Review Article
Journal of Smart Systems Research (JOINSSR) 3(2), 97-119, 2022
Received: 4-November-2022 Accepted: 25-November-2022

Nesne Tespiti İçin Derin Öğrenme Kütüphanelerinin İncelenmesi

Süleyman AKTÜRK [1], Kasım SERBEST[2*]

[1] Mekatronik Mühendisliği Bölümü, Lisansüstü Eğitim Enstitüsü, Sakarya Uygulamalı Bilimler Üniversitesi, Türkiye.
[2] Mekatronik Mühendisliği Bölümü, Teknoloji Fakültesi, Sakarya Uygulamalı Bilimler Üniversitesi, Türkiye.

ÖZ

Yapay zekâ kavramı 1900'lü yılların ikinci çeyreğinden itibaren hayatımıza girmeye başlamıştır. Bugüne kadar ki süreçte bazen çok popüler olmuş, bazen de unutulmaya yüz tutmuştur. 2000'li yıllarda bilgisayar donanımlarının gelişmesi ve yapay sinir ağlarındaki gelişmeler Yapay zekâ araçlarının tekrardan yoğun kullanılmasına sebep olmuştur. Derin öğrenme çalışmaları yapay zekâ çalışmalarının lokomotifi konumundadır. Derin öğrenme sadece bilgisayar biliminde ve nesne algılama görevinde değil, birçok disiplinde ve birçok görevde kullanılmaktadır. Derin öğrenme çalışmaları için farklı mimariler, farklı donanımlar, farklı yazılım çerçeveleri geliştirilmiştir. Derin öğrenme araçlarının ve mimarilerinin bu kadar fazla olması özellikle problemlerinin çözümü için derin öğrenme araçlarını kullanmak isteyen araştırmacıların kafasını karıştırabilmekte veya işini zorlaştırabilmektedir. Bu çalışmada derin öğrenme kavramı hakkında bir anlayış oluşturmayı hedefliyoruz. Derin öğrenme kavramını ve onu oluşturan alt bileşenleri sistematik olarak sunuyoruz. Akabinde günümüzde yaygın olarak kullanılan ana akım derin öğrenme çerçevelerini sunuyoruz. Bu çerçevelerin performans, zaman, doğruluk, Google da aranma, GitHub'da takip edilme gibi farklı ölçütlere göre incelendiği araştırmalardan bir derleme hazırlanmıştır. Bu çalışmanın özellikle derin öğrenme araçlarına aşina olmayan ancak çalışmalarında derin öğrenme araçlarını kullanmak isteyen araştırmacı ve okuyucular için bir kılavuz olmasını arzu ediyoruz.

Anahtar Kelimeler: Makine öğrenmesi, yapay sinir ağları, derin öğrenme kütüphaneleri

A Review of Deep Learning Libraries for Object Detection

ABSTRACT

The idea of artificial intelligence has been in our lives since the second quarter of the 20th century. Up to now, it has enjoyed great popularity in some cases, and in others it has been forgotten. In the 2000s, the development of computer hardware and developments in artificial neural networks led to an extensive use of artificial intelligence algorithms. Deep learning studies are the pioneers of artificial intelligence studies. Deep learning is not only used in computer science and object detection, but also in many disciplines and many tasks. Different architectures, hardware and software frameworks have been developed for deep learning studies. The fact that there are so many deep learning tools and architectures can confuse the work of researchers who want to use deep

* Sorumlu yazarın e-posta adresi: kserbest@subu.edu.tr

learning tools to solve their problems. In this study we want to create an understanding of the concept of deep learning. We present the concept of deep learning and its subcomponents systematically. We then introduce the common deep learning frameworks that are widely used recently. A review was created from the studies in which these frameworks were examined according to different criteria such as performance, time, accuracy, searched on Google, followed by GitHub. In this study, we would like to be a guide for researchers and readers who are unfamiliar with deep learning algorithms but want to use deep learning tools in their studies.

Keywords: Machine learning, neural network, deep learning libraries

1 Giriş

Yapay zekâ uygulamaları hava durumu tahmini, ekonomik göstergeler, günlük haberlerin yazılması, yeni sanat eserlerinin oluşturulması, kişisel sağlık takibi, mühendislik hesaplamaları, görüntü işleme gibi birçok alanda karşımıza çıkmaktadır. Bu uygulamalardan biri de nesne tespitidir. Nesne tespiti bir görüntü içindeki nesnenin belirginleştirilerek zeminden ayırt edilmesi işlemidir [1]. Diğer bir tanımla nesne tespiti görüntüde bulunan belirli nesnelerin konum ve sınıflarının belirlenmesidir. Nesne tespitinde nesneler genellikle dikdörtgen çerçevelerle işaretlenmektedir [2]. Nesne tespiti temel bir bilgisayarlı görü görevidir. Bu göreve dayalı olarak yüz tanıma, yaya tanıma, insan uzuvlarının tespiti, nesne takibi, yüzey alanı tespiti vb. birçok gerçek dünya problemi çözülmektedir. Tüm bu uygulamaların temelinde iyi bir nesne algılama algoritması bulunmaktadır.

İnsan beyni görüntüleri kolaylıkla tespit edebilmektedir. Ancak bir makine olan bilgisayar için binlerce pikselden oluşan görüntü içinden nesne tespiti bir dizi karmaşık süreçlerle gerçekleşmektedir. Bilgisayarlı görü uygulamalarında nesne tespit uygulamaları sığ ve derin nesne tespit uygulamaları olarak ele alınabilir. Genellikle makine öğrenmesi metotlarına dayalı sığ yöntemler Histogram Yönelimli Gradyan (HOG), Ölçekten Bağımsız Öznitelik Dönüşümü (SIFT), Destek vektör makineleri (SVMs), Adaptive Boosting (Ada-Boost) gibi sınıflandırıcılar kullanılmaktadır [3]. Derin öğrenme uygulamalarından önce nesne algılama uygulamaları teklif oluşturma, özellik vektörü çıkarma, bölge sınıflandırması aşamalarından oluşmaktaydı. Bu aşamalar uygulayıcı tarafından düzenlenmekteydi. Derin öğrenme ile klasik nesne algılama sürecindeki özelliklerin elle oluşturulması işi özellik oluşturucular tarafında gerçekleştirilmeye başlamıştır. Derin öğrenme yöntemlerinin 2012 yılından itibaren nesne tespit uygulamalarındaki başarı oranı giderek artmaktadır [4]. Derin öğrenme makine öğrenmesinin bir alt dalı olarak veri temsillerini öğrenebilen çok katmanlı ağlar olarak ifade edilir. Veri temsillerinin oluşturulması aşamasında derin öğrenme ağları içerisinde birden çok katmanda birden çok işlem gerçekleştirilmektedir [5].

Bu çalışmada öncelikle yapay zekâ, makine öğrenmesi ve derin öğrenme kavramları hakkında temel bilgiler verilerek okuyucular için bir ön hazırlık yapılmıştır. Daha sonra derin öğrenme çerçeveleri üzerine detaylı bilgiler verilmiş ve bu alandaki güncel araştırma sonuçları sunulmuştur. Bu çalışmanın birincil amacı derin öğrenme alanındaki yeni araştırmacılara ve konuya ilgi duyan okuyuculara derin öğrenme yönteminin temel prensiplerini aktararak nesne tespitine yönelik farklı algoritmalar hakkında bilgiler sunmaktır. Çalışmanın ikincil amacı ise derin öğrenme ile nesne tespiti alanında Türkçe bir kaynak oluşturmaktır. Derin öğrenme alanına yönelik Türkçe olarak hazırlanmış derleme kaynak sayısı son derece sınırlıdır. Hatta yaptığımız araştırmalara göre derin öğrenme ile nesne tespiti alanında son yıllarda hazırlanmış bir çalışma bulunmamaktadır. Çalışmanın ilerleyen bölümlerinde yapay sinir ağları, derin öğrenme modelleri ve derin öğrenme kütüphaneleri hakkında sistematik bilgiler sunulmaktadır.

2 Metodoloji

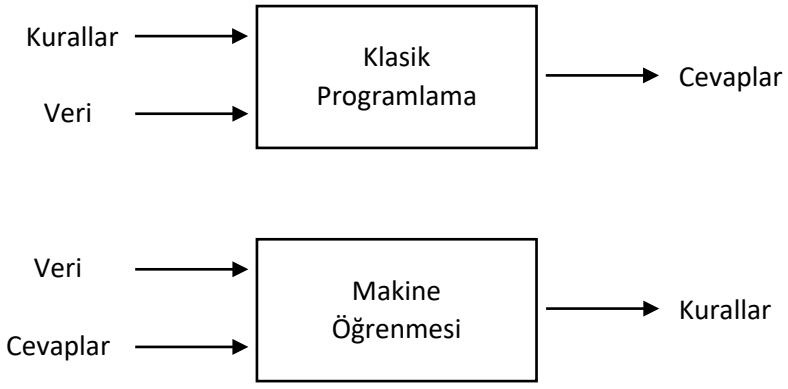
Derin öğrenme -özellikle de derin öğrenme ile nesne tespiti- alanında yapılan çalışmalara ulaşmak için Web of Science, Scopus, IEEE Xplore, Google Akademik ve Yök Akademik veri tabanları kullanılmıştır. Bu veri tabanlarında hem Türkçe hem de İngilizce olarak derin öğrenme (deep learning), nesne tespiti (object detection), duruş tahmini (pose estimation), derin öğrenme kütüphaneleri-algoritmaları (deep learning libraries-algorithms) gibi anahtar kelimeler ve bunların anlamlı kelime grupları ile araştırma yapılmıştır. Özellikle son beş yılda yapılan çalışmalar detaylı olarak incelenmiş ve sonuçları sistematik olarak derlenmiştir. Bu bölümde yapay zekâ, makine öğrenmesi, yapay sinir ağları, derin öğrenme, derin öğrenme katmanları, evrişimli sinir ağları, derin öğrenme modelleri ve derin öğrenme kütüphaneleri hakkında farklı çalışmalardan derlenen bilgiler yer almaktadır.

2.1 Yapay Zekâ

Yapay zekâ (AI) 1950'li yıllardan itibaren belirli duraklama evrelerini aşarak günümüze gelmiştir. Yapay zekanın çıkışı "Bilgisayarlar düşünebilir mi?" sorusuna cevap arama girişimiyle ilişkilidir. Yapay zekâ "Normalde insanlar tarafından yerine getirilen düşünsel faaliyetlerin otonom hale getirilmesi" olarak tanımlanabilir [6]. Bu bağlamda insan gibi düşünen, insan gibi davranışlar sergileyen uygulamalara yapay zekâ uygulamaları ismi verilebilir [7]. Yapay zekaya sahip sistemlerin insana özgü akıl yürütme, anlam çıkartma, genelleme yapma ve geçmiş deneyimlerden öğrenme gibi yüksek seviye zihinsel aktiviteleri yerine getirmesi beklenir. Problem çözümleme, oyunların modellenmesi, bilgilerin modellenmesi, uzman sistemler, doğal dil işleme, ses işleme, örüntü tanıma, makine öğrenmesi ve derin öğrenme, bilgisayar yaratıcılığı, robotik gibi konular yapay zekanın ilgi alanı olarak ifade edilebilir [8]. Uzman Sistemler, Bulanık Mantık, Genetik Algoritmalar, Yapay Sinir Ağları gibi yapay zekanın alt dalları, birçok problem alanında son yıllarda çok sık kullanılmaktadır [9].

2.2 Makine Öğrenmesi

Klasik programlamada yazılımcı bilgisayara ne yapacağını söyler. Klasik yapay zekâ uygulamalarında ise veri ve kurallar sisteme yüklenir ve sistemin bu verilere ve kurallara dayanarak sonuç üretmesi beklenir. Makine öğrenmesi (ML) tüm bu yöntemleri kökünden değiştirecek bir paradigma sunmaktadır. Makine öğrenmesi veri ve beklenen çıktıları kullanarak, yeni verilere özgün cevaplar üretebilecek kuralları öğrenebilen sistemler kurgular [6]. Makine öğrenmesi; bilgisayarın öğrenmesi ve sonrasında da tahminde bulunabilmesi için en uygun algoritmayı oluşturmayı amaçlamaktadır. Bu algoritmalar klasik bilgisayar programlaması için oluşturulan algoritmalar gibi sıralı talimatları uygulamak yerine eğitim setinde karşılaştığı örneklerden öğrenip genelleme yaparak öğrendiği bilgiyi yeni durumlara uyarlayabilmektedir [10]. Şekil 1'de makine öğrenmesi yaklaşımı ve klasik programlama arasındaki temel farklılık gösterilmektedir.



Şekil 1: Makine öğrenmesi ve klasik programlama arasındaki fark [6]

Makine öğrenmesi; gözetimli, gözetimsiz ve yarışmacı olarak üçe ayrılabilir. Gözetimli öğrenmede ağın gerçek çıkışı olması gereken çıkışla kıyaslanır. Gerçek çıkışla olması gereken çıkış arasındaki fark en aza indirilmeye çalışılır. Bu şekilde ağırlıklar değiştirilerek sürekli denemeler yapılır. Gözetimsiz öğrenmede ağa giriş verilir ancak çıkış verilmez. Giriş bilgilerine göre ağırlıklarının kendisinin ayarlaması beklenir. Yarışmacı öğrenmede ise girişteki verileri belirli sayıda sınıfa ayırması beklenir [9]. Şekil 2'de farklı makine öğrenmesi yöntemleri yer almaktadır.

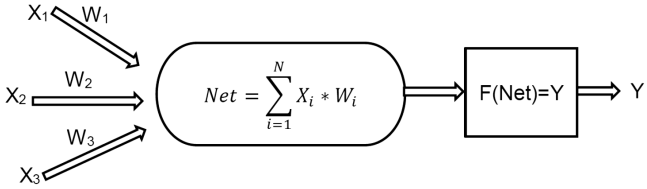
Şekil 2: Makine öğrenmesi yöntemleri [9]

Makine öğrenmesi istatistikle yakından alakalıdır. Ancak birçok yönden ayrılmaktadırlar. Makine öğrenmesi büyük veriyle çalışmaya yatkındır. İstatistik yöntemleri büyük veri karşısında yetersiz kalmaktadır [6]. Makine öğrenmesinde amaç deneyimlerden öğrenmek ve bu yolla performansı iyileştirmektir. Deneyim sayısı ne kadar artarsa öğrenme de o kadar iyi olmaktadır. Dolayısıyla Makine öğrenmesi artan veri miktarıyla daha da önemli bir hal almıştır [11]. Makine öğrenmesinde öğrenilecek olgunun temsilleri sisteme giriş olarak sunulur. Her temsilin çıkışı ne kadar etkileyeceği ağ tarafından belirlenir. Burada temsilcileri iyi belirlemek önemlidir. Klasik makine öğrenmesinde temsilciler elle oluşturulmaktadır [4].

2.3 Yapay Sinir Ağları

Yapay sinir ağları (YSA) insan beynindekine benzer sinir hücrelerinin birbiri ile ilişkilendirilmesiyle oluşur. En temel insani özelliklerden olan öğrenme yeteneğine sahiptir. Her proses elemanı birbirine ağırlıklarla bağlanmıştır. Öğrenme dediğimiz olgu bu ağırlıkların ağ tarafından değiştirilerek en doğru sonucu verecek hale getirilmesidir. Sonuçta girdilere göre ağırlıklar eğitilerek yeni durumları öğrenmesi sağlanır [12].

Bir yapay sinir ağı yapısı genel olarak giriş katmanı (girdi değerleri ya da öznitelikler), gizli katmanlar ve çıktı katmanı olmak üzere üç ana bölümden oluşur. Girdi katmanında başka bir sinir ağından gelen çıktı değerleri ya da işleme tabi tutulacak nesneye ait öznitelikler bulunur. Gizli katmanlar bu öznitelikleri ve belirli ağırlık değerlerini kullanarak özellik haritası çıkarırlar. Bu özellikler aktivasyon fonksiyonları ile çıktı katmanında anlamlı bir sonuç üretilmesini sağlarlar [10]. Şekil 3' te verilen model incelendiğinde x girişleri w ağırlıklarıyla çarpılarak toplam fonksiyonundan geçirilmektedir. Elde edilen toplam F(net) transfer fonksiyonuyla çıkışa aktarılmaktadır [7].



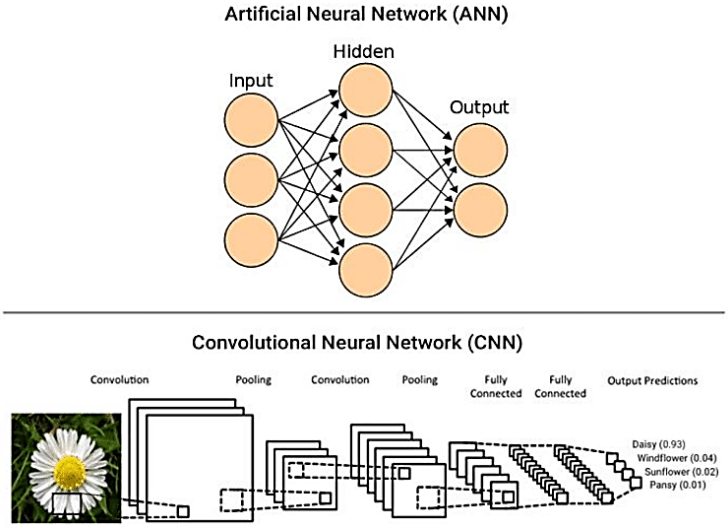
Şekil 3: Bir sinir hücresinin matematiksel modeli [7]

2.4 Derin Öğrenme

Derin öğrenme (DL), birden çok işleme katmanından oluşan hesaplama modellerinin, birden çok soyutlama düzeyiyle verilerin temsillerini öğrenmesine olanak tanır. Bu yöntemler, konuşma tanıma, görsel nesne tanıma, nesne algılama ve ilaç keşfi ve genomik gibi diğer birçok alanda son teknolojiyi önemli ölçüde geliştirmiştir. Derin öğrenme, bir makinenin önceki katmandaki temsilden her katmandaki gösterimi hesaplamak için kullanılan dahili parametrelerini nasıl değiştirmesi gerektiğini belirtmek için geri yayılım algoritmasını kullanarak büyük veri kümelerindeki karmaşık yapıyı keşfeder [13].

Derin öğrenme klasik makine öğrenmesindeki veri temsillerini elle oluşturma yükünden kurtulmaya imkân vermiştir. Derin öğrenme, birbirini takip eden katmanlarda veriler işlenirken giderek artan şekilde daha kullanışlı temsiller elde ederler. Derin öğrenmenin özellik çıkarım yükünü üzerine alması, onun popüler olmasını sağlamıştır. Günümüzde derin öğrenme çalışmaları hızla artmaktadır [14]. Büyük veri setlerinin artması, bilgisayar donanımlarının güçlenmesi derin öğrenme çalışmalarını desteklemektedir.

Derin öğrenmedeki derinlik ifadesi birbirini takip eden katmalardır [6]. Derin öğrenme, derin sinir ağları çok katmanlı yapay sinir ağları olarak da ifade edilebilir. YSA dönüşümü ileri beslemeli sinir ağları, geri beslemeli sinir ağları ve çok katmanlı sinir ağları şeklinde ifade edilebilir. En popüler derin sinir ağlarından biri Evrişimli Sinir Ağı'dır (CNN). Bu adı, evrişim adı verilen matrisler arasındaki matematiksel doğrusal işlemden alır. CNN'nin evrişimli katman, doğrusal olmayan katman, havuz katmanı ve tam bağlantılı katman gibi birden çok katmanı vardır [15]. CNN'ler özellikle bilgisayarlı görü alanında üst düzey başarılar elde etmiştir [4]. Şekil 4'te bir YSA ve CNN'in topolojik yapısı yer almaktadır. Bir sınıflama probleminde CNN'nin çıktıları girdinin ait olduğu sınıflardır. CNN'nin eğitim aşaması genellikle zaman alıcı ve kaynak tüketicidir. Eğitimden sonraki tahmin aşaması hızlıdır ve az kaynak tüketir [16].



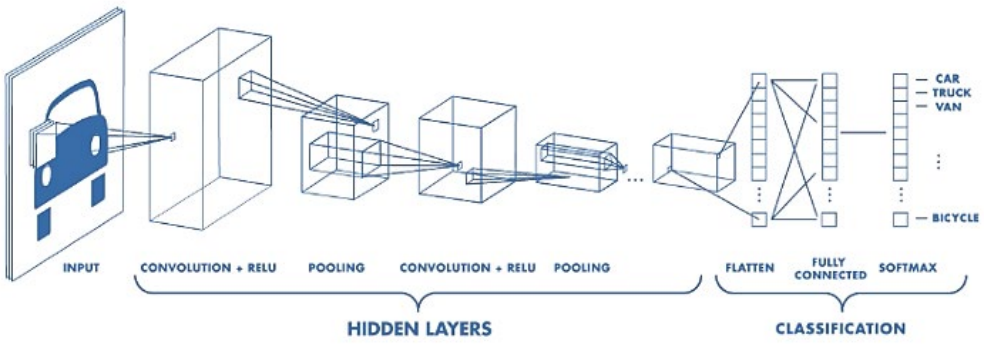
Şekil 4: YSA (üstte) ve CNN (altta) farklı topolojileri [17]

Bahsedilen kavramlar arasındaki ilişki şu şekilde ifade edilebilir; makine öğrenmesi yapay zekânın bir alt dalıdır. YSA ise makine öğrenmesinin bir sınıfı olarak kabul edilebilir. CNN'ler ise YSA'ların zamanla geliştirilmiş halidir ve aynı zamanda Derin Öğrenme kavramı altında kullanılmaktadır. Derin öğrenme ayrıca temelinde bulunan YSA ve CNN yapılarıyla beraber yapay zekânın bir alanı olarak kabul edilebilir (Şekil 5) ve makine öğrenmesi alanından ayrılabilir [18].

Şekil 5: Yapay zekâ yöntemlerinin temsili gösterimi [6]

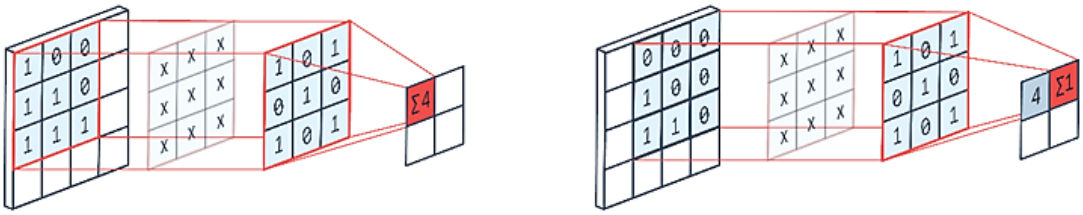
2.5 Derin Öğrenme Katmanları

Geleneksel makine öğrenmesi algoritmaları doğrusal yapıdadır. Ancak derin öğrenme algoritmaları çözümü aranan probleme göre değişiklik gösteren katmanlardan ve süreçlerden meydana gelir. Derin öğrenme yaklaşımları önişleme, boyutsal indirgeme, özellik çıkarma ve sınıflandırma katmanlarını bir arada bulundururlar (Şekil 6) [9]. Derin öğrenmenin temel mimarisi olan CNN ses işleme gibi alanlarda da kullanılmasına rağmen genellikle görüntü işleme, görüntü sınıflandırma, görüntü üzerinde örüntü algılama gibi alanlarda meşhur olmuştur [8].

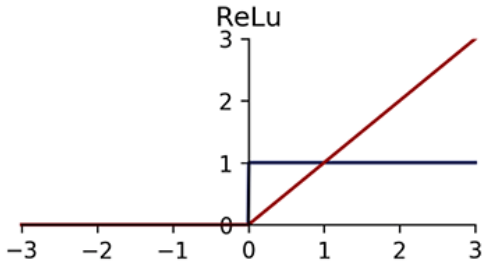


Şekil 6: CNN genel mimarisi [9]

- **Giriş (input) katmanı:** CNN'nin ilk katmanıdır. Ham verilerin bulunduğu katmandır. Giriş verilerinin ağın yapısına göre seçilmesi önemlidir. Eğer giriş veri boyutu yüksek seçilirse eğitim ve test aşamasında kullanılacak kaynak miktarı artırabilir. Veri boyutunun düşük seçilmesi ise ağın başarısını düşürebilmektedir. Giriş verilerinin optimizasyonu ağın çıktıları açısından önemlidir [18].
- **Evrişim (Convolution) katmanı:** Evrişim karmaşık işlemleri basitleştirmek için kullanılan matematiksel bir işlemdir. Evrişim katmanı giriş katmanından gelen veriler üzerinde küçük boyutlu filtreler uygulayarak özellik haritaları çıkarır. Bu şekilde giriş verisinden daha küçük matrisler elde edilir (Şekil 7) [7].
- **Aktivasyon katmanı:** YSA'da olduğu gibi CNN'ler de de aktivasyon fonksiyonu kullanılır. Aktivasyon işlemi giriş sinyali üzerinde yapılan doğrusal olmayan aktarmayı ifade eder. YSA'larda aktivasyon fonksiyonu olarak sigmoid tercih edilmektedir. CNN'de ise aktivasyon fonksiyonu olarak Düzeltilmiş Doğrusal Birimler (ReLU) tercih edilir [9]. ReLU fonksiyonu h(y) = max(0, y) şeklinde ifade edilebilir. ReLU aktivasyon fonksiyonu ve türevinin grafiği Şekil 8' de verilmiştir [19].

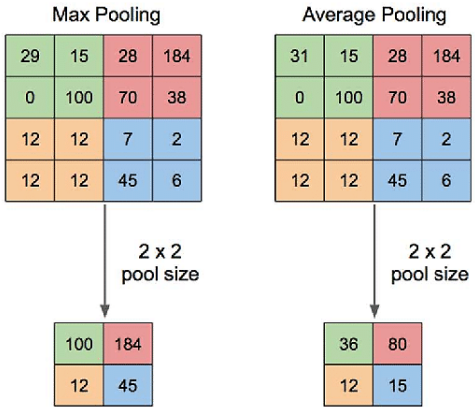


Şekil 7: Evrişim işlemi [7]



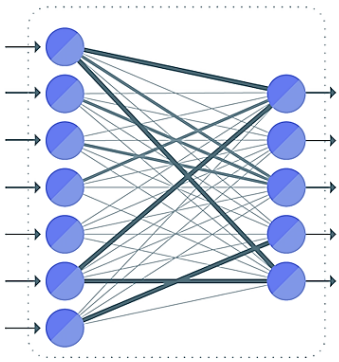
Şekil 8: ReLU fonksiyonu ve türevinin grafiği [19]

- **Havuzlama (Pooling) Katmanı:** Havuzlama katmanında amaç girdi boyutlarını düşürmektir. Giriş boyutları bazı filtre matrisleri kullanılarak (maksimum, minimum, ortalama vb.) azaltılır. Havuzlama esnasında veri kayıpları oluşur. Veri kayıplarının oluşması nedeniyle sonuç performansının düşmesi söz konusudur. Fakat havuzlama sonucunda ağın aşırı ezberlemesi de engellenmiş olur. Havuzlama katmanında filtre matrisinin ne kadar adım atacağı değişebilir [8]. Havuzlama katmanına ait gösterim Şekil 9'da sunulmuştur.

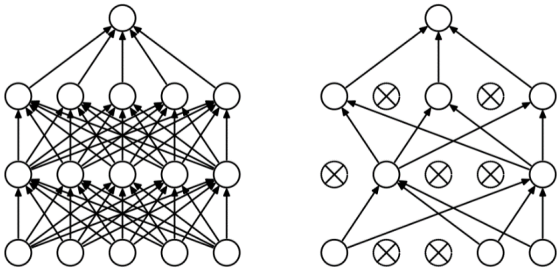


Şekil 9: Havuzlama katmanın şematik gösterimi [20]

- **Tam Bağlı (Fully Connected) Katman:** Evrişimli sinir ağlarında genellikle konvolüsyon, ReLu ve havuzlama katmanından sonra tam bağlantı katmanı yerleştirilir. Bu katmandaki tüm nöronlar bir dizi şeklindedir. Kendinden önceki katmanların tüm nöronları bu katmana bağlıdır. Kendinden önceki tüm katmanların sınıf skorlarını optimize eder. Bu şekilde baskın ağırlıklara bakılarak sınıf tahmini yapılabilir (Şekil 10) [7].
- **Seyreltme (Dropout) katmanı:** Çok katmanlı sinir ağlarında ağın ezberlemesi problemine karşı geliştirilmiş bir yöntemdir. Bir eşik değer kullanılarak gereksiz ya da zayıf bilgiler unutulur. Böylelikle ağın aşırı ezberlemesi engellenmiş olur [8]. Seyreltme işlemi sinir silme, bağlantı silme veya her ikisini silme şeklinde olabilir (Şekil 11).

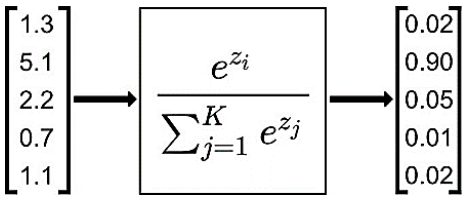


Şekil 10: Tam bağlı katman yapısı [7]



Şekil 11: Seyreltme katmanı sonrası bağlantı şekli [9]

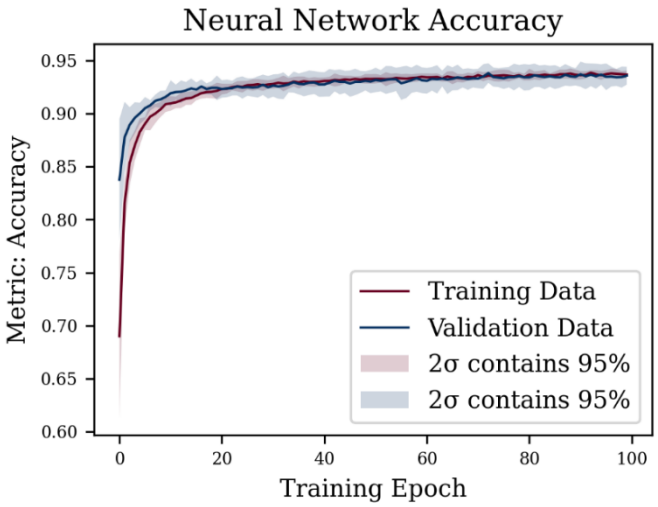
- **Sınıflandırma (Classification) katmanı:** Derin öğrenmede sınıflandırma işleminin yapıldığı katmandır. Bu katmanın çıkış değeri sınıf sayısı x sınıflandırılacak nesne adedi şeklinde bir matristir. Sınıflandırılacak her nesne için sınıf adedi kadar 0-1 aralığında çıkışlar elde edilir. Her nesne için 1'e en yakın değer ait olduğu sınıfı işaret etmektedir. Başarısından dolayı genellikle softmax sınıflandırıcısı tercih edilmektedir (Şekil 12) [18].



Şekil 12: Softmax fonksiyonu [21]

2.6 CNN Ağının Eğitilmesi

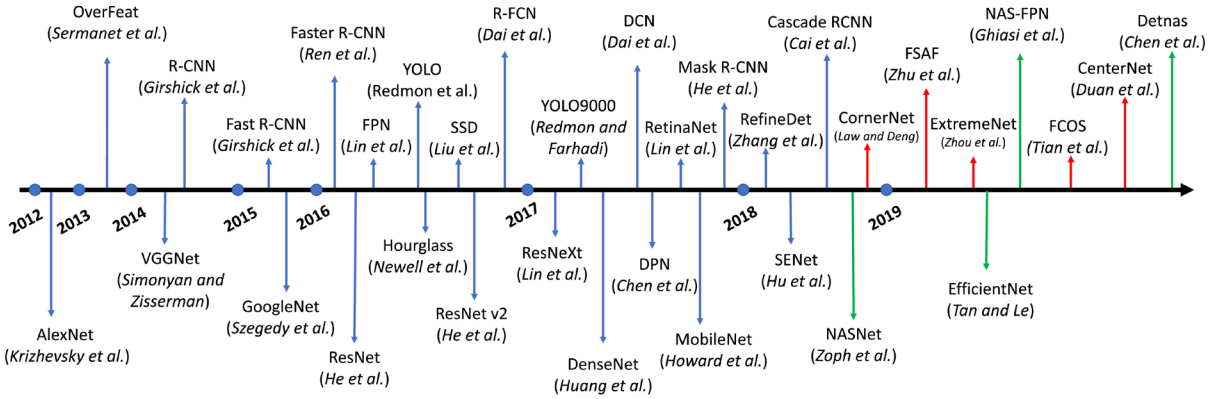
CNN eğitimi için öncelikle ağın yapısı planlanır. Ağda hangi katmanların olacağı, hangi sıraya göre dizileceği belirlenir. Akabinde giriş parametre sayısı, filtre sayısı ve boyutları belirlenir. Eğitim setinden bir veri alınarak sıralanan katmanlarda işlenerek çıkış değeri okunur (ileri yayılım). Elde edilen çıkış ile olması gereken çıkış kıyaslanır. Kıyaslama sonucunda hesaplanan hata tekrar katmanlardan bu defa ters yönde geçirilerek ağırlıklara dağıtılır (geri yayılım). Ağırlıklar güncellenerek toplam hata düşürülmeye çalışılır. Eğitim kümesindeki tüm elemanlar ağa uygulanarak ağırlıklar dinamik olarak güncellenir. Tüm veri seti belirlediğimiz döngü adedince ağa tekrardan uygulanarak toplam hata düşürülmeye çalışılır. Eğitim aşaması bittikten sonra test sırasında sadece ileri yayılım gerçekleştirilip toplam hata hesaplanarak ağın öğrenme başarısı puanlanır. Şekil 14'te bir CNN ağının eğitim ve test işlemine yönelik bir grafik yer almaktadır.



Şekil 13: CNN'nin eğitim ve test işlemlerinin karşılaştırması [19]

2.7 Derin Öğrenme Modelleri

Derin öğrenme nesne algılamadaki başarısından sonra popüler olmaya başlamıştır. Günümüze kadar birçok derin öğrenme modeli bilgisayarlı görme görevleri için tasarlandı ve bu alanlarda birçok başarı elde ettiler. Bu modellerden ilki LeCun tarafından hazırlanmış olan LeNet tir. 2012 yılındaki ImageNet yarışmasında başarılı olan AlexNet derin öğrenmenin yeniden parlamasına sebep olmuştur. Zaman içerisinde birçok mimari geliştirilmiştir. Şekil 14'te derin öğrenme mimarilerinden bazılarına yer verilmiştir. Bu mimariler özellikle bilgisayar görmesi görevlerinde giderek artan miktarda başarılara imza atmışlardır. Derin öğrenme modellerinin zaman içerisinde katman sayıları artırılmıştır. Algılayıcı yapıları iyileştirilerek daha iyi sonuçlar elde edilmeye çalışılmıştır. Özellikle son dönemki mimarilerde çapasız (kırmızı) algılayıcılara ve AutoML (yeşil) mimarilere ilgi artmıştır [4].



Şekil 14: Bazı derin öğrenme mimarilerinin zaman çizelgesi [4]

2.7.1 LeNet

1998 yılında Yann LeCun ve arkadaşları tarafından geliştirilmiştir [22]. Başarılı İlk CNN olarak bilinir. 0 ile 9 arasındaki sayıların görsellerini kullanarak sınıflandırma yapmaktadır. Ağın çıkışı 10 adet sınıftan oluşmaktadır. Giriş olarak 32x32 pikselden oluşan 1024 parametreden oluşmaktadır. Toplamda 7 katmandan oluşmaktadır (Şekil 15.A).

2.7.2 AlexNet

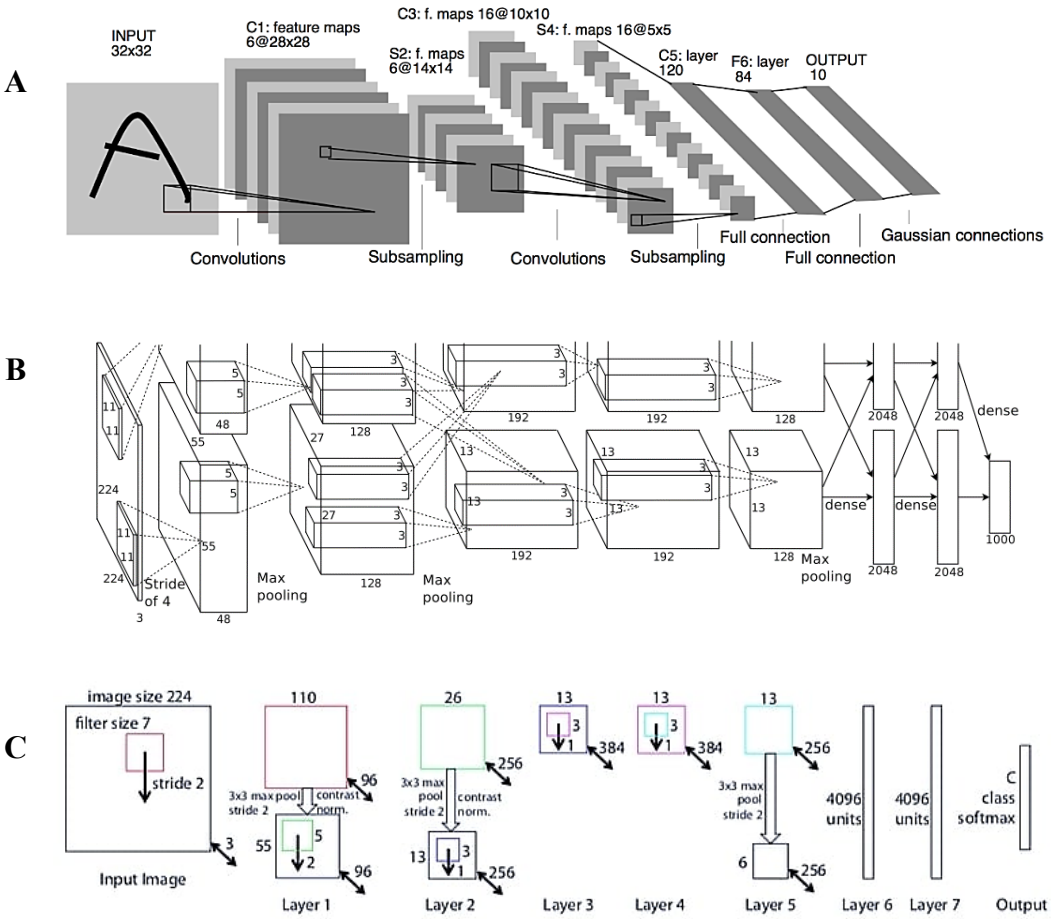
Krizhevsky ve arkadaşlarının çalışması 2012 ImageNet görsel tanıma yarışmasında birinci olmuştur. Örüntü tanıma hata oranını %15 e kadar düşüren bu çalışma [23] derin öğrenme mimarilerinin görüntü tanıma uygulamalarında kullanılmasının popüler olmasına neden oldu. AlexNet mimarisi 5 konvolüsyon katmanı, havuzlama katmanı ve 3 tam bağlantılı katmandan oluşmaktadır (Şekil 15.B)

2.7.3 ZF Net

AlexNet'in İmageNet yarışmasındaki başarısının ardından yapılan yarışmalarda derin öğrenme yöntemleri daha sık kullanılmaya başlanmıştır ve her defasında daha yüksek başarılar elde edilmiştir. Zeiler ve Fergus'un çalışması ZF Net [24] 2013 yılında %11,2'lik hata oranı ile birinci olmuştur. ZF Net modelinde, ilk katmanda AlexNet'in uyguladığı 11x11 boyutlu filtreler kullanmak yerine, 7x7 boyutundaki filtreleri ve havuzlama katmanında 2 adım kayma miktarı kullanılmıştır (Şekil 15.C)

2.7.4 VggNet

VggNet 2014 İmageNet yarışmasında %7.3'lük hata oranıyla birinci olmuştur. Simonyan ve Zisserman tarafından Oxford üniversitesinde tasarlanan bu mimari 6 farklı mimari ortaya koymuşlardır. Bu 6 farklı modelde 11, 13, 16, 19 konvolüsyon katmanlıdan oluşmaktadır [25]. Literatürde vgg-16 vgg-19 gibi fraksiyonları da ter almaktadır.

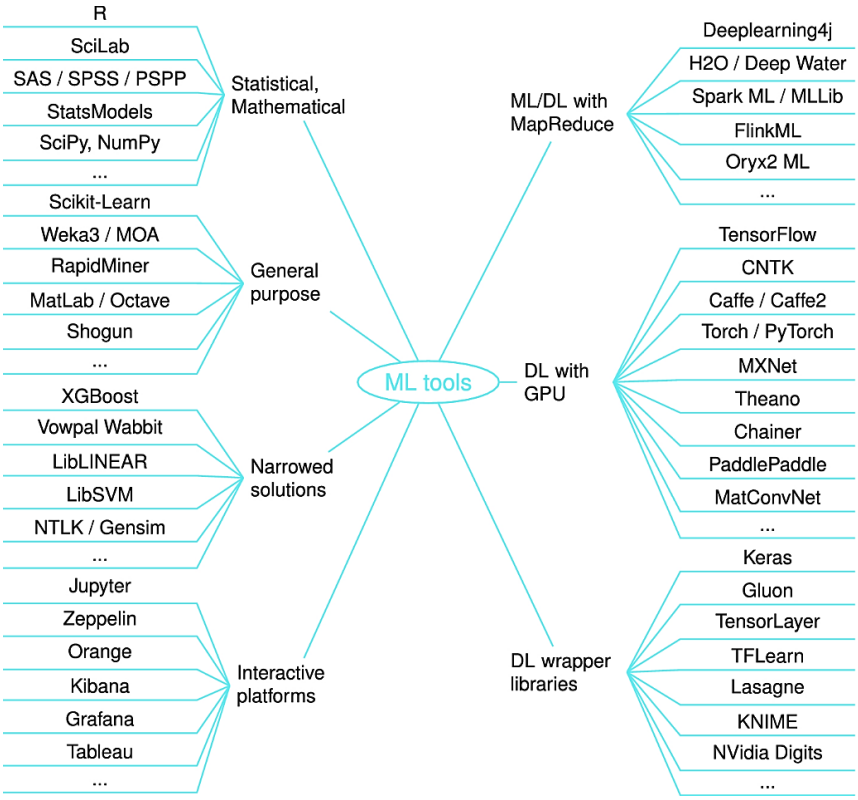


Şekil 15: Bazı derin öğrenme mimarilerinin şematik gösterimi. A; LeNet [22], B; AlexNet [23], C; ZF Net [24]

Literatürde, yukarıda sayılanlar haricinde birçok mimari bulunmaktadır. GoogleNet, ResNet, DenseNet, You only look once (YOLO) bunlardan bazıları olarak sayılabilir. Sonuç olarak derin öğrenme mimarileri hala nesne algılama uygulamalarında en gözde metotlar olarak karşımıza çıkmaktadır.

2.8 Derin Öğrenme Kütüphaneleri

Derin Öğrenme uygulamaları Makine Öğrenmesinin bir alt kümesidir. Makine öğrenmesi ve derin öğrenme uygulamalarını geliştirmek için birçok uygulama çerçevesi ortaya çıkmıştır. Bir problem genellikle bu çerçevelerden bir tanesiyle çözülememektedir. Kütüphanelerin birden fazlasının bir arada kullanılmasıyla çözüme gidilmektedir [26]. Şekil 16'da farklı makine öğrenmesi araçlarının sınıflandırması yer almaktadır.



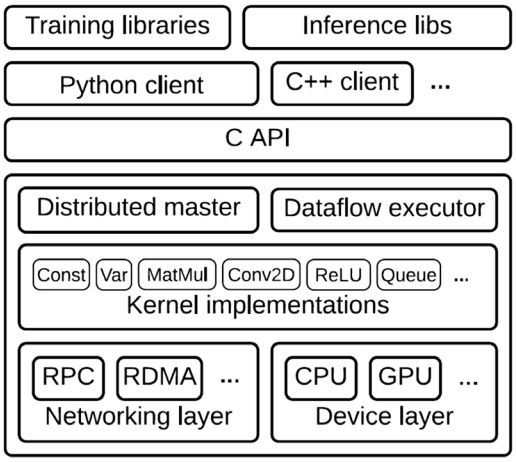
Şekil 16: Farklı makine öğrenmesi araçları [26]

Derin öğrenmenin popülerliği her geçen gün atmaktadır. Buna paralel olarak faklı derin öğrenme çerçeveleri geliştirilmektedir. TensorFlow (2014), Keras (2014), Theano (2008), caffe (2012), CNTK (2016) bunlara örnek olarak gösterilebilir [27], [28]. Bu derin öğrenme kütüphaneleri Python, c++, java gibi farklı programlama dilleri temel alınarak geliştirilmiştir [29].

2.8.1 TensorFlow

TensorFlow Makine öğrenmesi ve derin öğrenme projelerini baştan sona tasarlayıp sonuçları görebileceğimiz bir ortam sunar [30]. Google Brain ekibi tarafından 2015 yılında açık kaynaklı bir proje olarak yayınlanmıştır. Birbirinde farklı ortamlarda çalışabilir. Farklı programlama dilleri ve farklı donanımlarda çalışabilir. Bu kütüphanedeki yapı temel olarak bir dizi hesaplamalardan meydana gelen veri akış grafiklerinden oluşmaktadır. Bu akış grafikleri düğümlerin durumunu kaydetmek, güncellemek için dallanma ve döngü kontrolüne izin veren bir veri akışı hesaplaması sunar [31]. TensorFlow, otomatik farklılaştırma ve parametre paylaşma yetenekleri nedeniyle farklı mimari türlerini destekler. TensorFlow, paylaşılan parametreleri güncellemek için iş birliği yapan çoklu hesaplama kaynaklarını kullanarak veri akış grafiği modelinin paralel yürütülmesi yoluyla paralelliği destekler. Paralel çalışma yeteneği ciddi performans artışlarına izin verir [14]. TensorFlow ayrıca mobil ve Nesnelerin İnterneti (IoT) cihazlar için

optimize edilmiş çıkarım yapma sürümünü de sunmaktadır. Düşük donanım özelliklerinde etkin sonuçlar vermesi hedeflenmiştir [32]. TensorFlow ile Google ve Amazon gibi bulut ortamlarında model oluşturmak, eğitmek ve test etmek mümkündür. Şekil 17'de TensorFlow mimari katmanlarına yer verilmiştir.



Şekil 17: TensorFlow mimari katmanlarının şematik gösterimi [31]

2.8.2 Keras

Keras, Python'da geliştirilmiş bir derin öğrenme kütüphanesidir. Derin öğrenme modellerinin geliştirilmesi eğitim ve test edilmesi görevlerini yerine getirir [6]. Kullanıcılara yönelik daha kolay araçlar geliştirmeye odaklanmıştır bu nedenle kullanıcı dostudur. TensorFlow ve Theano üzerinde üst düzey uygulama programlama arayüzü (API)'ler sunar. 2021'in başlarında 400.000'den fazla bireysel kullanıcıyla Keras, hem endüstri hem de araştırma topluluğu genelinde güçlü bir şekilde benimsenmiştir [33]. Aynı kodun değişmeden hem merkezi işlem birimi (CPU), hem de grafik işlem birimi (GPU)'da çalışmasını sağlar. Derin öğrenme modellerinin hızlıca prototipleştirilmesini sağlayan API'ler sunar. Evrişimli ağlar, yinelemeli ağlar ve her ikisinin birlikte çalışabilmesi için önceden tanımlı araçlar sunar.

2.8.3 Theano

Theano, Montreal Üniversitesi'nden araştırmacılar ve geliştiriciler tarafından geliştirilmiştir. Theano, Berkeley Software Distribution (BSD) lisansı altında lisanslanan ücretsiz, açık kaynaklı bir yazılımdır. Dünya çapında geniş ve çok aktif bir geliştirici ve kullanıcı topluluğuna dayanır. Theano, derin öğrenme modelleri oluşturmayı kolaylaştıran temel bir matematiksel ifade kütüphanesidir. Theano, çok boyutlu dizileri içeren matematiksel ifadeleri verimli bir şekilde tanımlamaya, optimize etmeye ve değerlendirmeye izin veren bir Python kitaplığı olarak tanımlanabilir. Piyasaya sunulduğundan bu yana, özellikle makine öğrenimi topluluğunda çok fazla kullanılan CPU ve GPU matematiksel derleyicilerinden biri olmuştur [34].

Theano, birçok modern makine öğrenimi modelinde inşa edilen ve kullanılan çok sayıda üstyapı ile 2008'den beri aktif ve sürekli olarak geliştirildi. Ancak, Theono'nun geliştiricilerinden Yoshua Bengio, 2017'de Theano'daki geliştirmenin sona ereceğini duyurdu [35].

2.8.4 Caffe ve Caffe2

Caffe, derin öğrenme algoritmalarını uygulamak için Kaliforniya Üniversitesi, Berkeley tarafından geliştirilen açık kaynaklı bir platformdur. Kodlar, GPU hesaplama için kullanılan Bilgisayar Birleşik Cihaz Mimarisi (CUDA) kütüphanesi ile C++'da yazılır. Ayrıca CUDA kütüphanesi, Python/Numpy ve MATLAB'ı destekler. Caffe, ilk olarak görüntü işleme için tasarlanmasına rağmen daha sonradan ses tanıma, robotik, astronomi ve sinir bilimi için kullanılmıştır [36]. Derin Sinir Ağları (DNN), Caffe'de katman katman tanımlanır. Katman, bir modelin özü ve hesaplamanın temel birimidir. Veriler, Caffe'ye

veri katmanları aracılığıyla girer. Kabul edilen veri kaynakları, verimli veri tabanları (LevelDB veya LMDB), Hiyerarşik Veri Formatı (HDF5) veya yaygın görüntü formatlarıdır (örn. GIF, TIFF, JPEG, PNG, PDF). Nisan 2017'de son kararlı sürümü yayımlanmıştır [26]. Caffe2, Caffe'nin geliştiricisi Yangqing Jia tarafından geliştirilmiştir. Yangqing jia, Facebook'da çalışmaya başladıktan sonra, NVIDIA ve Facebook ile birlikte caffe tabanlı Caffe2 çerçevesini geliştirdiler. Caffe2, büyük ölçekli dağıtılmış eğitim desteği, mobil dağıtım, yeni donanım desteği nicelenmiş hesaplama gibi bazı Caffe sınırlılıklarını iyileştirdi. Caffe2, NVIDIA desteğine sahiptir. Ayrıca python ve c++ API desteği de mevcuttur.

2.8.5 Microsoft CNTK

CNTK, Microsoft Araştırma Ekibi tarafından geliştirilmiştir. Birden çok GPU veya sunucu üzerinde birçok Sinir Ağı türünü eğitmek ve test etmek için geliştirilmiş açık kaynaklı bir DL çerçevesidir. CNTK, İleri Beslemeli, Evrişimli, Tekrarlayan, Uzun Kısa Süreli Bellek (LSTM) gibi farklı DL mimarilerini destekler. CNTK arayüzü, hem GPU (CUDA) hem de CPU platformlarında Python, C ++ ve C # gibi çeşitli dillerin farklı API'lerini destekler. CNTK ileri ve geri işlemlerde yinelenen hesaplamaları kaldıracak şekilde ve minimum bellek kullanacak şekilde c++ ile yazılmıştır [37]. CNTK, Python, C # veya C ++ programlarınıza bir kitaplık olarak dahil edilebilir veya kendi model açıklama dili (BrainScript) aracılığıyla bağımsız bir makine öğrenimi aracı olarak kullanılabilir. Ek olarak, Java programlarında CNTK model değerlendirme işlevini kullanılabilir [38]. Geliştirme ekibi 2.7 sürümü ile CNTK'nın geliştirilme sürecinin sonlandırıldığını duyurmuştur. Ve yeni özellik eklenmeyeceğini duyurmuştur [39].

2.8.6 Torch

Makine öğrenimi algoritmaları için destek sunan BSD lisanslı bilimsel hesap kütüphanesidir. Facebook, Twitter, Google gibi kuruluşlar tarafından desteklenmektedir. Hem CPU hem de GPU'da çalışabilir. Esas olarak büyük ölçekli öğrenme (konuşma, görüntü ve video uygulamaları), denetimli öğrenme, denetimsiz öğrenme, pekiştirmeli öğrenme, Sinir Ağları, optimizasyon, grafik modeller, görüntü işleme için kullanılır. Son sürüm olan Torch7 ile geliştirmesi durdurulmuştur [26].

2.8.7 PyTorch

PyTorch, Facebook'un AI araştırma grubu tarafından Ekim 2016'da tanıtıldı. PyTorch, API aracılığıyla derin öğrenme modelleri oluşturmayı kolaylaştıran Python tabanlı bir derin öğrenme çerçevesidir. Statik hesaplama grafikleri kullanan diğer popüler derin öğrenme çerçevelerinin çoğunun aksine PyTorch, karmaşık mimariler oluşturmada daha fazla esneklik sağlayan dinamik hesaplama kullanır [14]. PyTorch, birçok kütüphaneyle entegre çalışır. Büyük, küçük birçok sinir ağında yüksek performans sunar. PyTorch'daki bellek kullanımı, Torch veya bazı alternatiflere kıyasla son derece verimlidir. Derin öğrenme modellerinizin maksimum düzeyde bellek açısından verimli olmasını sağlamak için GPU'ya özel bellek ayırıcıları mevcuttur. Buda, eskisinden daha büyük derin öğrenme modellerini eğitmenizi sağlar [40]. Kütüphane bir BSD lisansı altında ücretsiz olarak sunulur. Facebook, Twitter, NVIDIA ve diğer birçok kuruluş tarafından desteklenmektedir.

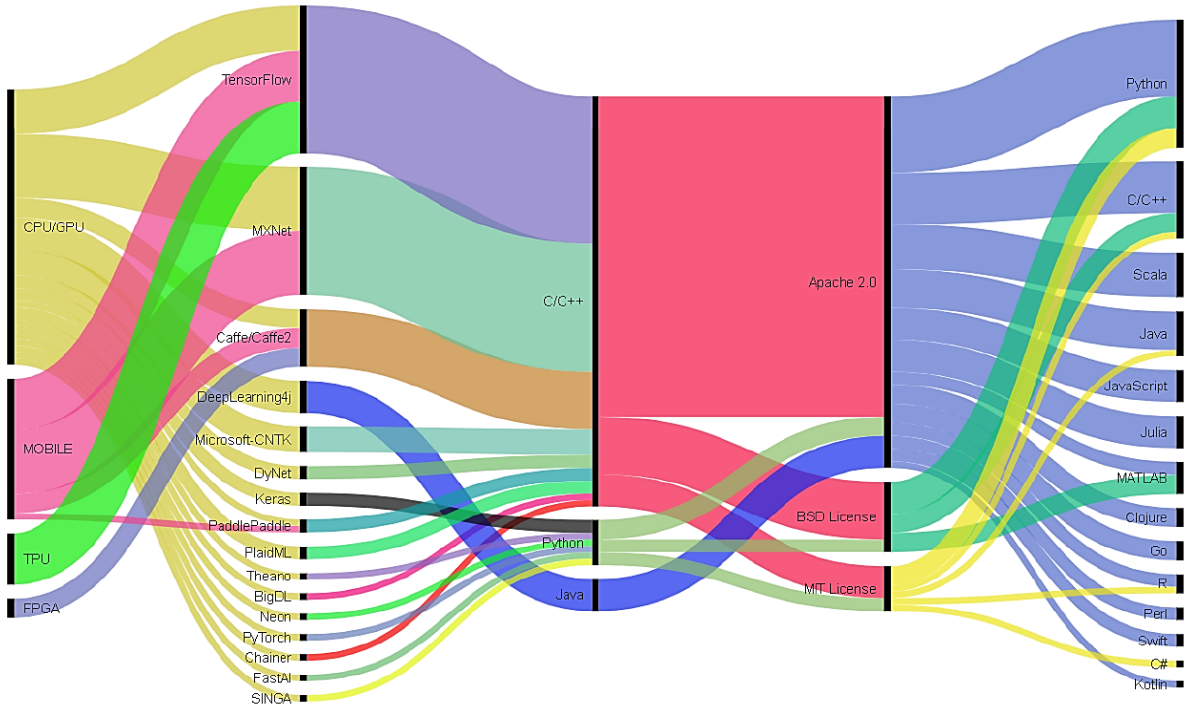
2.8.8 MXNet

MXNet, Carnegie Mellon Üniversitesi, Washington Üniversitesi ve Microsoft arasındaki iş birliği ile ortaya çıkmış açık kaynaklı bir derin öğrenme çerçevesidir. C, Python, MATLAB, JavaScript, R, Julia ve Scala dahil olmak üzere farklı programlama dillerini kullanarak derin sinir ağlarının eğitimine izin veren ölçeklenebilir bir çerçevedir. MXNet, çoklu CPU'larda veya GPU'larda veri paralelliğini destekler ve model paralelliğine de izin verir. MXNet, özellikle derin sinir ağları için makine öğrenimi algoritmalarının geliştirilmesini kolaylaştıran çok dilli bir makine öğrenimi (ML) kitaplığıdır. MXNet, hesaplama ve bellek açısından verimlidir ve mobil cihazlardan dağıtılmış GPU kümelerine kadar çeşitli heterojen sistemlerde çalışır [41].

2.8.9 Chainer

Chainer, DL modelleri için Python tabanlı, bağımsız bir açık kaynaklı çerçevedir. Chainer çekirdek geliştirici ekibi, çoğunlukla Tokyo Üniversitesi'nden mühendislerle bir ML girişimi olan Preferred Networks, Inc.'de çalışmaktadır. CNN, Tekrarlayan Sinir Ağı (RNN), pekiştirmeli öğrenme (RL) ve değişken otomatik kodlayıcılar dahil olmak üzere eksiksiz bir DL modelleri yelpazesi sunar [42]. Chainer, model oluşturmak ve eğitmek için üst düzey API'ler sunar. Aynı zamanda endüstriyel uygulamalar için kütüphaneler içerir. Chainer, CUDA hesaplamasını destekler. Bir GPU'dan yararlanmak için yalnızca birkaç satır kod gerektirir. Ayrıca çok az çabayla birden fazla GPU üzerinde çalışır [43]. Temmuz 2020 de son versiyon olan 7.7.0 versiyonu yayınlanmıştır [44].

Şekil 18'de farklı derin öğrenme kütüphanelerine ait bir grafik yer almaktadır. Sırasıyla soldan sağa kullanılan donanımlar, derin öğrenme çerçeveleri, kütüphanelerin yazıldığı programlama dilleri, kütüphanelerin lisansları ve kütüphanelerle geliştirme yapılabilecek programlama dilleri. Araçların hemen hepsi CPU/GPU üzerinde çalışabilmektedir. Python, c/c++, java dilleri kütüphaneleri geliştirmek için kullanılmıştır. Yine birçok dil bu kütüphanelerle geliştirme yapmaya müsaade etmektedir [45].



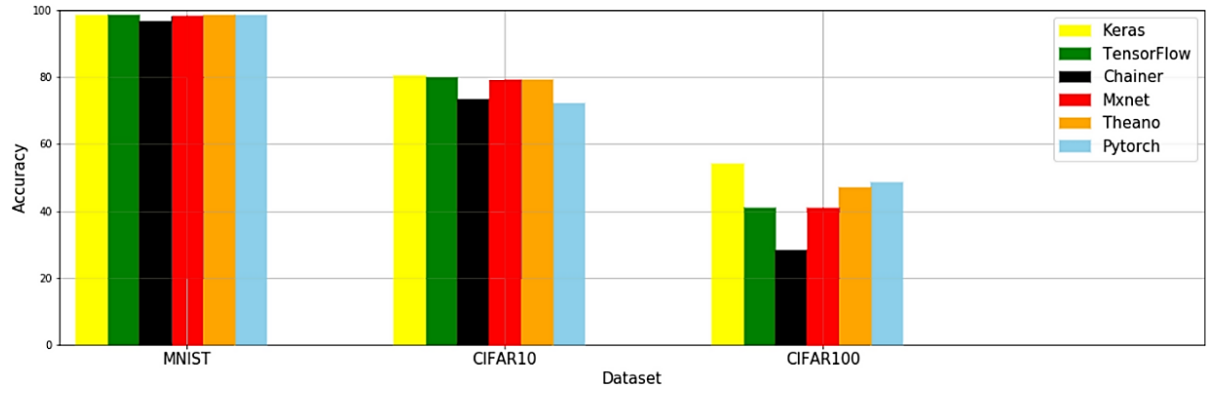
Şekil 18: Derin öğrenme kütüphaneleri ve API'lerin gösterimi [45]

3 Sonuç ve Tartışma

Derin öğrenme kütüphaneleri birçok görev için sıklıkla kullanılmaktadır. Araştırma ekiplerinden, bireysel çalışmalara, ticari uygulamalara kadar birçok alanda derin öğrenme çalışmaları mevcuttur. Özellikle bilgisayarlı görü uygulamalarındaki nesne algılama görevi derin öğrenme algoritmalarının geliştirilmesi için motivasyon olmuştur. Nesne algılama uygulamalarında parametrelerin fazlalığı veri boyutunun büyük olması bu alandaki iş yükünü bir hayli artırmıştır. Nesne tespitindeki zorlukların aşılması için daha fazla çabalar ortaya koyulmuştur ve bu çabalar derin öğrenme ekosistemini geliştirmiştir. Nesne algılamadaki çalışmaların başarısı diğer problem alanlarındaki başarıyı da olumlu yönde etkilemiş ve artırmıştır. Günümüzde derin öğrenme ekosistemi devasa bir büyüklüğe sahiptir. Bu çeşitlilik yeni uygulamalar geliştirecek kişi veya ekipler için bir karmaşaya da sebep olmaktadır. Sürekli genişleyen derin öğrenme kütüphaneleri arasında tercih yapmak başlı başına bir beceri haline gelmiştir.

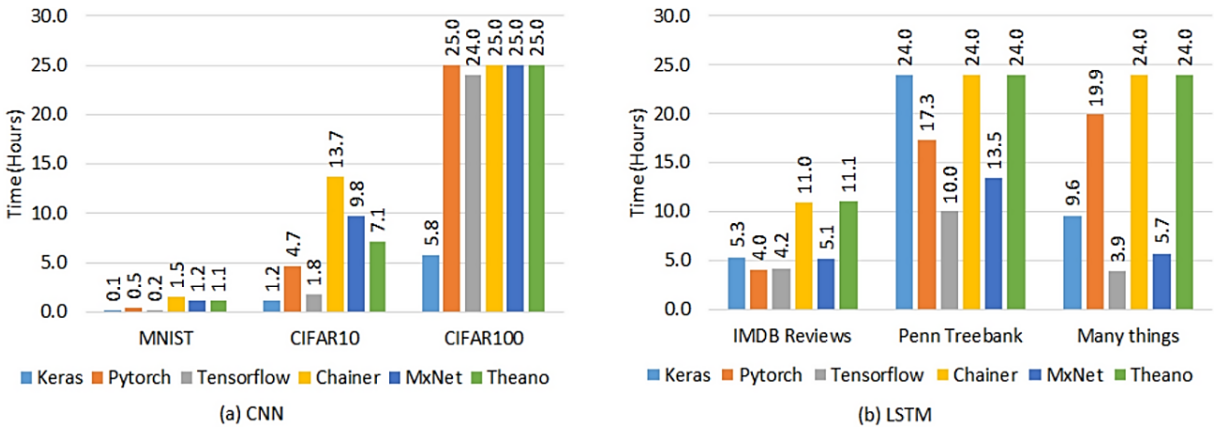
Kütüphanelerin genişlemesi beraberinde bu kütüphanelerin incelendiği araştırmaları getirmiştir. Birçok araştırmacı zaman içerisinde yaygın kullanılan derin öğrenme kütüphanelerini farklı açılardan ele alarak kıyaslamışlarıdır. Bu bölümde literatürdeki derin öğrenme kütüphanelerinin karşılaştırmalı incelemesi yer almaktadır.

Elshawi ve arkadaşlarının [14] gerçekleştirdiği çalışmada, TensorFlow , MXNet , PyTorch , Theano , Chainer ve Keras kütüphaneleri hem CPU hem de GPU üzerinde çalıştırılmıştır. Kütüphanelerin eğitim süresi, doğruluk, yakınsama, kaynak kullanımı açısından performansları karşılaştırılmıştır. Ayrıca farklı derin öğrenme mimarileri ve farklı veri setlerinin kaynak kullanımı ve performans üzerine etkisi incelenmiştir. Elshawi'nin çalışmasında MNIST veri setinin diğer veri setlerine göre doğruluk oranının yüksek olduğu ifade edilmiştir (Şekil 19). Keras kütüphanesinin tüm veri setlerinde daha iyi bir doğruluğa sahip olduğunu gözlemişleridir.



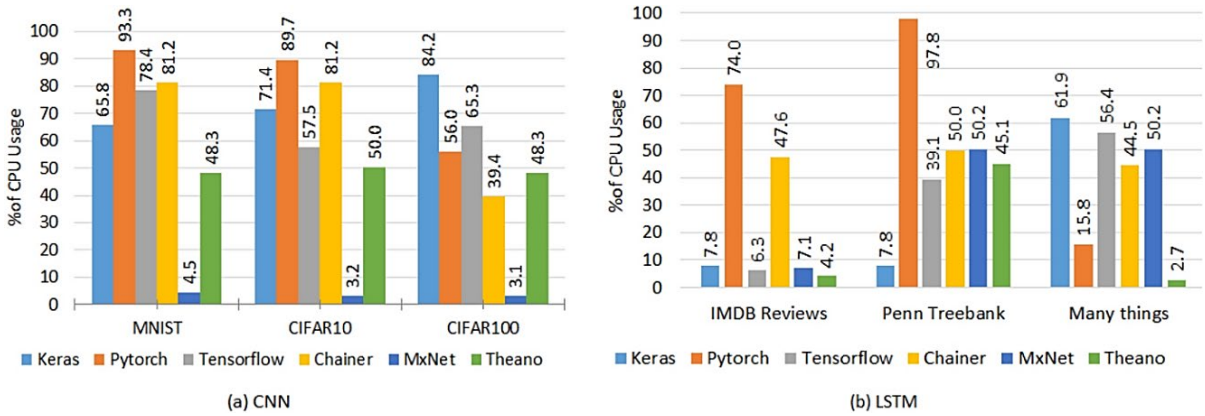
Şekil 19: Farklı veri setleri ile farklı derin öğrenme kütüphanelerinin doğruluk karşılaştırması [14]

Şekil 20'de CNN ve LSTM mimarilerinin farklı farklı veri setlerindeki eğitim süreleri ele alınmıştır. MNIST veri seti eğitim süresi olarak hem CNN hem de LSTM ağında, tüm kütüphanelerde, en az zamana sahip veri seti olarak gözlenmiştir.



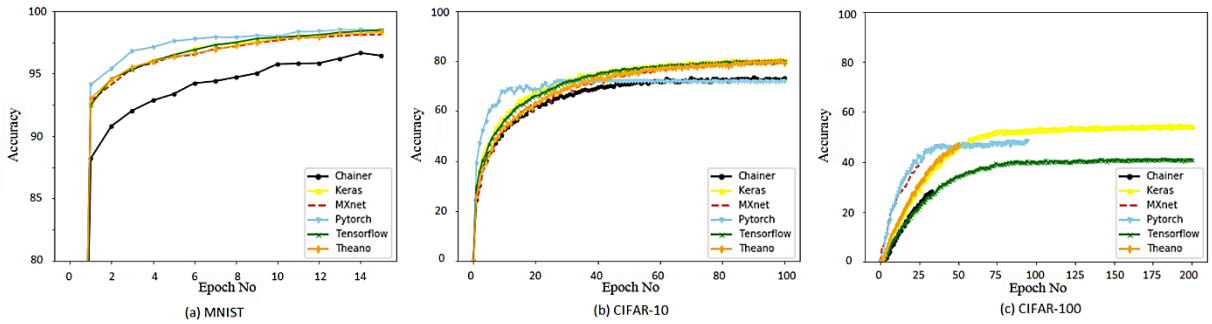
Şekil 20: Derin öğrenme kütüphanelerinin CNN ve LSTM mimarileri ile farklı veri setleri üzerinde eğitim süresi kıyaslaması [14]

Yine aynı çalışmada CPU yüzde kullanım oranları kıyaslanmıştır. Şekil 21'de özellikle CNN ağının CPU kullanımı LSTM' ye göre daha fazla olduğu gözlenmiştir. PyTorch her iki ağ için ortalamada CPU'yu en çok kullanan çerçevedir.



Şekil 21: Derin öğrenme kütüphanelerinin CNN ve LSTM mimarileri ile farklı veri setleri üzerinde ortalama CPU kullanım oranı [14]

Farklı veri setlerinde çerçevelerin en yüksek doğruluğa ulaşma başarısına bakıldığında (Şekil 22) En yüksek performansı PyTorch' un gösterdiği gözlenmiştir. CIFAR-100 veri setinde en düşük doğruluk oranına sahip çerçeve TensorFlow olarak gözlenmiştir.



Şekil 22: Derin öğrenme kütüphanelerinin farklı veri setleriyle eğitimindeki iterasyon-doğruluk grafiği [14]

Bahrampour [46] ve arkadaşları gerçekleştirdikleri çalışmada: Caffe, Neon, TensorFlow, Theano ve Torch adlı beş derin öğrenme çerçevesini genişletilebilirlik (diğer işlevleri etkilemeden yeni işlev ekleyebilme), hız ve kaynak kullanımı açısından karşılaştırmışlardır. Araştırma sonucunda Theano ve Torch'un en kolay genişletilebilir çerçeveler olduğunu, CPU için sırasıyla Torch ve Theano'nun daha iyi performans verdiğini belirtmişlerdir. Theano'nun, LSTM ağlarının eğitimi ve dağıtımı için GPU'da en iyi performansı elde ettiğini gözlenmiştir. Tensorflow'un esnek olduğunu ancak performans açısından rekabetçi olmadığını ifade edilmiştir.

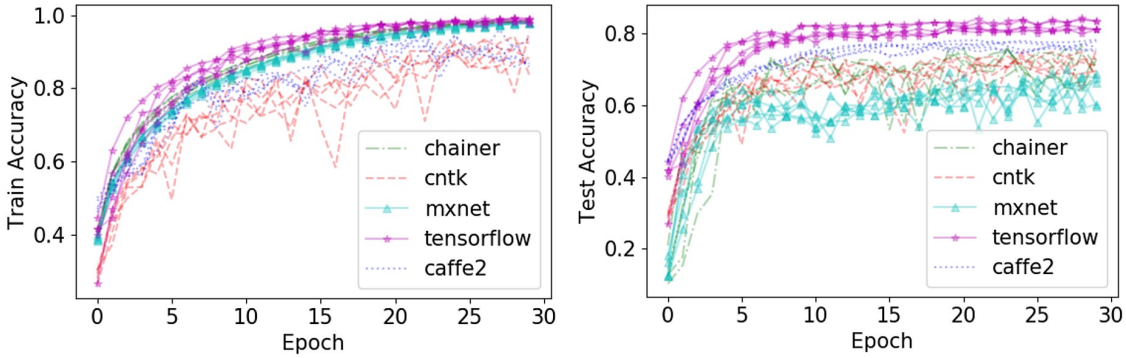
Wu ve arkadaşlarının [47] gerçekleştirmiş olduğu çalışmada: Tensorflow, Caffe, Torch ve Theano çerçeveleri performans, doğruluk ve kaynak kullanımı açısından karşılaştırılmıştır. Karşılaştırmalar MNIST ve CIFAR-10 datasetleri kullanılarak gerçekleştirilmiştir. MNIST veri seti üzerinde Eğitim süresi en az olan CPU kullanarak Caffe çerçevesi, GPU kullanarak ise TensorFlow olduğu gözlenmiştir (Tablo 1). Test aşamasında ise GPU'da Theano, CPU'da ise TensorFlow olarak gözlenmiştir. CIFAR-10 veri setinin doğruluk oranlarının MNIST' a kıyasla bir hayli düşük olduğu gözlenmiştir.

Tablo 1: Sırasıyla MNIST ve CIFAR10 veri seti değerlendirme sonuçları [47]

Frameworks	Training Time (s)	Testing Time (s)	Accuracy (%)
TF-CPU	1,114.34	**2.73**	**99.24±0.05**
Caffe-CPU	**512.18**	3.33	99.04±0.02
Torch-CPU	9,647.34	56.52	**99.24±0.00**
Theano-CPU	11,555.43	4.49	99.08±0.00
TF-GPU	**68.51**	0.26	**99.21±0.03**
Caffe-GPU	97.02	0.55	99.14±0.03
Torch-GPU	338.46	1.73	**99.22±0.00**
Theano-GPU	560.04	**0.19**	99.05±0.00

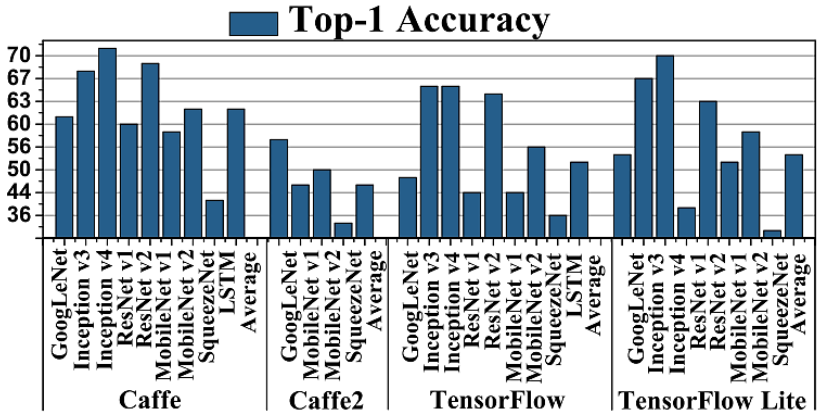
Frameworks	Training Time (s)	Testing Time (s)	Accuracy (%)
TF-CPU	219,169.14	4.80	**86.90**
Caffe-CPU	1,730.89	14.35	75.39
Torch-CPU	54,830.26	114.48	66.20
Theano-CPU	**646.9**	**0.91**	56.04
TF-GPU	12,477.05	2.34	**87.00**
Caffe-GPU	163.51	1.36	75.52
Torch-GPU	1,906.56	3.77	65.96
Theano-GPU	**105.27**	**0.10**	54.49

Liu ve arkadaşlarının [48] gerçekleştirdiği çalışmada: popüler derin öğrenme çerçevelerinin çeşitli dağıtılmış sürümleri araştırılmıştır. Derin öğrenme çerçeveleri zaman, hafıza kullanımı ve doğruluk açısından kıyaslanmıştır. Bu çalışmada TensorFlow sürümlerinin hem eğitim hem de test aşamasında daha yüksek performans gösterdikleri gözlenmiştir (Şekil 23).



Şekil 23: Derin öğrenme kütüphanelerinin iterasyon sayısına göre eğitim ve test performansları [48]

Shi ve arkadaşları [49] gerçekleştirdikleri çalışmada: tek GPU, çoklu GPU ve çok düğümlü ortamlar üzerinden Caffe-MPI, CNTK, MXNet ve TensorFlow derin öğrenme çerçevelerinin çalışma performansı değerlendirilmiştir. Önce DNN'leri SGD ile eğitirken standart süreçlerin performanslarını ölçmüşlerdir. Ardından bu çerçevelerin çalışma performansını AlexNet, GoogleNet ve ResNet-50 ile karşılaştırmışlardır. Dai ve arkadaşları [50] mobil cihazlarda derin öğrenme araçlarının kullanımı ile ilgili bir çalışma gerçekleştirmişlerdir. Bu çalışmada bazı derin öğrenme çerçevelerinin kaynak kullanımı, çıkarım süresi, enerji tüketimi gibi konularda kıyaslamalar sunulmuştur (Şekil 24).



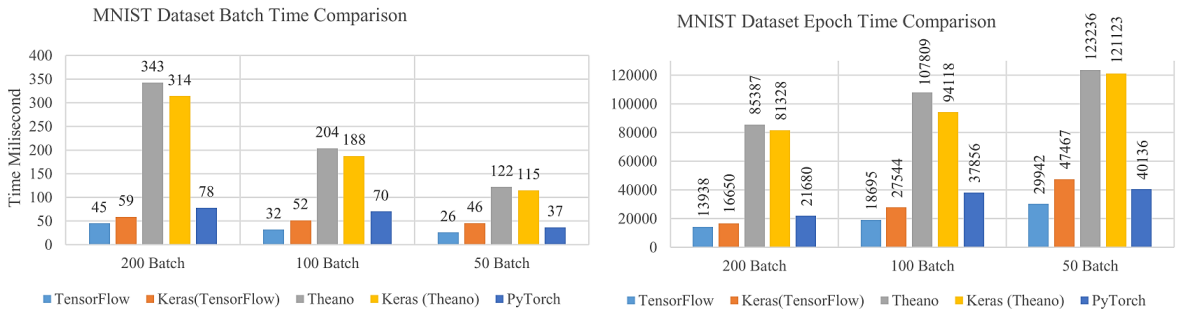
Şekil 24: Mobil cihazlarda farklı mimarilerin farklı çerçevelerdeki performansı [49]

Liu ve arkadaşlarının [29] gerçekleştirdiği çalışmada, derin öğrenme çerçevelerinin hataları araştırılmıştır. Çalışma sonucunda 1) incelenen tüm derin öğrenme çerçevelerinde önemli sayıda teknik hata mevcuttur. 2) derin öğrenme çerçevelerinde tasarım hatası, arıza hatası, belge hatası, test hatası, gereksinim hatası, uyum hatası ve algoritma hatası vardır. 3) Derin öğrenme çerçevesindeki teknik hatanın çoğunluğu tasarım hatası (% 24.07 - % 65.27), bunu ihtiyaç hatası (% 7.09 -% 31.48) ve algoritma hatası (% 5.62 - % 20.67) izlemektedir. Bazı projelerde, uyumluluk hatası %10'dan fazladır (Tablo 2). Bu bulguların derin öğrenme çalışmalarını olumsuz etkilediği vurgulanmıştır.

Tablo 2: Her bir derin öğrenme çerçevesinin teknik hatalarının kendi içinde dağılımı [29]

	TensorFlow	Keras	Caffe	PyTorch	MXNet	CNTK	DL4J
%design	**57.03%**	**24.07%**	**48.75%**	**59.73%**	**63.13%**	**65.27%**	**55.90%**
%requirement	**17.56%**	7.40%	5.62%	**12.80%**	**11.01%**	7.41%	**20.67%**
%algorithm	7.09%	**31.48%**	10.00%	**10.45%**	**10.16%**	**10.53%**	**13.92%**
%compatibility	3.78%	**35.18%**	**11.87%**	7.37%	2.96%	3.95%	0.21%
%documentation	1.27%	0.00%	**15.62%**	0.81%	0.42%	0.83%	0.42%
%test	**7.70%**	0.00%	3.12%	4.61%	3.81%	4.50%	0.63%
%defect	5.53%	1.85%	5.00%	4.20%	8.47%	**7.48%**	8.22%

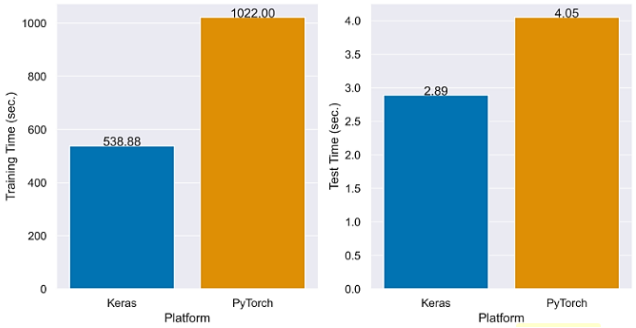
Yapıcı ve arkadaşı [35] Torch, Theano, Caffe, Caffe2, MXNet, Keras, TensorFlow ve Computational Network Tool Kit (CNTK) gibi en sık kullanılan DL çerçevelerinin performans karşılaştırmasını yapmışlardır (Şekil 25). Ayrıca kütüphanelerin GPU performanslarını da test etmişlerdir. TensorFlow kütüphanesinin tüm öbeklerde en iyi performansı verdiğini gözlemlemiştir.



Şekil 25: Derin öğrenme kütüphanelerinin öbek büyüklüğü ve iterasyon karşılaştırması [34]

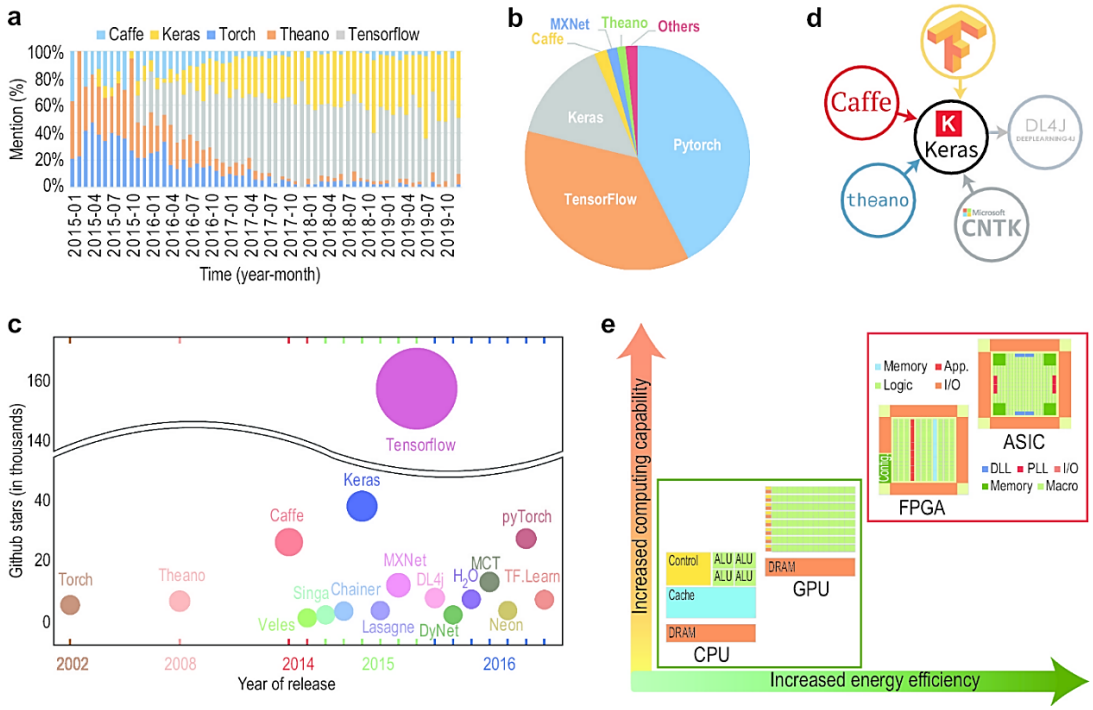
Kabakuş'un [27] gerçekleştirdiği çalışmada Keras ve Pytorch derin öğrenme çerçeveleri farklı mimariler kullanılarak karşılaştırılmıştır. Şekil 26'da CNN için Keras ve PyTorch için eğitim ve test süreleri ve doğruluğu verilmiştir. Keras PyTorch' a kıyasla hem daha hızlı sonuç vermiştir hem de doğruluğu daha yüksek çıkmıştır.

Platform	Accuracy (%)
Keras	78.43
PyTorch	76.54



Şekil 26: VGG16 ile Keras ve PyTorch karşılaştırması [27]

Mahmud ve arkadaşının [51] yaptığı çalışmada DL araçlarının göreceli karşılaştırması sunulmuştur (Şekil 27). Yapılan çalışmada Google Trend' de derin öğrenme araçlarının aranma oranı beş yıllık dönemde araştırılmıştır. Son yıllara doğru Keras ve TensorFlow' un popülerliğinin arttığı gözlenmiştir (Şekil 27.a). Yine aynı çalışmada arXiv'deki makalelerde 2020'nin ilk çeyreğinde bahsedilme oranları ele alınmıştır. TensorFlow, PyTorch ve Keras en çok bahsedilen çerçevelerdir (Şekil 27.b). GitHub üzerindeki popülerlik araştırmasında ise yine TensorFlow en çok ilgiyi gören çerçeve olmuştur. Keras ve Caffe'nin onu takip ettiği gözlenmiştir (Şekil 27.c). Keras diğer çerçevelerle uyumlu çalışma anlamında en esnek olan araç olarak belirtilmiştir (Şekil 27.d). Son olarak bu çalışmada donanım tabanlı ölçeklenebilirliğe vurgu yapılmıştır. Gelecekte DL sistemleri için donanımsal gelişmelerin önem kazanacağı ifade edilmiştir (Şekil 27.e)



Şekil 27: a) Google Trendlerinde yıllara göre aranma, b) 2020'nin ilk çeyreğinde gönderilen makale sayısına göre, c) github toplulukları, d) birlikte çalışabilirlik, e) donanım tabanlı ölçeklendirile bilirlik [51]

Sonuç olarak TensorFlow şu anda en popüler Derin Öğrenme çerçevelerinden biridir. Keras, daha basit bir ara yüz sağlamak için TensorFlow'un üzerinde çalışan yüksek seviyeli bir API olarak sunulmuştur. Caffe, görüntü işleme alanına yönelik bir başka popüler Derin Öğrenme çerçevesidir, PyTorch ise daha genç bir çerçevedir ancak popülerlik kazanmakta ve TensorFlow'un rakibi olarak kabul edilmektedir. Bahsettiğimiz bu çerçeveler, geliştiriciler tarafından model eğitimi ve çıkarım için en çok kullanılan çerçevelerdir [50]. Derin öğrenme çerçeveleri her geçen gün geliştirilmektedir. Geliştirmelerin artması daha yüksek performans, daha az kaynak kullanımı, daha kolay uygulanabilirlik gibi amaçlar taşımaktadır. Bununla beraber araçların sayısının artması kafa karışıklığı ve uygulama seçim zorluğu getirmektedir. Karmaşayı engellemek ve tercihleri olumlu yönde etkilemek için çerçevelerin kıyaslandığı araştırma çalışmaları yapılmaktadır. Gerçekleştirdiğimiz bu çalışmayla derin öğrenme alanına geniş bir bakış açısı sunulmuştur. Literatürde ilgili alanda yapılan çalışmalar karşılaştırmalı bir şekilde ele alınmıştır. Derin öğrenme çalışmaları özellikle bulut tabanlı platformlara doğru kaymaktadır. Bir diğer önemli platform ise IoT ve mobil algılama çalışmalarıdır. Bundan sonraki süreçte araştırmacıların bulut platformların derin öğrenme çalışmalarındaki performanslarını kıyaslayan çalışmalara yöneleceği düşünülmektedir.

4 Beyanname

4.1 Rakip Çıkarlar

Bu çalışmada herhangi bir çıkar çatışması yoktur.

4.2 Yazarların Katkıları

Süleyman AKTÜRK: Derleme için fikirlerin oluşturulması, araştırma sırasında literatür taraması ile ilgili sorumluluk almak, yazının tümü veya asıl bölümün oluşturulması için sorumluluk almak, makaleyi teslim etmeden önce sadece imla ve dil bilgisi açısından değil aynı zamanda entelektüel içerik açısından yeniden çalışma yapmak.
Sorumlu Yazar Kasım SERBEST: Bulguların mantıklı açıklanması ve sunumu için sorumluluk almak, araştırma sırasında literatür taraması ile ilgili sorumluluk almak, yazının tümü veya asıl bölümün oluşturulması için sorumluluk almak, makaleyi teslim etmeden önce sadece imla ve dil bilgisi açısından değil aynı zamanda entelektüel içerik açısından yeniden çalışma yapmak.

Kaynakça

[1] E. Şimşek, Ö. Barış, and G. tümüklü Özyer, "Foto-kapan Görüntülerinde Hareketli Nesne Tespiti," *Erzincan Üniversitesi Fen Bilim. Enstitüsü Derg.*, vol. 12, no. 2, pp. 902–919, Aug. 2019, doi: 10.18185/erzifbed.509571.

[2] E. Kiliç, S. Öztürk, E. Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği, A. Kelimeler Evrişimli Sinir Ağları, A. Sayımı, and H. Görüntüleme, "İnsansız Hava Aracı Görüntülerinde Evrişimli Sinir Ağı Kullanarak Araç Sayımı için Yeni Bir Haritalama Yöntemi."

[3] Ö. Er and H. Ş. Bilge, "Bir Küçük Nesne Tespit Zorluğu Olarak Hava Görüntülerinden Araç Tespiti Vehicle Detection From Aerial Imagery As A Small Object Detection Difficulty VERİ BİLİMİ DERGİSİ www.dergipark.gov.tr/veri," Jan. 2021. Accessed: Mar. 29, 2021. [Online]. Available: www.dergipark.gov.tr/veri.

[4] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 2020, doi: 10.1016/j.neucom.2020.01.085.

[5] A. Şeker, B. Diri, and H. H. Balık, "Derin Öğrenme Yöntemleri Ve Uygulamaları Hakkında Bir İnceleme," *Gazi Mühendislik Bilim. Derg.*, vol. 3, no. 3, pp. 47–64, Dec. 2017, Accessed: May 18, 2021. [Online]. Available: https://dergipark.org.tr/en/pub/gmbd/372661.

[6] F. Chollet, *Deep Learning with Phyton*. 2018.

[7] F. DOĞAN and İ. TÜRKOĞLU, "Derin Öğrenme Modelleri ve Uygulama Alanlarına İlişkin Bir Derleme," *DÜMF Mühendislik Derg.*, vol. 10, no. 2, pp. 409–445, Jun. 2019, doi: 10.24012/dumf.411130.

[8] V. V Nabiyev, "Yapay Zeka (6. baskı)," *Ankara: Seçkin Yayıncılık*, 2021.

[9] Prof.Dr. Çetin Elmas, *Yapay Zeka Uygulamaları*, 4th ed. Ankara: Seçkin Yayıncılık, 2018.

[10] S. Murat, B. Mühendisliği, A. Dalı, and Y. Lisans, "İNSANSIZ HAVA ARACI GÖRÜNTÜLERİNDEN DERİN ÖĞRENME YÖNTEMLERİYLE NESNE TANIMA YÜKSEK LİSANS TEZİ," Maltepe Üniversitesi, Lisansüstü Eğitim Enstitüsü, 2021. Accessed: May 02, 2021. [Online]. Available: http://openaccess.maltepe.edu.tr/xmlui/handle/20.500.12415/7379.

[11] N. Gürsakal, "Makine Öğrenmesi," *Baskı, Bursa Dora Basım Yayın Dağıtım Ltd. Şti*, 2018.

[12] E. Öztemel, "Yapay sinir ağlari," *PapatyaYayincilik, Istanbul*, 2003.

[13] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015, doi: 10.1038/nature14539.

[14] R. Elshawi, A. Wahab, A. Barnawi, and S. Sakr, "DLBench: a comprehensive experimental evaluation of deep learning frameworks," *Clust. Comput. J. Networks, Softw. Tools Appl.*, p. 1, 2021, doi: 10.1007/s10586-021-03240-4.

[15] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, Mar. 2018, vol. 2018-Janua, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.

[16] C. Min, J. Xu, L. Xiao, D. Zhao, Y. Nie, and B. Dai, "Attentional graph neural network for parking-slot detection," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3445–3450, Apr. 2021, doi: 10.1109/LRA.2021.3064270.

[17] I. Gogul and V. S. Kumar, "Flower species recognition system using convolution neural networks and transfer learning," Oct. 2017, doi: 10.1109/ICSCN.2017.8085675.

[18] Ö. İnik and E. Ülker, "Derin öğrenme ve görüntü analizinde kullanılan derin öğrenme modelleri," *Gaziosmanpaşa Bilim. Araştırma Derg.*, vol. ISSN, no. 6.3, 2017.

[19] J. S. Dramsch, "70 years of machine learning in geoscience in review," Jun. 2020, doi: 10.1016/bs.agph.2020.08.002.

[20] M. Yani, B. Irawan, and C. Setiningsih, "Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail," in *Journal of Physics: Conference Series*, May 2019, vol. 1201, no. 1, doi: 10.1088/1742-6596/1201/1/012052.

[21] N. Nabiyev and S. Malekzadeh, *Anomalous Sound Localization Estimation*. 2021.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1097–1105, 2012.

[24] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8689 LNCS, no. PART 1, pp. 818–833, doi: 10.1007/978-3-319-10590-1_53.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2015, Accessed: May 18, 2021. [Online]. Available: http://www.robots.ox.ac.uk/.

[26] G. Nguyen *et al.*, "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," *Artif. Intell. Rev.*, vol. 52, pp. 77–124, 2019, doi: 10.1007/s10462-018-09679-z.

[27] A. T. KABAKUŞ, "A Comparison of the State-of-the-Art Deep Learning Platforms: An Experimental Study," *Sak. Univ. J. Comput. Inf. Sci.*, vol. 3, no. 3, pp. 169–182, Sep. 2020, doi: 10.35377/saucis.03.03.776573.

[28] K. Dinghofer and F. Hartung, "Analysis of Criteria for the Selection of Machine Learning Frameworks," in *2020 International Conference on Computing, Networking and Communications, ICNC 2020*, Feb. 2020, pp. 373–377, doi: 10.1109/ICNC47757.2020.9049650.

[29] J. Liu, Q. Huang, X. Xia, E. Shihab, D. Lo, and S. Li, "Is Using Deep Learning Frameworks Free? Characterizing Technical Debt in Deep Learning Frameworks," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 2020, pp. 1–10.

[30] "Neden TensorFlow." https://www.tensorflow.org/about?hl=tr (accessed May 19, 2021).

[31] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," 2016.

[32] "TensorFlow Lite | Mobil ve Uç Cihazlar için Makine Öğrenimi." https://www.tensorflow.org/lite/?hl=tr (accessed May 19, 2021).

[33] "Neden Keras'ı seçmelisiniz?" https://keras.io/why_keras/ (accessed May 19, 2021).

[34] The Theano Development Team *et al.*, "Theano: A Python framework for fast computation of mathematical expressions," May 2016, Accessed: May 19, 2021. [Online]. Available: http://arxiv.org/abs/1605.02688.

[35] M. M. Yapıcı and N. Topaloğlu, "Performance comparison of deep learning frameworks," 2021. Accessed: May 18, 2021. [Online]. Available: https://dergipark.org.tr/tr/pub/ci.

[36] A. Uçar, Ö. H. Üniversitesi, and M. Bölümü, "Derin öğrenmenin Caffe kullanılarak grafik işleme kartlarında değerlendirilmesi Mehmet Safa BİNGÖL," 2018.

[37] G. Al-Bdour, R. Al-Qurran, M. Al-Ayyoub, and A. Shatnawi, "Benchmarking open source deep learning frameworks," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 5, pp. 5479–5486, 2020, doi: 10.11591/ijece.v10i5.pp5479-5486.

[38] "The Microsoft Cognitive Toolkit - Cognitive Toolkit - CNTK | Microsoft Docs." https://docs.microsoft.com/tr-tr/cognitive-toolkit/ (accessed May 20, 2021).

[39] "CNTK_2_7_Release_Notes - Bilişsel Araç Seti - CNTK | Microsoft Docs." https://docs.microsoft.com/en-us/cognitive-toolkit/releasenotes/cntk_2_7_release_notes (accessed May 20, 2021).

[40] "pytorch/pytorch: Tensors and Dynamic neural networks in Python with strong GPU acceleration." https://github.com/pytorch/pytorch (accessed May 20, 2021).

[41] T. Chen *et al.*, "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," Dec. 2015, Accessed: May 20, 2021. [Online]. Available: http://arxiv.org/abs/1512.01274.

[42] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a Next-Generation Open Source Framework for Deep Learning."

[43] "Chainer – A flexible framework of neural networks — Chainer 7.7.0 documentation." https://docs.chainer.org/en/stable/ (accessed May 20, 2021).

[44] "chainer/chainer: A flexible framework of neural networks for deep learning." https://github.com/chainer/chainer (accessed May 20, 2021).

[45] W. Dai and D. Berleant, "Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics," in *Proceedings - 2019 IEEE 1st International Conference on Cognitive Machine Intelligence, CogMI 2019*, Dec. 2019, pp. 148–155, doi: 10.1109/CogMI48466.2019.00029.

[46] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative Study of Deep Learning Software Frameworks," Nov. 2015, Accessed: May 19, 2021. [Online]. Available: http://arxiv.org/abs/1511.06435.

[47] Y. Wu *et al.*, "A Comparative Measurement Study of Deep Learning as a Service Framework," *ieeexplore.ieee.org*, doi: 10.1109/TSC.2019.2928551.

[48] J. Liu *et al.*, "Usability Study of Distributed Deep Learning Frameworks For Convolutional Neural Networks," 2018. Accessed: May 21, 2021. [Online]. Available: https://caffe2.ai/.

[49] S. Shi, Q. Wang, and X. Chu, "Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2018, pp. 949–957, doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.000-4.

[50] Y. Dai, R. Zhang, R. Xue, B. Liu, and T. Li, "Towards Efficient Execution of Mainstream Deep Learning Frameworks on Mobile Devices: Architectural Implications," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 40, no. 3, pp. 453–466, Mar. 2020, doi: 10.1109/TCAD.2020.3003233.

[51] M. Mahmud, M. S. Kaiser, T. M. McGinnity, and A. Hussain, "Deep Learning in Mining Biological Data," *Cognitive Computation*, vol. 13, no. 1. Springer, p. 3, Jan. 01, 2021, doi: 10.1007/s12559-020-09773-x.