Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi
Bilecik Şeyh Edebali University Journal of Science
e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

Araştırma Makalesi – Research Article
Geliş Tarihi / Received: 01/07/2025
Kabul Tarihi / Accepted: 15/10/2025
Yayın Tarihi / Published: 30/11/2025

Derin Öğrenme Modellerinin Testi için Ortak Platform Tasarımı

Designing a Common Platform for Testing Deep Learning Models

İbrahim Kuru[1], Hatice Küpeli[2*], Kerim Kürşat Çevik[3]

[1]Akdeniz Üniversitesi/ Sosyal Bilimler Enstitüsü/ Yönetim Bilişim Sistemleri Anabilim Dalı/ Antalya, Türkiye/ ibo.kuru@gmail.com, https://orcid.org/0000-0003-3362-3725

[2*]Akdeniz Üniversitesi/ Sosyal Bilimler Enstitüsü/ Yönetim Bilişim Sistemleri Anabilim Dalı/ Antalya, Türkiye/ haticekupeli46@gmail.com/ https://orcid.org/0009-0005-2872-5841

[3]Akdeniz Üniversitesi/ Uygulamalı Bilimler Fakültesi/ Yönetim Bilişim Sistemleri Bölümü/ Antalya, Türkiye/ kcevik@akdeniz.edu.tr/ https://orcid.org/0000-0002-2921-506X

Etik Beyan: Bu çalışmanın hazırlanma sürecinde bilimsel ve etik ilkelere uyulduğu ve yararlanılan tüm çalışmaların kaynakçada belirtildiği beyan olunur.

Yapay Zeka Etik Beyanı: Yazar bu makalenin hazırlanma sürecinin hiç bir aşamasında yapay zekadan faydalanılmadığını; bu konuda tüm sorumluluğun kendisine(kendilerine) ait olduğunu beyan etmektedir.

Çıkar Çatışması: Çıkar çatışması beyan edilmemiştir.

Finansman: Bu araştırmayı desteklemek için dış fon kullanılmamıştır.

Ethical Statement: It is declared that scientific and ethical principles were followed during the preparation of this study and that all studies used are stated in the bibliography.

Artificial Intelligence Ethical Statement: The author declares that artificial intelligence was not utilized at any stage of the preparation process of this article and accepts full responsibility in this regard.

Conflicts of Interest: The author(s) has no conflict of interest to declare.

Grant Support: The author(s) acknowledge that they received no external funding to support this research.

Derin Öğrenme Modellerinin Testi için Ortak Platform Tasarımı

ÖZ

Bu araştırmanın amacı, kullanıcıların herhangi bir yazılım bilgisine ihtiyaç duymaksızın farklı derin öğrenme modellerini mobil cihazlar üzerinden test edebilecekleri bir uygulama geliştirmektir. Geliştirilen platform, Flutter yazılım çatısı kullanılarak hazırlanmış olup, TensorFlow Lite (TFLite) formatındaki modellerin çeşitli veri kümeleri ile denenmesini mümkün kılmaktadır. Uygulamanın temel işlevi, DeepLabV3, MobileNet ve YOLOv2 gibi yaygın kullanılan derin öğrenme mimarilerini destekleyerek model doğruluğu, işlem süresi ve bellek kullanımı gibi performans ölçütlerine dair kullanıcıya geri bildirim sunmaktır. Böylece, yalnızca belirli modellerle sınırlı kalan mevcut uygulamaların aksine, daha genel ve genişletilebilir bir yapı önerilmektedir. Çift platform desteği (Android ve iOS), sade kullanıcı arayüzü ve kolay kullanılabilirlik özellikleri sayesinde teknik bilgisi sınırlı kullanıcılar için erişilebilir bir çözüm sunulmaktadır. Elde edilen bulgular, geliştirilen platformun model test süreçlerini demokratikleştirerek derin öğrenme teknolojilerinin daha geniş kullanıcı gruplarına ulaşmasında etkili olabileceğini göstermektedir. Gelecek çalışmalarda, PyTorch ve ONNX formatlarının entegrasyonu, bulut tabanlı veri işleme sistemleriyle uyumluluk ve mobil donanım sınırlamalarına yönelik optimizasyon çalışmaları hedeflenmektedir.

Anahtar Kelimeler- Derin Öğrenme, Mobil Platform, TensorFlow, Flutter

Öne Çıkanlar

Bu çalışmada, kullanıcıların herhangi bir yazılım bilgisine ihtiyaç duymaksızın farklı derin öğrenme modellerini mobil cihazlar üzerinden test edebilecekleri bir platform geliştirilmiştir.

- Flutter yazılım çatısı kullanılarak geliştirilen platform, TFLite formatındaki modelleri destekler ve Android ile iOS için çapraz platform uyumludur.
- Uygulama, DeepLabV3, MobileNet ve YOLOv2 gibi farklı mimarileri destekleyerek model doğruluğu, işlem süresi ve bellek kullanımı gibi performans ölçütleri sunar.
- Test sonuçları, MobileNet gibi sınıflandırma modellerinde mobil platformun referans sistemlerle (%94-%97,7) birebir aynı doğrulukta sonuçlar verdiğini göstermiştir.
- Geliştirilen bu genel ve genişletilebilir platform, mevcut tekil model uygulamalarının aksine, derin öğrenme model test süreçlerini demokratikleştirerek teknolojinin daha geniş kitlelere ulaşmasını hedeflemektedir.

Designing a Common Platform for Testing Deep Learning Models

ABSTRACT

The aim of this study is to develop a mobile application that enables users to test various deep learning models on mobile devices without requiring any programming knowledge. The platform was developed using the Flutter framework and allows for the testing of models in TensorFlow Lite (TFLite) format with different datasets. The core functionality of the application includes supporting commonly used deep learning architectures, such as DeepLabV3, MobileNet, and YOLOv2, while providing feedback on performance metrics, such as accuracy, processing time, and memory usage. Unlike existing applications that are often limited to specific models, the proposed platform offers a generalized and extensible structure. With its cross-platform compatibility (Android and iOS), user-friendly interface, and ease of use, the application provides an accessible solution for users with limited technical background. The findings indicate that the platform can contribute to democratizing the model testing process and facilitating broader access to deep learning technologies. Future work will focus on extending compatibility to other model formats, such as PyTorch and ONNX, integrating with cloud-based processing systems, and optimizing performance for mobile hardware limitations.

Keywords- Deep learning, mobile application, TensorFlow, Flutter

Highlights

This study develops a mobile application that enables users to test various deep learning models on mobile devices without requiring any programming knowledge.

- The platform, developed using the Flutter framework, supports models in TFLite format and is cross-platform compatible for Android and iOS.
- The application supports various architectures like DeepLabV3, MobileNet, and YOLOv2, providing feedback on performance metrics such as accuracy, processing time, and memory usage.
- Test results showed that classification models like MobileNet produced identical accuracy (94%-97.7%) on the mobile platform compared to reference systems.
- Unlike existing single-model applications, this generalized and extensible platform aims to democratize the model testing process and facilitate broader access to deep learning technologies.

I. GİRİŞ

Derin öğrenme (DL), çoklu işlem katmanlarından oluşan hesaplama modellerinin çoklu soyutlama seviyelerine sahip veri temsillerini öğrenmesine olanak tanımaktadır. Bu yöntemler, konuşma tanıma, görsel tanıma, nesne algılama, tıbbi görüntü işleme, ilaç keşfi ve genomik gibi birçok alanda en son teknolojiyi önemli ölçüde geliştirmiştir. Derin öğrenme, büyük veri kümelerindeki karmaşık yapıları keşfetmek için geriye yayılma algoritmasını kullanarak bir makinenin her katmandaki temsilini, bir önceki katmandaki temsilden hesaplamak için dahili parametreleri nasıl değiştirmesi gerektiğini belirler. Derin konvolüsyonel ağlar görüntü, video, konuşma ve ses işlemede başarılıyken, tekrarlayan ağlar metin ve konuşma gibi sıralı verilere etkili olmuştur [1]. Derin öğrenme teknikleri, çeşitli son teknoloji multimedya sistemlerinin ve bilgisayarla görmenin ana parçaları haline gelmiştir. Daha spesifik olarak, CNN'ler görüntü işleme, nesne algılama ve video işleme dahil olmak üzere farklı gerçek dünya görevlerinde önemli sonuçlar göstermiştir [2].

DL, günümüzde birçok son teknoloji multimedya sisteminin ve bilgisayarla görme alanının temelini oluşturmaktadır. Derin sinir ağları (DNN), büyük veri kümelerinden anlamlı örüntüler öğrenme yetenekleri sayesinde görüntü işleme, ses tanıma, doğal dil işleme ve video işleme gibi çeşitli alanlarda olağanüstü performans göstermektedir [3]. Derin öğrenme modellerinin en bilinen türlerinden biri olan Evrişimsel Sinir Ağları (CNN), özellikle görüntü sınıflandırma, nesne algılama ve semantik segmentasyon gibi görevlerde başarı sağlamaktadır [4]. CNN'ler, katmanlar arasında görüntüdeki mekânsal ilişkileri koruyarak derin özellik çıkarımı yapabilmektedir.

Bir diğer DL modeli olan; Yinelenen Sinir Ağları (RNN) ve türevi olan Uzun Kısa Süreli Bellek (LSTM) ağları, özellikle zaman serisi verilerle, yani ardışık verilerle çalışan görevlerde kullanılır. RNN'ler, doğal dil işleme, makine çevirisi ve video sıralarını analiz etme gibi alanlarda öne çıkmaktadır [5]. Ayrıca, son yıllarda büyük ilgi gören Üretken Çekişmeli Ağlar (GAN), iki sinir ağının (üretici ve ayırt edici ağlar) birbirlerine karşı yarıştığı bir öğrenme süreciyle, gerçekçi görüntü ve video üretiminde çığır açmıştır [6].

DL modellerinin başarısı büyük oranda kullanılan mimari ve optimizasyon tekniklerine bağlıdır. Doğru model mimarisi ve uygun eğitim stratejileri, daha hızlı ve doğru sınıflandırmalar yapılmasını sağlamaktadır. Örneğin, optimizasyon sürecinde Stokastik Gradient Descent (SGD) gibi algoritmalar kullanılırken, ReLU ve Batch Normalization gibi teknikler de model performansını artıran önemli bileşenlerdir [7]. Transfer öğrenimi de başka bir önemli yaklaşımdır; eğitim öncesinde büyük bir veri kümesinde eğitilmiş modeller, daha küçük ve özel veri kümelerinde yeniden eğitilerek daha verimli sonuçlar elde edilmesini sağlar [8]. Bu yöntemlerin başarıları, ImageNet gibi büyük veri kümeleri üzerinde test edilmiş ve özellikle CNN'lerin görüntü sınıflandırmadaki doğruluğu büyük ölçüde artırdığı gözlemlenmiştir [9].

Eğitim öncesinde uygun bir derin öğrenme model mimarisine ihtiyaç duyulmaktadır. İyi bir model mimarisi, daha doğru ve hızlı sınıflandırmayı sağlamaktadır. Günümüzde kullanılan derin öğrenme ana ağ türleri CNN, RNN ve üretken çekişmeli ağlardır (GAN) [10].

Gerçekleştirilen çalışmada, Yapay Zekâ (YZ) alanında çalışan akademisyenler ve firmaların oluşturdukları derin öğrenme (DL) modellerinin test edilebilmesi için ortak bir platform tasarlanması amaçlanmıştır. Bu platform, DL modellerinin daha geniş bir kitle tarafından değerlendirilmesini ve geliştirilmesini kolaylaştırmayı hedeflemektedir. Tasarlanan bu platform, kullanıcıların herhangi bir yazılım bilgisine ihtiyaç duymadan eğitilmiş modelleri yükleyip test edebilmesine olanak tanır. Bu, özellikle yapay zekâ ve derin öğrenme alanında teknik bilgisi sınırlı olan kullanıcıların bile modelleri deneyimlemesine ve test sonuçlarını gözlemlemesine imkân sunarak, model performansının gerçek dünya senaryolarında değerlendirilmesini sağlayacaktır. Ayrıca bir model üzerinde çalışan akademisyen veya firmaların yayınladıkları modellerin son kullanıcı tarafından kolaylıkla test edilmesi tasarlanan yazılım ile sağlanabilecektir.

A. Literatür Taraması

Çalışmanın bu bölümünde, benzer amaçlara hizmet eden akademik çalışmalar detaylı bir şekilde incelenmiş ve literatür bilgisi sunulmuştur.

2021 yılında Karar ve arkadaşları yaptıkları çalışmada uzmanları ve çiftçileri desteklemek için derin öğrenmeyi kullanarak tarımsal zararlıları tanımaya yönelik yeni bir mobil uygulama geliştirmişlerdir. Çalışmanın ana hedefi hem büyük çiftlikler hem de seralar gibi açık alanlarda belirli mahsullere yönelik tarımsal zararlıların çevrimiçi olarak tanınmasına yönelik uygulama oluşturmaktır. Uygulama geliştirme içi arayüz olarak Apache Cordova kullanılmıştır. Geliştirilen uygulamada, bulut bilişime dayalı böcek zararlılarını tanıma görevini gerçekleştirmek için Faster R-CNN kullanılmıştır ve en iyi sonuçların %99,0 doğruluk oranı ile Faster R-CNN de elde edildiği belirtilmiştir [11].

2020 yılında Ngugi ve arkadaşlarının yaptığı çalışmada derin öğrenme ile yaprak hastalıklarının tanınması için bir uygulama geliştirilmiştir. Yapılan çalışmada kontrollü aydınlatma ve tekdüze arka plan koşulları altında çekilen laboratuvar görüntülerine dayanan çalışmaların gerçek saha koşullarında test edildiğinde performanslarının keskin bir şekilde düştüğü belirtilmiştir. Bu çalışmada arka planın kaldırılmasının hastalık tanıma doğruluğunu artırdığı için yaprak görüntüleri için otomatik arka plan çıkarımı gerçekleştirmek üzere evrişimli sinir ağları kullanılmıştır. 1.408 domates yaprağı görüntüsü üzerinde yapılan çalışmada kullanılan SegNet, UNet, KijaniNet mimarileri ile yüksek doğruluk oranları elde edilmiştir. Yapılan çalışmada önerilen ağlar da, 0,96 doğruluk elde edilmiştir [12].

Şahin ve arkadaşları tarafından 2022 yılında yapılan başka bir çalışmada maymun çiçeği hastalığını hızlı bir şekilde tespit etmek derin öğrenmeyi kullanan bir Android mobil uygulamasını geliştirilmiştir. Uygulama, Java programlama dili ve Android SDK 12 kullanılarak Android Studio ile geliştirilmiştir. Mobil cihazın kamerası aracılığıyla toplanan video görüntüleri, aynı cihaz üzerinde çalışan derin evrişimli bir sinir ağına gönderilerek görüntüler pozitif ve negatif olarak sınıflandırılmaktadır. Eğitim ve test işlemleri Matlab üzerinde önceden eğitilmiş ağlar kullanılarak yapılmıştır. Model %91,11 doğrulukla sınıflandırabildiği belirtilmiştir. TensorFlow kullanılarak oluşturulan en iyi doğruluğa sahip ağ TensorFlow Lite modeline dönüştürülerek mobil cihazlara uyarlanmıştır. Geliştirilen uygulama üç cihazda başarıyla çalıştırılmıştır [13].

Gençtürk ve arkadaşları tarafından 2023 yılında yapılan çalışmada fındığın ayıklama sürecinde harcanan emek, zaman ve maliyetin en aza indirilmesi amacıyla derin öğrenme algoritmaları kullanılarak sınıflandırılması için bir mobil uygulama geliştirilmiştir. Çalışmada Giresun, Ordu ve Van fındık çeşitlerinden toplam 3627 görselden oluşan veri seti oluşturulmuştur. Sınıflandırma InceptionV3 ve ResNet50 gibi derin öğrenme modelleri kullanılarak yapılmıştır ve model %100 sınıflandırma doğruluğuna sahiptir [14].

Awasthi ve arkadaşları 2021 yılında yaptıkları çalışmada akciğer görüntülerini kullanarak COVID-19'un tespiti için hafif, mobil uyumlu, verimli bir derin öğrenme modeli geliştirmişlerdir. COVID-19, zatürre ve sağlıklı olmak üzere üç farklı sınıf verileri ile eğitimi yapılan Mini-COVIDNet son teknoloji ürünü ağır modellerin yanı sıra diğer hafif sinir ağı modelleriyle karşılaştırılmıştır. Önerilen ağın en yüksek doğruluk oranı olan %83,2'ye ulaşabileceği ve yalnızca 24 dakikalık bir eğitim süresi gerektirdiği belirtilmiştir. Önerilen Mini-COVIDNet, bir sonraki en iyi performansa sahip ağa kıyasla 4,39 kat daha az sayıda parametreye sahiptir ve yalnızca 51,29 MB'lık bir bellek gerektirdiği vurgulanmıştır. Bu durum mobil platformlarda bu ağın optimum performans sergileyeceğini göstermektedir [15].

Cabbar ve arkadaşları 2020 yılında yaptıkları çalışmada derin sinir ağı tabanlı yöntemler kullanılarak mikro uyku ve uyuşukluğun tespitine yönelik model geliştirmişlerdir. Geliştirilen derin öğrenme modeli, Ulusal Tsing Hua Üniversitesi (NTHU) Sürücü Uyuşukluğu Tespit Veri Kümesinden elde edilen videolarla eğitilmiştir. Makalede uykululuğu sınıflandırmak için kamera tarafından tespit edilen ve Evrişimli Sinir Ağına (CNN) aktarılan yüz işaretlerinden yararlanılarak doğruluk artırılmıştır. Bu çalışmanın başarısı, gözlüksüz kategori için %88'in üzerinde, gözlüksüz gece kategorisi için %85'in üzerinde ve modelin maksimum boyutun 75 KB olarak diğer ağır sınıflandırma modellerine göre daha hafif bir alternatif sunabilmektedir. Geliştirilen android mobil uygulama ile sürücünün fotoğrafları alınmaktadır. Dlib kütüphanesi görüntü verilerini aldıktan sonra ön işlemeyi yapmakta ve yüz işaretlerini çıkararak verileri geliştirilen CNN modeline göndermektedir. Bu veriler sinir ağı üzerinden geçerek algoritma, sürücünün uykulu olup olmadığını değerlendirmektedir. Görüntülerin sonuçları uygulamada gerçek zamanlı olarak gösterilmekte; Sürücünün uykulu olduğunun tespit edilmesi halinde uygulama, görsel ve sesli mesajlarla bildirim göndermektedir [16].

2021 yılında Loyani ve Machuve tarafından yapılan çalışmada, tarımsal üretimde önemli kayıplara neden olan Tuta Absoluta zararlısının domates yapraklarındaki tahribatını tespit edebilen derin öğrenme tabanlı bir mobil uygulama geliştirilmiştir. Yapılan çalışmada U-Net mimarisi kullanılarak 1212 domates yaprağı görüntüsünden oluşan veri seti üzerinde eğitim yapılmış, model %82,86 Dice katsayısı ile yüksek segmentasyon başarısı göstermiştir. Eğitim sonrası model TensorFlow Lite formatına dönüştürülerek Android tabanlı akıllı telefonlara

entegre edilmiştir. "TutaSegmenter" adı verilen uygulama, yalnızca 5 saniye içerisinde %70 güven aralığında sonuç üretebilmekte ve çevrimdışı çalışabilmektedir. Bu sayede özellikle internet erişiminin kısıtlı olduğu kırsal bölgelerde çiftçiler için pratik ve düşük maliyetli bir çözüm sunulmuştur [17].

2021 yılında Ahmed ve Reddy tarafından yapılan başka bir çalışmada, bitki hastalıklarının hızlı ve doğru teşhisi için derin öğrenme tabanlı mobil bir uygulama geliştirilmiştir. Çalışmada 14 farklı bitki türüne ait 38 sınıftan oluşan toplam 96.206 görüntü kullanılmıştır. Geliştirilen CNN modeli %94,13 doğruluk oranı ile sınıflandırma başarısı göstermiştir. Model, mobil cihazlar üzerinden çekilen yaprak görüntülerini bulut destekli bir sistemde analiz ederek sonuçları kullanıcıya vermektedir. Böylelikle çiftçilerin yanlış teşhis ve tedavi yöntemlerinden kaynaklanan maliyetlerini azaltmak ve tarımsal verimliliği artırmak hedeflenmiştir. Geliştirilen sistem Android tabanlı bir uygulama üzerinden kullanıma sunulmuş ve gerçek saha koşullarında test edilmiştir [18].

2024 yılında Kimeu ve arkadaşları tarafından yapılan çalışmada, pnömoni hastalığının tespit ve sınıflandırılması amacıyla derin öğrenme tabanlı bir mobil uygulama geliştirilmiştir. Çalışmada VGG, ResNet ve EfficientNet gibi farklı evrişimli sinir ağı mimarileri kullanılmıştır. Mobil cihazlar üzerinden çekilen akciğer görüntüleri bu modellerle analiz edilerek pozitif ve negatif sınıflandırma yapılmıştır. Geliştirilen sistem, mobil cihazlarda çalışacak şekilde optimize edilmiş ve gerçek zamanlı tespit yapılabilmiştir. Çalışma sonucunda, EfficientNet mimarisinin diğer modellere kıyasla daha yüksek doğruluk oranı sağladığı ve mobil uygulamalara entegre edilmek üzere uygun performans sergilediği belirtilmiştir [19].

Literatür ile ilgili yapılan ayrıntılı incelemelerde, geliştirilen mobil uygulamaların çoğunlukla tek bir modele özgü olduğu ve genel, yaygın kullanıma geçemedikleri görülmektedir. Bu durum, günümüzde kullanılan derin öğrenme (DL) modellerinin test edilebilmesi için ortak bir uygulamaya duyulan ihtiyacı açıkça ortaya koymaktadır. Ayrıca literatürde dikkat çeken bir diğer sorun, hazırlanan veri setleriyle eğitilen modellerin ürünleştirilme sürecindeki zorluklardır. Çoğu çalışmada test ve tahmin (prediction) işlemleri programcılar tarafından Python kodları aracılığıyla gerçekleştirilmektedir. Ancak eğitilmiş bir modelin farklı araştırmacılar veya kullanıcılar tarafından test edilebilmesi için bu kişilerin yazılım bilgisine sahip olmaları gerekmektedir. Bu ihtiyacı karşılamak amacıyla yürütülen çalışmada, günümüzde yaygın olarak kullanılan bazı DL modellerinin test edilmesine imkân veren bir uygulama arayüzü tasarlanmıştır. Geliştirilen arayüz, eğitilmiş modelleri TensorFlow Lite (TFLite) formatına dönüştürerek herhangi bir kullanıcıya gönderme olanağı sunmaktadır. Kullanıcı, bu modeli uygulamaya yükleyerek herhangi bir kod bilgisine ihtiyaç duymadan test işlemini gerçekleştirebilmektedir. Bu yaklaşım, yapay zekâ modellerinin daha geniş kullanıcı kitleleri tarafından erişilebilir olmasını sağlayarak kullanım yaygınlığını artıracaktır. Böylece eğitilen modellerin çok daha geniş bir kesim tarafından test edilmesi, geri bildirimlerle geliştirilmesi ve gerçek yaşam uygulamalarındaki kullanım oranının yükseltilmesi mümkün olacaktır. Geliştirilen arayüz, farklı DL modelleri ile test edilmiş ve elde edilen sonuçlar makale kapsamında sunulmuştur.

II. MATERYAL VE METOT

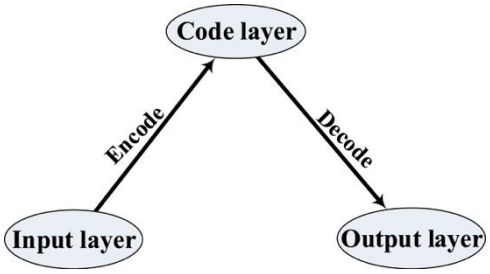
A.Derin Öğrenme

Son yıllarda makine öğrenimi alanında, gelişmiş öğrenme algoritmaları ve verimli ön işleme tekniklerinde önemli ilerlemelere tanık olunmuştur. Bu ilerlemelerden biri ise yapay sinir ağlarının (YSA) derin sinir ağı mimarilerine evrilerek Derin Öğrenme (DL) olarak özetlenen gelişmiş öğrenme yeteneklerine sahip olmasıdır. Kapalı ortamlardaki belirli uygulamalar için DL insanüstü performans gösterdiği görülmektedir [20]. DL, özellikle büyük ve yüksek boyutlu verilere sahip alanlarda kullanışlıdır. Bu sebeple, derin sinir ağları, metin, görüntü, video, konuşma ve ses verilerinin işlenmesi gereken çoğu uygulama için sığ Makine Öğrenimi (ML) algoritmalarından daha iyi performans sergilemektedir [21].

DL'de ilk adım 1943'te nörofizyolog Warren McCulloch ve matematikçi Walter Pitts'in nöronların nasıl işleyebileceğine dair bir makale yazmalarıyla gerçekleşmiştir. Makalede elektrik devrelerine sahip temel bir sinir ağı önerilmiştir. 1949'da Donald Hebb, sinir yollarının her kullanıldığında güçlendirildiğine dair bir teori ortaya koymuştur [22]. 1950'lerde, IBM'den Nathanial Rochester, IBM 704 bilgisayarlarında simüle edilmiş soyut sinir ağı üzerine çalışma yapmıştır [23]. 1956'da John McCarthy, Marvin L. Minsky, Nathaniel Rochester ve Claude E. Shannon yapay zekâ üzerine Dartmouth Yaz Araştırması projesi olarak bilinen bir yaz projesinde birlikte çalışmışlardır. Bu çalışma yapay zekâ araştırmalarında bir sıçrama sağlamıştır [24]. Bu çalışmadan sonra 1957'deki John Von Neumann, basit nöron fonksiyonunu taklit etmek için telgraf rölelerinin veya vakum tüplerinin kullanılabileceğini öne sürmüştür. 1958'de Cornell'in nörobiyoloğu Frank Rosenblatt Perceptron derin öğrenmenin temelini oluşturan ilk sinir ağı olan Perceptron'u donanımsal olarak inşa etmiştir ve günümüzde kullanılmaya devam etmektedir [10]. Gerçek dünyadaki bir soruna uygulanabilecek ilk sinir ağı 1959 yılında Stanford'da Bernard Widrow ve Marcian Hoff tarafından oluşturulan ADALINE ve MADALINE isimli model olmuştur. 1997 yılında Schmidhuber ve Hochreiter tarafından tekrarlayan bir sinir ağı yapısı olan LSTM önerilmiştir. Önceki

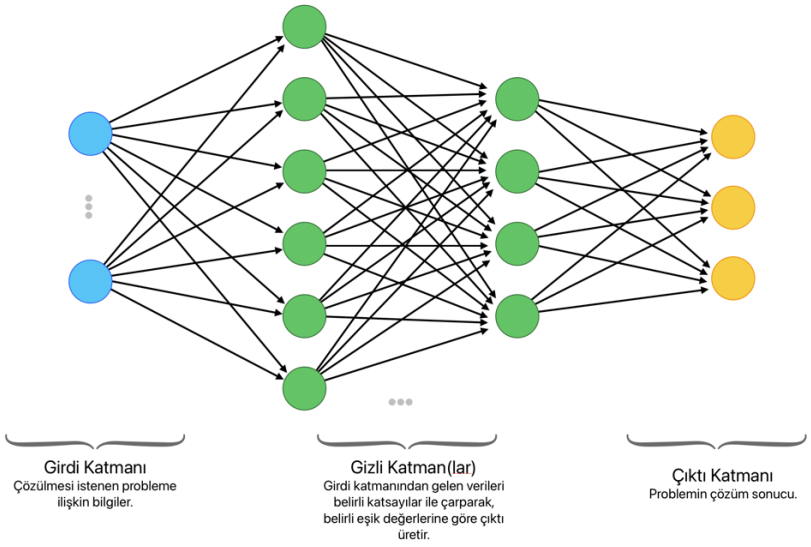
çalışmalarda kullanılan ileri beslemeli bağlantıların yerine, bir RNN mimarisi olan LSTM'de geri besleme bağlantıları bulunmaktadır [25].

Günümüzde kullanılan temel DL mimarileri ise yığılmış otomatik kodlayıcı, derin inanç ağı, derin Boltzmann makinesi, evrişim sinir ağı vb. içermektedir. Şekil 1'de temel modellerin yapısı verilmiştir.

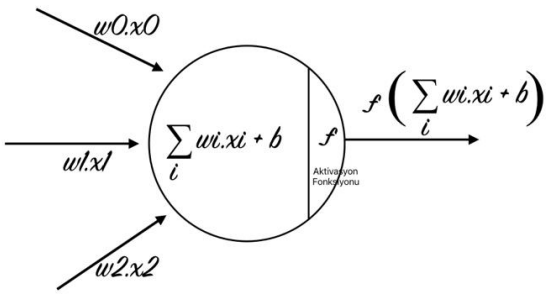


Şekil 1. DL Model Yapısı

DL canlıların sinir sistemi vasıtasıyla öğrenme yapısının modellenmesi/taklit edilmesi üzerine kurulu yapay öğrenme tekniğidir. DL girdi, gizli ve çıktı katmanlarında organize edilmiş işlem birimlerinden oluşur. Şekil 2 de bu katmanlar gösterilmiştir. Her katmandaki düğümler veya birimler, bitişik katmanlardaki düğümlere bağlanmaktadır. Her bağlantının bir ağırlık değeri bulunmaktadır. Girdiler ilgili ağırlıklarla çarpılmakta ve her birimde toplanmaktadır. Elde edilen toplam daha sonra, sigmoid fonksiyon, tan hiperbolik veya düzeltilmiş doğrusal birim (ReLU) olan aktivasyon fonksiyonuna dayalı bir dönüşüme tabi tutulmaktadır. Bu dönüşüm sonucuna göre nöronun aktif olup olmayacağı belirlenir. Şekil 3' de bu işleme ait görsel verilmiştir. Giriş bilgisi Sigmoid fonksiyonu ile [0,1] aralığına, TanH fonksiyonu ile [-1, 1] aralığına ve ReLu fonksiyonu ile [0, □] aralığına dönüştürülmektedir. Fonksiyonun çıktısı bir sonraki katmandaki bir sonraki birime girdi olarak gönderilir. Son çıktı katmanının sonucu problemin çözümü olarak kullanılır [26].

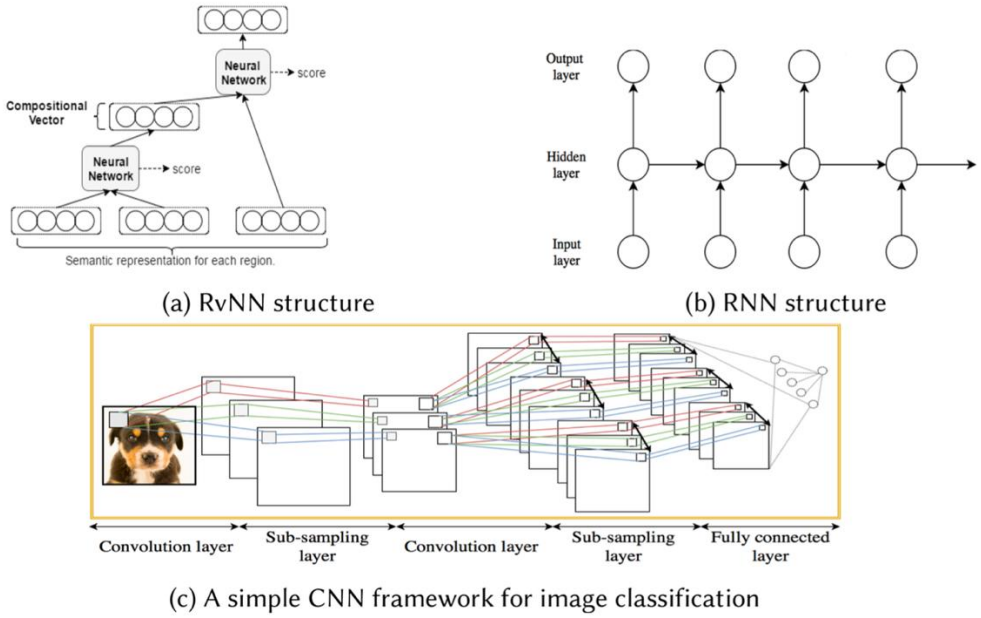


Şekil 2. DL Katmanları



Şekil 3. DL için her bir sinir hücresi yapısı

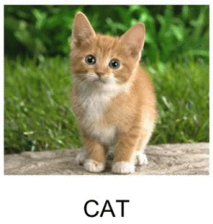
Çok fazla ve çeşitte DL algoritmaları bulunmaktadır. Bunlardan bazıları RvNN, RNN ve CNN'dir. Bu algoritmaların mimarileri Şekil 4'te gösterilmiştir.



Şekil 4. RvNN, RNN ve CNN Mimarileri [2]

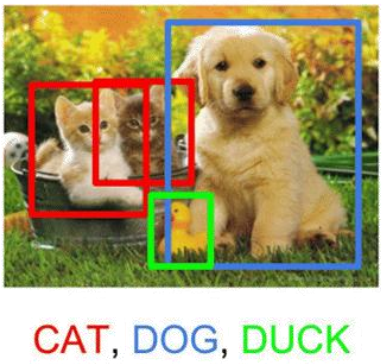
Günümüzde DL teknikleri, ileri seviye multimedya sistemleri ve bilgisayarla görme alanlarında temel bileşenler haline gelmiştir. Özellikle, Evrişimli Sinir Ağları ile (CNN'ler), görüntü işleme, nesne tanıma ve video analizi gibi çeşitli uygulamalarda önemli sonuçlar elde edildiği görülmektedir. [2]. DL teknikleri Image Classification (Görüntü sınıflandırma), Object Detection (Nesne Tanıma), Semantic segmentation (Anlamsal Bölütleme), Human pose estimation (İnsan Pozu Tahmini) gibi bazı alanlarda kullanılmaktadır.

Görüntü Sınıflandırma (Image Classification): Görüntü sınıflandırma görevi, giriş görüntülerinin belirli bir görsel nesne sınıfının varlığına ilişkin bir olasılıkla etiketlenmesinden oluşmaktadır. Lenet – 5, AlexNet, VGGNet, Resnet, Inception v3 bu alanda en çok kullanılan modellerdir [27] (Şekil 5).



Şekil 5. Görüntü Sınıflandırma

Nesne Tanıma (Object Detection): Nesne tespiti, görüntü sınıflandırma görevinden farklıdır ancak onunla yakından ilişkilidir. Görüntü sınıflandırma için tüm görüntü girdi olarak kullanılır ve görüntüdeki nesnelerin sınıf etiketi tahmin edilir. Nesne tespiti için, belirli bir sınıfın varlığı bilgisinin çıktısının yanı sıra, konumunun da tahmin edilmesi gerekir. Çıktısı alınan sınırlayıcı kutu, temel gerçek nesnesiyle yeterince büyük bir örtüşmeye sahipse (genellikle %50'den fazla) bir algılama penceresi doğru olarak kabul edilir. R-CNN, YOLO, SSD, RetinaNet, Mask R-CNN, EfficientDet nesne tanımada en çok kullanılan modellerdir [28] (Şekil 6).



Şekil 6. Nesne Tanıma

Anlamsal Bölütleme (Semantic Segmentation): Görüntü düzeyinde sınıflandırma ve nesne düzeyinde tespitten farklı olarak anlamsal bölütleme, 2 boyutlu uzamsal dağılıma sahip çıktı maskeleri gerektirir. Fully Convolutional Networks, U-Net, SegNet, DeepLab, PSPNet, Mask R-CNN, Enet anlamsal bölütlemede en çok kullanılan modellerdir [29] (Şekil 7).

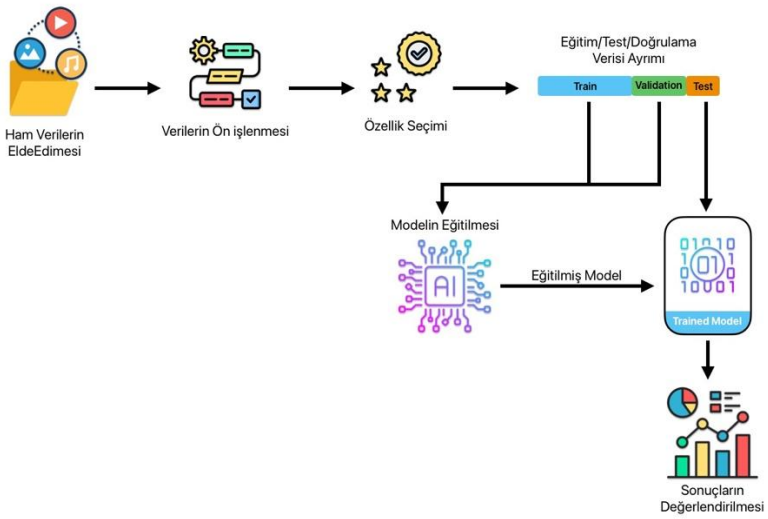
Şekil 7. Anlamsal Bölütleme

İnsan Pozu Tahmini (Human Pose Estimation): Hareketsiz görüntülerden veya görüntü dizilerinden insan eklemlerinin konumunu tahmin etmeyi amaçlamaktadır. Video gözetimi, insan davranışı analizi, insan-bilgisayar etkileşimi [30] gibi çok çeşitli potansiyel uygulamalar için çok önemlidir ve son zamanlarda kapsamlı bir şekilde incelenmektedir. OpenPose, DeepPose, Convolutional Pose Machines (CPM), Hourglass Network, PoseNet, AlphaPose, DensePose, HRNet insan pozu tahmininde en çok kullanılan modellerdir [31] (Şekil 8).

Şekil 8. İnsan Pozu Tahmini

B.DL Model Eğitim/Test Süreci:

DL modelinin geliştirilmesi ve uygulanması sürecinde izlenen temel aşamalar Şekil 9'da gösterilmiştir. Şekil 9'da belirtilen her bir adım, modelin genel başarısını ve genelleme yeteneğini artırmak için önemlidir.



Şekil 9. DL modellerinin eğitim/test süreçleri

Ham Verilerin Elde Edilmesi: Bu aşamada, modelin eğitimi için gerekli olan veriler temin edilir. Veriler, çeşitli kaynaklardan elde edilebilmekte ve farklı formatlarda (örneğin, görüntü, ses, metin) olabilmektedir. Potansiyel veri kaynakları arasında sensörler, internet, veri tabanları ve manuel veri toplama yöntemleri bulunmaktadır [32].

Verilerin Ön İşlenmesi: Ham veriler, modelin eğitimi için uygun hale getirilebilmesi amacıyla çeşitli ön işleme adımlarından geçirilir. Bu adımlar arasında veri temizleme (eksik veya hatalı verilerin düzeltilmesi ya da çıkarılması), normalizasyon (veri değerlerinin belirli bir aralığa ölçeklenmesi), veri dönüştürme (veri formatlarının değiştirilmesi) ve veri artırma (veri setini genişletmek amacıyla yeni verilerin oluşturulması) yer almaktadır [33].

Özellik Seçimi: Bu adımda modelin performansını artırmak amacıyla, ham verilerden anlamlı ve modelin öğrenme sürecini kolaylaştıracak özellikler (features) seçilmektedir. Ham verilerden daha yüksek seviyeli temsil biçimlerinin oluşturulması için işlemler yapılır. Bu süreç, modelin verilerden daha iyi öğrenmesini sağlamakta ve genelleme yeteneğini artırmaktadır [34].

Eğitim/Test/Doğrulama Verisi Ayırımı: Veriler, modelin eğitimi, doğrulanması ve test edilmesi için farklı setlere ayrılmaktadır. Eğitim seti, modelin öğrenme sürecinde kullanılırken, doğrulama seti modelin hiperparametrelerini ayarlamak ve aşırı öğrenmeyi (overfitting) önlemek için kullanılmaktadır. Test seti ise modelin genel performansını değerlendirmek için ayrılır [35].

Modelin Eğitilmesi: Bu aşamada, seçilen özellikler kullanılarak model eğitilir. DL modelleri, eğitim verisi üzerinde ileri ve geri yayılım (forward and backward propagation) yöntemlerini kullanarak parametrelerini optimize eder. Bu süreç, modelin veri üzerindeki desenleri ve ilişkileri öğrenmesini sağlamaktadır [36].

Eğitilmiş Model: Eğitim sürecinin sonunda, model eğitim verisi üzerinde en iyi performansı gösterecek şekilde optimize edilir. Eğitilmiş model, yeni ve daha önce görmediği veriler üzerinde tahminler yapmak için kullanılır.

Sonuçların Değerlendirilmesi: Modelin performansı, test verisi kullanılarak değerlendirilir. Bu aşamada çeşitli performans ölçütleri (doğruluk, kesinlik, geri çağırma, F1 skoru, vb.) kullanılarak modelin tahmin yeteneği analiz edilir. Değerlendirme sonuçlarına göre, modelin başarısı ve kullanılabilirliği belirlenir ve gerekli görülmesi durumunda model iyileştirme adımları tekrarlanabilir [37].

Eğitim öncesinde uygun bir DL model mimarisine ihtiyaç vardır. Etkili bir model mimarisi, daha yüksek doğruluk ve hızda sınıflandırma yapabilmeyi sağlar. Günümüzde yaygın olarak kullanılan DL ağ türleri arasında Evrişimsel Sinir Ağları (CNN), Tekrarlayan Sinir Ağları (RNN) ve Üretici Çekişmeli Ağlar (GAN) bulunmaktadır [10].

Model mimarisi oluşturulduktan sonra eğitim ve değerlendirme için farklı hiperparametreler ayarlanır. Eğitim sırasında veriler ağın ilk katmanına yerleştirilir ve her nöron çıktının etikete eşit olup olmadığına göre geri yayılım yoluyla nöronun ağırlığını günceller. Bu süreç eğitim verilerinden öğrenme işlemi tamamlanana kadar devam eder. Modelin performansı doğruluk, kesinlik, geri çağırma ve F1 puanı kriterlerine göre değerlendirilir. Doğruluk, kesinlik ve geri çağırmanın değerleri yükseldikçe ağın sonuçları iyileşmektedir. Ancak Belirli bir

aralıkta F1 puanının değeri ne kadar küçük olursa eğitilen modelin genelleme performansı o kadar iyi olmaktadır. Eğitim ve değerlendirme işlemleri tamamlandıktan sonra model yeni verilerle test edilebilir [38].

DL modelinin yeteneği yeni verilerle hızlı bir şekilde uygulama ve daha önce görmediği verilere dayanarak doğru cevabı hızlı bir şekilde sağlama üzerinden test edilir. Eğitim süreci tamamlandıktan sonra ağlar, daha önce hiç görmedikleri verilerden bir sonuç çıkarmak için kullanılır. Ancak o zaman eğitilmiş DL modelleri gerçek ortamlarda uygulanabilir. Ayrıca eğitilen modelin sahada daha iyi performans gösterebilmesi için modelin genelleme yeteneğinin geliştirilmesi gerekmektedir ve genelleme yeteneğini geliştirmek için modelleri yeni etiketli veri kümeleri ile sürekli olarak güncellemek gerekmektedir [39].

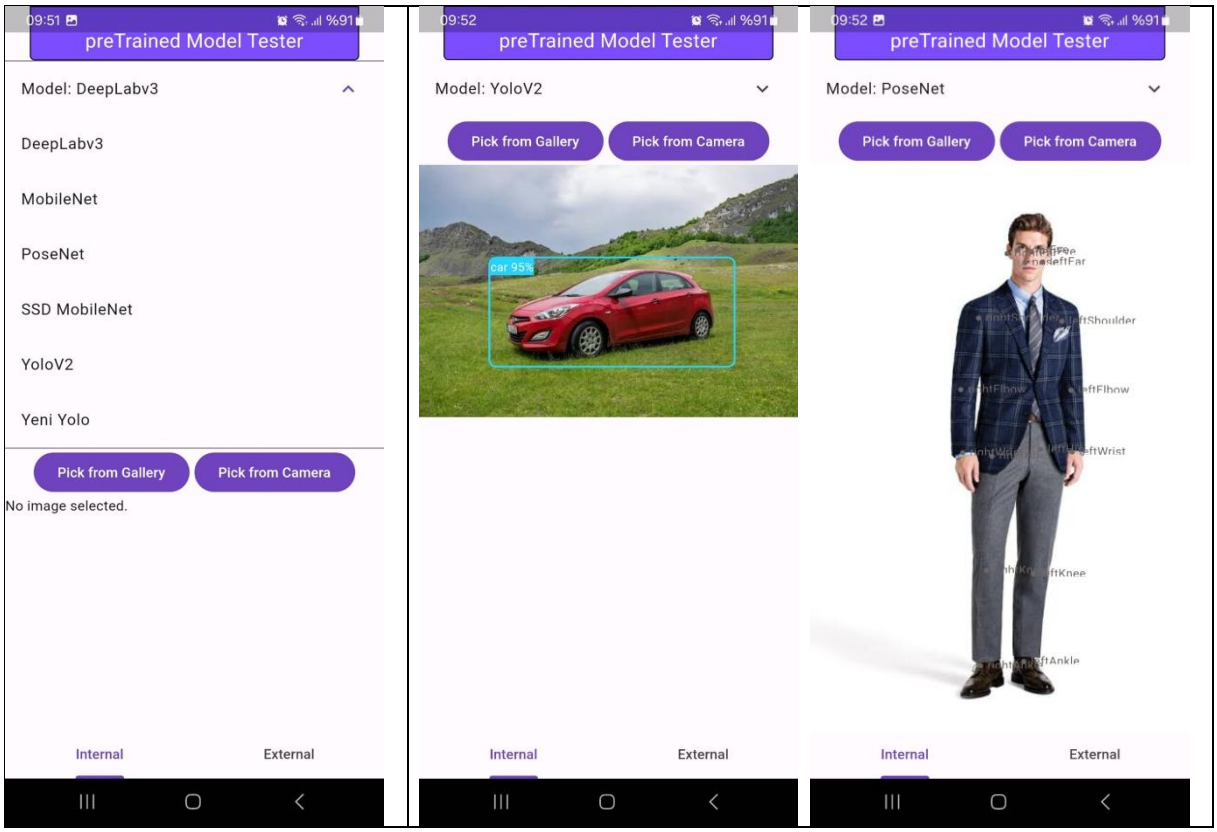
Gelişen yapay zekâ teknolojileriyle birlikte, eğitilmiş yapay zekâ modellerinin kullanımı giderek yaygınlaşmaktadır. Bu modeller, çeşitli alanlarda kullanılmak üzere özel veri kümeleriyle eğitilir ve belirli görevleri gerçekleştirmek için kullanılabilir hale gelir. Ancak, bu modellerin geniş kitlelere erişimini kolaylaştırmak için uygun platformlar gerekmektedir [40].

Yapay zekâ modellerinin geliştirilmesi ve kullanılması aşamasında çeşitli sorunlar ortaya çıkmaktadır. İlk olarak, erişilebilirlik ve demokratikleşme konusu önem arz etmektedir. Yapay zekâ teknolojileri genellikle sınırlı bir uzman kitlesi tarafından geliştirilmekte ve kullanılmaktadır. Ancak, bu teknolojilerin yaygınlaşması ve demokratikleşmesi için geniş bir kullanıcı kitlesinin bu modellere erişebilmesi gerekmektedir. İkinci olarak, topluluk katkısı ve paylaşım konusu dikkate alınmalıdır. Yapay zekâ modelleri genellikle büyük veri setleri üzerinde eğitilmekte olup, bu süreçler uzun zaman almaktadır. Birçok araştırmacı, geliştirici ve meraklı, kendi modellerini eğitmek için yeterli kaynağa veya yeteneğe sahip değildir. Üçüncü olarak, hız ve verimlilik meselesi öne çıkmaktadır. Yapay zekâ modellerinin geliştirilmesi ve dağıtılması zaman alıcı bir süreçtir. Son olarak, tarafsızlık konusu ele alınmalıdır. Eğitilmiş modeller genellikle belirli topluluklar tarafından test edilmekte ve bu durum veri taraflılığına yol açmaktadır [41]. Bu sorunların üstesinden gelmek için, yapay zekâ teknolojilerinin daha erişilebilir hale getirilmesi ve topluluk katkısını teşvik eden ortamların oluşturulması gerekmektedir. Bu ortamlar, modellerin daha objektif olmasını da sağlayacaktır [41].

Yapılan literatür incelemeleri, geliştirilen yapay zekâ uygulamalarının çoğunlukla tek bir modele özgü olduğunu ve bu modellerin geniş kullanıcı kitlelerine ulaşamadığını göstermektedir. Bu durum, derin öğrenme (DL) modellerinin test edilmesi ve yaygın kullanımı açısından önemli bir eksiklik yaratmaktadır. Ayrıca, eğitim aşamasından sonra bu modellerin ürünleştirilmesi sürecinde de zorluklarla karşılaşılmaktadır. Genellikle test ve tahmin işlemleri Python kodları üzerinden gerçekleştirilmekte, bu da kodlama bilgisi olmayan kullanıcılar için erişim ve kullanım zorlukları oluşturmaktadır. Bu eksiklikleri gidermek amacıyla, çalışmamızda yaygın olarak kullanılan bazı DL modellerinin test edilebilmesini sağlayan bir uygulama arayüzü geliştirilmiştir. Geliştirilen bu platform, eğitilmiş modelleri TensorFlow Lite (TFLite) formatına dönüştürerek, kullanıcıların herhangi bir kod bilgisine ihtiyaç duymadan bu modelleri test edebilmesine olanak tanımaktadır. Bu sayede, yapay zekâ modelleri daha geniş bir kullanıcı kitlesine ulaştırılabilecek ve bu kullanıcılar tarafından geri bildirimlerle daha da geliştirilebilecektir. Ayrıca, geliştirilen arayüz birçok farklı DL modeli ile test edilmiş ve sonuçlar karşılaştırmalı olarak sunulmuştur. Sonuç olarak, bu yenilikçi platform, yapay zekâ modellerinin erişimini artırarak hem akademik hem de endüstriyel alandaki projelerin yaygınlaşmasına ve geri bildirimlerle iyileştirilmesine katkı sağlamaktadır.

III. UYGULAMA

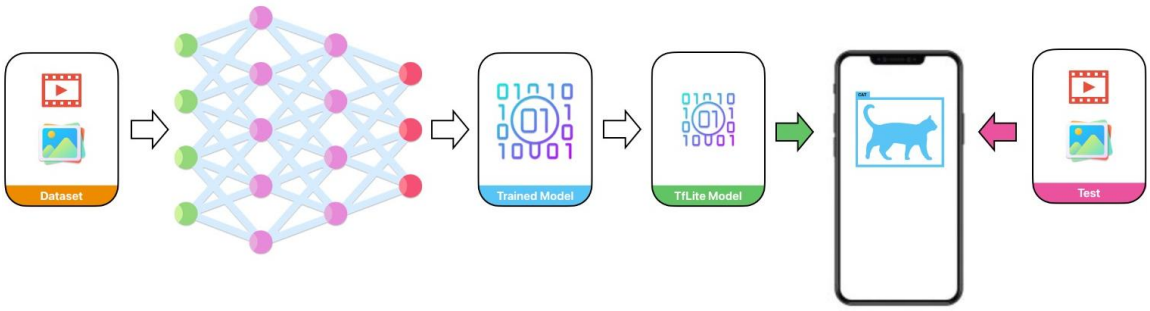
Bu çalışmada, AndroidStudio platformunda Flutter kullanılarak bir uygulama platformu geliştirilmiştir. Flutter, Google tarafından 2017 yılında ücretsiz olarak kullanıma sunulan açık kaynak kodlu bir mobil, web ve masaüstü uygulamaları geliştirme uygulamasıdır. Flutter, tek bir kod tabanı ile hem Android hem de iOS tabanlı uygulamalar geliştirmeye olanak tanımaktadır. Flutter ile yazılım geliştirmek için Dart programlama dili kullanılmaktadır [42]. Gerçekleştirilen yazılımın görselleri Şekil 10'da gösterilmiştir.



Şekil 10. Tasarlanan DL test yazılımının ekran görüntüleri

Geliştirilen uygulama, kullanıcıların çeşitli yapay zekâ ve makine öğrenimi modellerini test edebilecekleri bir platform olarak tasarlanmıştır. Uygulama, model yükleme, veri işleme ve sonuç görselleştirme gibi ana işlevlere sahiptir. Kullanıcılar, geliştirdikleri yapay zekâ modellerini uygulamaya entegre edebilir ve bu modelleri mobil cihazlarda test edebilirler. Bu platform, özellikle teknik bilgiye sahip olmayan kullanıcıların bile modelleri kolayca deneyimlemesine ve test etmesine olanak sağlamaktadır. Bu çalışma, sadece geliştiricilere değil, aynı zamanda yapay zekâ modellerini test etmek isteyen geniş bir kullanıcı kitlesine de hitap etmektedir. Uygulamanın temel hedefi, yapay zekâ ve makine öğrenimi modellerinin daha kolay test edilebilir ve erişilebilir hale getirilmesini sağlamaktır.

A.Eğitilmiş Derin Öğrenme Modeli Test İşlemleri



Şekil 11. Tasarlanan yazılımın çalışma akış şeması

Şekil 11'de tasarlanan yazılımın test edilmesi için gerekli işlemler sıralanmıştır. Bu işlemler sırasıyla literatürde sıklıkla kullanılan DL modelleri üzerinde, yine literatürde kullanılan ve test sonuçları sabit veri setleri ile test edilmiştir. Test işlemlerinin gerçekleştirilme aşamaları bu bölümde anlatılmıştır:

Veri Setinin Hazırlanması: Çalışmanın ilk aşamasında, kullanılacak veri setinin temizlenmesi, etiketlenmesi ve model eğitimi için uygun bir formata dönüştürülmesi gerekmektedir. Bu süreçte, veri setindeki eksik veya hatalı veriler tespit edilip ayıklanmalı, her bir veri noktası için doğru sınıf etiketleri belirlenmelidir. Böylece modelin eğitimi için kullanılacak veri kümesi, tutarlı ve uygun hale getirilmiş olacaktır.

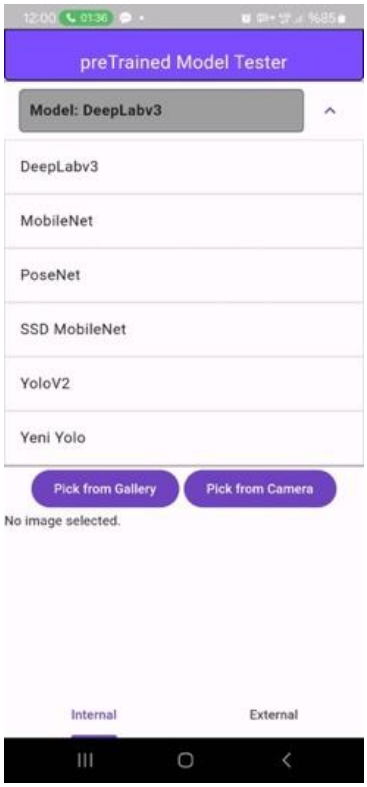
Model Eğitimi: Modelin eğitimi aşamasında YOLOv2, MobileNet, DeepLabv3, PoseNet, SSD MobileNet mimarisi tercih edilmiştir. Eğitim süreci, DarkNET framework'ü kullanılarak yürütülmüştür. Modelin

parametreleri, doğruluk ve kayıp (loss) fonksiyonları dikkate alınarak optimize edilmiştir. Hiperparametrelerin ayarlanması da bu süreçte gerçekleştirilmiş olup, eğitimin sonlandırılması model doğruluk değerinin stabil bir seviyeye ulaştığı noktada yapılmıştır.

Test Aşaması: Eğitim aşaması tamamlanan model, ayrılan test veri seti üzerinde değerlendirilmiştir. Modelin performansı, ortalama hassasiyet (mean Average Precision- mAP) gibi performans metrikleri ile ölçülmüştür. Elde edilen sonuçlar, genel doğruluk oranı, yanlış pozitif ve yanlış negatif oranları gibi performans göstergeleriyle birlikte detaylı bir şekilde raporlanmıştır.

Sonuçların Yorumlanması: Test sonuçları, modelin sınıflandırma kapasitesini ve hatalı tahminlerini analiz etmek için incelenmiştir. Bu aşamada, YOLOv2, MobileNet, DeepLabv3, PoseNet, SSD MobileNet modelinin hız ve doğruluk açısından sağladığı avantajlar ile sınırlılıkları değerlendirilmiştir. Modelin hem doğruluk hem de işlem hızı bakımından ne ölçüde etkili olduğu detaylı bir şekilde analiz edilmiştir.

Sonuçların Sunumu ve Karşılaştırma: Elde edilen bulgular, literatürde yer alan benzer yöntemlerle karşılaştırılarak sunulmuştur. YOLOv2, MobileNet, DeepLabv3, PoseNet, SSD MobileNet 'nin performansı, önceki modellerle karşılaştırıldığında hız ve doğruluk oranları açısından değerlendirilmiş, bu sayede modelin avantajları ve geliştirilmesi gereken yönleri ortaya konmuştur. Bu işlem adımları, platformun veri hazırlama, model eğitimi ve test aşamalarındaki işlevselliğini ve performans değerlendirmesini bütüncül bir şekilde ele almaktadır (Şekil 12).

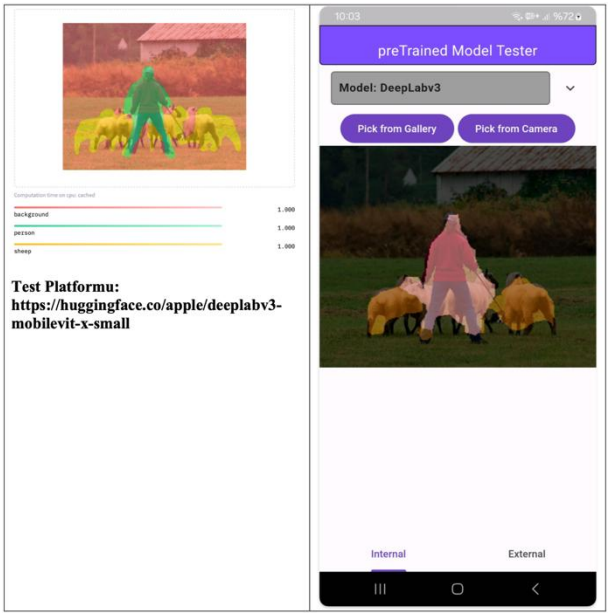


Şekil 12. Tasarlanan yazılımın test ekranı

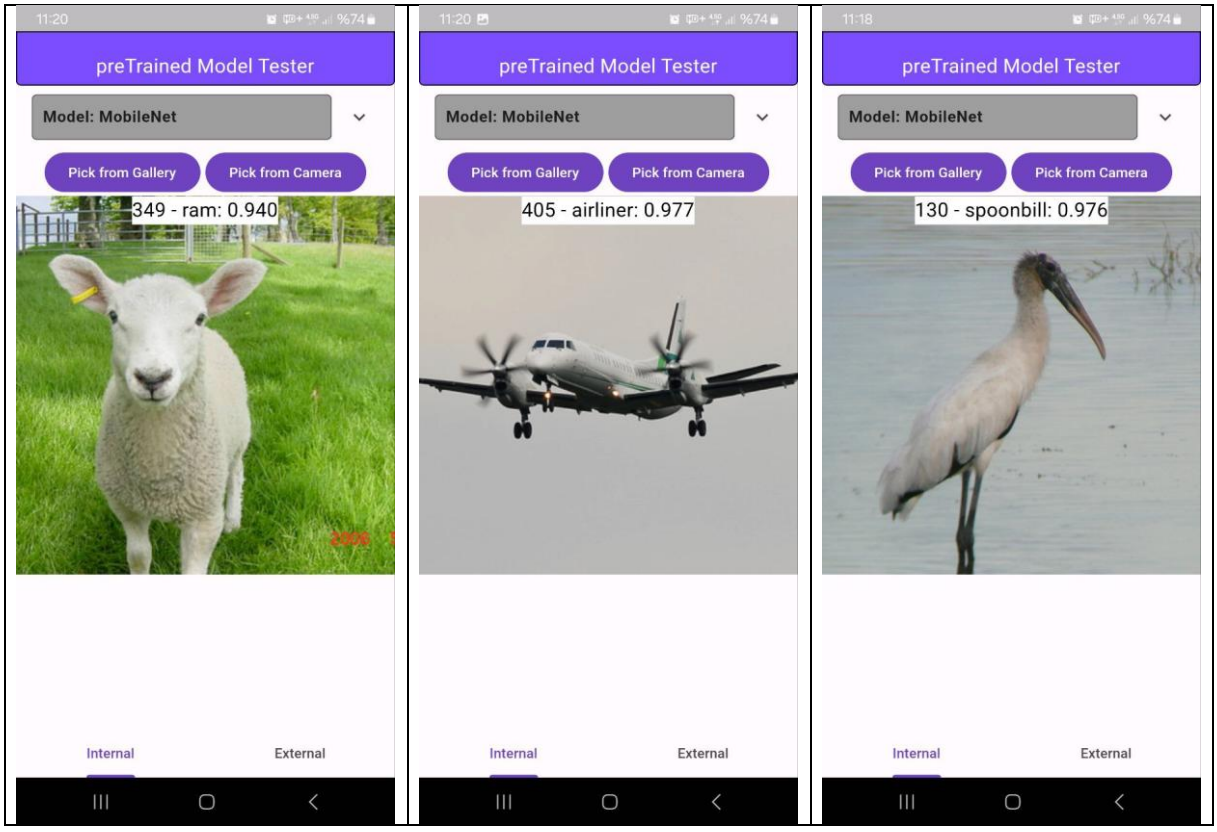
DL modelini oluşturmak için açık kaynak kodlu kütüphaneler kullanılmaktadır. Tensorflow en çok kullanılan python tabanlı kütüphanelerden biri olarak gösterilmektedir [43]. DL modelinin eğitim süreci belirli aşamalar çerçevesinde gerçekleştirilmektedir. İlk aşamada, Python ortamında TensorFlow gibi açık kaynaklı kütüphaneler kurulmakta ve gerekli bağımlılıklar sağlanmaktadır. Daha sonra, modelin eğitimi için uygun bir veri seti seçilmekte ve yüklenmektedir. Örnek olarak rakam tanıma için MNIST, kıyafet tanıma için FASHION-MNIST ya da nesne tanıma için YOLO veri setlerinden biri seçilebilir. Bu veri setleri eğitim ve test verileri olmak üzere iki alt kümeye ayrılmaktadır. Modelin kurulumu aşamasında Keras Sequential yapısı kullanılarak yapay sinir ağı mimarisi tanımlanmakta; giriş ve çıkış boyutları belirlenmektedir. Aktivasyon fonksiyonu ve kayıp fonksiyonu seçilerek model derlenir. Ardından, model.fit yöntemi aracılığıyla eğitim verisi üzerinde parametre optimizasyonu gerçekleştirilir. Son aşamada ise model.evaluate yöntemi ile test verisi kullanılarak modelin doğruluk ve kayıp değerleri ölçülerek performans değerlendirmesi yapılır. Bu adımlar yazılım bilgisi, zaman ve teknolojik altyapı gerektirmektedir. Bu süreç sonucunda ortaya çıkan eğitilmiş model ise ürüne dönüştürülememektedir. Bu problemlere çözüm amacıyla çalışmamızda eğitilmiş modellerin ürüne dönüştürülebileceği ve herkesin kullanımına sunulabileceği bir uygulama geliştirilmiştir.

IV. SONUÇLAR

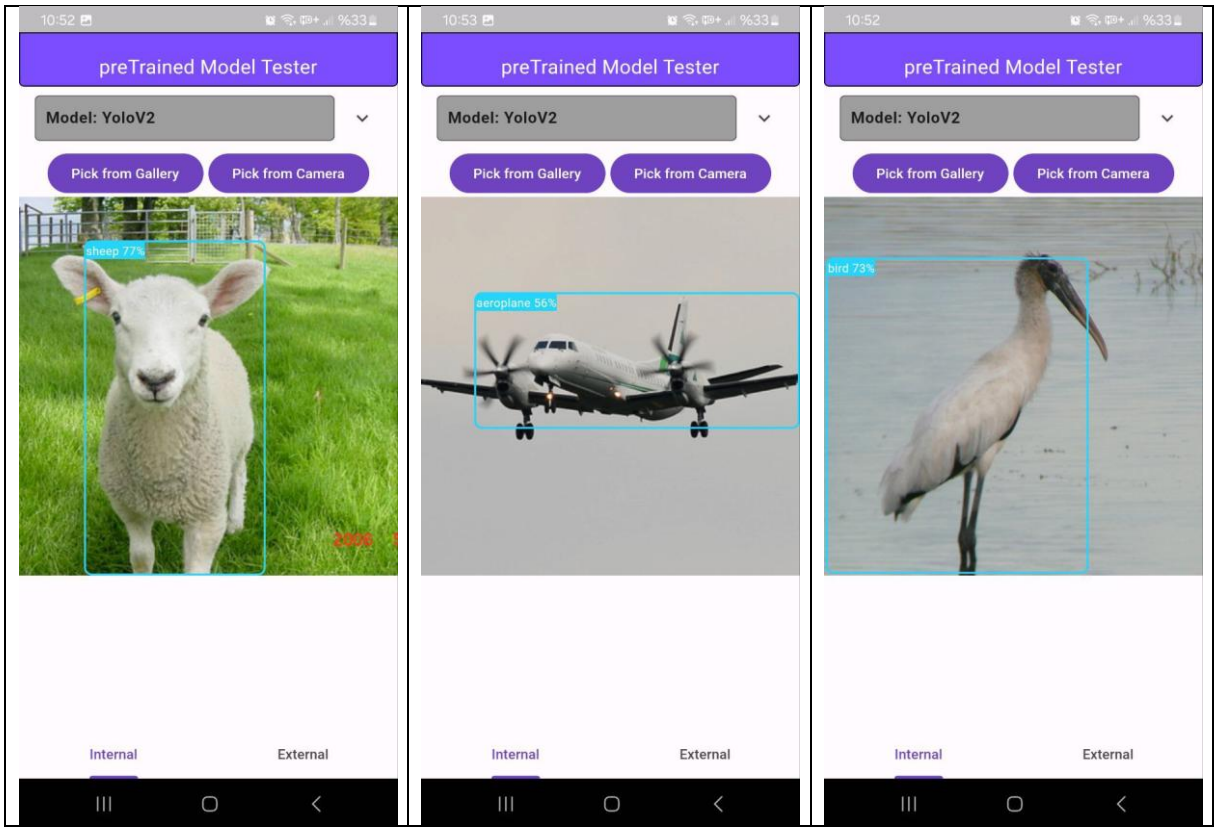
Bu çalışmada geliştirilen mobil uygulamanın performansı, önceden eğitilmiş üç farklı derin öğrenme modeli (DeepLabV3, MobileNet, YOLOv2) üzerinden test edilmiştir. Her bir model üç farklı görsel üzerinde hem uygulamamızda hem de HuggingFace/yerel test ortamlarında çalıştırılmış, sonuçlar karşılaştırmalı olarak değerlendirilmiştir. Test süreci görseller üzerinden Şekil 13–15'te, özet veriler ise Tablo 1'de sunulmuştur.



Şekil 13. DeepLabV3 test ekranı



Şekil 14. MobileNet test ekranı



Şekil 15. Yolov2 test ekranı

DeepLabV3 modeli için segmentasyon çıktıları mobil uygulamada başarılı bir şekilde yeniden üretilmiş ve referans sistemle uyum göstermiştir. MobileNet modelinde üç farklı görsel için elde edilen sonuçlar mobil ve web ortamında tamamen aynı doğruluk oranlarını (%94, %97,7 ve %97,6) üretmiştir. Bu durum, mobil uygulamanın sınıflandırma görevlerinde masaüstü sistemlerle eşdeğer performans gösterdiğini ortaya koymaktadır.

YOLOv2 modelinde ise kısmi farklılıklar gözlenmiştir. Örneğin, koyun görselinde mobil uygulama %77 doğruluk sağlarken yerel test %81 doğruluk vermiştir (fark %4). Kuş görselinde bu fark %16'ya, uçak görselinde ise %35'e kadar çıkmıştır. Bu farkların, kullanılan model dosya formatlarındaki (mobilde TFLite, yerelde H5) çevrim farklılıklarından ve mobil cihazların donanım sınırlamalarından kaynaklandığı düşünülmektedir.

Tablo 1. Test Sonuçları

	Geliştirilen Mobil Uygulama	**Yerel Bilgisayar / Hugging Face**
MobileNet (Koyun görseli)	%94	%94
MobileNet (Uçak görseli)	%97,7	%97,7
MobileNet (Kuş görseli)	%97,6	%97,6
YoloV2 (Koyun görseli)	%77	%81
YoloV2 (Uçak görseli)	%56	%91
YoloV2 (Kuş görseli)	%73	%89

Genel olarak bakıldığında, test edilen modellerin sonuçları mobil uygulama ile referans sistemler arasında büyük ölçüde uyumludur. Mobil uygulama, doğruluk oranlarını kabul edilebilir sapmalarla yeniden üretmiş ve gerçek kullanıcılar için güvenilir test imkânı sunmuştur. Elde edilen bulgular, geliştirilen platformun yalnızca bir prototip olmadığını, aynı zamanda derin öğrenme modellerinin mobil cihazlarda test edilmesine yönelik pratik, erişilebilir ve demokratik bir çözüm sunduğunu göstermektedir.

V. TARTIŞMA

Bu çalışmada, derin öğrenme (DL) modellerinin geniş bir kullanıcı kitlesi tarafından erişilebilir ve test edilebilir olması amacıyla ortak bir mobil platform geliştirilmiştir. Literatürdeki incelemeler, genellikle belirli bir modele özgü veya dar bir amaca yönelik uygulamaların geliştirilmiş olduğunu ve bu uygulamaların çoğunlukla yaygın bir kullanıcı kitlesine ulaşamadığını göstermektedir. Bu durum, DL modellerinin farklı alanlarda test edilmesi ve gerçek dünya uygulamalarında benimsenmesi için önemli bir eksiklik yaratmaktadır.

Geliştirilen platform, Flutter çerçevesi kullanılarak mobil cihazlar için oluşturulmuş ve kullanıcıların herhangi bir yazılım bilgisi olmadan DL modellerini yükleyip test edebilmelerine olanak tanımıştır. Bu özellik, akademisyenler, araştırmacılar ve geliştiriciler tarafından hazırlanan modellerin son kullanıcılar tarafından deneyimlenmesi ve geri bildirim sağlanabilmesi açısından önemli bir avantaj sunmaktadır. Platformun kullanıcı dostu arayüzü ve çapraz platform desteği (Android ve iOS) sayesinde geniş bir erişim imkânı sunulmaktadır.

Önceki çalışmalarda, DL modellerinin test edilmesi için geliştirilen mobil uygulamaların genellikle tek bir modele veya belirli bir probleme odaklandığı görülmektedir. Örneğin, bazı uygulamalar tarımsal hastalıkların tespiti veya spesifik hastalıkların tanısı için belirli DL modellerini kullanmıştır. Bu uygulamalar, genellikle tek bir bitki hastalığı, cilt kanseri veya göz rahatsızlığı gibi sınırlı alanlarda optimize edilmiştir ve bu nedenle geniş bir kullanıcı kitlesine hitap edememektedir. Oysaki bu çalışmada geliştirilen platform, farklı DL modellerini (örneğin, DeepLabV3, MobileNet, YOLOv2) destekleyerek kullanıcıların farklı görevler için modelleri test etmesine olanak tanımaktadır. Ayrıca, platformun TFLite formatında modelleri kabul etmesi, çeşitli DL modellerinin hızlıca ve kolayca yüklenebilmesine olanak tanır.

Literatürde, Cabbar ve arkadaşları (2020) tarafından geliştirilen sürücü uykululuk tespiti uygulaması gibi benzer çalışmaların genellikle belirli bir amaca yönelik ve tek bir modelle sınırlı kaldığı görülmektedir. Awasthi ve arkadaşları (2021) da COVID-19 tespiti için hafif bir DL modeli geliştirmiş, ancak bu modelin genel kullanıma sunulması veya farklı modellerin entegrasyonuna uygun bir platform sağlamamıştır. Bu çalışmadaki platform ise DL modellerinin demokratikleşmesini ve daha geniş bir kullanıcı kitlesi tarafından erişilebilir hale getirilmesini sağlamaktadır. Bu sayede hem akademik hem de endüstriyel araştırmacılar ve geliştiriciler modellerini son kullanıcılara ulaştırabilir ve pratik geri bildirimler alabilirler. Platform, DL modellerinin gerçek dünya senaryolarında test edilmesi ve iyileştirilmesi sürecine önemli bir katkı sağlamaktadır.

Platformun en belirgin avantajı, farklı DL modellerinin tek bir mobil uygulama üzerinden test edilebilmesidir. Bu, kullanıcı deneyimini iyileştirirken DL modellerinin yaygınlaşmasına katkıda bulunmaktadır. Flutter kullanılarak geliştirilmesi sayesinde, uygulama hem Android hem de iOS cihazlarında geniş bir kullanıcı kitlesine ulaşabilmektedir.

Bununla birlikte, platformun bazı sınırlamaları da vardır. Mobil cihazların donanım kısıtlamaları nedeniyle, karmaşık ve büyük modellerin gerçek zamanlı olarak çalıştırılması zor olabilmektedir. Bu sorunun üstesinden gelebilmek için optimize edilmiş hafif model sürümlerinin kullanılması gerekmektedir. Ayrıca, mevcut durumda platform yalnızca TFLite formatındaki modelleri desteklemektedir; diğer framework'lerde (örneğin, PyTorch, ONNX) eğitilmiş modellerin entegrasyonu için ek geliştirmeler yapılması gerekmektedir.

Bu çalışma, DL modellerinin test edilmesi ve yaygınlaştırılması açısından önemli bir adım atmaktadır. Literatürdeki benzer çalışmalardan farklı olarak, geliştirilen platform genelleştirilmiş bir çözüm sunarak farklı alanlardaki DL modellerinin geniş bir kullanıcı kitlesi tarafından test edilmesini ve geri bildirimlerle geliştirilmesini olanaklı kılmaktadır. Geliştirilen uygulamanın eğitim ve test süreçleri, literatürdeki benzer çalışmalara kıyasla daha esnek ve genelleştirilebilir bir yapı sergilemektedir. Örneğin Loyani ve Machuve (2021) çalışmasında yalnızca Tuta Absoluta zararlısının tespiti için U-Net tabanlı bir model eğitilmiş ve mobil uygulamaya entegre edilmiştir. Ahmed ve Reddy (2021) ise 96.206 görüntüden oluşan geniş bir veri seti üzerinde CNN modeli eğitmiş, ancak bulut tabanlı sistemlere bağımlı kalmıştır. Kimeu ve arkadaşları (2024) ise pnömoni tespiti için EfficientNet mimarisini optimize etmiş, fakat yalnızca sağlık alanına odaklanmıştır. Buna karşılık, bu çalışmada geliştirilen platform; YOLOv2, MobileNet, DeepLabV3 gibi farklı derin öğrenme mimarilerini aynı arayüzde test edebilme özelliğiyle öne çıkmakta, TFLite formatına dönüştürülen modellerin herhangi bir yazılım bilgisine gerek duyulmadan denenmesine olanak tanımaktadır [17-19]. Bu yönüyle uygulama, literatürdeki dar kapsamlı örneklerden farklı olarak, derin öğrenme modellerinin daha geniş kullanıcı grupları tarafından erişilebilirliğini ve gerçek yaşamda test edilebilirliğini artırmaktadır.

Gelecekte, platformun daha fazla model ve formatı desteklemesi hedeflenmektedir. Ayrıca, performans optimizasyonları ve bulut tabanlı işlem yeteneklerinin entegrasyonu platformun verimliliğini artırabilir. Kullanıcı geri bildirimlerine dayalı arayüz ve kullanım kolaylığının iyileştirilmesi, platformun benimsenme oranını artıracaktır. Bu geliştirmelerle DL modellerinin gerçek dünya uygulamalarında test edilmesi ve kullanıma sunulması sürecine katkı sağlanacaktır.

KAYNAKLAR

[1] Bengio, Y., Lecun, Y., & Hinton, G. (2021). Deep learning for AI. Communications of the ACM, 64(7), 58–65.

[2] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M. L., Chen, S. C., & Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys (CSUR), 51(5), 1–36.

[3] Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N., & Fox, E. A. (2020). Natural language processing advancements by deep learning: A survey. arXiv preprint, arXiv:2003.01200.
[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.
[5] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.
[6] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139–144.
[7] Ioffe, S. (2017). Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. Advances in Neural Information Processing Systems, 30.
[8] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? Advances in Neural Information Processing Systems, 27.
[9] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115, 211–252.
[10] Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. Computer Science Review, 40, 100379.
[11] Karar, M. E., Abd Elaziz, M., Abdelwahab, A., & Elsheikh, A. H. (2021). A new mobile application of agricultural pests recognition using deep learning in cloud computing system. Alexandria Engineering Journal, 60(5), 4423–4432.
[12] Ngugi, L. C., Abdelwahab, M., & Abo-Zahhad, M. (2020). Tomato leaf segmentation algorithms for mobile phone applications using deep learning. Computers and Electronics in Agriculture, 178, 105788.
[13] Sahin, V. H., Oztel, I., & Yolcu Oztel, G. (2022). Human monkeypox classification from skin lesion images with deep pre-trained network using mobile application. Journal of Medical Systems, 46(11), 79.
[14] Gencturk, B., Cinar, A., Taspinar, Y., & Sahin, C. (2024). Detection of hazelnut varieties and development of mobile application with CNN data fusion feature reduction-based models. European Food Research and Technology, 250(1), 97–110.
[15] Awasthi, N., Paul, A., Kumar, S., Kumar, M., Kaur, T., & Sethi, N. (2021). Mini-COVIDNet: Efficient lightweight deep neural network for ultrasound-based point-of-care detection of COVID-19. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 68(6), 2023–2037.
[16] Chen, Y., Li, C., & Cheng, L. (2020). Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. ACM Computing Surveys (CSUR), 53(4), 1–37.
[17] Loyani, L., & Machuve, D. (2021). A deep learning-based mobile application for segmenting Tuta absoluta's damage on tomato plants. Engineering, Technology & Applied Science Research, 11(5), 7730–7737.
[18] Ahmed, R. T., & Reddy, P. V. G. D. (2021). Development of a deep learning model for plant disease detection on mobile devices. AgriEngineering, 3(1), 32–45.
[19] Kimeu, J. M., Kisangiri, M., Mbelwa, H., & Leo, J. (2024). Deep learning-based mobile application for the enhancement of pneumonia medical imaging analysis: A case-study of West-Meru Hospital. Informatics in Medicine Unlocked, 50, 101582.
[20] Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. Electronic Markets, 31(3), 685–695.
[21] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press, Cambridge, MA.
[22] Hebb, D. O. (2005). The Organization of Behavior: A Neuropsychological Theory. Psychology Press, New York, NY.
[23] Crevier, D. (1993). AI: The Tumultuous History of the Search for Artificial Intelligence. Basic Books, New York, NY.
[24] McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the Dartmouth summer research project on artificial intelligence, August 31, 1955. AI Magazine, 27(4), 12–12.
[25] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.
[26] Shrestha, A., & Mahmood, A. (2019). Review of deep learning algorithms and architectures. IEEE Access, 7, 53040–53065.
[27] Chen, L., Zhang, Y., & Li, H. (2021). Review of image classification algorithms based on convolutional neural networks. Remote Sensing, 13(22), 4712.
[28] Padilla, R., Netto, S. L., & Da Silva, E. A. (2020). A survey on performance metrics for object-detection algorithms. International Conference on Systems, Signals and Image Processing (IWSSIP), IEEE.
[29] Mo, Y., Lin, Y., & Zhou, J. (2022). Review of the state-of-the-art technologies of semantic segmentation based on deep learning. Neurocomputing, 493, 626–646.
[30] Ustundag, M. T., Gunes, E., & Bahçivan, E. (2017). Turkish adaptation of digital literacy scale and investigating pre-service science teachers' digital literacy. Journal of Education and Future, 12, 1–17.

[31] Wang, J., Li, Z., & Liu, W. (2021). Deep 3D human pose estimation: A review. Computer Vision and Image Understanding, 210, 103225.
[32] Janssen, M., Brous, P., Estevez, E., Barbosa, L. S., & Janowski, T. (2020). Data governance: Organizing data for trustworthy artificial intelligence. Government Information Quarterly, 37(3), 101493.
[33] García, S., Luengo, J., & Herrera, F. (2015). Data Preprocessing in Data Mining. Springer, Cham.
[34] Cateni, S., Colla, V., & Vannucci, M. (2012). Variable selection and feature extraction through artificial intelligence techniques. Multivariate Analysis in Management, Engineering and the Science, 6, 103–118.
[35] Kärkkäinen, T. (2014). On cross-validation for MLP model evaluation. Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+SSPR 2014, Joensuu, Finland, August 20–22, 2014, Proceedings. Springer.
[36] Mathew, A., Amudha, P., & Sivakumari, S. (2021). Deep learning techniques: An overview. Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020. Springer, 599–608.
[37] Khalid, S., Khalil, T., & Nasreen, S. (2020). Evaluation of deep learning models for identifying surgical actions and measuring performance. JAMA Network Open, 3(3), e201664.
[38] Lu, J., Tan, L., & Jiang, H. (2021). Review on convolutional neural network (CNN) applied to plant leaf disease classification. Agriculture, 11(8), 707.
[39] Sameen, M. I., Pradhan, B., & Lee, S. (2020). Application of convolutional neural networks featuring Bayesian optimization for landslide susceptibility assessment. Catena, 186, 104249.
[40] Gunda, N. S. K., Gautam, S. H., & Mitra, S. K. (2019). Artificial intelligence based mobile application for water quality monitoring. Journal of The Electrochemical Society, 166(9), B3031.
[41] Sánchez-Morales, L. N., López-Juárez, I., Muñoz-Arteaga, J., & Estrada, M. (2020). Generating educational mobile applications using UIDPs identified by artificial intelligence techniques. Computer Standards & Interfaces, 70, 103407.
[42] Tashildar, A., Pandey, P., & Deshmukh, V. (2020). Application development using Flutter. International Research Journal of Modernization in Engineering Technology and Science, 2(8), 1262–1266.
[43] Saabith, A. S., & Vinothra, T. (2020). Flutter-based mobile applications: A review. International Journal of Advanced Computer Science and Applications, 11(9).