölünmektedir. Keras'ta ve Tensorflow'un mobil ve Javascript kütüphanesinde üzere eğitimi tamamlanmış YSA modelleri H5 formatında çıktıyı kullanabilmektedir. Şekil 10'da el yazısı tanıma veri seti üzerine eğitilmiş bir YSA modelinin statik hesaplama çizgesi, Tensorboard üzerinde gösterilmektedir.



Şekil 10. Tensorboard ile statik çizge gösterimi

Tensorflow'da kodlama aşamasında model statik bir hesaplama çizgesi olarak tanımlanmaktadır. Tensorflow'da tanımlanan çizge disk üzerinde dosyalarda tutulmaktadır. Oluşturulan çizgelerin Tensorflow geliştirme ortamı dışında işlenmesi ve izlenmesinin mümkün olmadığı çizgelerdir. Eğitimin yapıldığı sırada çizgelerdeki veriler ve ilişkilerdeki veriler gözlemlenememektedir. Tensorflow TensorBoard adında eğitim metriklerinin izlenebildiği, statik çizgenin görüntülenebildiği, histogram grafiklerine erişilebildiği ve ağ içindeki veri dağılımını gözlemlemek adına dağıtım grafiklerinin bulunduğu bir yazılım aracını içinde barındırmaktadır. Tensorflow'daki kod geliştirmeleri zamanda Google'ın Colab adındaki grafik kartı destekli bulut servisinde yapılabilmekte ve çalıştırılabilmektedir. Colab üzerinde birden çok araştırmacının beraber çalışacağı ortamda eğitimi tamamlanmış donmuş modelin yüklemesi yapılması gerekmekte bu da gerçek dünya problemlerinde gerçeklemesi zor bir çalışma ortamı oluşturmaktadır. Kod parçalarının değiştirilmesi ve model üzerinde tekrardan işletilmesi ile her seferinde sıfırdan model eğitimi yapılmaktadır. Sadece kodlama ile oluşturulabilir olan YSA katmanları işletilirken tensörler olarak hesaplamalara katılmaktadır. YSA meta modeli eğitilmiş modelin çıktısı verilmediği sürece saklanmamaktadır. Model topolojisinde ise katmanlar arasındaki ilişkiler gözlemlenememektedir.

Tensorflow'un dezavantajları, muadili olan diğer yazılım çatılarından yavaş çalışması, alt seviye kodlama gerektirdiği için araştırmacının Tensorflow'a hakimiyetinin fazla olmasını gerektirmekte olması, sadece Nvidia grafik kartlarında model eğitimini desteklemesi, tüm özelliklerinin kullanılabilir olması için Python yazılım diliyle oluşturulmuş kütüphanesinin kullanılması zorunluluğu, model döngü desteğinin olmaması, derin öğrenmede uzmanlaşmış ve odak noktası olarak çalışması ve daha basit modellerde yeterince kullanım kolaylığı sağlamamasıdır.

2.5.2. Keras

Chollet ve diğerleri (2015) tarafından ONEIROS (Açık Uçlu Sinir Elektronik Akıllı Robot İşletim Sistemi) projesi kapsamında geliştirilmiş ve geliştirilmeye devam eden sinir ağı yazılım kütüphanesidir. Keras diğer yazılım kütüphaneleri gibi bağımsız şekilde kendi yazılım ortamını oluşturmak yerine bir arayüz görevi görerek diğer yazılım kütüphaneleriyle entegre çalışan bir arayüz sağlamaktadır. Keras sunduğu sanallaştırma arayüzü ile Tensorflow, Microsoft Cognitive Toolkit, Theano, CNTK gibi yazılım kütüphaneleriyle entegre çalışabilmektedir. Keras Tensorflow'un 2.0 versiyonuyla beraber standart olarak kurulmakta ve çalışmaktadır.

Keras derin sinir ağlarının karmaşık ve yeni araştırmacıların derin öğrenmede

yaşayacakları zorluklar göz önüne alınarak geliştirilmiş yüksek seviyeli bir uygulama geliştirme arayüzü sunmaktadır. Sinir ağı modellerinin araştırmacılar tarafından yüksek seviyeli arayüz olan Keras ile gerçekleştirilmesiyle çalışmalarını daha hızlı ve sorunsuz gerçeklemesini amaçlamaktadır. Python programlama diline destek vermektedir.

Keras kullanıcı dostu, genişletilmesi ve geliştirilmesi kolay, makinelere nazaran insanların sinir ağlarını anlamasına odaklı prensipleri benimseyerek geliştirilmiştir. Sinir katmanları, maliyet fonksiyonları, optimizasyonlar, başlangıç şemaları, aktivasyon fonksiyonları, düzenleme şemaları gibi tek başına çalışabilecek Keras içindeki diğer modüllerle bir bağı olmayan modüllerden oluşarak yeni bir model oluşturulurken ihtiyaç duyulacak temel bileşenleri içermektedir. Sinir ağı öğrenimi konusunda Amazon, Apple, Nvidia, Uber, Microsoft ve Google gibi firmalarda kullanılmakta ve geliştirilmektedir.

Keras alt seviye işlemler olan konvolüsyon işlemi, tensör işlemleri gibi işlemleri kendi kütüphanesi içinde yapmamakta ve bağlı olduğu diğer yazılım kütüphanelerinde yapılmasını sağlamaktadır. Keras'ın sanallaştırma ve yüksek seviyeli işlemlere destek verme prensibinden dolayı araştırmacı alt seviye model değişiklikleri ve geliştirmeler yapmak istediğinde Keras'ı kullanamamaktadır. Bu durumda araştırmacıların oluşturdukları modellerde Tensorflow gibi alt seviye işlemlerin desteklendiği yazılım çatılarını kullanmaları gerekmektedir. Keras Sequential ve Functional olmak üzere iki tür uygulama geliştirme arayüzü sunmaktadır. Sequential arayüz ile doğrusal biçimde eklenecek sinir katmanlarının oluşturulduğu arayüzle kısıtlı bir model geliştirme imkanı sunmaktadır. Functional arayüz ile Sequantial'da olduğu gibi her katmanda tek bir girdi ve tek bir çıktının olduğu kısıtlamadan kurtarılmıış, böylelikle esneklik kazanması amaçlanmaktadır. Keras platformunda, Tensorflow'da olduğu gibi performans istatistiklerinin takip edileceği ya da oluşturulan modelin görüntülenebileceği bir araç bulunmamaktadır. Model görselleştirmesinin yapılamaması araştırmacının modeli görüntüleyememesini ve kara kutu yapının korunmasını sağlamaktadır. Oluşturulan modellerdeki katmanlar girdi ve çıktılarının tanımlanmasıyla çalışmakta ve çalışma anında işletilmektedir. Oluşturulan modelin işlem sonucunda çıktısı alınmak isterse dosya olarak çıktı verebilmektedir. Dolayısıyla işlem anında herhangi bir topoloji değişikliğinin desteklenmediği gibi, her seferinde sıfırdan eğitime başlamak gereklidir.

2.5.3. PyTorch

Paszke ve diğerleri (2019) tarafından geliştirilen PyTorch projesi, Torch yazılım çatısı tabanlı Facebook yapay zeka araştırma laboratuvarı tarafından desteklenen açık kaynak kodlu bir makine öğrenme kütüphanesidir. Python programlama dilini temel alarak

geliştirilmesine rağmen C++ diliyle geliştirmeye de destek vermektedir. PyTorch kütüphanesi tensörleri kullanarak grafik kartları ile sinir ağı hesaplamaları yapmaya destek vermektedir. PyTorch Tensorflow ile çalışılırken tüm modelin tanımlanması gereken statik hesaplama çizgesi yaklaşımı yerine dinamik hesaplama çizgesi yöntemiyle çalışmaktadır.

PyTorch ile oluşturulan model çizgesi kodlama yapılırken dinamik değiştirilmesi sağlanmaktadır. Tensorflow'a göre geliştiren topluluk kitlesi az olması, yazılım çatısının gelişmesini yavaşlatmaktadır. PyTorch'un Tensorboard gibi oluşturulan modelin takibinin sağlanabileceği bir aracı bulunmamaktadır, oluşturulan sinir ağı modelinin görselleştirilmesi desteklenmemektedir. Tensorflow ile son ürün aşamasında, büyük projelerde kullanılacak dağıtıklaştırılabilecek modeller üretmek konusunda başarılı modeller üretilmek amaçlanmaktayken, PyTorch ile geliştirmesi hızlı, örnek modeller oluşturma konusunda özelleşmektedir.

PyTorch için model kaydetme örneğini kullanarak eğitimdeki verilerin değişme kaydının sağlanması ile sinir ağlarının eğitiminin daha iyi anlaşılması ve kara kutu yapısının önüne geçilmesi amaçlanmaktadır. Tensorflow'un statik çizge dosyası oluşturma yapısında, eğitime devam edileceği zaman yeni katmanlar eklenecekse yeniden bir modelin oluşturulması gerekmektedir ancak PyTorch'ta hali hazırda geliştirilmiş bir modele yeni bir katman eklenebilmesine olanak sağlanmaktadır. Dinamik yapısı sayesinde kod hata ayıklamalarında hataları bulmak çok daha kolay olmaktadır.

PyTorch işlemci ve grafik kartı tabanlı model hesaplama işlemleri için ayrı kütüphaneler içererek ayırmakta ve çalışma ortamına göre odaklanmış ve kodlanmış kütüphaneler içermesi ile verimli çalışmak üzere geliştirilmiş olduğunu göstermektedir. PyTorch'un nn modülü kullanılarak katmanlar evrişim, pooling, tek boyutlu matrise çevrim işlemlerini gerçekleştiren katmanlar tanımlandıktan sonra nn modülüyle kullanılması gereken forward fonksiyonu ile katmanların nasıl bağlanacağı bileşke fonksiyon düzeniyle birbiri içinde tanımlanmaktadır ve model iskeleti oluşturulmaktadır. Oluşturulan bu model eğitim ve test için kullanılabilmektedir.

2.5.4. Caffe

Jia ve diğerleri (2014) tarafından geliştirilen Caffe bir derin öğrenme yazılım kütüphanesidir. C++ programlama dilini temel alarak, Python ve MATLAB dilleri ile de kullanılabilmektedir. Yoğunlukla görsel üzerine yoğunlaşmış öğrenme problemleri için kullanılmaktadır ve grafik kartı ile hesaplama yapmaya destek vermektedir. Açık kaynak kodlu olarak kullanıma sunulmaktadır ve topluluğu tarafından geliştirilmeye devam etmektedir.

Caffe ile modeller tanımlanırken protobuf dosyaları olan prototxtler kullanılmaktadır. YSA modelindeki katmanlar birbirileri arasındaki ilişkiler bir prototxt'de eğitim ve performans parametreleri Solver adı verilen başka bir prototxt tanımlanmakta ve tüm model yapısı oluşturma işlemleri dosyalar ile gerçekleştirilmektedir. Caffe'de araştırmacıların katman içerisinde değişiklik yapabilmesinin bir yolu bulunmamaktadır. Prototxtler ile tanımlanabilecek ve Caffe tarafından tanınacak alanlar dışında katmanlar içine müdahale edilmesi zordur.

Caffe, son kullanım ortamında gerçek dünya verileriyle ve görsel verilerin yoğun olduğu problem öğrenim alanlarında derin öğrenme işlemleri yapmak üzere yoğunlaşmıştır. Bu nedenlerden dolayı esnekliği bulunmamaktadır. Tensorflow'daki Tensorboard gibi bir aracı bulunmamakla beraber, dağıtık grafik kartı bulut servislerinde çalışmak için idealdir. Araştırmacının yeni bir problem öğrenme alanı üzerinde çalışması yerine, büyük organizasyonların büyük son kullanım verileriyle çalışması için uygun görülmektedir. Ayar dosyaları ile çalışarak, programlama kodları ile çalışmaması sağlanarak grafik kartı destekli bulut sunucularında ve mobil cihazlarda YSA modellerinin çalışmasını amaç edinmektedir. Bulut servislerine verdiği destek sayesinde performansının yüksek olduğu ve endüstri standartlarında çalışmaya yetecek hızda olduğu öne sürülmektedir. Eğitilmiş modellerin çıktıları caffemodel ve solverstate adında iki dosyada alınabilmektedir. Caffemodel ile modelin son durumu katmanlar arasındaki ağırlıklarla alınabilmektedir. Solverstate çıktısı ile oluşturulan ikili dosya ile modelin eğitim sırasındaki son hali alınabilmekte ve daha sonra eğitime devam edilebilmektedir. Araştırmacıların işbirlikçi çalışması playbook isimli bir araç ile gerçekleştirilebilmektedir. Araştırmacılar önceden belirli metriklerle modelin durumunu takip edebilmektedir. Kara kutu yapının incelenmesine yönelik herhangi bir araç bulundurmamaktadır. Caffe'de örnek bir prototxt katmanı Json formatında oluşturulduğunda "Type" alanında katmanın hangi tipte olduğu, bottom ve top anahtarlarıyla da önce ve sonra gelen katmanın hangi katmanlar olduğu, katmanın tipine göre de alacağı parametrelerin "convolution_param" anahtar ifadesiyle verilmesi gerekmektedir. "convolution_param" anahtar ifadesinde kernel boyutları çıktı sayısı gibi parametrelerin de katman tanımı yaparken verilmesi gereklidir.

BÖLÜM 3
MATERYAL VE YÖNTEM

Bu bölüm yapay sinir ağ modellerini çizge veritabanlarına aktarma ve saklamak için önerilen yöntem ve yazılımın bileşenleri ve seçilen teknolojilerin seçilme nedenleri içermektedir. YSA süreçlerindeki tahminleme, öğrenim aktarımı, yapay sinir ağı tahminleme süreç gözlemi gibi örnek senaryoları gösterilmiştir.

3.1. Çalışmada Kullanılan Teknolojilerin Belirlenmesi

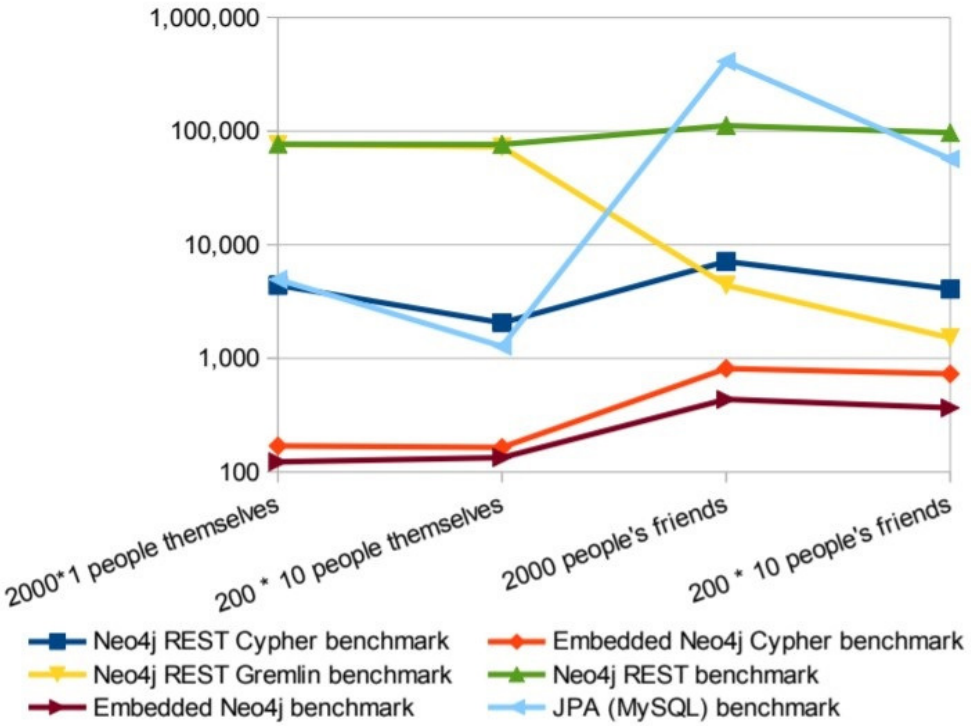
Birbirinden bağımsız çalışabilen yazılım bileşenleri bakış açısıyla geliştirilen yazılım üç ana kısımdan oluşmaktadır.

- Kullanıcı ile çizge veri tabanı arasında bağı kuran ve kullanıcıların kimlik ve yetkilendirme işlemlerinin kontrolünü sağlayan arka yüz servisi
- Kullanıcıların YSA modellerini dinamik olarak web arayüzü üzerinden oluşturabildiği, testini, eğitimini ve son durumunu gözlemleyebildiği ön yüz servisi
- YSA modellerinin tüm verilerinin tutulduğu, işlemlerinin gerçekleştirildiği çizge veri tabanı servisi

Şekil 12'de yazılımın üç temel parçası ve aralarındaki iletişim şeması gösterilmektedir. Arka yüz sunucusunun farklı işletim sistemlerinde çalışmasını desteklemek için çapraz platform desteği sunan .NET Core 2.2 teknolojisi ile CSharp dili kullanılmıştır. Ön yüz sunucusunda sayfa yenilemeye gerek kalmadan dinamik veri akışına izin veren Javascript ve Typescript dillerini kullanan Angular 8 kütüphanesi kullanılmıştır. Ön yüz ile arka yüzün YSA modelleri için haberleşmesinde, çift yönlü sürekli veri akışına izin veren soket iletişim kanalları kullanılmıştır.

YSA'larda hücre sayısından daha çok bu hücreler arasında bağlantılar bulunmaktadır. Bir YSA modelinin çıktı üretebilmesi için tüm bu bağların işletilmesi gerekmektedir. Hesaplama zamanını hücre sayısından daha çok bağ sayısı etkilemektedir. Tam bağlı YSA'larda katman sayısı ve katmandaki hücre sayısı arttıkça bağlantı sayısı katlanarak artmaktadır. Neo4j veritabanında bağlantı sayılarının katlanarak artması sorgu performansını, ilişkisel veritabanlarındaki dramatik düşüş kadar değiştirmemektedir (Holzschuher ve Peinl, 2013). Şekil 11'de bu çalışmadaki karşılaştırma sonuçları gösterilmiştir. Neo4j sorgularının anlaşılırlığı ve kullanım kolaylığı da çizge veritabanı seçiminde rol oynamıştır. Tasarlanan YSA'lar çizge olarak tasarlandığı şekliyle Neo4j'de tutulabilmektedir. Bahsedilen özelliklerinin yanında çizge veri tabanları arasında en çok kullanılan veritabanı olması ve geliştirici topluluğunun büyüklüğü nedeniyle Neo4j

YSA'ların saklanacağı veri tabanı olarak seçilmiştir.

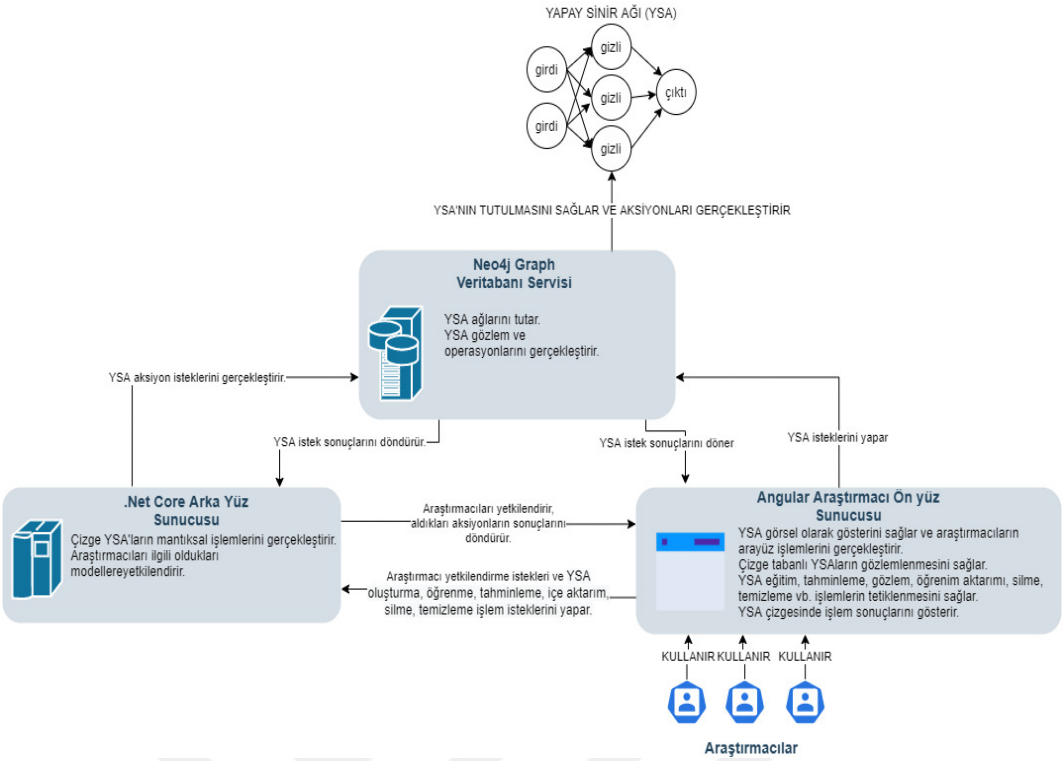


Şekil 11. Neo4j bağlantı sayısının performansa karşılık etkisinin Sql ile karşılaştırılması

3.2. Geliştirilen Yazılım Çatısının Mimarisi

Çalışmada geliştirilen yazılım çatısında her bir yazılım bileşeninin tek başına ayakta kalıp kendi sorumluluğunu gerçekleştirebilmesi amaçlanmıştır. Bu nedenle ön yüz sunucusu, arka yüz sunucusu ve veri tabanları kendi başlarına çalışarak geliştirilen ekosistemde sorumluluklarını yerine getirmektedir. Sistemin mimarisi oluşturulurken, açık kaynak kodlu kütüphaneler, bireysel kullanımı ücretsiz yazılım çatıları tercih edilmiştir.

Şekil 12'de oluşturulan mimari ve arasındaki ilişkiler gösterilmiştir. Araştırmacılar sadece önyüz servisiyle etkileşime girerek süreçleri tetiklemektedir. Önyüz sunucusu kullanıcı girişini onayladıktan sonra YSA model görüntüleme işlemlerinde gerekli yönlendirmeleri yaparak soket iletişimi ile çizge veri tabanının bolt protokolüyle konuşmaktadır. Bolt protokolü Neo4j geliştiricileri tarafından oluşturulmuş veri tabanı ağ protokolüdür. Bolt protokolüyle herhangi bir uygulama ile Neo4j veri tabanı arasında iletişim kurulabilmektedir.



Şekil 12. Geliştirilen yazılım çatısı bileşenleri ve aralarındaki iletişim

Arkayüz sunucusu, Neo4j'de tutulan YSA modelleri ile kullanıcının etkileşimini gerçekleştirmek üzere Model-View-Controller yazılım deseni ile model, gösterim ve kontrolcü katmanlarından oluşmaktadır. Model katmanında, kullanıcıların giriş veri yapısı, yapay sinir ağ bileşenlerinin yapısı, ön yüz ve veri tabanları ile iletişim halinde kullanılması gereken tüm yapıların protokolleri yer almaktadır. Gösterim katmanı son kullanıcı sayfalarının hazırlanmasından sorumludur. Kontrolcü katmanında çalışma mantıkları ve akışlarının yönetilmesi sağlanmaktadır. Bu şekilde oluşturulmuş mimaride veri tabanı sunucusu arka yüz ve ön yüz sunucularına bağlı kalmadan, oluşturulmuş sinir ağ modellerinin saklanmasına ve incelenmesine olanak sağlamaktadır.

Neo4j[2]'nin kendi sorgulama dili olan Cypher diğer çizge veri tabanı sorgulama dillerine göre çok daha açıklayıcı ve üst seviye sorgulama dilidir. Cypher'da sorgular SQL'deki gibi tablo mantığında değil, objeler ile aralarındaki ilişkiler ile çizgenin işlem yapılacak bölümünün ifadesi şeklindedir. Örneğin "(:dugum)" ifadesi düğüm etiketine sahip bir düğümü ifade ederken, "-" çift yönlü bağlantı ifadesi, "->" tek yönlü bağlantı ifadesi gibi olabildiğince kendini açıklayan ifadeler düğüm ile ilişkisi olan başka bir düğümü ya da belirtilecek ise ilişkiyi işaret eden ifadelerdir. "[:iliski]" ifadesi düğümler arasında ilişki etiketiyle bir ilişkinin sorgulanmasında kullanılmaktadır. Örneğin "MATCH (n:Dugum)-[r:Iliski]->() RETURN n,r" Cypher sorgusuyla yukarıda bahsedilen ifadeler ile

[2]Önerilen yazılımda Neo4j'nin 3.5.16 versiyonu kullanılmıştır.

anlatılmış düğüm ilişki örüntüsünde eşleşen çizge getirilmektedir.

Çalışma kapsamında Neo4j'de tutulan çizgelerde YSA katmanları ve ilişkiler label (etiketler), düğüm ve ilişkilerde tutulacak veriler property (özellikler) olarak ifade edilmektedir. Çalışmada ele alınan YSA model türlerine bağlı olarak bağlı olarak yapay sinir ağlarındaki hesaplama matris değerleri ilişkilerde ya da düğümlerin kendilerinde tutulmaktadır. İlk yöntemde ilkel skaler veri tipleri ile hesaplamalar yapılacak YSA'larda hesaplama değerleri hücre bağlantılarında tutulmaktadır. İkinci yöntemde ESA ağlarında olduğu gibi bir boyutludan büyük hesaplama matrisleriyle işlemler yapılırken her katmanın hesaplama matrisini temsil eden hücrelerde tutulmaktadır. Hücrelerde çok boyutlu matrisler saklanmaktadır. Üçüncü yöntemde ise katmanlardaki hesaplama matrislerindeki her bir hücre değeri hesaplama matrisindeki hücreyi temsil eden düğümlerde tutulmaktadır.

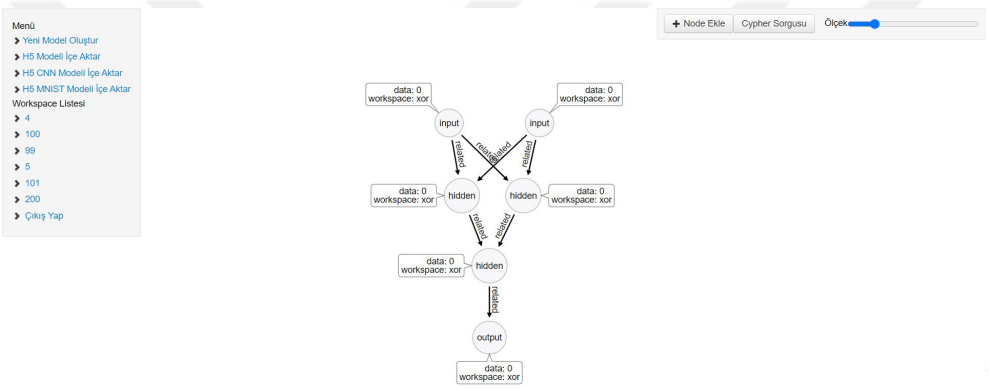
Neo4j yapılan çalışmada YSA model deposu olarak görev almakta ve modellerle alakalı hesaplama işlemleri, tutulma işlemlerinin tamamı Neo4j üzerinde yapılmaktadır. Çalışma gerçekleştirilirken ESA gibi büyük YSA'larda hesaplamalar yapılırken performans kayıpları yaşandığı için, Neo4j veri tabanında "workspace" özelliği üzerinde index tanımlanmıştır. Veriler yoğunlaştıkça Neo4j üzerindeki okuma hızlarının düşmemesi için verilerin adreslendiği yerleri tutan index tanımı yapılması önerilmektedir. Index kullanımı ile verilerin yerleri işaretlendiğinden Neo4j verileri disk üzerinde daha fazla yer kaplayacaktır. Index eklendiğinde okuma hızı artarken yazma hızı yavaşlamaktadır.

Neo4j çizge veri tabanındaki sorguların hızını etkileyen bir diğer unsur ise Neo4j veri tabanının, verileri disk üzerinden ya da ön bellekten sunmasıdır. Neo4j ilk kez çalıştırılıp YSA modelleri üzerinden işlemler yapılmaya başlandığında daha yavaş çalışmakta, kullanıldıkça hızlanmaktadır. Bunun nedeni Neo4j'nin ısınma denilen verileri çalışmaya başladığı zaman disk üzerinden sunması, işlemler yapıldıkça bellek üzerinden sunmaya başlamasıdır. Büyük YSA kümelerine ulaşılmaya başlandığında veri tabanında ısınma işleminin yapay olarak uygulanması Neo4j tarafından önerilmektedir, böylece veri tabanı çalışmaya başladığında YSA modelleri ön belleklenecek ve hızlı şekilde kullanıcıya işlem yapması için sunulmaya başlanacaktır.

3.3. Model Oluşturma

Kullanıcılar site üzerinden sisteme giriş yaptıktan sonra ortak çalışma ekranı üzerinden görsel olarak YSA modelini tasarlayabilmektedir. Yazılım üzerinde kodlama yapılmadan sinir hücrelerini temsil eden düğümler ve düğümleri birbirleriyle tek ya da çift yönlü bağlayan ilişkiler oluşturulabilmektedir. Düğümlerin bulunduğu katmanın belirlenmesi etiketlerin (label) atanması ile yapılmaktadır. Örneğin girdi, gizli ve çıktı

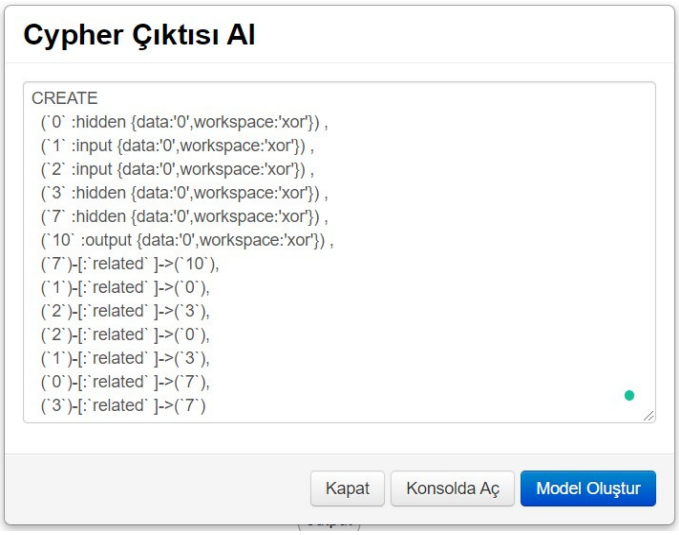
katmanları başlıklarda belirtilmelidir. Düğümlerde ve ilişkilerde YSA başlangıç parametreleri (ağırlıklar, bias vb.) model oluşturma aşamasında verilebilmektedir. Örneğin mantık kapısı öğrenen bir YSA tasarlarken düğümlerde "data" özelliği tanımlanarak ikili başlangıç değerleri verilebilirken ilişkilerde ise weight, kernel ve bias değerlerinin başlangıç değerleri verilebilmektedir. Kernel dizisi bir evrişim filtresinin tamamını veya bir parçasını temsil eden matristir. Hesaplamayı kolaylaştırmak için kernel matrisleri değişik yapıda olabilmektedir. YSA modellerinin öğreniminde kullanılacak öğrenme oranı, eşik değeri vb. öğrenme parametreleri bir düğümde tutularak YSA öğrenimini ilgilendiren parametrelerin de çizgede tutulması sağlanmaktadır. Şekil 21'de öğrenim parametrelerinin tutulması "setting" düğümünde gösterilmektedir. Şekil 13'te örnek bir XOR mantık kapısını öğrenen YSA modelinin görsel olarak sistem üzerinden oluşturulmuş hali gösterilmektedir.



Şekil 13. Xor mantık kapısının yazılım üzerinden oluşturulmuş örneği

YSA modelleri tasarlanırken XML formatında tutulmaktadır. Tasarımı bitmiş olan model sisteme kaydedilirken Cypher sorgu formatına çevrilmektedir. Şekil 14'te tasarlanmış YSA'nın Cypher sorgu ekranı gösterilerek sisteme kaydedilmesi sağlanmaktadır. Oluşturulmuş Cypher sorgusu herhangi bir Neo4j veri tabanında çalıştırılarak modelin oluşturulması sağlanabilir.

Cypher sorgusu "CREATE" komutu ile yeni bir modelin oluşturulacağını ifade etmektedir. Sorguda öncelikle düğümler sanal kimlik numaralarıyla beraber etiketleri yani katmanları, içerdikleri özellikler yani hücre özellikleri tanımlanmaktadır. Sorgunun ikinci bölümünde ise sanal kimlik numaralı düğümlerin arasındaki bağlantılar ifade edilmektedir. Düğüm ve bağlantıların özellik alanında Json veri türünde diğer gerekli veriler tanımlanmaktadır.



Şekil 14. Oluşturulan xor yapay sinir ağ modelinin cypher sorgusu çıktı ekranı

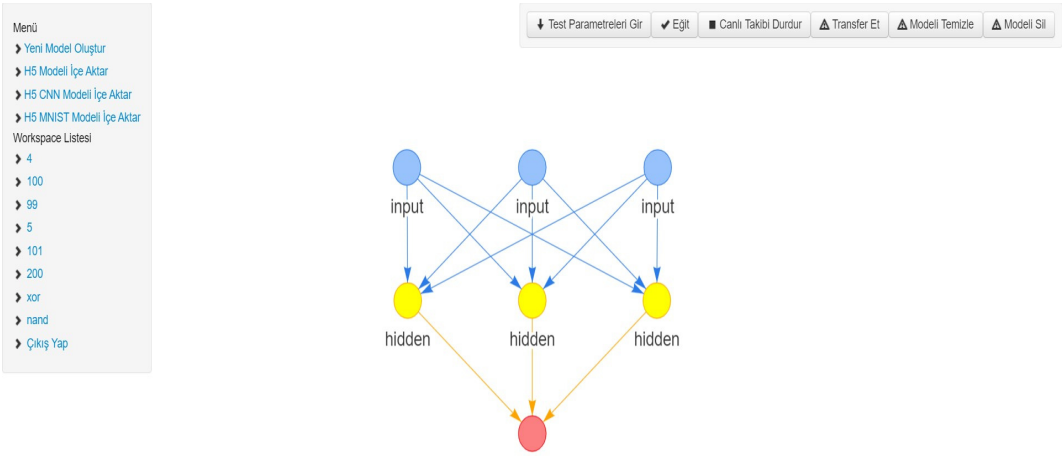
Tasarlanmış YSA'nın sisteme kaydedilmeden Neo4j simulatörüyle ön izlemesi görüntülenebilmektedir. Böylece tasarımda oluşabilecek olası bir hatanın önüne geçilebilmektedir.

3.4. Model Gözlemi

YSA modeli oluşturulduktan sonra geliştirilen yazılım üzerinden kullanıcılar tarafından eş zamanlı olarak gözlemlenebilmektedir. YSA tahminleme, sınıflandırma ve eğitim süreçleri görsel olarak tetiklenebilmekte ve izlenebilmektedir. YSA gözlemi sırasında aşağıdaki işlemler kodlama yapmaya gerek olmadan gerçekleştirilebilmektedir.

- Test verileri girilerek tahminleme süreci başlatılabilir.
- Canlı izleme durdurularak modelin o anki durumu (ağırlıkları ve çıktı değeri) detaylı incelenebilir.
- Öğrenim aktarımı gerçekleştirilebilir.
- YSA model versiyonlaması gerçekleştirilebilir.
- Modelin başlangıç durumundaki parametrelerine dönmesi sağlanabilir.
- YSA modeli silinebilir.
- Eğitim süreci başlatılabilir.

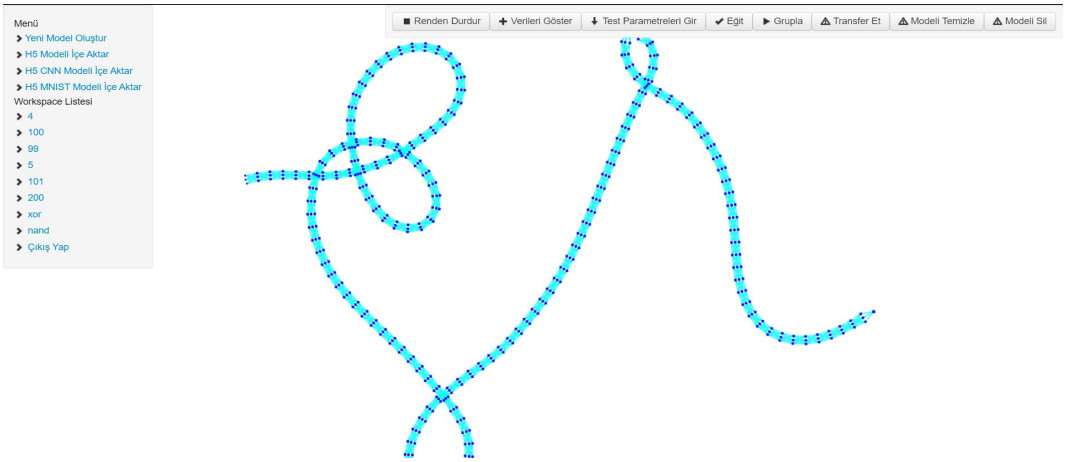
Şekil 15'te oluşturulan örnek bir modelin gözlem sırasındaki görüntüsü gösterilmiştir.



Şekil 15. Gözlemlenen ileri beslemeli yapay sinir ağı modelin görüntüsü

Küçük ağlarda canlı gösterim ve çizge veri tabanı sorgulaması fazla performansa ihtiyaç duymadığından alan üzerinde Javascript desteğiyle yapılan SVG, yani ölçeklenebilir vektör grafik çizimleri yeterli olmaktadır. Ancak evrişimli sinir ağları gibi onlarca ya da yüzlerce katman ve ilişkiden oluşan ağların gösteriminde bu yöntem yetersiz kalmaktadır. Bu yüzden WebGL (Leung ve Salga, 2010) tabanlı tarama ve grafik işleme yöntemi kullanılmıştır. YSA izlemede iki yöntemle de izleme yapılabilmektedir.

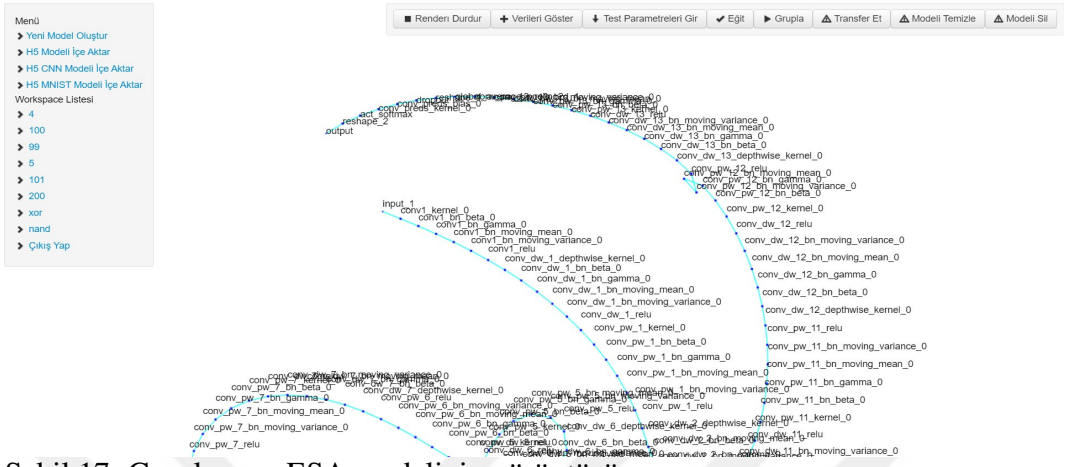
Evrişimli sinir ağlarında YSA gözlemini örneklemek için MobileNet (Howard ve diğerleri, 2017) modeli geliştirilen sisteme aktarılmıştır. MobileNet modeli mobil cihazlarda nesne tespit problemini çözmek üzere eğitilmiş bir ESA modelidir. Şekil 16'da çizge veri tabanına aktarılmış olan ESA modelinin gözlemi yapılmaktadır. MobileNet'in görsel üzerinde nesne tespitini yapacağı evrişim ve filtre hesaplama matrisleri katmanlardaki sinir hücrelerinde tutulmaktadır.



Şekil 16. Gözlemlenen ESA modelinin görüntüsü

YSA modelleri gözlemlenirken her düğümün görüntülenmesi her zaman istenen yöntem olmayabilmektedir. Bu durumlarda ise, "Grupla" düğmesi ile düğümlerdeki

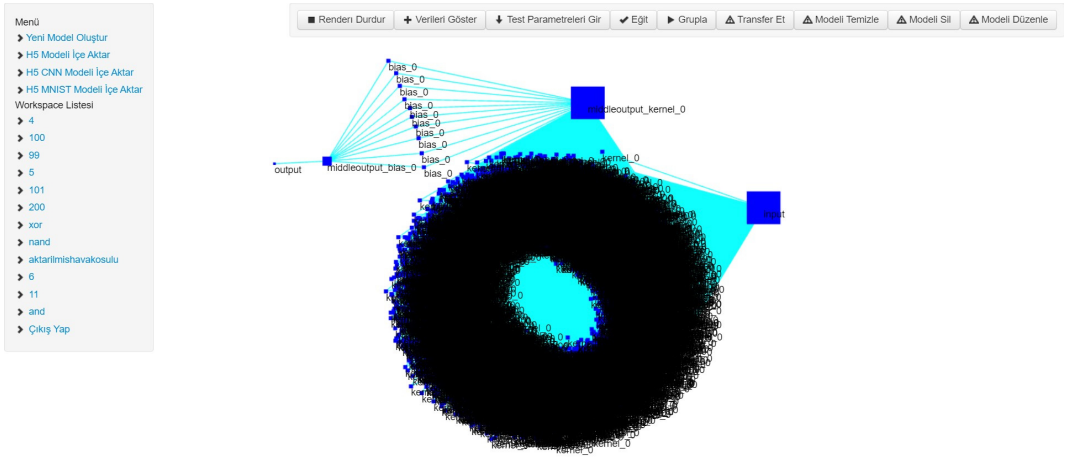
etiketler yani katman isimleri üzerinden gruplama yapılarak sadece katmanların görüntülenmesi sağlanabilmektedir. Şekil 17'de gruplanmış ESA modelinin gösterimi yapılmaktadır.



Şekil 17. Gruplanmış ESA modelinin görüntüsü

ESA gibi çok katmanı ve ilişkisi olan yapay sinir ağlarında kernel, bias, görüntülere uygulanan gama, beta, moving variance, relu gibi işlem matrisleri çok büyük olabilmektedir. Neo4j tek boyutlu diziler haricinde matris tutulmasına standart haliyle destek vermemektedir. ESA'larda kullanılan işlem matrisleri ise çoğu zaman bir boyutlu dizilerden fazladır. Bu nedenle ESA modellerinde düğümlerde veya ilişkilerde işlem matrislerini tutarken Json veri tipinde veri tutma ile gerçekleştirilmiştir.

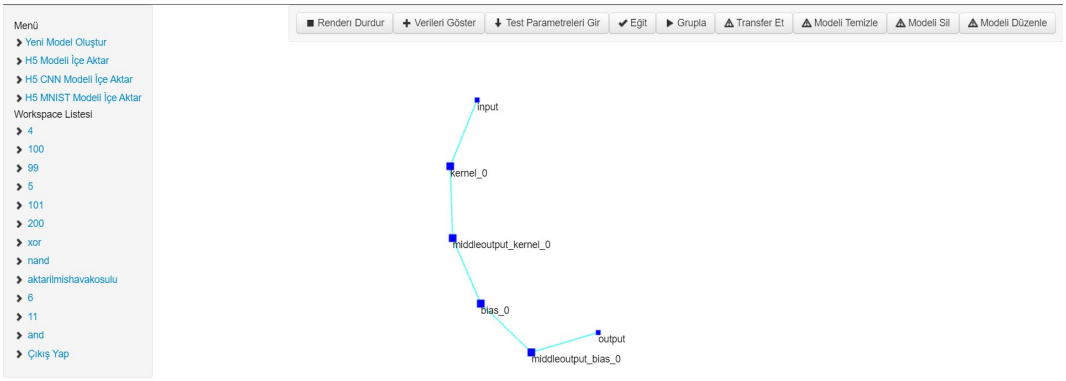
Şekil 18'de popüler olan Mnist el yazısı rakam tanıma veri seti için oluşturulan çok katmanlı YSA modelinin ağırlıklar matrisindeki her bir hücrenin ayrı düğümde tutulmuş hali gösterilmektedir.



Şekil 18. Mnist modelinin gösterimi

Şekil 19'da Mnist veri setinin eğitilmiş modelinin hesaplama aşamalarına göre gruplanmış hali gösterilmektedir. Bu örnekte "input" girdi katmanına karşılık gelmektedir.

"kernel_0", "middleoutput_kernel_0", "bias_0", "middleoutput_bias_0" gizli katmanı ifade etmektedir.



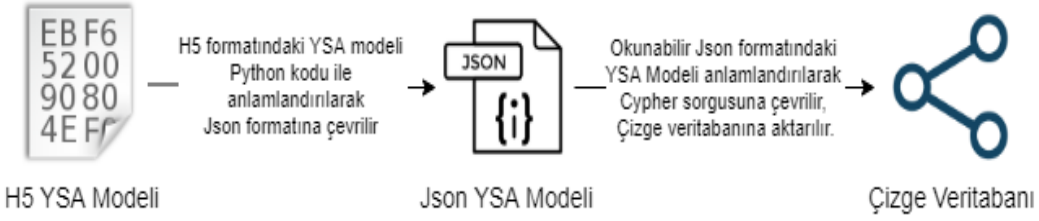
Şekil 19. Gruplanmış Mnist modelinin gösterimi

Model gözlemi yapılan sinir hücreleri ile düğümler arasındaki ilişkilerin kalınlıkları tüm kernel sayıları içerisindeki oranlarına göre gösterilmiştir. Farklı özelliklere göre ilişkiyi ifade eden çizginin kalınlığının ayarlanması mümkündür. İlişkinin ya da hücrenin üzerine imleç ile gelindiğinde değeri gösterilmektedir. Şekil 23'te gözlem sırasındaki çizim özellikleri gösterilmektedir.

3.5. Model Aktarımı

Model aktarımı, başka ortamlarda geliştirilmiş YSA modelinin birebir çizge veritabanında oluşturulmasıdır. Geliştirilmiş modellerin saklanmasında H5 standartlaşmış bir dosya biçimidir. Bu çalışmada H5 biçimindeki dosyaların çizge veritabanına aktarılması gerçekleştirilmiştir. Model aktarıldıktan sonra geliştirilen yazılımın tüm özellikleri bu model için kullanılabilir hale gelmektedir. Model aktarımını örneklemek için Bölüm 3.7'de anlatılan hava durumu ile oyun oynama tahmini gerçekleştiren YSA modeli Keras çatısı üzerinde eğitilmiş ve çizge veri tabanına aktarımı tamamlanmıştır.

Model aktarım sürecinde H5 model dosyaları önce Json işlenebilir formatına Python kod parçacığıyla çevrilmekte daha sonra ise Json dosyası işlenerek Cypher sorgusuna çevrilip çizge veri tabanına yazılmakta ve kullanıcıların çalışmasına sunulmaktadır. Şekil 20'de aktarım süreç adımları gösterilmektedir.



Şekil 20. Model aktarım süreci

3.6. Model Öğrenimi

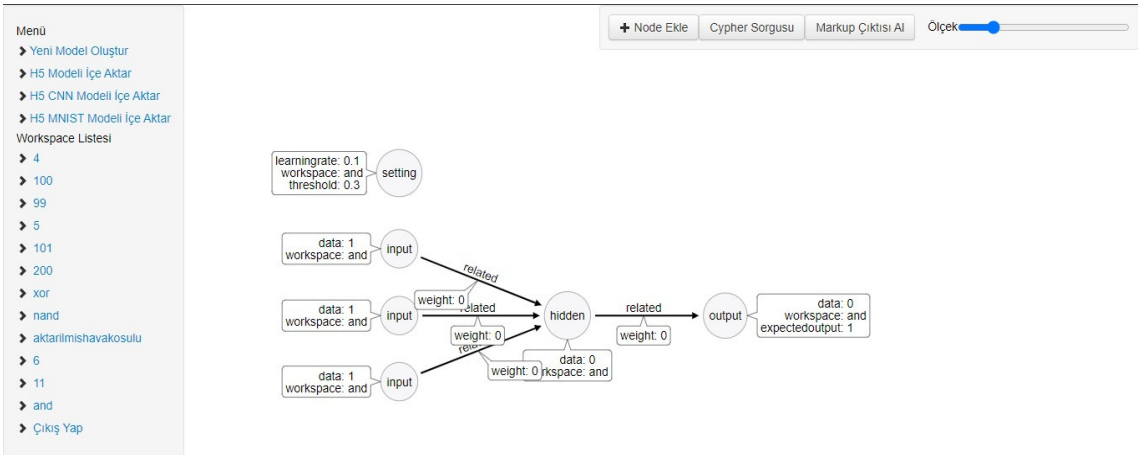
Model öğrenimi oluşturulan veya aktarılan modelin girdi verileriyle eğitilmesini anlatmaktadır. Geliştirilen yazılım ile eğitim yapılabilmektedir ancak performanslı bir eğitim ortamı oluşturmak gibi bir amacı yoktur.

Model öğreniminde oyuncak problemlerden olan üç girdili AND mantık kapısını öğrenen sinir ağı modeli oluşturulmuş ve öğrenme gerçekleştirilmiştir. Tablo 1'de üç girdili AND kapısının doğruluk tablosu gösterilmektedir.

Tablo 1
Üç girdili AND doğruluk tablosu

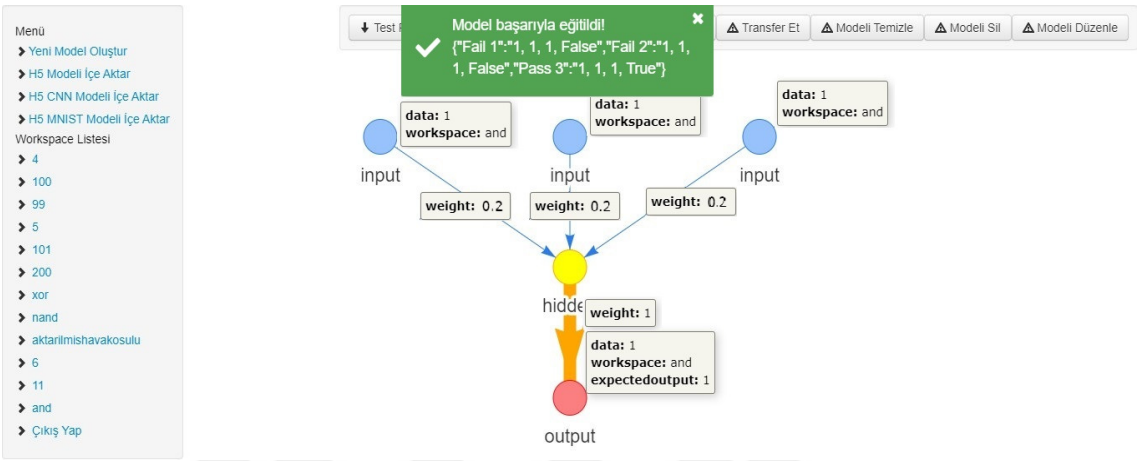
Girdi 1	Girdi 2	Girdi 3	Çıktı
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Şekil 21'de AND öğrenimi gerçekleştiren çizge yapay sinir ağının oluşturulması gösterilmektedir. Eşik değeri ve öğrenme oranı ayrı bir düğüm olarak tanımlanmaktadır. Bu oluşturulan düğüm çizge veritabanında saklanıp istenilen modelle ilişkilendirilip kullanılabilir. Girdi verileri ve beklenen değerler de çizge veritabanında tutulduğundan doğrudan modele bağlanıp eğitim verisi olarak kullanılması sağlanabilmektedir. Örnek model Perceptron öğrenmesi ile ağırlıkların güncellenmesi sağlanmaktadır.



Şekil 21. AND YSA modelinin oluşturulması

Şekil 22'de ortak çalışma sayfası üzerinden "and" modeli seçildikten sonra "Eğit" düğmesine tıklanarak canlı şekilde takibi yapılan modelin 3 iterasyon sonrasında istenen başarımı elde ederek durduğu gösterilmektedir. Düğümler arasındaki ağırlıkların kalınlığının ölçütü "weight" parametresi olarak belirtilmiştir bu nedenle girdi ve gizli düğüm arasındaki "0.2" değerli ilişkiler ince, gizli ve çıktı düğümü arasındaki ilişki kalın olarak temsil edilmektedir.



Şekil 22. Eğitilmiş AND YSA modelinin eğitim takibi ve sonucu

3.7. Model Çalıştırılması

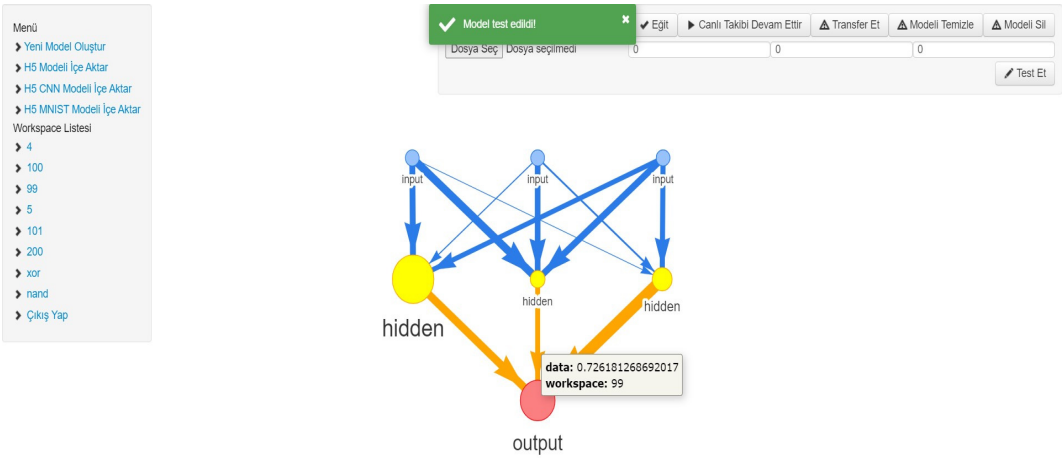
Model çalıştırılması eğitilmiş YSA modelinin girdiler gösterilerek çıktı üretmesini sağlamaktır. Yapılan çalışmada örnek bir eğitilmiş model ile model çalıştırma denemesi yapılmıştır. İleri beslemeli bir YSA modelinin çalıştırılarak doğru çıktılar üretmesi beklenmektedir. Bu amaçla kullanılan oyuncak problemin veri seti Tablo 2'de gösterilmiştir. Çıktılar evet, hayır şeklinde ikili olarak tanımlanmıştır. YSA modelinin sayısal olarak verileri işleyebilmesi için hava koşullarının sayısal karşılıklarına ihtiyaç duyulmaktadır. Hava durumu güneşli ise 1, kapalı ise 0, yağmurlu ise -1, rüzgar güçlü ise 1, zayıf ise 0, nem normal ise 0, yüksek ise 1 şeklinde dönüştürülmüştür.

Tablo 2
Hava şartı oyun oynama veri seti

Hava Durumu	Rüzgar	Nem	Oyun
Güneşli	Zayıf	Yüksek	Hayır
Güneşli	Güçlü	Yüksek	Hayır
Güneşli	Güçlü	Normal	Evet
Güneşli	Zayıf	Yüksel	Hayır
Güneşli	Zayıf	Normal	Evet
Kapalı	Zayıf	Yüksek	Evet
Kapalı	Güçlü	Normal	Evet
Kapalı	Güçlü	Yüksek	Evet
Kapalı	Zayıf	Normal	Evet
Yağmurlu	Zayıf	Yüksek	Evet
Yağmurlu	Zayıf	Normal	Evet
Yağmurlu	Güçlü	Normal	Hayır
Yağmurlu	Zayıf	Normal	Evet
Yağmurlu	Güçlü	Yüksek	Hayır

Yazılım çatısına aktarılmış olan hava şartlarına göre oyun oynama durumunu tahmin eden eğitilmiş ileri beslemeli YSA modeli üç farklı hava durumunu ifade eden üç nöronu, üç gizli nöronu, bir çıktı nöronunu içermektedir.

Yeni girdi verileriyle çıktı ürettirmek için test parametreleri arayüz üzerinden girilerek modelin çalışması sağlanabilir. Örnek model ile kapalı hava durumu yani 0, zayıf rüzgar yani 0, normal nem yani 0 değerleri girilerek model çalıştırılmıştır. Şekil 23'te çalıştırılmış model 0.72 değerini üretmiştir Softmax fonksiyonu uygulayarak çıktı değerinin beklenen çıktı değerlerine dönüşmesi sağlanır. Yani oyun oynanabilir çıktı değerine ulaşılır. Gizli katmanda arasında Tanh aktivasyon fonksiyonu, çıktı katmanında Sigmoid aktivasyon fonksiyonu kullanılmıştır.

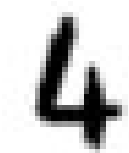


Şekil 23. Hava şartlarına göre oyun oynama tahmini yapan YSA modelinin tahminleme sonucu

3.7.1. Bir Model İncelemesi Mnist El Yazısı Rakam Tanıma

Mnist, LeCun ve Cortes (2010) tarafından 28 piksele 28 piksel gri skalada 60 bini eğitim, 10 bini test el yazısı rakam görselinden oluşan veri setidir. Literatürde derin öğrenme konusunda yapılan çalışmalarda sıklıkla kullanılmaktadır. Siyah beyaz skaladaki 20 piksele 20 piksel Nist veri setinin başarımını arttırmak üzere örtüşme önleme tekniği kullanılarak 28 piksele 28 piksellik görsellerin elde edilmesi ile Mnist veri seti oluşturulmuştur.

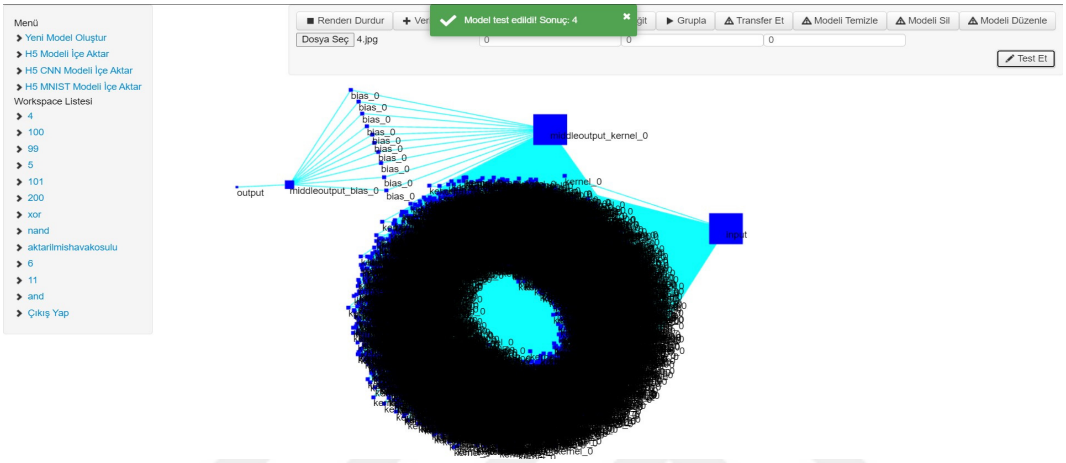
Bu bölümde incelenen evrişimli sinir ağında kullanılan örnek veriseti olan Mnist verisetinin eğitimi Keras üzerinde gerçekleştirilmiştir. Aktivasyon fonksiyonu olarak Relu, son adımda çok sınıflı sınıflandırıcılar için kullanılan Softmax aktivasyon fonksiyonu eğitimde kullanılmıştır. Eğitilmiş model çizge veri tabanına aktarılırken H5 formatındaki Keras model çıktısı öncelikle okunabilir Json formatına daha sonra Cypher sorgusuna dönüştürülmüştür. Evrişim matris çarpımlarındaki her bir hücre bir düğüm olarak ifade edilecek şekilde Cypher sorgusuna dönüştürülmüştür. YSA modelinin testleri geliştirilen yazılım üzerinde, gri skaladaki 28 piksele 28 piksel alan üzerine çizim programları yardımıyla çizilen rakamlarla yapılmıştır. Şekil 24'te tahminlemesi gerçekleştirilen çizim programları üzerinden çizilen rakamlar gösterilmiştir.



Şekil 24. Mnist tahminlemesi için çizilen 4 rakamı

Çizilen rakam görselleri site üzerinden test parametresi sekmesi ile gelen dosya seç bölümünden sisteme, 1'e 784'lük matris haline dönüştürülerek ve her bir piksel değeri gri skalada çalışması için 255'e bölünerek normalize edildikten sonra arka yüz sunucusuna

gönderilmektedir. Aktarılan modelde kernel matrisini ifade eden 784 hücre üzerindeki "data" değerleri ile çarpılarak bir sonraki hücredeki "data" alanına yazılmaktadır. "middleoutput_kernel_0" hücresinde kernel matrisi ile çarpılmış görsel, bias değerleri ile toplanarak "middleoutput_bias_0" hücresine aktarılmakta ve en küçük değerin bulunduğu matris indisi modelin tahminlediği rakam olarak bulunmaktadır. Şekil 25'te tahminleme sonucu yazılım üzerinden gösterilmektedir. Araştırmacı ilgilendiği nöron ve ilişkilerinde model verilerini gözlemleyebilmektedir.



Şekil 25. Mnist yapay sinir ağının 4 rakamını tahminleme sonucu

3.7.2. Çizgelerde Örnek Bir Evrişim Hesaplaması

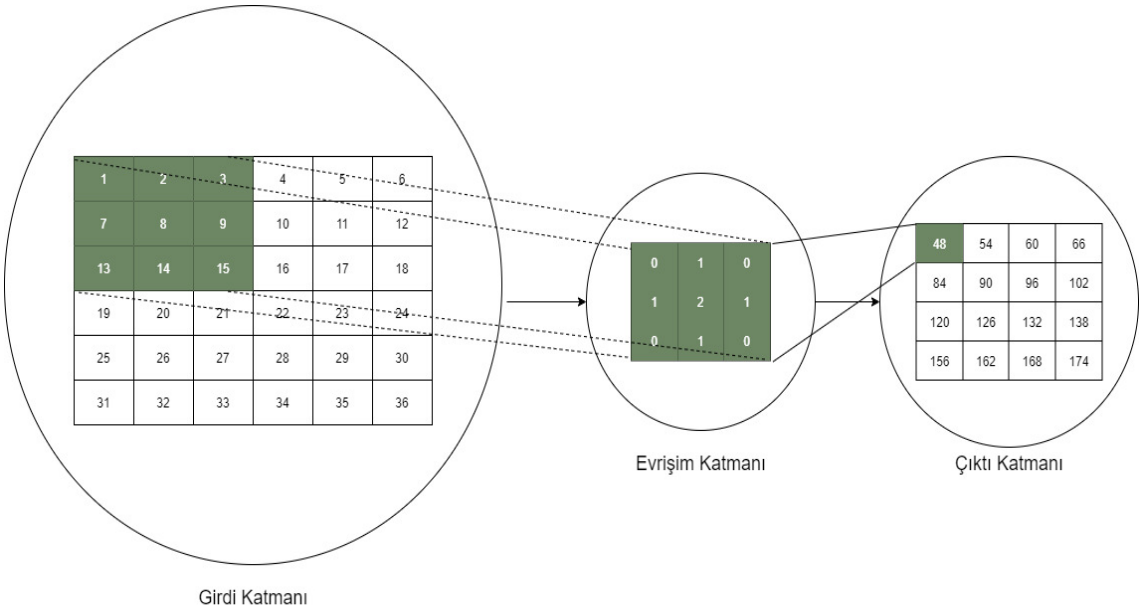
Evrişimli sinir ağlarında her katmanda bir ya da birden daha fazla hesaplama matrisi bulunmaktadır. Bu katmanlar evrişim, havuzlama vb. matrisler içerebilmektedir. Birden çok hesaplama matrisinin olduğu katmana filtre denilmektedir. Evrişim matrisleri genellikle bir boyutludan fazla boyut içeren matrislerdir. Evrişim matrislerinde "stride" olarak adlandırılan evrişim matrisinin girdi matrisi üzerinde her taramada kaç birim kaydırılacağı parametresinin belirlenmesi gerekmektedir. Girdi matrisi üzerinde kenar özelliklerinin öneminin kaybolmaması istenirse matrisin dört kenarına da belli sayıda sıfır değeri girilmesi "padding" parametresinin belirlenmesiyle olmaktadır. "padding" değeri genellikle evrişim matrisinin orta hücresinin, girdi matrisinin ilk hücresine gelecek şekilde belirlenmesiyle ayarlanmaktadır. Evrişim hesaplaması sonucunda çıktı matrisinin boyutları girdi matrisinin sütun ya da satır sayısı i, "padding" p, "stride" s, evrişim matrisinin sütun ya da satır sayısı k olarak ifade edildiğinde Denklem 3.1'de göründüğü şekilde hesaplanmaktadır.

$$o = \frac{i + 2p - k}{s} + 1 \tag{3.1}$$

Çalışmada kullanılan Neo4j özellik alanlarında temel veri tipleri olan "integer",

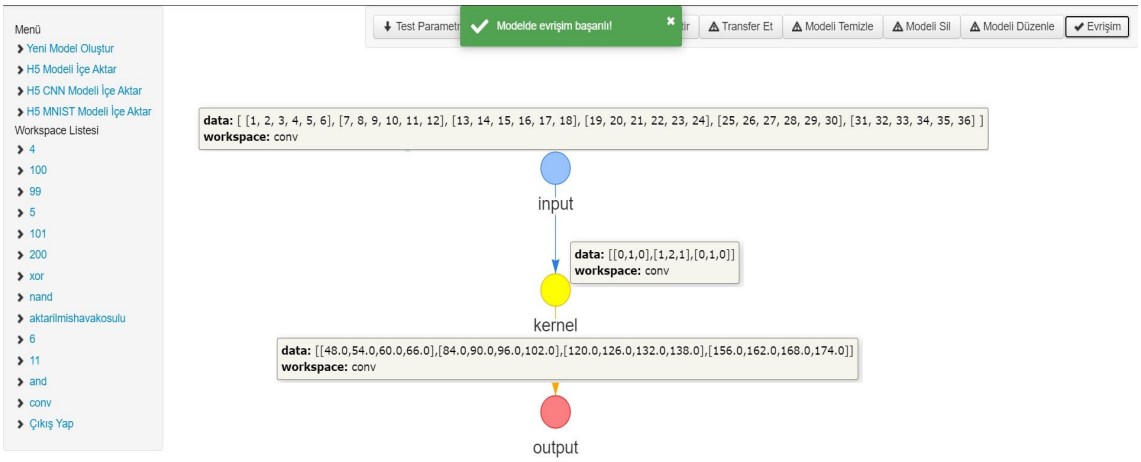
"float", "string", bir boyutlu dizi vb. değerleri saklamaya destek vermektedir. Çalışma kapsamında bir boyuttan fazla boyut içeren matrisler Json veri türüne dönüştürülerek düğümlerde tutulmaktadır. Bu sayede istenilen boyutta matrisin düğümlerde tutulması sağlanmıştır.

Örnek bir 6x6 matris üzerinde "stride" değeri bir, "padding" değeri sıfır olan 3x3 evrişim matrisi ile bir evrişim hesaplaması yapılarak çıktıları Şekil 26'da gösterilmiştir.



Şekil 26. Girdi katman hücresi üzerine evrişim uygulanması

Geliştirilen yazılımda ele alınan evrişim örneğindeki girdi matrisi "input", evrişim matrisi "kernel" ve çıktı matrisi "output" isimli düğümlerde tutulmaktadır. Evrişim hesaplaması arka yüz sunucusunda yapılmaktadır. Şekil 27'de yazılım üzerinde evrişim hesaplama çıktısı gösterilmektedir.



Şekil 27. Geliştirilen yazılımda evrişim hesaplaması

3.8. Model Versiyonlama

Yapay sinir ağlarında genellikle modelin en yüksek doğrulukta olduğu hali dosya formatında saklanmaktadır. Ancak daha sonra modelin yeni veriler ile eğitilmesi veya aynı modelin farklı öğrenme algoritması ve parametreleriyle eğitilmesi gibi durumlarda her bir modelin ayrı bir versiyon olarak tutulması önem kazanmaktadır. Bu sayede modeller arası karşılaştırma olanağı sağlanmış olur. YSA model eğitimi tek sefer gerçekleştirilecek bir süreçten çok yinelenen bir süreç olarak değerlendirilmelidir.

Geliştirilen yazılım ile YSA modelleri çizgede tutulduğundan kullanıcı tarafından belirlenen herhangi bir anda modelin bir versiyonu çizge veritabanında tutulabilmektedir. Bu sayede geleneksel yöntemlerdeki dosya formatlı YSA'ların tekrar eğitim için gereken dönüşüm ihtiyacı engellenmiştir. Model versiyonlarının aynı çizge ortamında bulunması sağlanmaktadır. Böylece YSA'daki herhangi bir hata durumunda ya da performans kaybında kullanıcının eski versiyona dönmesine olanak sağlanmıştır.

3.9. Öğrenim Aktarımı

Transfer öğrenimi diğer adıyla öğrenim aktarımı eğitilmiş bir YSA modelinde edinilen öğrenimin yeni bir modele aktarılması veya bu öğrenimin kullanılmasıdır. Öğrenim aktarımı ile geliştirilmiş bir modelin üzerine yeni sinir ağı katmanları eklenerek farklı modeller oluşturulabilir. Bu sayede temel alınan modelin başarısı yeni model ile geçilmeye çalışılır. Aynı problem tipinde eğitilmiş bir modelin bu şekilde kullanılması zaman ve kaynak tasarrufu sağlar.

Popüler YSA yazılım kütüphaneleri ile büyük veri setleri üzerinde oluşturulmuş modeller ince ayarlamalar yapılması için araştırmacılara sunulmaktadır. Bu modeller dosya olarak tutulmaktadır. Bu model dosyalarının bilgisayarda kullanılabilir hale getirilmesi ve değiştirilmesi süreci manuel olarak kodlama ile yapılabilmektedir. Öğrenim aktarımı yapıldıktan sonra kullanıcının model katmanları hakkında derin bilgisi olmadan herhangi bir katmanda değişikliği kodla yapmak zorunda olması pratikte zorluklara yol açmaktadır. Donmuş modellerde sadece hesaplama için gerekli bilgi bulunmaktadır. Katman bilgisi ve diğer üst bilgiler silinmiş olduğundan değişiklik yapmaya uygun değildir. Sadece ağırlık ve hesaplama verileri değiştirilebilmektedir. Aktarılmış modellerin doğrudan görselleştirilebilmesi sayesinde kullanıcılar model hakkında bilgi sahibi olabilmektedir.

Geliştirilen yöntem ile çizge YSA modelinin öğrenme aktarımı yeni oluşturulacak modelin adı girilerek kodlamaya ihtiyaç duymadan yapılabilmektedir. Şekil 36'da arayüz üzerinden öğrenim aktarımı süreci gösterilmektedir. Geliştirilen yazılım ile dışarıdan dosya ile modelin aktarılması sağlanabilmektedir. Geliştirilen yöntem ile veritabanındaki

modeller dosya taşıma işlemine gerek olmadan doğrudan kullanılabilmektedir. Aktarım yapıldıktan sonra yeni modelin yapısı kodlama yapmadan görsel olarak değiştirilmeye hazır hale gelmektedir.

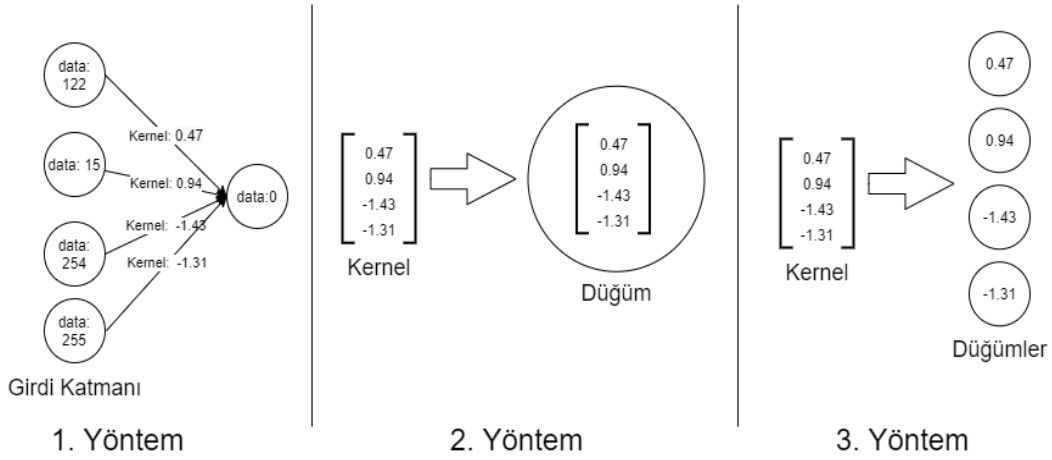
BÖLÜM 4
ARAŞTIRMA BULGULARI VE TARTIŞMA

4.1. Önerilen Yöntem Oluşturulurken Karşılaşılan Zorluklar

ESA modelleri çok boyutlu matrisler içerdiğinden Neo4j'nin özellik alanlarında desteklenen bir veri tipi olmadığı için Json formatına çevrilerek tutulması sağlanmıştır. Bu şekilde ESA modellerinin temsili Neo4j'de gerçeklenmiştir. Ancak Json halindeki matrisler doğrudan hesaplama için kullanılamaz. Bu yüzden hesaplama aşamasında bazı durumlarda veritabanında hesaplanabilir çizgeye dönüştürülmüştür. Bazı durumlarda da arka yüz sunucusunda çalışacak şekilde matris veri tipine dönüştürülmüştür.

H5 dosya tiplerinde kaydedilen modellerin her bir katmanı bir veya birden fazla matris olarak saklanmaktadır. Mnist veriseti üzerinde oluşturulan modeli aktarırken her bir matris hücresinin bir düğüm olarak çizge veritabanında tutulması modelin gözlemlenmesinde zorluklar yaşattığı tespit edilmiştir. Bu durumu çözmek için düğümlerin gruplandırılarak gösterilmesi yöntemi seçilmiştir. Bu gruplar birebir bir katmanı ifade etmeyebilir. Bir kaç grup birleşerek bir katmana karşılık gelebilmektedir. Bir diğer çözüm olarak her bir matrisin bir düğüm olarak tutulması ile daha etkin görselleştirme sağlanabilmiştir.

YSA'ların çizge veri tabanlarında temsili için üç farklı yöntem belirlenmiştir. İlk yöntem YSA hücreleri üzerindeki verilerin (hücre girdisi ve aktivasyon fonksiyonu) düğümlerde tanımlanan özellikler alanında tutulması şeklinde yapılmıştır. Sinir hücreleri arasındaki bağlantı ağırlıkları düğümler arasındaki bağlantılarda (kenarlarda) tutulmuştur. Örneğinde Bölüm 3.7'da bu yöntem gözlemlenebilmektedir. İkinci çok boyutlu matrislerle hesaplamalar yapılan onlarca katmana sahip olabilen modellerde her bir hesaplama matrisi bir düğüm olarak tutulmuştur. Bu durumda düğümler arasındaki bağlantılarda veri tutulmamıştır. Örneğin Şekil 16'da bu yöntem gözlemlenebilmektedir. Bu yöntem ile hücreler arasındaki hesaplama sonuçları sonraki düğüme girdi olarak aktarılmaktadır. Üçüncü yöntemde hesaplama matrislerindeki her bir hücrenin bir düğüm olarak tutulması sağlanmaktadır. Bu durumda düğüm bağlantılarında veri tutulmasına gerek yoktur. Örneğin Şekil 18'de bu yöntem gösterilmektedir. Şekil 28'de soldan sağa birinci yöntemden üçüncü yönteme örnekler gösterilmiştir.



Şekil 28. Yapay sinir ağlarının çizgelerde üç farklı temsili

İlk yöntem ile düğümler arasında iki sinir hücresinin, örneğin girdi ve kernel hücrelerini ele alırsak, girdi hücresindeki veri ile ilişki üzerinde tutulan kernel değerlerinin çarpım sonucunun kernel hücresinde yer alması sağlanabilmektedir. Çalışmada incelenmemesine rağmen geri yayılımlı ağlar gerçeklendiğinde de, örnekten yola çıkıldığında, kernelden girdiye ters yönlü ilişkiler üzerindeki hesaplama verileriyle geri yayılım sonucu, girdi hücrelerine yazılabilecektir. Modellerin temsil ve sunumu açısından ilk modelin kullanılması önerilmektedir. İkinci ve üçüncü yöntemler kendine has avantaj ve dezavantaj takasları içermektedirler. Üçüncü yöntem temsili ve sunumu güçlendirirken sistem kaynak gereksinimini arttırmakta, ikinci yöntemde ise kullanılan çizge veri tabanında varsayılan olarak desteklenmemektedir.

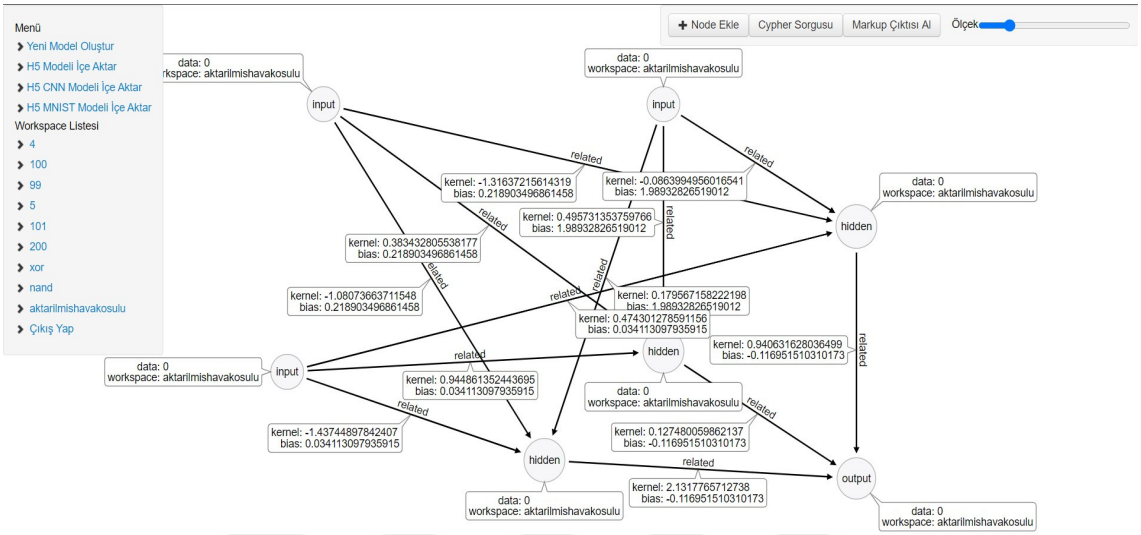
Her tür YSA modelinin oluşturulabilmesi geliştirilen sistem üzerinden teorik olarak mümkündür. Ancak bu çalışma kapsamında tüm öğrenme yöntemlerini ve YSA modelleri gerçekleştirilmemiştir. Yapılabilirliğini göstermek adına örnek öğrenme yöntemleri ve örnek modeller oluşturulmuştur. Önerilen yöntem ile model ve verinin aynı ortamda olması sağlanarak literatüre katkı sağlamak amaçlanmıştır. Bu fikrin endüstriyel hayatta gerçeklenmesine yardımcı olmak adına yaptığımız çalışma açık kaynak kodlu olarak araştırmacılara sunulmuştur(*Graph Based Neural Network*, 2020)[3].

4.2. Model Gösterimi ile Elde Edilen Kazanımlar

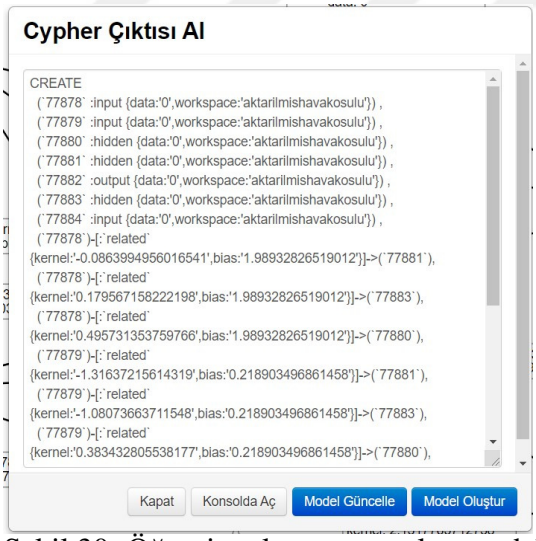
Geliştirilen sisteme aktarılan veya sistem üzerinde oluşturulan her model görselleştirildiği ortamda üzerinde değişiklik yapılmasına da olanak sağlamaktadır. Bu sayede modellerin değişik katmanlarının başka modellerin farklı katmanlarına bağlanması gibi karmaşık işlemler görsel olarak yapılabilmektedir. Aynı şekilde veriler de modele girdi olması için görsel olarak bağlanabilir. Bu sayede kullanıcıların oluşturdukları modellerin

[3]Graph Based Neural Network `https://github.com/dogabaris/GraphBasedNeuralNetwork`

girdi çıktı ilişkilerini rahatlıkla görmesi sağlanmaktadır. Modellere üst seviyede bakmaya olanak verir. Geliştirilmiş modeller canlı olarak kullanılmak üzere kolaylıkla veriyle buluşturularak sonuç üretebilmektedir. Bu sayede kodlama yapmaya ihtiyaç duyulmadan bir modelin kullanılması sağlanmaktadır. Örnek bir modelin bu şekilde kullanılması Şekil 29'da ve Şekil 30'da gösterilmiştir.



Şekil 29. Hava koşulu ile oyun tahmini yapan modelin öğrenim aktarımı sonrası düzenlenmesi



Şekil 30. Öğrenim aktarımı yapılan modelin düzenlendikten sonraki güncellenme ekranı

4.3. Yaygın Kullanılan Kütüphaneler İle Sunulan Yazılımın Karşılaştırılması

Bu bölümde Bölüm 2'de irdelenen yaygın kullanılan kütüphaneler olan Tensorflow, Keras, PyTorch, Caffe kütüphaneleri ile tezde önerilen çizge veritabanlı yapay sinir ağlarının özellikleri karşılaştırılmıştır.

Tensorflow, Keras ve Pytorch, Caffe kütüphanelerinde modeller sembolik hesaplama

çizgeleri olarak tanımlanmaktadır. Sembolik denilmesinin sebebi oluşturulan YSA mimarilerinin bir yerde saklanmamasından ve tensörler olarak işletilmesinden kaynaklanmaktadır. Hesaplama kelimesi ise çizgelerde nöronların ve bağlarının birebir varlık olarak tanımlanmasının yapılmadan tensörlerde birer hesaplama elemanı olarak kullanılmasından ileri gelmektedir. Bahsedilen kütüphanelerde tam bağlı YSA'lara odaklı programlama arayüzleri sunulmuştur. Tam bağlı olmayan ağ katmanlarının doğrudan oluşturulması desteklenmemektedir. Tez çalışmasında sunulan görsel arayüz oluşturucu ile çizgelerde oluşturulan modellerin tam bağlı ya da kısmen bağlı mimarilerde çalışması sağlanmıştır. Bir YSA modelinin birebir çizge olarak tutulması bu özelliği getirmiştir.

Tensorflow ve Keras statik hesaplama çizgelerini kullandığı için model mimarisinin oluşturulmasından sonra kod üzerinde sembolik tanımlanan YSA modelinin öğrenim, sınıflandırma ve tahminleme aşamaları devam ederken değişimine izin verilmemektedir. Model topolojisi değiştiğinde öğrenimin yeniden başlaması gerekmektedir.

Pytorch, dinamik hesaplama çizgesi kullanmaktadır. Bu sayede model öğrenimi sırasında hesaplama katmanları dinamik oluşturulmaktadır. Statik ve dinamik hesaplama çizgelerini birbirinden ayıran temel fark statik hesaplama çizgelerinin öğrenim ve hesaplamalardan önce tüm modeli tanımlaması, dinamik hesaplama çizgelerinin ise ilgili katmanlar ve model yapılarının işletilme zamanları geldiğinde tanımlanmasıdır.

Çizge veritabanlı yapay sinir ağlarında ise öğrenim ve test aşamalarında model mimarisi ve verileri veritabanındaki çizgelerde işlendiğinden her işlemde güncellemeler yapılmakta ve saklanmaktadır. Böylece herhangi bir anda topoloji değişikliği ve parametre değişikliği yapılmasını sağlayacak ortam oluşturulmuştur. Tez çalışmasında katman çizgeleri öğrenim ve test aşamasında herhangi bir zamanda tanımlanabilir. Tüm süreçte önce tanımlanarak öğrenim başlatıldıktan sonra herhangi bir zamanda tasarım değiştirilerek işletilmeye devam edebilebilir.

Tez çalışmasında çevrim içi olarak YSA modellerine her an ulaşılabilmekte ve işlemler gerçekleştirilebilmektedir. Eğitilmiş model dosyaya aktarılıp indirilmesi gerekmektedir. İş birlikçi model geliştirme kısmen yapılmaktadır. Tezde sunulan yöntem ile YSA süreç gözlemleri YSA topolojisi üzerinden neredeyse canlı olarak yapılabilmektedir. Smilkov, Carter, Sculley, Viégas ve Wattenberg (2017) yaptıkları çalışma ile Tensorflow modellerini canlı olarak manipüle edebilecekleri bir yöntem ortaya koyarak Playground yazılımı geliştirmişlerdir. Böylece YSA uzmanı ve araştırmacısı olmayan kişilerin de YSA süreçlerini hızla öğrenebilecekleri görsel etkileşimi Tensorflow altyapısıyla sunmuşlardır.

Tensorflow, Keras, Pytorch, Caffe ile geliştirilen YSA'larda öğrenim ve tahminleme yapılırken katmanlar ve aralarındaki bağıntı ağırlıkları görüntülenememektedir. Geliştirilen

araçla eğitim sırasında bağlantıların ağırlık değerinin düşük veya yüksek olmasına göre değişik şekillerde gösterimi yapılabilmektedir. Böylece oluşturulan modelde katmanlarda bulunan hücre sayılarının gereğinden fazla olma durumu belirlenebilir. Eğitim başarısını ölçen maliyet fonksiyonunun eğitim süresince değeri azalmadığı durumda eğer ağırlıkları çok düşük değere sahip olan katmanlarda hücre sayısının fazla olduğu söylenebilir.

Sunulan tez çalışmasında YSA'ların tutulduğu çizge veritabanında yaygın kütüphanelerden farklı olarak ekran kartı üzerinde hesaplamalar yapılmamaktadır. Büyük veri setlerinde YSA eğitimleri yapılması gerektiğinde hesaplamaları grafik kartları üzerinde gerçekleştirilmesi ileriki çalışmalarda yapılabilir.

4.4. Veritabanında Yapay Sinir Ağlarını Kullanan Çalışmalar İle Tez Çalışmasının Karşılaştırılması

Yapay sinir ağlarının veritabanlarında saklanma fikri daha önce de düşünülmüş ve gerçekleştirilmiştir. Daha önceki çalışmalarda genel olarak ilişkisel veritabanlarında meta modelin ve işlem verilerinin tutulması ile YSA varlıkları nesne yönelimli olarak saklanmaktadır. Bu yöntemle araştırmacılar geliştirdikleri model mimarilerini nesne yönelimli olarak sisteme tanımlamaları gerekmektedir. Örneğin tek yönlü bağlantılar içeren bir YSA mimarisini ilişkisel veritabanında tanımladıktan sonra çift yönlü bağlantılar içeren yeni bir mimari için veritabanında yeniden tanımlama yapılması gerekmektedir. Model mimarisinden bağımsız biçimde YSA varlıklarını birer veritabanı verisi olarak işlemek genelleştirilebilir bir model saklama ortamı oluşturmaktadır. Bölüm 2.4.3'te veritabanlarında saklanan ve işlenen YSA çalışmaları incelenmiştir.

NeuDB (Schikuta ve diğerleri, 1996) çalışmasında YSA varlıkları bir seferliğine tablo olarak tanımlanmalıdır. Örneğin sinir ağ katmanları bir tabloda, katmanlardaki olası bağlantıları ifade edebilecek bağlantılar bir tabloda, eğitim algoritmalarını işleyecek parametreler başka bir tabloda tanımlanmalıdır. Birden çok ve birbirinden farklı YSA modelini saklaması beklenen veritabanı, her modelde aynı özellikleri miras almak zorundadır. Öngörülemeyen ya da yeni geliştirilecek bir model NeuDB ile gerçekleştirileceğinde tablolarda yapılacak bir değişiklik tüm modelleri etkilemektedir. Her YSA için ayrı tablolar ile modellerin saklanması yöntemi uygulanırsa süreç genelleştirilebilir olmaktan çıkmaktadır. Nesne yönelimli tanımlama ile YSA modellerinin genelleştirmek istenmesine karşın tablo tanımlarında değiştirilen özellikler tüm modelleri etkilediğinden yönetmesi zor ve değişime kapalı bir yapı ortaya koymaktadır. İlişkisel veritabanlarında model saklayan tüm çalışmalarda bu sorun mevcuttur. NeuDB'de model oluşturulduktan sonra model mimarisi görsel olarak görüntülenememektedir. Eğitim ve

çalıştırma aşamasında modelde yapılan hesaplama sonuçları veritabanı üstünde gözlemlenememektedir. YSA modeli oluşturmak, eğitmek ve test etmek için araştırmacılara bir arayüz sağlanmamaktadır. Araştırmacılar modeli oluşturmak ve eğitmek için veritabanı sorguları yazmak zorundadır.

NeuroOracle (Schikuta ve Glantschnig, 2007) çalışmasıyla NeuDB çalışmasındaki yöntemi Oracle veritabanında gerçekleştirilmiştir. NeuroOracle'da da YSA varlıklarını ifade eden nesneler bir kez tanımlanmalıdır. YSA modeli oluşturmak ve eğitmek için veritabanı sorguları çalıştırılmak zorundadır. Araştırmacılara görsel bir arayüz sunulmamakta ve eğitim süreçleri görsel olarak gözlemlenememektedir.

NeuroAccess (Schikuta, Brunner ve Schultes, 1998a) çalışmasında NeuDB'de ve NeuroOracle'da gerekliliği vurgulanan bir ön yüz programı içermektedir. Bu çalışma ile veritabanında tutulan yapay sinir ağlarının yapısı görselleştirilmekte, eğitimin her adımındaki katman çıktıları liste şeklinde incelenebilmektedir. Ancak görsel olarak YSA oluşturulamamaktadır. Schikuta (2002) yaptıkları NeuroWeb çalışmasıyla veritabanlarında tutulan YSA'lara çevrimiçi erişim yöntemi sağlanmıştır. YSA'lara hem yerel bilgisayar üzerinden hem de çevrimiçi olarak erişip eğitim gerçekleştirmek mümkündür. NeuroWeb ile YSA oluşturulurken katmanlar kullanılmakta ve temsili olarak görselleştirilmektedir. Katmandaki nöron sayıları ön yüzden değiştirilebilmektedir. Katman bağlantıları sadece tam bağlı olabilmekte ve katman sayısı değiştirilememektedir. NeuroWeb çalışması ile YSA konusuna hakim olmayan kişilerin dahi kolaylıkla YSA modeli oluşturabilmesinin modellerin nesne yönelimli olarak veritabanlarında tutulması ile olacağına değinilmiştir.

N2Cloud (Huqqani ve diğerleri, 2010) çalışmasında YSA saklayan ve eğitimini gerçekleştiren sunucuları bulut ortamında tutarak ölçeklenebilir ortam sağlamışlardır. Böylece modelleri çevrime ihtiyaç duymadan paylaşabilmek mümkün olmuştur. N2Sky (Schikuta ve Mann, 2013) çalışmasıyla ilişkisel veritabanlarındaki statik yapının dışında büyük veride sıklıkla kullanılan NoSQL döküman veritabanı olan MongoDb kullanılmıştır. Böylece genelleştirilebilir veri saklama şeklinin yapı bağımsız çalışması gerekliliğini göstermişlerdir. Mongodb'de veriler yarı yapılandırılmıştır. İlişkisel veritabanlarındaki tablolar Mongodb'de koleksiyonlara denk gelmektedir. Koleksiyon içindeki her bir veri döküman olarak ifade edilmektedir. Koleksiyonlarda YSA varlıkları için özellikler ön tanımlı olsa da farklı türde nöron verisi ve bağı içeren varlıklar saklanabilmektedir. Bu yaklaşım YSA'lara ilişkisel veritabanlarından daha genelleştirilebilir bir saklama ortamı sunmaktadır.

N2Sky (Schikuta ve Mann, 2013) ile araştırmacıların kendi üyeliklerine girerek araştırma gruplarını ayırmıştır. Ancak tez çalışmasında önerilen sistemdeki gibi

araştırmacıların eğitim süreçlerini aynı anda görüntülemeye imkan vermemektedir. N2Sky ile eğitim ve test süreçlerinin sonucu araştırma grubu tarafından görüntülenebilmektedir.

YSA süreçlerinde önemli bir yeri olan öğrenim aktarımına karşılaştırılan hiç bir çalışmada değinilmemiştir. Önerilen yöntem ile hem versiyonlama hem de öğrenim aktarımı yapılabilmektedir. Önerilen yöntem ile kodlama ya da sorgu yazmadan herhangi bir anda modelin mimarisi, bağların ve nöronların özellikleri görsel olarak değiştirilebilmektedir. Görsel model değişikliği yapılabilmesiyle nörondaki ve bağdaki veriler kaybolmadan yeni mimari tasarlanabilmektedir. Tablo 3'te bu bölümde anlatılan karşılaştırmalar özetlenerek sunulmuştur.

Tez çalışmasında oluşturulan araçlarda çizge veritabanı kullanılmaktadır. Çizgede tasarlanan düğüm ya da ilişkilerde özellikler ve özellikte tutulacak veriler kullanıcıya bırakılmıştır. Böylece çizgelerde tasarlanan YSA model varlıkları olan hücre ve bağlantılarının zorunlu bir veri yapısı içermesi gerekliliği bulunmamaktadır. İlişkisel veritabanlı çalışmalarda YSA model varlıklarının tüm özellikleri tanımlanarak yapılandırılmaktadır. Bu tür YSA model saklaması yapmak statik veri yapısı olarak adlandırılmaktadır. İlişkisel olmayan ve çizge veritabanlarında tutulan model varlıklarında ise yapılandırma zorunlu değildir ve ilgili YSA model yapısı her an değiştirilebilmektedir. Bu yüzden çizge veritabanlarında veri yapısı dinamiktir. Herhangi bir YSA modelinin sistemde bulunan diğer modellerden farklı özellikler ve bağlantılar içerebilmesiyle farklı tür YSA model süreçlerinin işletilmesi sağlanmıştır. Ancak kullanıcılar sistemde tanımlı olmayan mimari tipinde bir model geliştireceklerinde eğitim ve test süreçlerinin bir kez kodlanması gerekmektedir. Her çeşit YSA mimarisinde tahminleme ve öğretim teorik olarak tez çalışmasında sunulan yöntem ile gerçekleştirebilmektedir. Önerilen yöntem ile yeni bir mimari gerçekleştirileceğinde YSA modelini tekil tanımlayan bir özellik ve katmanları ayırt edecek etiketler tanımlamak gereklidir. Önerilen yöntem ile diğer çalışmaların aksine tüm model görsel olarak tasarlanabilmektedir, model oluşturmak için veritabanı sorguları yazmaya gerek yoktur. Diğer çalışmalardan farklı olarak önerilen yöntemde eğitim ve test sürecinde hesaplama sonuçları hücrelerde anlık olarak model çizgesi üzerinden izlenebilmektedir. İlişkisel veritabanlarında çoktan çoka bağı olan hücreler arasında fazladan bir tabloya ihtiyaç duyulmaktadır. Bu yöntem araştırmacının veritabanındaki verilere bakarak gözlem yapmasını zorlaştırmaktadır. Çizge veritabanında sorguların kendisi ve sonuçları doğal olarak görselleştirmeye yatkındır. Böylece gözlem yaparken model varlıkları görsel olarak gözlemlenebilirken aynı zamanda hücrelerdeki hesaplama sonuçları gözlemlenebilmektedir. Geliştirilen yöntem ile ön yüz sunucusu, arka yüz sunucusu ve çizge veritabanı olarak üçe ayrılan önerilen sistem ile fazla yük alan

sunucu kaynakları arttırılarak hesaplama ölçeklenebilir hale getirilebilmektedir.

Tablo 3
Veritabanında Saklanan Yapay Sinir Ağları İle Önerilen Çizge Veritabanlı Ağların Karşılaştırılması

Özellik	Çizge Veritabanlı YSA	N2Sky	N2Cloud	NeuroWeb	NeuroAccess	NeuroOracle	NeuDB
Model Görselleştirme	Var	Katman Seviyesinde	Yok	Var	Var	Yok	Yok
Model Saklama Veritabanı	Çizge	İlişkisel ve İlişkisel Olmayan	İlişkisel	İlişkisel	İlişkisel	İlişkisel	İlişkisel
Model Varlık Yapılandırması	Dinamik	Dinamik	Statik	Statik	Statik	Statik	Statik
Görsel YSA Modeli Oluşturma	Var	Yok	Yok	Yok	Yok	Yok	Yok
Kodlama Yapmadan YSA Modeli Oluşturma	Var	Yok	Yok	Yok	Yok	Yok	Yok
Model Süreçlerini Gözlemleme	Model Üzerinden	Metriklerle	Metriklerle	Metriklerle	Metriklerle	Metriklerle	Yok
Hesaplama Ölçeklenebilirliği	Arka yüz sunucusu ile	Var	Var	Yok	Yok	Yok	Yok
Kullanıcı Seviyesinde Eş Zamanlı Model Çalıştırılması	Var	Kısmen	Kısmen	Kısmen	Yok	Yok	Yok
Mimari Bağımsız YSA Modeli Oluşturma	Var	Var	Var	Var	Var	Var	Var
Model Aktarımı	Var	Yok	Yok	Yok	Yok	Yok	Yok
Model Versiyonlama	Var	Yok	Yok	Yok	Yok	Yok	Yok
Modellerin Düzenlenmesi	Görsel	Kodlamayla	Kodlamayla	Kodlamayla	Kodlamayla	Kodlamayla	Kodlamayla

4.5. Sunulan Yöntem ile Bir Modelin Tahminleme Süresinin Deneysel Olarak Ölçülmesi

Çalışmanın öncelikli hedeflerinden birisi hesaplama performansı olmamasına karşın YSA model süreçleri geliştirilen yazılım üzerinde gerçekleştirilirken kabul edilir sürelerde sonuçlar üretmesi beklenmektedir. Bu bölümde kabul edilir sürede çıktı üretme denemeleri deneysel olarak karşılaştırılmıştır. Bölüm 3.7.1'de incelenen ileri beslemeli görsel tanıma modeli, geliştirilen yazılımda ve Keras üzerinde tahminleme çıktı üretme süreleri dikkate alınarak gerçekleştirilmiştir.

Karşılaştırmalar Intel i7-7700HQ işlemcisine sahip ve 16 GB DDR4 bellekli, Nvidia 1050Ti ekran kartlı bir dizüstü bilgisayarda yapılmıştır. Tahminleme girdisi olarak 28 piksele 28 piksellik rakam görseli kullanılmıştır. Keras'ın 2.3.0 sürümü ile uyumlu ekran kartı destekli Tensorflow kütüphanesinde arka arkaya yapılan on denemenin sonuçları elde edilmiştir.

Keras ile olan denemelerde Tensorflow'un ekran kartı ile işlem yapan versiyon kullanılmıştır. Kerasla yapılan her denemede, kod üzerinden modelin dönüşümleri yapılarak Keras'a yüklenmesi, ekran kartına dağıtılması gerekmektedir. Önerilen yazılımda YSA verisi üzerinde modelin tümünü kapsayacak bir dönüşüm işlemine gerek olmamakla birlikte sadece hesaplamalar için işleme ihtiyaç bulunmaktadır.

Önerilen sistemde ilk sorgular önceki bölümlerde anlatılan ön ısıtma konusundan dolayı yavaş çalışmakta ancak her sorgu geldiğinde daha hızlı cevaplar dönmeye başlamaktadır. Keras tarafında ortalama korunmaktadır. Ekran kartının o anki yüküne göre artmalar olmakla birlikte genel olarak ortalamaya yakın tahminleme sonuç zamanları elde edilmektedir. Önerilen sistemde ön yüzden giden internet isteğinden tahminleme sonucu elde edilene kadarki süre ölçümü yapılmıştır. Önerilen sistemde ekran kartı üzerinden herhangi bir işlem yapılmamasına rağmen model çevrimi veya ekran kartına aktarımı için dönüşüm ihtiyacı bulunmadığı için çok az bir fark ile hızlı sonuç ürettiği gözlemlenmiştir. Deney sonucunda kabul edilebilir sürede tahminleme çıktısı üretilmiştir. Yapılan testlerin sonucunda önerilen yazılımda en yavaş 3.76 saniyede, en hızlı ise 3.09 saniyede çıktı üretilmiştir. Keras üzerinde gerçekleştirilen deneylerde en yavaş 3.66 saniyede, en hızlı ise 3.26 saniyede çıktı üretilmiştir.

BÖLÜM 5
SONUÇ VE ÖNERİLER

Tez çalışması ile yapay sinir ağlarını çizge veritabanlarında çalıştırmak için yeni bir yöntem geliştirilmiştir. Geliştirilen bir yazılım ile araştırmacıların yapay sinir ağlarını çizgelerle görsel olarak oluşturabilmeleri sağlanmıştır. Geliştirilen yazılım ile kullanıcılar çizge veritabanında saklanan YSA modellerine çevrimiçi olarak her yerden erişebilmektedir. Çizge veritabanında tutulan yapay sinir ağ modelleri kodlama gerekmeden görsel olarak düzenlenebilmektedir. Sunulan yöntem ile modellerdeki eğitim ve test süreçleri canlı olarak gözlemlenebilmektedir. Araştırma gruplarının aynı model üzerinde ortaklaşa çalışabilmesi sağlanmıştır. Eğitimi tamamlanmış modelleri dosya formatlarına dönüştürmeye gerek kalmadan paylaşmak mümkün hale getirilmiştir.

Yapay sinir ağı ile test ve eğitim yapılacak verilerin aynı ortamda olması sağlanmıştır. Günümüzde veri saklarken sıklıkla kullanılan veritabanı verilerini veri seti olarak kullanmak için ortam oluşturulmuştur. Önerilen yöntem ile yapay sinir ağları teoride anlatıldığı şekilde birebir çizge veritabanında saklanmaktadır. Çizge veritabanlarında saklanan yapay sinir ağları ile kodlama yapmadan ve dönüşüme ihtiyaç olmadan öğrenim aktarımı yapılabilmektedir. Öğrenim aktarımı yapılmış modeller kodlama yapmadan görsel olarak düzenlenebilmektedir. Böylece diğer yöntemlerdeki ön eğitimli modellere hakimiyet zorunluluğu ve kodlamayla değişiklik yapmak zorunda olunmasının önüne geçilmiştir.

Yaygın kullanılan yapay sinir ağ yazılımlarında bazı eğitilmiş modelleri çizge veritabanına aktarmak için bir yöntem oluşturulmuştur. Evrişimli sinir ağları önerilen sisteme aktarılarak tahminleme gerçekleştirilmiştir. Model mimarisinden bağımsız biçimde YSA varlıklarını birer çizge veritabanı verisi olarak işlenmesi ile genelleştirilebilir bir model saklama ortamı oluşturulmuştur. Çizge veritabanlarında sinir ağlarını tutmak ilişkisel veritabanlarında tutmaya göre daha esnek ve genelleştirilebilir olduğu sonucu elde edilmiştir. Literatürde veritabanında yapay sinir ağları tutmayla ilgili çalışmalar yöntemin gelişime açık yönlerinin olduğunu göstermektedir. Çalışmanın asıl amacı yapay sinir ağlarını performanslı şekilde işletmek olmasa da yaygın kütüphanelerin performansında test hesaplaması gerçekleştirebilmektedir. Tez kapsamında yapay sinir ağlarının çizge veritabanlarında gerçeklenmesiyle genelleştirilebilir bir saklama ortamı oluşturulmuş ve literatüre katkı sunulmuştur. Önerilen yazılım açık kaynak kodlu paylaşılarak geliştirmeye açık haldedir. Önerilen yöntem ile çizge veritabanlı yapay sinir ağları olarak tanımlanan yeni bir ekosisteminin ilk adımları atılmıştır.

Gelecek çalışmalarda çizge veritabanlarında saklanan yapay sinir ağlarındaki

hesaplamaların ekran kartlarında yapılmasıyla performans arttırılabilir. Genelleştirilebilir yayılım algoritmaları çizge veritabanlarına uygulanarak çizge yapısının verimliliği arttırılabilir. Gerçek hayat problemlerinden veri setleri çizge veritabanlarında oluşturularak önerilen yöntemin büyük ve gürültülü veri setlerindeki yetkinliği karşılaştırılabilir.

KAYNAKLAR

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Erişim adresi: https://www.tensorflow.org/ (Software available from tensorflow.org)

Bergtholdt, M., Kappes, J., Schmidt, S. ve Schnörr, C. (2010, 03). A study of parts-based object class detection using complete graphs. *International Journal of Computer Vision*, *87*, 93-117. doi: 10.1007/s11263-009-0209-1

Bianchini, M., Gori, M. ve Scarselli, F. (2005). Inside pagerank. *ACM Transactions on Internet Technology (TOIT)*, *5*(1), 92–128.

Bijari, K., Zare, H., Kebriaei, E. ve Veisi, H. (2020, Apr). Leveraging deep graph-based text representation for sentiment polarity applications. *Expert Systems with Applications*, *144*, 113090. Erişim adresi: http://dx.doi.org/10.1016/j.eswa.2019.113090 doi: 10 .1016/j.eswa.2019.113090

Chollet, F., et al. (2015). *Keras.* https://keras.io.

Cun, Y. L., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D. ve Henderson, D. (1990). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems 2* (s. 396–404) içinde. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Duchi, J., Hazan, E. ve Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Elman, J. L. (1990). Finding structure in time. *Cogn. Sci.*, *14*(2), 179-211. Erişim adresi: http://dblp.uni-trier.de/db/journals/cogsci/cogsci14.html#Elman90

Frosst, N., ve Hinton, G. (2017). *Distilling a neural network into a soft decision tree.*

Ginde, G., Saha, S., Mathur, A., Vamsi, H., Dey, S. R. ve Gambhire, S. S. (2018). Use of nosql database and visualization techniques to analyze massive scholarly article data from journals. *arXiv preprint arXiv:1805.00390*.

Gori, M., Monfardini, G. ve Scarselli, F. (2005). A new model for learning in graph domains. *Proceedings. 2005 ieee international joint conference on neural networks, 2005.* (Cilt. 2, s. 729–734) içinde.

Graph based neural network. (2020). Erişim adresi: https://github.com/dogabaris/GraphBasedNeuralNetwork

Grover, A., ve Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery*

and data mining (s. 855–864) içinde.

Guia, J., Soares, V. G. ve Bernardino, J. (2017). Graph databases: Neo4j analysis. *Iceis (1)* (s. 351–356) içinde.

Hamilton, W. L. (t.y.). Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *14*(3), 1-159.

Hamilton, W. L., Ying, R. ve Leskovec, J. (2017). Inductive representation learning on large graphs. (s. 1025–1035) içinde. Red Hook, NY, USA: Curran Associates Inc.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley. Hardcover.

Holzschuher, F., ve Peinl, R. (2013). Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. *Proceedings of the joint edbt/icdt 2013 workshops* (s. 195–204) içinde.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications.*

Humphreys, P. (2009, 08). The philosophical novelty of computer simulation methods. *Synthese*, *169*, 615-626. doi: 10.1007/s11229-008-9435-2

Huqqani, A. A., Li, X., Beran, P. P. ve Schikuta, E. (2010). N2cloud: Cloud based neural network simulation application. *The 2010 international joint conference on neural networks (ijcnn)* (s. 1–5) içinde.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the 22nd acm international conference on multimedia* (s. 675–678) içinde. New York, NY, USA: ACM. Erişim adresi: http://doi.acm.org/10.1145/2647868.2654889 doi: 10.1145/2647868.2654889

Jordan, M. I. (1989). Serial order: A parallel, distributed processing approach. J. L. Elman ve D. E. Rumelhart (Ed.), *Advances in connectionist theory: Speech* içinde. Hillsdale, NJ: Erlbaum.

Kingma, D. P., ve Ba, J. (2014). *Adam: A method for stochastic optimization.* Erişim adresi: http://arxiv.org/abs/1412.6980 (cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015)

Kipf, T. N., ve Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *CoRR*, *abs/1609.02907*. Erişim adresi: http://arxiv.org/abs/1609.02907

Krizhevsky, A., Sutskever, I. ve Hinton, G. E. (2012). Imagenet classification with

deep convolutional neural networks. F. Pereira, C. J. C. Burges, L. Bottou ve K. Q. Weinberger (Ed.), *Advances in neural information processing systems 25* (s. 1097–1105) içinde. Curran Associates, Inc. Erişim adresi: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Lecun, Y., Bottou, L., Bengio, Y. ve Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi: 10.1109/5.726791

LeCun, Y., ve Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Erişim adresi[2016-01-14 14:24:11]: http://yann.lecun.com/exdb/mnist/

Leung, C., ve Salga, A. (2010). Enabling webgl. *Proceedings of the 19th international conference on world wide web* (s. 1369–1370) içinde. New York, NY, USA: Association for Computing Machinery. Erişim adresi: https://doi.org/10.1145/1772690.1772933 doi: 10.1145/1772690.1772933

Mcculloch, W., ve Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 127–147.

Minsky, M., ve Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA, USA: MIT Press.

Nekhaev, D., ve Demin, V. (2017, 01). Visualization of maximizing images with deconvolutional optimization method for neurons in deep neural networks. *Procedia Computer Science*, *119*, 174-181. doi: 10.1016/j.procs.2017.11.174

Neo4j graph platform. (2007). Erişim adresi: http://www.neo4j.org/

Neuhaus, M., ve Bunke, H. (2004). A probabilistic approach to learning costs for graph edit distance. *Proceedings of the 17th international conference on pattern recognition, 2004. icpr 2004.* (Cilt. 3, s. 389–393) içinde.

Neuhaus, M., ve Bunke, H. (2005). A graph matching based approach to fingerprint classification using directional variance. *Avbpa* içinde.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox ve R. Garnett (Ed.), *Advances in neural information processing systems 32* (s. 8024–8035) içinde. Curran Associates, Inc. Erişim adresi: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Perozzi, B., Al-Rfou, R. ve Skiena, S. (2014). Deepwalk: Online learning of social representations. *Proceedings of the 20th acm sigkdd international conference on*

knowledge discovery and data mining (s. 701–710) içinde.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408.

Rumelhart, D. E., Hinton, G. E. ve Wilson, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. ve Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. ve Monfardini, G. (2009, January). The graph neural network model. *Trans. Neur. Netw.*, *20*(1), 61–80. Erişim adresi: https://doi.org/10.1109/TNN.2008.2005605 doi: 10.1109/TNN.2008.2005605

Schikuta, E. (2002). Neuroweb: an internet-based neural network simulator. *14th ieee international conference on tools with artificial intelligence, 2002.(ictai 2002). proceedings.* (s. 407–412) içinde.

Schikuta, E. (2008). Neural networks and database systems. *arXiv preprint arXiv:0802.3582*.

Schikuta, E., Brunner, C. ve Schultes, C. (1998a). The neuroaccess system. *International conference on artificial neural networks* (s. 1183–1188) içinde.

Schikuta, E., Brunner, C. ve Schultes, C. (1998b). A relational neural network database model. *Nc* (s. 937–943) içinde.

Schikuta, E., ve Glantschnig, P. (2007). Neurooracle: Integration of neural networks into an object-relational database system. *International symposium on neural networks* (s. 1115–1124) içinde.

Schikuta, E., ve Mann, E. (2013). N2sky—neural networks as services in the clouds. *The 2013 international joint conference on neural networks (ijcnn)* (s. 1–8) içinde.

Schikuta, E., et al. (1996). Neudb'95: An sql based neural network environment. *Progress in neural information processing, proc. int. conf. on neural information processing, iconip* (Cilt. 96, s. 1033–1038) içinde.

Simonyan, K., ve Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*. Erişim adresi: http://arxiv.org/abs/1409.1556

Smilkov, D., Carter, S., Sculley, D., Viégas, F. B. ve Wattenberg, M. (2017). Direct-manipulation visualization of deep networks. *arXiv preprint arXiv:1708.03788*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. *Computer vision and pattern recognition (cvpr)* içinde. Erişim adresi: http://arxiv.org/abs/1409.4842

The HDF Group. (2000). *Hierarchical data format version 5.* Erişim adresi: http://www.hdfgroup.org/HDF5

Tompson, J., Jain, A., LeCun, Y. ve Bregler, C. (2014). *Joint training of a convolutional network and a graphical model for human pose estimation.*

Varda, K. (2008, 6). *Protocol buffers: Google's data interchange format* (Tech. Rep.). Google. Erişim adresi: http://google-opensource.blogspot.com/2008/07/protocol-buffers-googles-data.html

Yahia, M. E., ve Elsawi, A. M. (t.y.). Neural database model.

Yao, Y., ve Holder, L. B. (2015). Scalable classification for large dynamic networks. *2015 IEEE International Conference on Big Data (Big Data)*, 609-618.

Zednik, C. (2019, 12). Solving the black box problem: A normative framework for explainable artificial intelligence. *Philosophy and Technology*. doi: 10.1007/s13347-019-00382-7

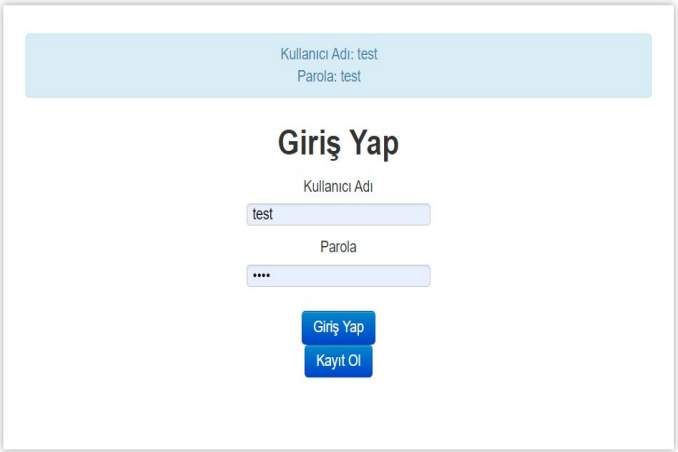
EKLERİ

EK 1. Önerilen Yazılımın Diğer Görselleştirme Arayüzleri

Şekil 31'de araştırmacıların kayıt olma ekranında araştırmacıdan kullanıcı adı, ad, soyad, parola bilgilerinin girilmesi istenmektedir.

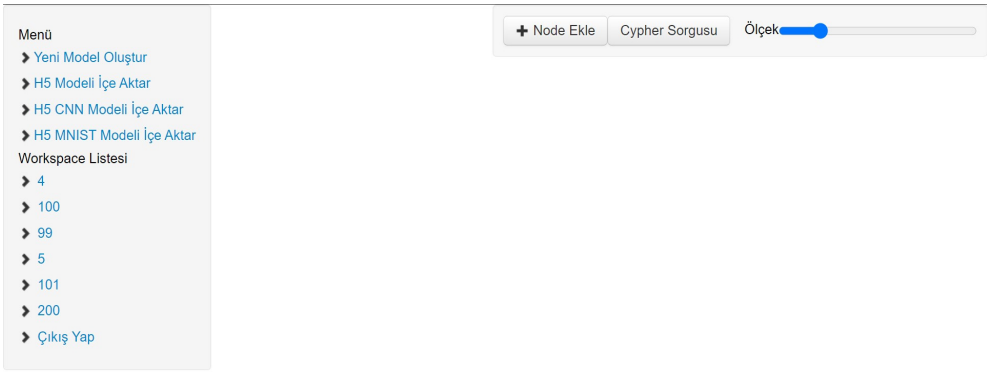
Şekil 31. Araştırmacının sisteme kayıt olma ekranı

Bilgilerle kayıt olan kullanıcı Şekil 32'de kullanıcı adı ve parolası ile ortak çalışma sistemine giriş yapabilmektedir. Örnek olarak, projenin hızlıca incelenebilmesi adına "test" kullanıcı adı ve "test" parolasıyla kullanıcı oluşturulmuştur.



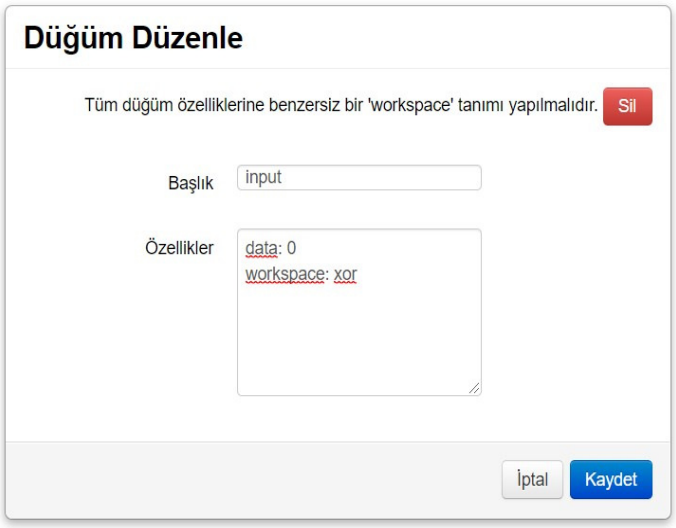
Şekil 32. Araştırmacının sisteme giriş ekranı

Şekil 33'te ortak çalışma ekranı gösterilmiştir. Site üzerinden ulaşılan bölümler temel olarak bu şekilde 3 ana bölüme ayrılmaktadır. Beyaz alanda model düzenlemeleri ve gösterimi görsel olarak yapılırken, sağ üstteki menüden model işlemleri, soldaki menüden ise örnek modellerin sisteme aktarımı, yeni model oluşturulması, modellerin listelenmesi yapılmaktadır.



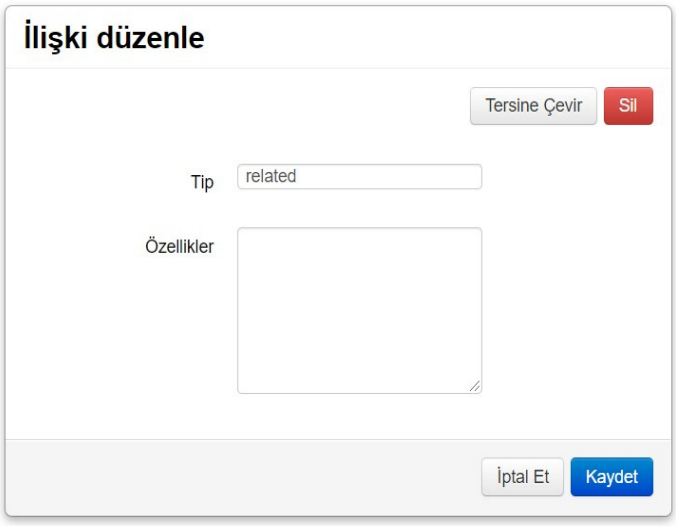
Şekil 33. Araştırmacıyı giriş yaptıktan sonra karşılayan ortak çalışma alanı

Şekil 34'te düğüm düzenleme ekranı üzerinde örnek "Başlık", "Özellikler" parametrelerinin girişi gösterilmiştir. "Kaydet" dendikten sonra görsel YSA oluşturucusu ilgili olan nörondaki verileri değiştirmektedir.



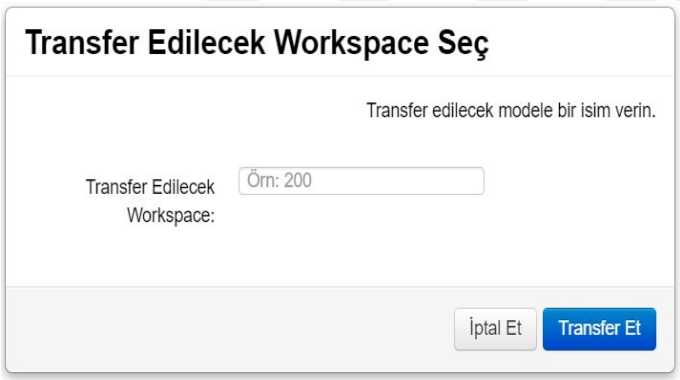
Şekil 34. Düğüm düzenleme ekranı

Şekil 35'te "Tip" ve "Özellikler" parametrelerinin örnek girişleri gösterilmiştir. "Tip" bölümü etiket olarak çalışmaktadır. "Özellikler" bölümünde benzersiz bir "workspace" verisinin girilmesine ihtiyaç bulunmamaktadır. "Özellikler" bölümünde genellikle ağırlıklar, sapma ve kernel değerleri yer almaktadır. Düğüm düzenle ekranıyla aynı mantıkta nöronlar arasındaki ilişkilere çift tıklama yapılarak erişilmektedir.



Şekil 35. İlişki düzenleme ekranı

Şekil 36'da öğrenim aktarımı yapılacak yeni modelin workspace ismi sorulmakta ve hali hazırdaki modelin bilgileri ile yeni model oluşturulmaktadır. Böylece yeni modelde öğrenim aktarımı yapılmış olan modelin hem yapısı hem de öğrendiği parametreler aktarılmış olmaktadır.



Şekil 36. Öğrenim aktarım ekranı

EK 2. H5 Model Yapısı ve Aktarımı

H5 dosyaları "groups" ve "datasets", "root", "apiVersion" anahtar değerlerini içermektedir. "groups" bölümü katmanların mimarisi, üst bilgileri ve birbirlerine olan bağlantılarını içermektedir. "datasets" bölümü katmanlardaki işlem matrislerini ve katman verilerini içermektedir. "root" bölümü YSA'nın başladığı düğümün GUID formatında kimlik bilgisini içermektedir, bu bölüm kullanılarak YSA modelinin başladığı katmandan itibaren ilerlenerek taraması yapılmakta ve Cypher sorgusuna çevrilmektedir. Cypher sorgusunda olduğu gibi her katmanın kendine özel tekil GUID kimlik numaraları bulunmaktadır. YSA model bağlantıları ağaç yapısına benzer şekilde tekillik sağlayan katmanlar arasında GUID değerler ile yapılmaktadır. "groups" bölümünde YSA modelinin her katmanının adı listelenmektedir ancak sırası her zaman doğru değildir. Sıra farkından dolayı Cypher sorgusu oluşturulurken, "groups" alanındaki liste rehber liste olarak kullanılmakta ancak YSA'yı ve ilişkilerini oluşturacak sorgu oluşturulurken, "root" alanından başlayarak bağlı olduğu katmanların sırası ile geliştirilen yazılım çatısı ile çizge veri tabanında oluşturulmaktadır.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Doğa Barış Özdemir

Doğum Yeri :

Doğum Tarihi :

EĞİTİM DURUMU

Lisans Öğrenimi : Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 2017

Yüksek Lisans Öğrenimi : Çanakkale Onsekiz Mart Üniversitesi, Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, 2020

Bildiği Yabancı Diller : İngilizce

BİLİMSEL FAALİYETLERİ

a) Yayınlar

1) SCI

b) Bildiriler

1) Uluslararası

2) Ulusal

The Use of Graph Databases for Artificial Neural Networks, Çanakkale Onsekiz Mart University Journal of Graduate School of Natural and Applied Sciences

Sıcaklık Nem Endeksi (SNI) Kullanılarak Arduino Tabanlı Düşük Maliyetli Bir Klima Otomasyon Cihazının Tasarımı, Türk Tarım ve Doğa Bilimleri Dergisi

Görüntülerdeki Araba Nesnelerinin Belirlenmesi İçin Derin Öğrenme ile Bir Model Eğitilmesi, Akademik Bilişim Konferansı 2018

Bir E-Burun Sisteminin Arduino Tabanlı Dönüşümünün Yapılması, 1st International Congress on Agricultural Structures and Irrigation

İŞ DENEYİMİ

Çalıştığı Kurumlar ve Yıl : Argelog, 2017-2019

Shopi, 2019-2020

Hepsiburada, 2020-Halen

İLETİŞİM

E-posta Adresi

ORCID :