
GAZİ ÜNİVERSİTESİ
Gazi
1926

ANDROID KÖTÜCÜL YAZILIM ANALİZİNDE DERİN ÖĞRENME MODELLERİNİN PERFORMANSININ KARŞILAŞTIRILMASI

Taylan KURAL

YÜKSEK LİSANS
BİLGİ GÜVENLİĞİ MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2022

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Taylan KURAL

01/07/2022

ANDROID KÖTÜCÜL YAZILIM ANALİZİNDE DERİN ÖĞRENME MODELLERİNİN PERFORMANSININ KARŞILAŞTIRILMASI

(Yüksek Lisans Tezi)

Taylan KURAL

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Temmuz 2022

ÖZET

Android işletim sistemi, açık kaynak olan yapısı, geniş uygulama marketiyle telefonlarda, televizyonlarda, saatlerde, arabalarda ve diğer nesnelerin interneti uygulamalarında yaygın olarak kullanılmaktadır. Yaygın kullanım ve açık kaynak yapısı, kötücül niyet barındıran saldırganlar için işletim sistemini ve sahip olduğu cihazları kolay ve kazançlı hedefler haline getirmektedir. Saldırganlar tarafından sıklıkla tercih edilen yöntem, kötücül yazılım uygulamalarının, kullanıcı cihazlarına yüklenmesidir. Bu yazılımların sayıları gün geçtikçe artmakta, kötücül yazılımların tespitinde geleneksel yöntemler yetersiz kalabilmektedir. Kötücül yazılım tespitinde makine öğrenmesi ve derin öğrenme tabanlı yöntemler umut veren sonuçlar elde etmişlerdir. Özellikle derin öğrenme tabanlı yöntemler, alan uzmanlık bilgisi gereksiniminin azlığı ve kendi kendine özellik çıkarabilen yapıları sayesinde, kötücül yazılım tespitinde artan bir kullanıma sahiptirler. Bu tez çalışmasında, kötücül yazılımların görsel imajlara dönüştürülerek bu imajlar üzerinde evrişimsel sinir ağları (ESA) tabanlı derin öğrenme modelleriyle görsel kötücül yazılım analizleri gerçekleştirilmektedir. Çalışmada, popüler ESA modelleri olan Xception, ResNet, VGG, Inception, MobileNet, DenseNet, NasNet, EfficientNet, sunulan toplu ince ayar öğrenim aktarma yöntemiyle eğitilmiş ve elde edilen sonuçlara göre modeller doğruluk, kesinlik, geri çağırma, hassaslık, F1 skoru metriklerine göre karşılaştırılmaktadır. Eğitilen modellerden performanslarına göre birleşimler oluşturulmakta ve birleşimlerin performansları karşılaştırılmaktadır.

Bilim Kodu : 92403
Anahtar Kelimeler : Android, Derin Öğrenme, Kötücül Yazılım Analizi, Görsel Analiz
Sayfa Adedi : 166
Danışman : Doç. Dr. Murat DENER
İkinci Danışman : Prof. Dr. Yusuf SÖNMEZ

BENCHMARKING OF DEEP LEARNING MODELS FOR ANDROID MALWARE ANALYSIS

(M. Sc. Thesis)

Taylan KURAL

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

July 2022

ABSTRACT

Android operating system has been widely used in mobile phones, televisions, smart watches, cars and other Internet of Things applications with its open source structure and wide application market. This widespread use and open-source nature make this operating system and its devices easy and lucrative targets for cyber attackers. One of the most used methods often preferred by attackers is to install malware applications on user devices. As the number of malware programs is increasing, the traditional methods can be insufficient in detecting. Machine learning-based and deep learning-based methods have achieved promising results in malware detection and classification. Deep learning-based methods have an increasing use in malware detection, thanks to the low need for domain expertise and their feature extracting capabilities. Convolutional neural networks (CNN) are popular deep learning methods that are widely used in visual analysis of malware by transforming them to images. In this thesis work, a batch fine-tune transfer learning method was proposed and used on popular CNN models, Xception, ResNet, VGG, Inception, MobileNet, DenseNet, NasNet, EfficientNet. According to the results, the models were analyzed and compared with metrics like accuracy, specificity, recall, precision, F1-score. Ensembles were created from models according to their performances and the performance of ensembles were compared.

Science Code : 92403
Key Words : Android, Deep Learning, Malware Analysis, Visual Analysis
Page Number : 166
Supervisor : Assoc. Prof. Dr. Murat DENER
Co-Supervisor : Prof. Dr. Yusuf SÖNMEZ

TEŞEKKÜR

Bu tezi hazırlamamda bana destek olan, danışmanlarım Doç. Dr. Murat DENER ve Prof. Dr. Yusuf SÖNMEZ hocalarıma, akademik hayatımda bana destek olarak daha ileriye gitmeme motivasyon sağlayan değerli hocam Prof. Dr. Halil İbrahim BÜLBÜL'e, hayatımda önemli etkisi olan, bilimsel ve kritik düşünme konusunda bana yol gösteren Nejat ALPAR'a ve çalışmalarım sırasında bana desteklerini esirgemeyen, çok sevgili aileme, eğitim hayatım boyunca katkı sağlayan tüm öğretmenlerime saygılarımı sunar, teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
ms	milisaniye
sn	saniye

Kısaltmalar	**Açıklamalar**
API	Uygulama programlama arayüzü
APK	Android uygulama dosyası
DEX	Davik çalıştırılabilir
DSA	Derin sinir ağı
DVM	Destek vektör makinesi
ESA	Evrişimsel sinir ağı
GB	Milyar bayt
KA	Karar ağacı
LSTM	Uzun kısa dönem hafıza
PIL	Python görüntüleme kütüphanesi
RO	Rastgele orman
SH	Sıkıştırma ve heyecanlandırma
TSA	Tekrarlayan sinir ağı

1. GİRİŞ

Akıllı telefonların çıkışıyla beraber dünya genelinde değişimler yaşanmaktadır. Android, akıllı telefonlarda en çok kullanılan işletim sistemi olmaktadır. Android mobil işletim sistemi ilk kez 2008 yılında, HTC Dream modeliyle kullanıcılara sunulmuştur. Çıkışını yaptığı 2008 yılından, günümüze sürekli gelişmekte ve değişmektedir. Akıllı telefonlar, önceki nesil telefonlara göre daha hızlı ve kolay şekilde yeni yazılımlar yüklenerek geliştirilebilmektedir. Mobil telefonları hedef alan, başta iletişim olmak üzere yeni uygulamalar geliştirilmektedir. Geliştirilen uygulamaların milyar dolarlık ekonomiler yaratması hem kişisel hem kurumsal kullanıcı düzeyinde akıllı telefonları günlük yaşamın ayrılmaz parçası haline getirmektedir. Kullanıcılar artık telefonlarına daha bağımlı kalmakta ve işlerini akıllı telefonları sayesinde yapmaktadırlar. Tüm bu gelişmeler, yeni kurulan girişimlerin ve eski şirketlerin mobil uygulama marketlerinde yer almak istemelerine, bu sayede kullanıcılarıyla sürekli bağlantı halinde olmak istemelerine ve müşterilerine hizmet sunacakları dijital temsilciliklerini, web öncelikli anlayıştan, mobil öncelikli anlayışa doğru değiştirmelerine neden olmaktadır. Android işletim sistemi, akıllı telefonlarla başladığı yolculuğunu, arabalar, televizyonlar, saatler, tablet bilgisayarlar ve diğer nesnelerin interneti uygulamalarıyla sürdürmekte ve günümüzde Android ekosistemi olarak anılmaktadır.

Bu değişimin, siber güvenlik, bilgi güvenliği açısından takip edilmesi ve değerlendirilmesi gerekmektedir. Özel ve kamu kurumlarının artan mobil hizmetleri, kullanıcı hayatlarını kolaylaştırırken, büyük miktarda kişisel, hassas veri oluşturmaktadır. Akıllı telefonların ilk çıktığı yıllardaki bankacılık uygulamaları, günümüze kıyasla sayıca az kalmaktadır. Yıllar içerisinde bankacılık uygulamaları artmakta ve günümüzde mobil bankacılık kavramı kullanılmaktadır. Bankalar hizmetlerini mobile taşımakta ve özellikle 2019 yılında başlayan ve süregelen kovid-19 pandemisiyle birlikte fiziksel şubelerden hizmet almak ve vermek daha az tercih edilir hale gelmektedir. Yeni teknolojilerin çıkması blok zincir, kripto para gibi yeni kavramları insanların hayatlarına sokmakta ve gündelik yaşamda kullanılır kılmaktadır. Mobil cüzdanlar bu yeni metaların saklandığı yerler olmaktadır. Benzer şekilde uygulama içi satın alma, mobil ödeme, yakın alan temassız ödeme gibi hizmetlerde kullanıcı hayatlarında yer almaktadır. Bu gelişmeler, mobil cihazları ekonomik faaliyetlerin merkezi olmasına doğru götürmektedir. Cihazların sahip olduğu iletişim

kapasiteleriyle birlikte kimi kullanıcılar için akıllı telefonlar hayatlarının merkezi konumuna gelebilmektedir. Bu durum siber saldırganlar için, mobil ortamları kazançlı ve cazip birer hedef haline getirmektedir.

Siber saldırganlarla, siber güvenlik uzmanlarının aralarında sürdürdüğü mücadele sonucunda, saldırı yöntemleri oldukça gelişmekte ve bunu da yeni nesil güvenlik yaklaşımları izlemektedir. Tehdit sayısının hızlı artışı, sıfırıncı gün olarak adlandırılan keşfedilmemiş güvenlik açıklarından ortaya çıkan saldırıların varlığı ve tehditlerin mevcut analiz yöntemlerini aşan yaklaşımları, araştırmacıları ve uzmanları yeni arayışlara, çalışmalara yönlendirmektedir. Derin öğrenme alanında yaşanan gelişmeler ve bu gelişmelerin siber güvenlik alanında da uygulanması oldukça umut verici sonuçlar sunmaktadır. Tez çalışmasında, derin öğrenme alanında sıklıkla kullanılan modeller, güncel tehditleri barındıran veri seti üzerinde kıyaslanmakta ve birleşik performansları araştırılmaktadır.

Bu çalışmanın katkıları şu şekildedir;

- Android ekosisteminin, siber güvenlik açısından mevcut durumu güvenlik firmalarının sunduğu güncel raporlar doğrultusunda değerlendirilmekte ve iki potansiyel güvenlik riskine dikkat çekilmektedir.
- CICMalDroid2020 [1] veri setinden, DroidMalImg görsel veri seti oluşturulmakta ve açık erişime sunulmaktadır [2]. Bu aşamada, görsel veri seti oluşturmada kullanılan iki yöntem birbirleriyle karşılaştırılmaktadır.
- 25 adet son teknoloji evrişimsel sinir ağı modeli, geliştirilen toplu ince ayar öğrenim aktarımı yöntemiyle eğitilmekte ve model performansları karşılaştırılmaktadır.
- Eğitilen modellerden 24 adet birleşik model oluşturulmakta ve model performansları karşılaştırılmaktadır.

Tez çalışması, kavramsal çerçeve, ilgili çalışmalar, yararlanılan araçlar ve veri seti, geliştirilen yöntem, modellerin eğitilmesi ve değerlendirilmesi, modellerin birleştirilmesi ve birleşimlerin değerlendirilmesi, tartışma, sonuç ve öneriler bölümlerinden oluşmaktadır. Kavramsal çerçeve bölümünde, çalışmanın anlaşılması için gerekli kavramlar ve bilgiler sunulmaktadır. İlgili çalışmalar bölümünde, alanla ilgili öncül çalışmalar ve gerçekleştirilen tez kapsamındaki çalışmalar incelenmektedir. Yararlanılan araçlar ve veri

seti bölümünde, yararlanılan araçlar, çalışmada kullanılan veri seti, veri setinin üretilmesi aşamaları açıklanmaktadır. Geliştirilen yöntem bölümünde, çalışmayı gerçekleştirmek için kullanılan, geliştirilen yöntem ve deney ortamı açıklanmaktadır. Modellerin eğitilmesi ve değerlendirilmesi bölümde, açıklanan yönteme göre eğitilen modeller tek tek değerlendirilmekte, modellere ait eğitim grafikleri ve karmaşa matrisleri sunulmaktadır. Modellerin birleştirilmesi ve birleşimlerin değerlendirilmesi bölümünde, eğitilen modellerden birleşimlerin oluşturulması açıklanmakta ve bu birleşimlerin temel model değerlendirme metriklerine göre değerlendirilmesi yapılmaktadır. Son bölümde, çalışma sonunda varılan sonuçlar açıklanmakta, ilgili çalışmalarla karşılaştırılmaktadır.

2. KAVRAMSAL ÇERÇEVE

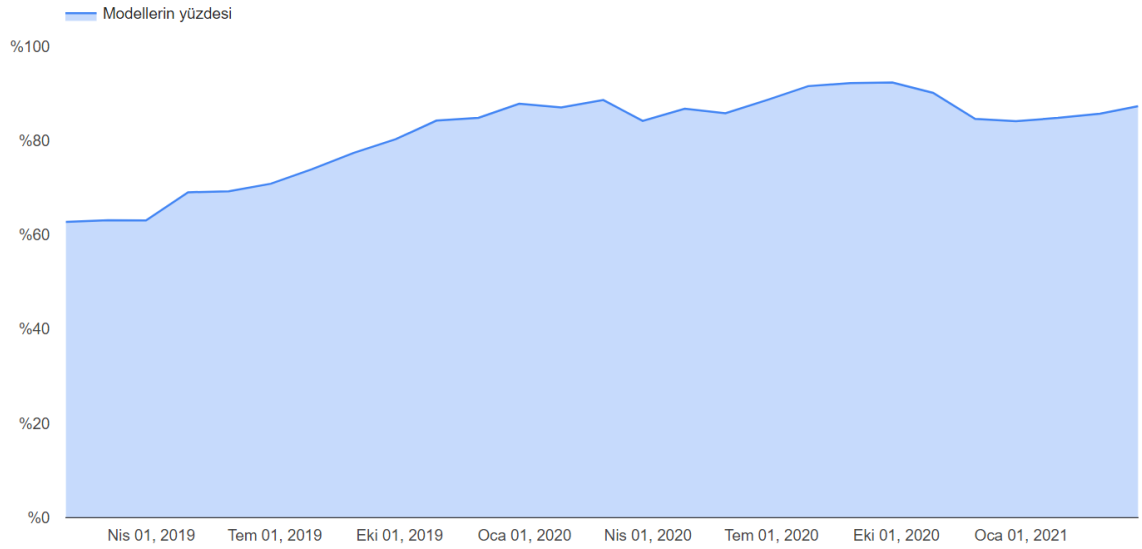
Bu bölümde, tez çalışmasının anlaşılması için gerekli olan kavramsal çerçeve sunulmaktadır. Android ekosistemi mevcut ve potansiyel riskleri, Android işletim sistemi ve dosya yapısı, makine öğrenmesi ve derin öğrenme, evrişimsel sinir ağları ve evrişimsel sinir ağı modelleri açıklanmaktadır.

2.1. Android Ekosistemi Mevcut ve Potansiyel Riskler

2008 yılında ilk çıkışını yapan Android işletim sisteminde, bilinen ilk zararlı olan FakePlayer [3] adlı zararlının ortaya çıkması yaklaşık 2 yıl sürmüştür. İlk zararlının çıkması 2 yıl sürmesine rağmen, günümüzde her yıl binlerce yeni zararlı yazılım sistemde yerini almaktadır. Malwarebytes firmasının 2021 yılında yayınlamış olduğu raporda [4], kötücül yazılımların, yıllar geçtikçe, tespitinin daha güç, daha sessiz ve daha hileci olduğunu belirtilmektedir. Aynı raporda, 2019 yılında tespit edilen HiddenAds adlı reklam zararlısının 2,5 kata yakın arttığı, 288.233'ten 704.418'e çıktığı belirtilmektedir. Bu rapordan anlaşılacağı üzere reklam zararlılarının, siber saldırganlar için hala oldukça kolay ve kazançlı birer "iş" alanı olduğu gözlemlenmektedir. Raporda, sahte bankacılık yazılımlarında olan büyük artış dikkat çekmektedir. 2019 yılında tespit edilen BankBot yazılımlarının sayısı 5025 iken, bu sayı 2020 yılında 198.031 olmuştur. Veriler, saldırıların hızla arttığını ve saldırganların daha hassas alanları hedef aldığını göstermektedir. Kovid-19 pandemisinin yarattığı ortam, siber saldırganlar için yeni iş fırsatları sunmaktadır. Pandemi sebebiyle artan ve kimi zaman zorunlu olan mobil bankacılık hizmetleri ve mobil bankacılık uygulamaları, daha fazla insanın saldırganların hedefi olmasına neden olmaktadır. Son yıllarda ortaya çıkan ve yeni bir kategori olan riskli yazılımların oluşturduğu risk, raporda belirtildiği üzere gittikçe artmaktadır [4]. Fabrika ayarlarına dönülmesine rağmen, cihazda kalmaya devam eden tehditler de raporda belirtilmektedir [4]. İlgili raporda, zararlılar arasında iş birliği olduğu, bir zararlının bulaşması halinde zamanla diğer zararlıları da cihaza yüklediği ve bu şekilde yayılmaya devam ettiği ifade edilmektedir. Raporda, Kovid-19 pandemisinin neden olduğu haklı korkuyu, saldırganların sosyal mühendislik saldırılarında kullandığı belirtilmektedir. Bu durum yaşanan yeni gelişmelere ve trendlere göre saldırganlarının yeni yöntemler ve teknikler geliştirdiğini

göstermekte, aynı zamanda ilgili alanın saldırganlar için cazip olduğu ve yatırım yapmaya değer olduğu iddiasını güçlendirmektedir.

Akıllı telefonlar için mobil işletim sistemi istendiğinde, iOS ve Android olmak üzere iki tane işletim sistemi akla gelmektedir. Apple firmasının üretmiş olduğu iOS işletim sistemi kapalı kaynak kodlu ve kendi ürettiği cihazlara özel olmaktadır. Açık kaynak kodlu işletim sistemi olan Android, mobil cihaz üreticileri tarafından sıklıkla tercih edilmektedir. Dünya genelinde 2021 yılı itibariyle 3 milyardan fazla Android cihaz olduğu açıklanmıştır [5]. Bu cihazlar çok sayıda üretici tarafından üretilmektedir. Android işletim sistemi, sık sık güncellenmekte, neredeyse her yıl yeni bir sürüm duyurulmaktadır. Tez çalışmasının gerçekleştirildiği tarih itibariyle son sürüm Android 12 olmakta, Android 13 ise yakın zamanda çıkışını yapması beklenmektedir. Cihaza yeni özelliklerin ve geliştirme katmanlarının eklendiği uygulama programlama arayüzü (API) seviyeleri, Android ekosisteminin diğer bir sürüm takip sistemi olmaktadır. Güncel API seviyesi 32 olmakla birlikte Android 13 ile 33 seviyesine yükselmesi beklenmektedir. 2008'den günümüze 14 yıl geçmesine rağmen, 32 güncelleme alması, yıl içerisinde sürekli güncellendiğini göstermektedir. Fakat bu güncellemelerin kullanıcı cihazlarına ulaştırılması, cihaz üreticilerinin inisiyatifine kalmaktadır. Üreticiler, güncellemeleri kullanıcılarına ulaştırmada ya yetersiz kalmakta ya da isteksiz davranmaktadırlar. Benzer şekilde, önceki sürümlerin desteği Google tarafından sonlandırılmaktadır. Her yeni sürümde güncelleme desteği yaklaşık 3 yıl sürmektedir. Kullanıcıların, çalışan ve eskimemiş cihazlarını, sırf güvenlik güncelleştirmeleri alabilmek için değiştirmeleri beklenmemektedir. Cihazlarını yenileseler bile, önceki cihazları çalışır olduğundan çöpe atmayacakları düşünülmektedir. Google şeffaflık raporu [6] bu durumu doğrular niteliktedir. Şekil 2.1'de ilgili rapordan alınan grafik sunulmaktadır.



Şekil 2.1. 90 günlük dönem içerisinde güvenlik güncellemesi alan modellerin oranı

Şekil 2.1'de, son iki yıl içerisinde çıkıp, son 90 günde güncelleme alan cihazların oranı gösterilmektedir. Yatay eksende tarih bilgisi ve dikey eksende modellerin yüzdesi gösterilmektedir. Ocak 2019 tarihinde, güncelleme alan cihaz oranı %62,749 olarak açıklanmıştır, bu oran Mart 2021 tarihinde %87,311 olarak açıklanmıştır. Oranlarda artış olmasına rağmen bu oranların sadece son iki yılda çıkan cihazları kapsadığına ve iki yıldan daha eski cihazları dahil etmediğine dikkat etmek gerekmektedir.

Statista'dan [7] elde edilen verilere göre, 2021 yılı itibariyle, mevcut cihazların işletim sistemi dağılımları, Android 11 %17,73, Android 10 %36,47 olarak görülmektedir. Güncelleme desteği süren en eski sürümün Android 10 olduğu göz önünde bulundurulursa, 3 milyardan fazla cihazın en çok yarısına Google tarafından güvenlik güncellemeleri sunulduğu anlaşılmaktadır. 2022 yılı verilerinde, bu oranlarda Android 11 %37,92, Android 10 %26,02 olacak şekilde artış gözlenmektedir. Android 12 işletim sistemine sahip cihazların oranıysa henüz bilinmemektedir. Tüm bu hususlar değerlendirildiğinde, sayısı milyarları bulan ve güncelleme almayan mobil cihazların siber uzayda potansiyel bir tehdit olduğu görülmektedir.

Gerek reklam zararlıları olsun gerek sahte banka zararlıları, bu zararlıları oluşturan saldırgan motivasyonu incelendiğinde, kazanç sağlama amaçlı saldırılar olduğu, genellikle kişileri ve (özel kurumlar ağırlıklı olmak üzere) kurumları hedef aldıkları görülebilmektedir. Mevcut durumların yanında, gelecekte karşılaşılabilecek potansiyel

riskleri de değerlendirmek gerekmektedir. Para ve kazanç sağlama amaçlı saldırganlar yerine, kendilerince kutsal, önemli motivasyonlara sahip saldırganlar düşünüldüğünde, saldırılarında bu doğrultuda potansiyel riskler oluşturması beklenmektedir. Saldırganlar, toplumda siyasi değişimler yaratma, toplumları kendi düşünce ve inançları doğrultusunda yönlendirme motivasyonu taşımaktalarsa, saldırgan hedeflerinin kişisel, kurumsal boyuttan, toplumsal boyuta geçebileceği düşünülmektedir. Aslında bu durum oldukça yüksek ve olası bir potansiyel tehlike barındırmaktadır. Bu iddiayı desteklemek için Cambridge Analytica skandalına [8] bakmak gerekmektedir. Facebook adlı sosyal medya platformunun ve Cambridge Analytica adlı kurumun dahil olduğu düşünülen skandal, Amerikan seçimlerinde seçmenleri istedikleri adaya yönlendirmek için analiz ettikleri, onların görüşlerini değiştirecek şekilde içerik gösterdikleri iddiasını taşımakta ve bu hususta önemli deliller barındırmaktadır. Birçok ülkede çeşitli davalara ve soruşturmalara konu olmuş bulunmakta ve yasal süreçler devam etmektedir. Skandalda, kamu tarafından bilindiği kadarıyla siyasiler, firmalar söz konusu olmaktadır. Skandal, saldırgan kişi ya da grupların, kendi kutsal amaçları için hedef cihazlara yükledikleri zararlı yazılımlarla toplumları istekleri doğrultusunda manipüle edilebileceğini göstermekte ve potansiyel tehdidi de ortaya koymaktadır. Bu potansiyel tehdidi engellemek için, cihaz güvenliğine dikkat etmek, cihazların güncelleme almasına ve üreticilerin güncelleme vermesine devam edeceği zorlayıcı yasal yaptırımlar düzenlemek yazılım seviyesinde alınabilecek tedbirlerdir. Toplumları siber güvenlik konusunda bilinçlendirici faaliyetleri arttırmak ve güncel tehditlerin yanında, gelecekte karşılaşabileceği potansiyel tehditler konusunda da bir farkındalık yaratmak, uzun dönemde alınabilecek tedbirler olabileceği düşünülmektedir.

Güvenlik güncellemelerinden mahrum ve sayısı milyarları bulan cihazların, siber silah olarak kullanılma potansiyelleri bulunmaktadır. Bu cihazlar yüksek işlem kapasitesine sahip, gerekli enerjiyi karşılayacak bataryaları bulunan ve mobil şebekelere ve internet gibi daha büyük ağlara sürekli bağlı olan cihazlardır. Bu durum, ilgili cihazları büyük çaplı dağıtık hizmet engelleme saldırıları için ideal konaklar haline getirmektedir. Kurban cihazlara bulaşan zararlı yazılımla, cihazları birer zombi haline getiren saldırganlar, planladıkları zamanda gerçekleştirdikleri toplu bir saldırıyla büyük kesintilere ve toplumsal kargaşaya neden olabilecekleri düşünülmektedir. Deprem gibi doğal felaketlerde, yakınlarına ulaşmak isteyen insanların sebep olduğu yoğunluktan ötürü, mobil şebekelerin hizmet veremez duruma geldiği hatta internet erişiminin bile aksadığı, verilemediği olaylar

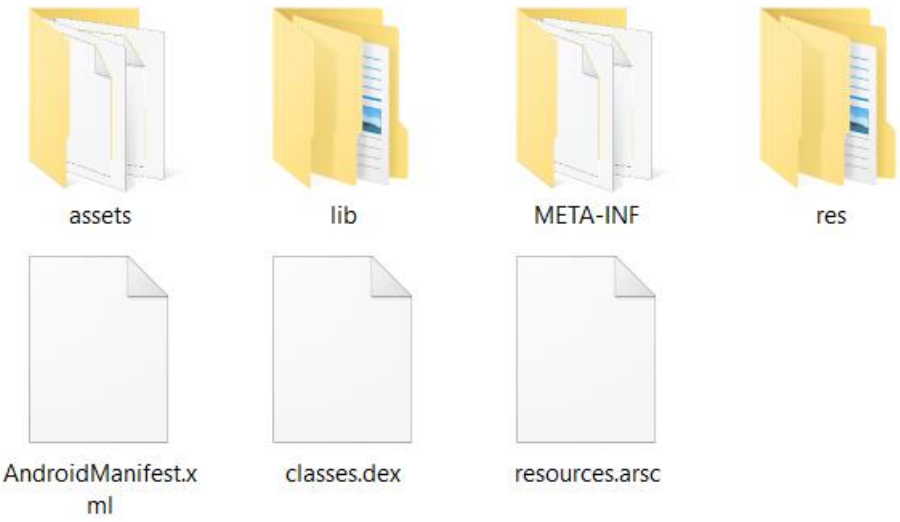
yaşanabilmektedir. Aksamalar kimi zaman bankacılık hizmetlerini de aksatabilmekte, ciddi ekonomik zararlara neden olabilmektedir. Kâğıt paranın giderek kullanımının azaldığı ve azalacağı düşünüldüğünde, bankacılık hizmetlerinin aksaması hazırlıksız bir toplumda ciddi kargaşaya neden olabileceğini düşünmek ve potansiyel tehditleri değerlendirirken göz önünde bulundurmak gerekmektedir. Nitekim Estonya'da yaşanan olaylarda, siber saldırılar sonucu bankacılık uygulamaları uzun süre aksamış ve halk büyük mağduriyetler yaşamıştır [9]. Bu durumları engellemek için, cihaz güvenliğinin sağlanması ve güvenliği olmayan cihazların siber uzaydan uzaklaştırılması, altyapı yatırımları gerçekleştirilirken bu ihtimallerin göz önünde bulundurulması gerekmektedir.

2.2. Android İşletim Sistemi ve Uygulama Dosyası Yapısı

Google'ın yöneticiliğini yaptığı, Open Handset Alliance adlı konsorsiyum tarafından geliştirilen, Linux tabanlı açık kaynak mobil işletim sistemi olan Android, ilk beta sürümünü 5 Kasım 2007 'de ve ilk yazılım geliştirme kitini de 12 Kasım 2007 yılında yayınlamıştır. Eylül 2008 yılında, mobil dünyaya çıkması ve son kullanıcılara ulaşması, ilk olarak HTC firmasının Dream modeli akıllı telefonuyla gerçekleşmesinden beri, Android işletim sistemi hem tasarım hem kapasite hem de kullanım amacı ve yeri olarak köklü değişiklikler geçirmiştir. Her yeni sürümle birlikte, yeni özellikler kazanmanta olan Android işletim sistemi, akıllı telefonlarla başladığı yolculuğunda, akıllı televizyonlara, saatlere, arabalara ve nesnelerin interneti olarak adlandırılan IoT cihazlara varınca kadar pek çok alanda kullanılmakta ve artık Android ekosistemi olarak adlandırılmaktadır. Android, işletim sisteminin gelişimini ortaya koyan sürümler ifade edilirken iki yaklaşım izlenmektedir. Birinci yaklaşım, son kullanıcının da anlamasını kolaylaştırdığı ve pazarlama faaliyetlerinde kullanımını kolaylaştırdığı için her büyük sürüme, bir sürüm numarası ve sürüm numarasını temsil eden ve alfabetik şekilde ilerleyen bir tatlı ismi verilmesidir. Bu yaklaşım, Nisan 2009 tarihinde çıkan Android 1.5 Cupcake sürümüyle başlayıp, Ağustos 2018 yılında çıkan Android 9.0 Pie sürümüne kadar sürmüştür. Eylül 2019 yılında çıkan, Android 10 ile tatlı isimleri verilmesi son bulmuştur. Genelde geliştiricilerin tercih ettiği ikinci yaklaşım, her bir Android sürümünü ilgili API sürümüyle anmaktır. İlk çıkan Android sürümünde tatlı adlandırmasına dayanan bir kod ismi bulunmazken, ilgili sürüm API 1 olarak adlandırılmakta ve Android 12L sürümüyle birlikte API 32 kadar gitmektedir. Android işletim sistemine hazırlanan uygulamalar, en düşük API seviyesi ve hedef API seviyesi seçilerek geliştirilmektedir. API seviyeleri

uygulamanın Android manifesto dosyasında belirtilmektedir. Uygulamada belirtilen en düşük API seviyesinden daha düşük sürüme sahip cihazlarda, uygulama kurulamamaktadır.

Akıllı cihazları ve bu cihazlardan çalışan işletim sistemlerini daha kullanışlı kılan en önemli unsur, bu sistemlere kurulabilen üçüncü parti uygumalar olmaktadır. Android cihazlar, üreticinin paket içerisinde sunduğu uygulamalarla birlikte satın alınmaktadır. Bu uygulamalar kullanıcılara yetersiz gelebilmektedir. Yeni kullanım imkanları elde etmek, akıllı cihazdan daha fazla yararlanmak için, kullanıcılar Google tarafından sunulan resmi uygulama marketi olan Google Play ya da diğer üçüncü parti uygulama marketleri üzerinden uygulamaları indirebilmekte, kurabilmektedirler. Android uygulamaları, APK dosya formatıyla gösterilen Android uygulama paketleri sayesinde, destekleyen cihazlara kurulabilmektedir. APK dosyası, uygulamanın çalışması için gerekli bileşenlerden oluşmaktadır. APK dosyaları sıkıştırılmış formattadır ve zip, winrar gibi bir araçla çıkartılabilmektedir. Çıkartıldığında, dosyalar ve klasörler gözükmektedir. Şekil 2.2'de bu dosyalar ve klasörler gösterilmektedir.



Şekil 2.2. Android uygulamasına ait dosya bileşenleri

Şekil 2.2'de gösterildiği üzere bir APK dosyası, assets, lib, META-INF, res klasörlerinden ve AndroidManifest.xml, classes.dex, resources.arsc dosyalarından oluşmaktadır. Assets klasöründe, uygulamada kullanılan logolar, fontlar, grafikler ve benzeri varlıklar bulunmaktadır. Lib klasöründe, uygulama içerinde kullanılan çoğunlukla C, C++ dilinde

yazılmış ve derlenmiş saf kod içeren kütüphaneler ve farklı işlemci mimarilerine göre uyarlamaları bulunmaktadır. META-INF klasöründe, uygulamaya ait sertifika bilgileri ve java paketlerinin çalışabilmesi için gereken diğer meta bilgileri bulunmaktadır. Res klasöründe, uygulamanın arayüzünü oluşturan XML formatında dosyalar bulunmaktadır. Android cihazlar bu XML formatında belirtilen birleşenleri kullanarak, kullanıcı arayüzünü gösterirler. Resources.arsc dosyası, Res klasörüne benzer fakat önceden derlenmiş bulunmaktadır. Uygulama çalışırken kullanılması gereken değerler, isim, özellik ve id formatında tablo şeklinde tutulmaktadır. AndroidManifest.XML dosyası, uygulamada bulunan ve uygulamanın parçacıkları, modülleri olarak düşünülebilecek aktivitelerin, arkaplanda çalışan ve uygulama boyunca kullanılacak olan servislerin, dinleyicilerin, istenilen izinlerin, hedeflenen, uyumlu cihazların, API sürümü gibi uygulamaya ait birçok statik özelliğin bildirildiği dosya olmaktadır. APK dosyasından çıktığında ikili dosya biçiminde ve okunamamaktadır. Bir dönüştürücü yardımıyla, okunabilir biçime çevrilebilmektedir. Classes.dex ve sonu ".dex" ile biten dosyalar, uygulamaların temel çalışma mantığını bulundurmaktadırlar. Dex, dalvik executable (Türkçe dalvik çalıştırılabilir) anlamına gelmektedir. Android uygulamaları çoğunlukla java ve kotlin yazılım dilinde yazılmaktadır. Kaynak kod dosyaları sırasıyla java ve kt dosya formatında olmaktadırlar. Kaynak kod dosyaları derlendiğinde byte-kod dosyalarına dönüştürülmekte ve .class dosya formatına sahip olmaktadırlar. Son olarak Android yazılım geliştirme kitinin içerisinde bulunan dx aracı ile "dex" dosya formatına dönüştürülmektedirler. Dex dosyaları Android işletim sistemi içerisinde bulunan sanal makine aracılığıyla çalıştırılmaktadır.

2.3. Makine Öğrenmesi ve Derin Öğrenme

Geliştirilen algoritmalar yardımıyla, makinelere, örüntüleri ayırt etme yeteneği kazandırılması çabaları makine öğreniminin temelini oluşturmaktadır. Bu algoritmaları geliştirirken canlılardan esinlenen araştırmacılar, tıpkı insan ve diğer canlı varlıkların yaptığı gibi, geçmiş tecrübelerden çıkarılan derslerle mevcut ya da yeni durumlara karşı bir sonuç üreten ve karşılaşılan her yeni durumda bu sonucun doğruluğuna ya da yanlışlığına göre tecrübelerini güncelleyen öğrenme algoritmaları tasarlamaya çalışmaktadırlar.

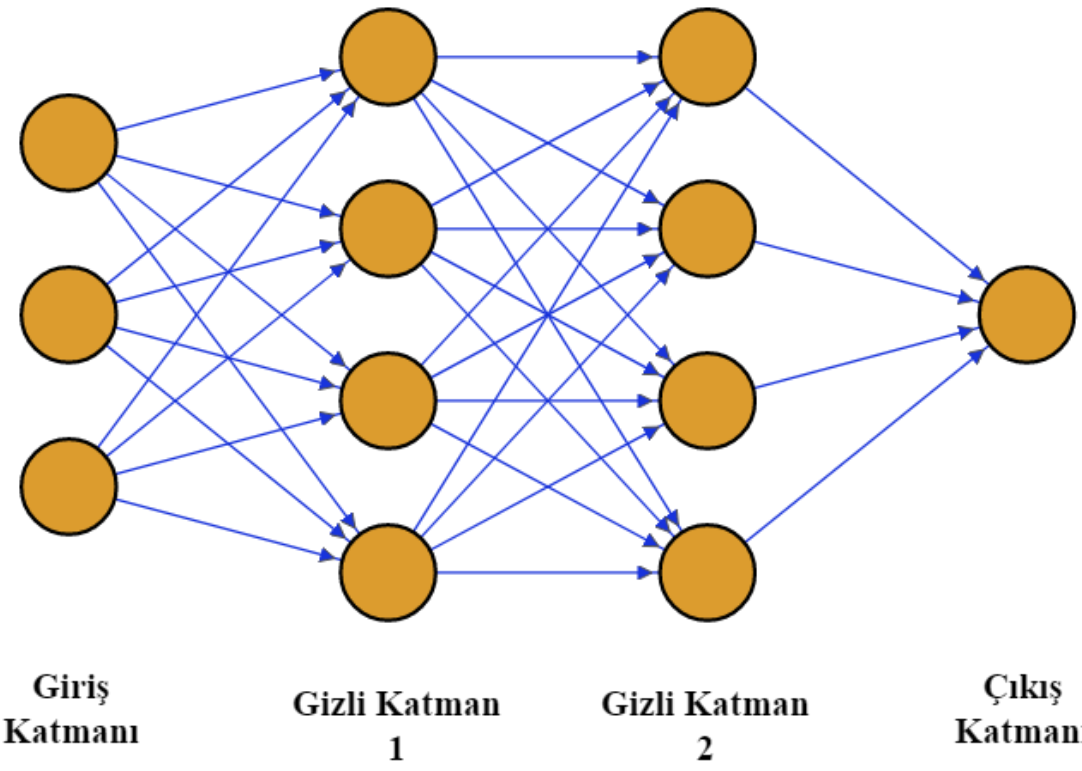
Makine öğreniminde, girdi değerlerine karşı, çıktı değerleri üreten, içerisinde bir ya da daha fazla öğrenme algoritması barındıran yapılara model denilmektedir. Modelin

eğitimler sonucunda örüntüleri öğrenmesi ve daha önce karşılaşmadığı örnekler için geçmiş örneklere bakarak tahminlerde bulunması istenmektedir. Makine öğrenmesinde eğitim aşaması, modelin istenilen davranışı, örüntüyü öğrenene kadar, sunulan veri setiyle davranışlarının (sonuçlarının) gözlemlendiği ve bu sonuçların kabul edilebilir, istenilen sonuçlara ulaşıncaya kadar davranışların güncellendiği süreci kapsamaktadır. Modele eğitim süresinde sunulan örnekler veri setini oluşturmaktadır. Veri seti alana özgü, soruna dair içerisinde genellikle tasniflenmiş, etiketlenmiş örnekler bulunduran yapılar olmaktadır. Veri seti oluşturmak için öncelikle alanla ilgili veriler toplamakta, toplanan veriler çeşitli araç ve yöntemlerle işlenmektedir. Veri seti, kullanılacak modeli, modelin yapısını, modele ait diğer değişkenleri belirleyen önemli bir bileşen olmaktadır. Bunun temel sebebi hayatın olağan akışı gereği öncelikle verinin bulunması gerekmektedir. Bir firmaya ait üretim miktarlarını, gelecek aylara göre tahmin edecek bir model tasarlamak istediğinde, önceki üretim miktarlarına ait verilerden yararlanılmaktadır. Buradaki veri hali hazırda gerçekleşmiş olan üretim miktarları olmaktadır. Gerçekleşen üretim miktarlarını gösteren veri bulunmadığında, gelecek aylardaki üretin miktarlarını tahmin edecek model kurulamamaktadır.

Veri seti üzerinde, veriler modele sunulmadan önce gerçekleştirilen işlemlere, ön işlemler denilmektedir. Ön işlemler model başarımını doğrudan etkilediği için çalışmalarda önemli yer tutmaktadırlar. Gerçekleştirilen ön işlemler çözüm aranan alana göre çeşitlenmektedir. Örneğin görsellerin sınıflandırılması için bir model oluşturulmuşsa, yapılacak ön işlemler bu görsellerin modele uygun piksel boyutlarına dönüştürülmesi, görsele ait piksel değerlerinin, modele uygun şekilde dağılım sergilemesinin sağlanması (normalize edilmesi) biçiminde olmaktadır.

Veri setindeki her bir örneğe karşılık, modelden beklenen çıktıya etiket (label), temel gerçek (grund truth) denilmektedir. Bu verilerin gözlemler sonucunda elde edildiği ve çıkarımla elde edilen verilerin aksine gerçek veriler olduğu varsayılmaktadır. Modelin, sunulan örneklere ait etiket karşılıklarını bildiği durumda yapılan eğitim denetimli öğrenme olarak adlandırılmaktadır. Model, veri setindeki örneklere ait etiketler, temel gerçekler sunulmadan eğitildiği durumda ya da bu temel gerçeklere hiç sahip olunmadan olası örüntüler araştırıldığı durumda yapılan eğitime denetimsiz öğrenme denilmektedir. Temel gerçeklere sahip olunmadan, olası örüntülerin, ilişkilerin araştırılması aynı zamanda veri madenciliği olarak adlandırılmaktadır.

Derin öğrenme, makine öğrenmenin alt kümesi olmakla birlikte, son yıllardaki başarılı uygulamaları ve kendini diğer makine öğrenmesi algoritmalarından ayıran özellikleri sayesinde kendi başına oldukça geniş bir kümeyi ifade etmeye başladığı görülmektedir. Evrişimsel sinir ağları gibi derin öğrenme modellerinin sahip olduğu kendi kendilerine özellik çıkartabilme yeteneği, derin öğrenmeyi makine öğrenmesinden farklılaştıran sebeplerden olmaktadır. En basit şekilde, özellik (öznitelik) bir örneği, başka bir örnekten ayırt etmeyi sağlayan bilgiler olmaktadır. Model açısından bu özellikler, insanların algıladığı şekilde olmayabilmektedir. Fakat önemli olan modelin bir örneği, diğer bir örnekten ayırt etmesini ve öğrenmesini sağlaması olmaktadır. Makine öğrenmesinde bu özellikler, bir ya da daha fazla alan uzmanın görüşü, geliştirilen çeşitli araç ve yöntemlerle sağlanmaktadır. Derin öğrenme modelleriyse, insan emeğine dayanan çalışmayı, teknolojinin sunduğu artan işlem kapasiteleriyle telafi etmekte ve hatta aşabilmektedir. Diğer önemli özellikse derin öğrenme kavramına ismini kazandıran derinlik olmaktadır. Derin öğrenme kavramındaki derinlik, birden fazla yapının, genellikle birbirini izleyen ardışık şekillerde birleştirilmesinden gelmektedir. Bu yapının bir yapay sinir ağı algoritması olduğu durumda, bu yapay sinir ağlarının ardışık birleştirilmeleriyle derin sinir ağları elde edilmektedir. Şekil 2.3'te basit bir derin sinir ağı gösterilmektedir.

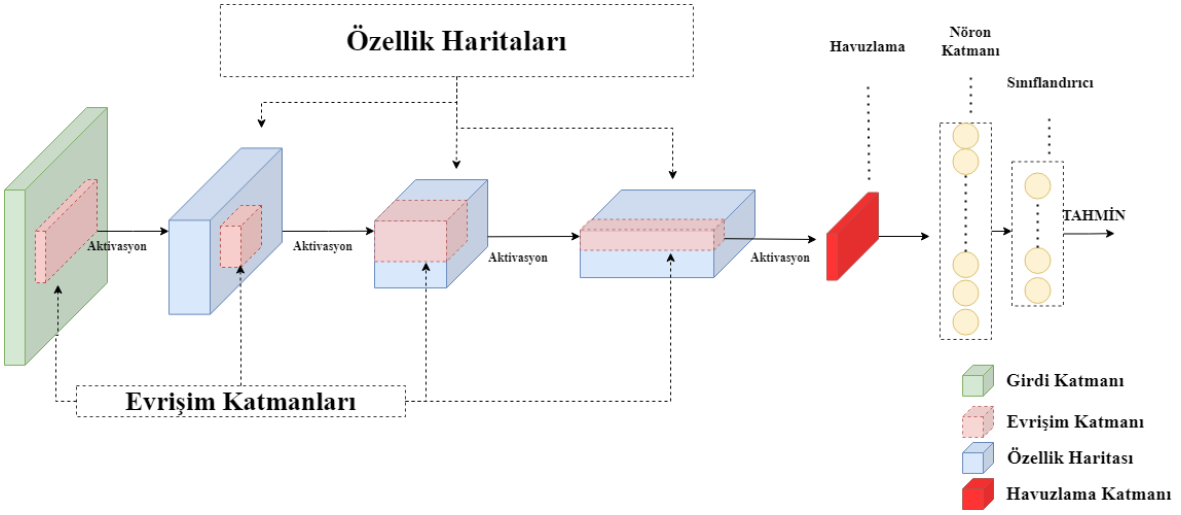


Şekil 2.3. Basit bir derin sinir ağı

Şekil 2.3'te her bir yuvarlak, bir yapay nöronu temsil etmektedir. Giriş katmanı modelin öğrenmesi için örneklerin sunulduğu katmandır. Örnekler genellikle vektör formatında sunulmaktadır. Giriş katmanındaki her bir nöron girdiyi, özelliği temsil etmektedir. Tam bağlı modellerde, birbirini izleyen katmanlar arasında önceki nörondan çıkan değer, diğer katmandaki tüm nöronlara iletilmektedir. Bu bağlantı şekilde oklarla gösterilmektedir. Oklar, aynı zamanda ağın ağırlıklarını temsil etmektedir. İki katman arasında, gösterilen ok sayısı kadar ağırlık değeri ve izleyen katmandaki nöron sayısı kadar bias değeri bulunmaktadır. Ağırlık değerleri ve bias değerleri parametre olarak adlandırılmaktadır. Önceki katmandaki, nöronlardan çıkan değer, ağırlık değeriyle çarpıldıktan sonra, bias değerinin eklenmesiyle, izleyen katmandaki nöronun giriş değerini oluşturmaktadır. Girdi katmanı haricinde, tüm nöronlarda aktivasyon adı verilen bir işlem uygulanmaktadır. Aktivasyon işleminde, nörona gelen değer, aktivasyon fonksiyonu adı verilen ve genellikle doğrusal olmayan bir fonksiyonda işlenmektedir. Aktivasyon fonksiyonu türevi alınabilir, hesaplaması kolay bir fonksiyon olması gerekmektedir. Aktivasyon fonksiyonu sonucunda elde edilen değer nöronun çıkış değeri olmaktadır. Bu işlem katmanlar boyunca giriş katmanından, çıkış katmanına ileriye doğru sürdürülür. Çıkış katmanında, tahmin edilecek sınıf sayısı kadar nöron bulunmaktadır. İki sınıf bulunduğu durumda tek bir nöron yeterli olmaktadır. Çıkış katmanındaki nöronlarda, sorun tipine uygun bir aktivasyon değeri tercih edilmelidir. Sigmoid aktivasyon fonksiyonu 0 ile 1 arasında değerler verdiğinden ikili sınıflandırma problemlerinde kullanılabilmektedir. İkiden fazla sınıf olduğunda SoftMax aktivasyon fonksiyonu kullanılabilmektedir. Sürekli değerler beklenen regresyon problemlerinde doğrusal aktivasyon fonksiyonları tercih edilebilmektedir. Çıkış katmanında elde edilen sonuç, etiket değeriyle karşılaştırılmaktadır. Bu karşılaştırma kayıp fonksiyonuyla gerçekleştirilmektedir. Kayıp değerine göre modelin parametre değerleri seçilen optimizasyon algoritmasına göre güncellenmektedir. Güncelleme aşaması model eğitiminde, geri yayılım (backpropagation) olarak adlandırılmaktadır. Eğitim süreci, döngüler şeklinde sürmekte ve her eğitim döngüsü epoch olarak adlandırılmaktadır. Derin sinir ağları, tam bağlı katman adıyla evrişimsel sinir ağlarında da kullanılmaktadır.

2.4. Evrişimsel Sinir Ağları

Yapısında evrişim işlemi gerçekleştiren katmanlar bulunan sinir ağlarına genel olarak evrişimsel sinir ağları denilmektedir. Şekil 2.4'te basit bir evrişimsel sinir ağı gösterilmektedir.



Şekil 2.4. Basit bir evrişimsel sinir ağı

Şekil 2.4'te gösterildiği üzere basit bir evrişimsel sinir ağı, girdi katmanı, evrişim katmanı, havuzlama katmanı, tam bağlı katman ve sınıflandırı katmandan oluşmaktadır. Evrişim işlemi, giriş katmanı ve izleyen diğer katmanlar boyunca girdilerin ve çıktıların, belirlenen boyut ve sayıdaki matrislerle, matris çarpımı uygulanarak özniteliklerinin çıkartılmasıdır. Eğitim sürecince, algoritmik bir şekilde değerleri güncellenen bu matrisler, zamanla soruna özgü ayırıcı öznitelikleri çıkarmayı kendi kendine öğrenmeye başlarlar. Her evrişim katmanından sonra tespit edilen özelliklerin soruna göre özgünlüğü artmaktadır. Sorunun nesnelere ait fotoğrafları sınıflandırılması olarak düşünüldüğünde, giriş katmanında ve bu katmana daha yakın olan katmanlarda çizgi, çember, elips ve benzeri basit geometrik şekillere ait özellikler çıkartılırken, çıkış katmanına doğru nesnenin sınıflandırılmasında daha önemli olan nesnenin şekli, nesnede bulunan kapılar, düğmeler gibi daha gelişmiş özelliklerin çıkartılması gerçekleşmektedir. Evrişimsel katmanlar boyunca, her katmanda yeni özellik haritaları, önceki katmana göre çıkartılmaktadır. Havuzlama katmanında (pooling layer), son katmandaki özellik haritasında örnek azaltımı gerçekleştirilmekte ve çok boyutlu özellik haritaları bir vektöre dönüştürülmektedir. Tam bağlı katman ve sınıflandırıcı bu özellik vektörüne ait ilişkileri öğrenmekte ve bu doğrultuda sınıflandırılmaktadır.

Kendi kendine özellik çıkartabilme yetenekleri, alan bilgisine olan ihtiyacı azalttığından evrişimsel sinir ağlarını oldukça faydalı araçlar haline getirmektedir. Günümüzde birçok sorunun derin öğrenmeyle çözümünde, evrişimsel sinir ağlarından veya evrişim katmanlarından yararlanılmaktadır. ImageNet [10] adlı görsel sınıflandırma yarışmasındaki

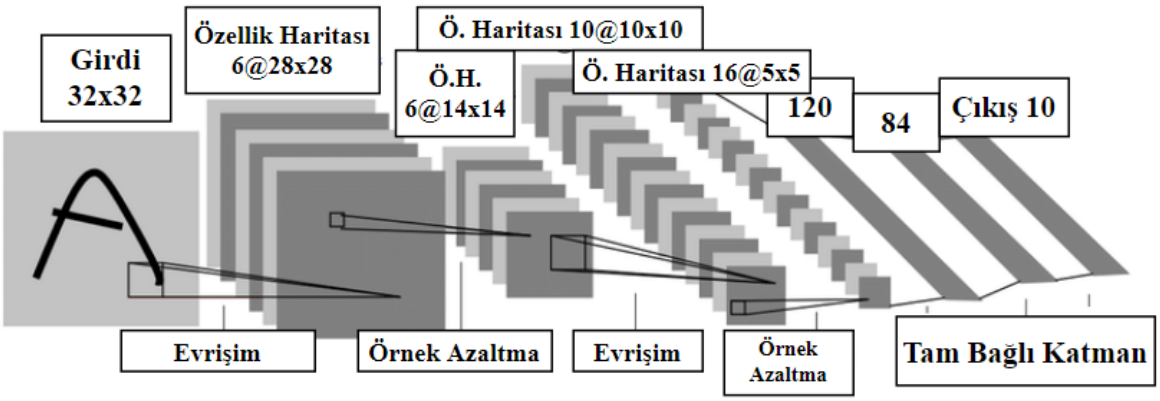
başarılarından dolayı, özellikle görsel sınıflandırma, tespit çalışmalarında evrişimsel sinir ağlarında yararlanılmaktadır. Bu tez çalışmasında gösterildiği üzere, çözülmesi amaçlanan sorun görsellere dönüştürülebildiğinde, bu sorunun çözümünde de evrişimsel sinir ağları kullanılabilmektedir.

2.5. Evrişimsel Sinir Ağı Modelleri

Bu bölümde, alanda öncü olan, ImageNet yarışmasında başarı sağlayan ve bu çalışmada yararlanılan modeller açıklanmaktadır.

2.5.1. LeNeT evrişimsel sinir ağı modeli

Evrişimsel sinir ağlarının tarihine bakıldığında LeNeT [11] ve AlexNeT [12] adı verilen iki model önemli yer tutmaktadır. LeNeT ilk evrişimsel sinir ağı içeren çalışmadır. LeNet modeline ait mimari gösterim Şekil 2.5'te verilmektedir.

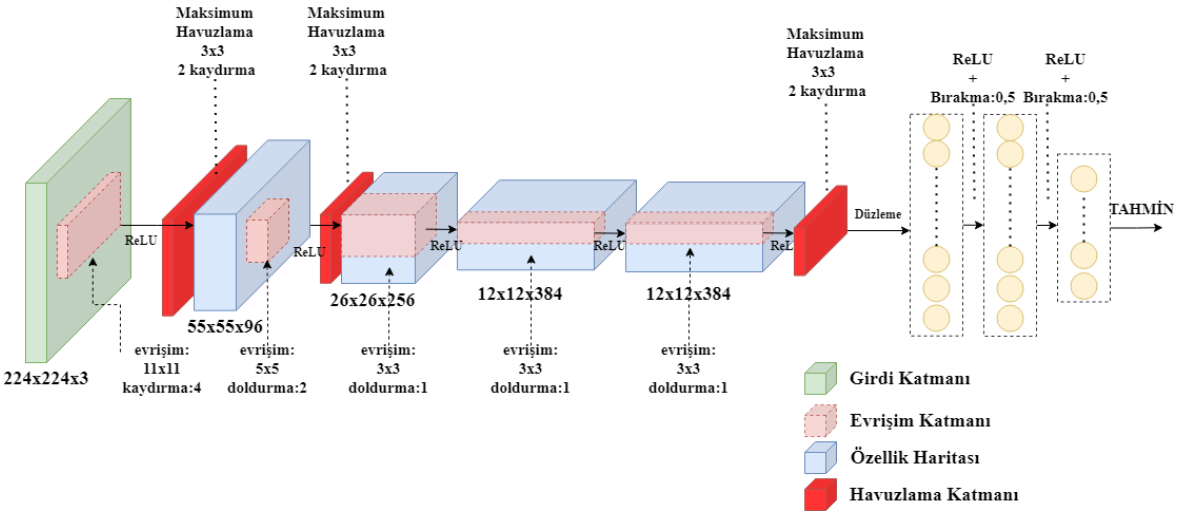


Şekil 2.5. LeNeT-5 modeli [11]

Şekil 2.5'te gösterildiği üzere LeNet modeli 32x32 piksel boyutlarındaki girdi katmanı, üç adet evrişim katmanı ve tam bağlı katman olmak üzere toplam beş adet katmandan oluşmaktadır. Evrişim katmanlarında, filtre büyüklüğü 5x5 olmak üzere, sırasıyla 6, 16 ve 120 adet özellik haritası oluşturulmaktadır. Tam bağlı katmanda, 84 nöron ve çıkış için 10 nöron bulunmaktadır. Örnek azaltma katmanlarında, ortalama havuzlama işlemi gerçekleştirilmektedir. Bu çalışmada LeNeT modeli kullanılmamaktadır.

2.5.2. AlexNet evrişimsel sinir ağı modeli

LeNeT ilk çalışma olmakla beraber, AlexNeT, ESA mimarilerinin bilinirliğini ve popülerliğini arttıran çalışma olmaktadır. ImageNet, milyonlarca görselden oluşan ve 1000 kategoriden oluşan bir görsel sınıflandırma yarışması olmaktadır. 2012 yılında düzenlenen yarışmada AlexNeT, makine öğrenme temelli diğer rakiplerine ciddi fark atarak birinci olmuştur. AlexNeT'in başarısından sonra ESA modelleri literatürde daha fazla yer almaya başlamıştır. Ekran kartlarının hesaplamalarda kullanılabilmesi, ekran kartlarının bellek kapasitesinin artması gibi teknolojik alanda yaşanan gelişmeler, çok katmanlı evrişimsel sinir ağı modellerini gerçekleştirilebilir kılmaktadır. AlexNet modeline ait mimari gösteril Şekil 2.6'da verilmektedir.



Şekil 2.6. AlexNet modeli mimari gösterimi

Şekil 2.6'da gösterildiği üzere AlexNet modeli, 224x224 boyutlarında girdi katmanından, beş adet evrişim katmanından, üç adet havuzlama katmanından ve tam bağlı katmandan oluşmaktadır. AlexNet, LeNeT ile kıyaslandığında kendisine başarıyı getiren önemli yeniliklere sahiptir. AlexNet'in sunduğu bu yenilikler günümüzde halen kullanılmaktadır. ReLu adını verdikleri aktivasyon fonksiyonu, aktivasyona uğrayacak değerle sıfır arasında en büyüğün seçilmesine dayanmaktadır. Bu fonksiyon diğer aktivasyon foskiyonlarına göre, işlem maliyeti açısından oldukça kolay hesaplanabilmektedir. Fonksiyon 0 ya da daha büyük değerleri verdiğinden ağdaki nöronların negatif değerlere düşmesini önlemektedir. Ağ derinleştikçe, modellerin öğrenme kapasiteleri oldukça artmaktadır. Bu

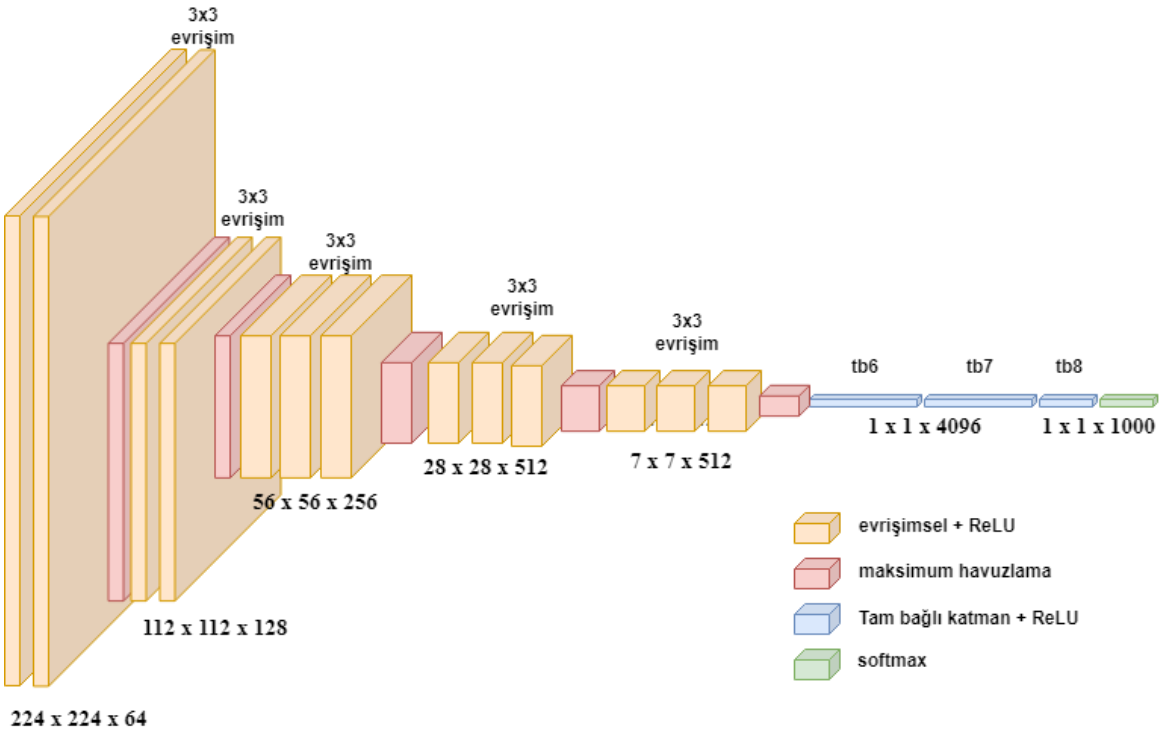
durum ilk bakışta olumlu gözükebilmektedir. Fakat modellerin bu istenmeyen, kontrol edilemeyen öğrenme performansı, modelin eğitiminde kullanılan örnekleri ezberlemesiyle sonuçlanabilmektedir. Bu durum literatürde aşırı uyum (overfitting) sorunu olarak geçmektedir. Eğitim veri setini ezberleyen ve oldukça iyi sonuç alan model, eğitim veri setinde bulunmayan örneklerle karşılaştığında düşük başarı göstermektedir. Modelin bu örneklere aşırı uyumunu engellemek için yazarlar, Dropout (Türkçe bırakma) adını verdikleri yeni bir katman geliştirmişlerdir. Bu katman ağdaki diğer katmanların arasında yer almakta ve iki katman arasındaki bağlantı değerlerini, belirlenen olasılık değerine göre sıfırlamaktadır. Bu oran %20 olarak belirlendiğinde, 100 nöron değerinden rastgele seçilecek olan 20 nöron değeri düşürülmekte, bir sonraki katmandaki nöronlara kalan 80 nörondan gelen değer ulaşmaktadır. Her eğitim örneğinde bu işlem tekrarlandığından, nöronların bir önceki nöronla kurduğu güçlü bağlar zayıflamakta, bu sayede diğer nöronlara da öğrenme fırsatı sağlanmaktadır. Yazarlar tarafından bu katmanın çok basit bir şekilde birleşim (ensemble) yarattığı söylenmektedir. Düşünüldüğünde, her örnekte farklı nöronlar aktif olduğundan aslında farklı bir ağın öğrenme süreci oluşmaktadır ve sayede birçok ağ eğitilmektedir. Bu katman sadece eğitim sırasında kullanılmakta, çıkarım (inference) sırasında bu katman aktif olmamaktadır. AlexNet modeli bu çalışmada kullanılmamaktadır.

2.5.3. VGG16 ve VGG19 evrişimsel sinir ağı modelleri

Çalışmada incelenen modeller arasında AlexNet bulunmasada, onu temsilen VGGNet [13] bulunmaktadır. VGGNet, AlexNet'in sunduğu yenilikleri ve tasarım prensiplerini aynen uygulamakta, farklı olarak sadece daha derin bir ağ kullanmaktadır. Sadece daha derin bir ağ ile 2014 yılında gerçekleştirilen ImageNet yarışmasında birinci olmuşlardır.

VGG16 evrişimsel sinir ağı modeli

VGG16 modeline ait mimari gösterim Şekil 2.7'de sunulmaktadır.

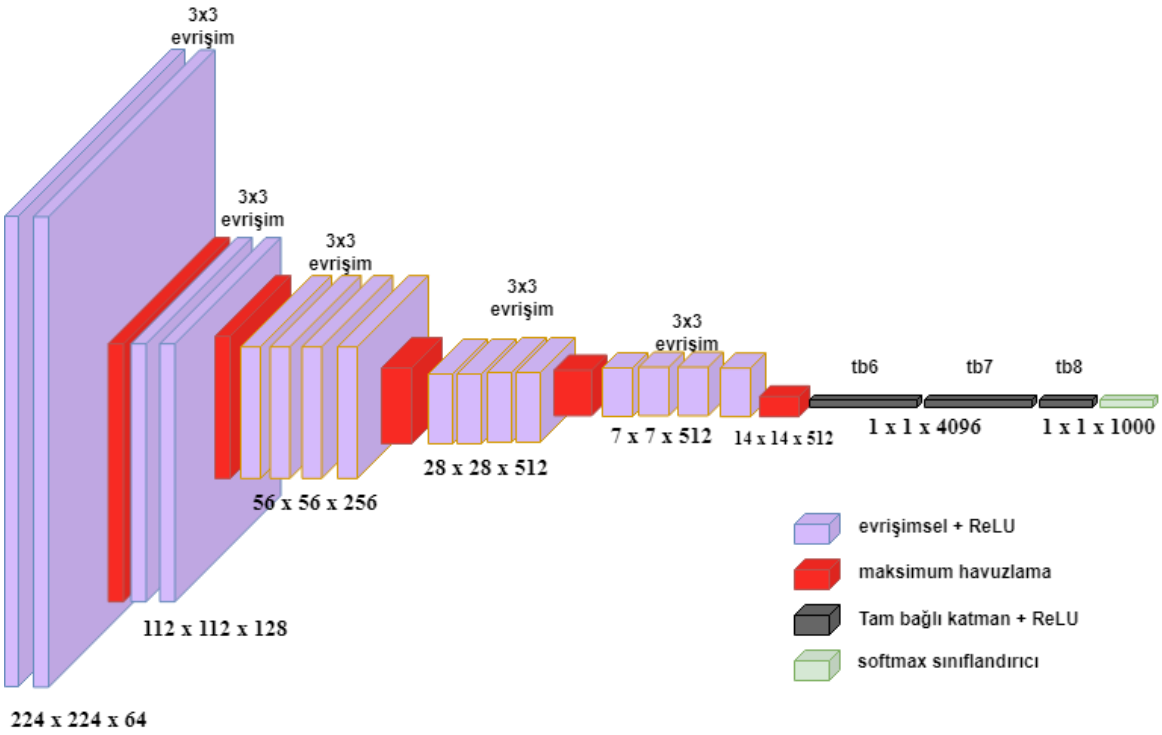


Şekil 2.7. VGG16 modeline ait mimari gösterim

Şekil 2.7'de gösterildiği üzere model, on üç katman evrişim ve son üç katman tam bağlı olmak üzere toplam 16 katmandan oluşmaktadır. Modelde aktivasyon fonksiyonu olarak ReLU kullanılmaktadır. İki kaydırmalı maksimum havuzlama kullanılarak, katmanlar arasında boyut azaltma işlemi uygulanmaktadır.

VGG19 evrişimsel sinir ağı modeli

VGG19 modeline ait mimari gösterim Şekil 2.8'de sunulmaktadır.



Şekil 2.8. VGG19 modeline ait mimari gösterim

Şekil 2.8'de gösterildiği üzere VGG19 modeli on altı katman evrişim ve son üç katman tam bağlı olmak üzere toplam 19 katmandan oluşmaktadır. Modelde aktivasyon fonksiyonu olarak ReLU kullanılmaktadır. İki kaydırmalı maksimum havuzlama kullanılarak, katmanlar arasında boyut azaltma işlemi uygulanmaktadır.

2.5.4. Inception evrişimsel sinir ağı modelleri

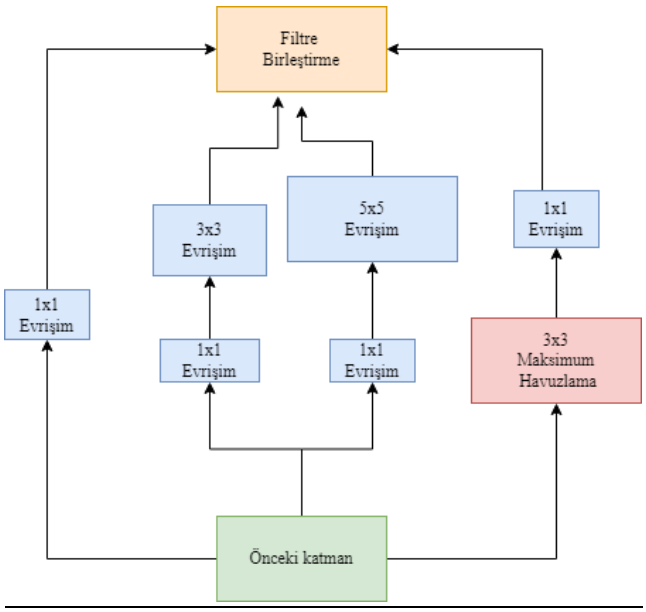
Inception modülleri, amacı hesap maliyeti daha düşük ağlar üretme gayretinden gelmektedir. Gerek AlexNet olsun gerek VGG olsun, Inception modüllerine kıyasla oldukça büyük evrişimsel filtreler kullanmaktadırlar. Her evrişimsel filtre, boyutlarının ve ürettiği özellik haritasının eleman sayısının çarpımı kadar matris çarpımı işlemi gerçekleştirmektedir. Eğer bir evrişim filtresinin boyutları 32x32 ve oluşturduğu özellik haritası 10x10 ise, toplamda 32x32x100 kez çarpım işlemi gerçekleştirilmektedir ve sayıyı ilgili katmandaki filtre sayısıyla da çarpmak gerekmektedir. Çarpım, hesaplama maliyeti

oldukça yüksek olan bir işlem olmaktadır. Katman sayısı arttıkça hesaplama maliyeti de artmaktadır. Bu durumu fark eden araştırmacılar, Inception modülleriyle bu durumu çözmeye çalışmaktadırlar. Nitekim küçük modüller, hücreler kullanarak daha büyük ağlar elde etme mantığı ileriki çalışmalarda da temel olmaktadır.

Inception [14] model ailesi yıllar içerisinde öne sürülen çeşitli modellerden oluşmaktadır. Genel isimlendirme Inception adından sonra sürüm numarası şeklinde olmaktadır. Modellerin Inception olarak adlandırılması, Inception modüllerinden gelmektedir. InceptionV1 [14], InceptionV2 ve InceptionV3 olmak üzere üç sürüm bu başlıkta açıklanmaktadır. InceptionV2 ve InceptionV3 aynı makalede sunulmuştur [15]. InceptionV3 modeli çalışmada kullanılmakta, diğer modeller kullanılmamaktadır.

InceptionV1 modeli

Ağ içinde ağa benzer yapılar olan Inception modülleri, modüller arasında birden fazla evrişimsel filtre ağları bulundurmaktadırlar. Bu ağlardan elde edilen özellik haritaları, ağların çıkış katmanında bulunan birleştirme katmanıyla birleştirilmektedirler. InceptionV1, diğer adıyla GoogleLeNet mimarisini oluşturan inception modülü Şekil 2.9'da gösterilmektedir.

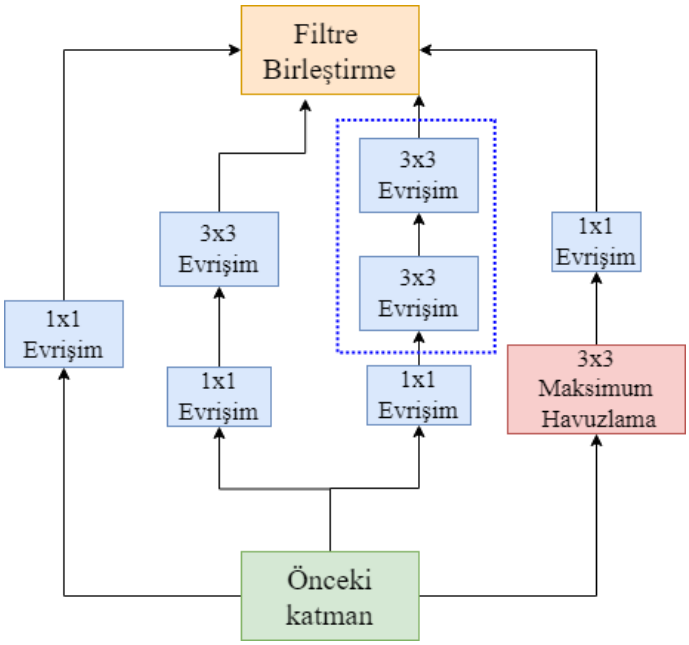


Şekil 2.9. Inception modülü (InceptionV1,GoogLeNet)

Şekil 2.9'da gösterildiği üzere önceki katmanla filtre birleştirme katmanına kadar dört yol bulunmaktadır. İlk yolda 1x1 evrişim işlemi gerçekleştirilmektedir. İkinci yolda önce 1x1 evrişim, ardından 3x3 evrişim gerçekleştirilmektedir. Üçüncü yolda, 1x1 evrişim ardından 5x5 evrişim gerçekleştirilmektedir. Dördüncü yolda 3x3 maksimum havuzlama ve 1x1 evrişim gerçekleştirilmektedir. Aslında her yolda farklı bağlantılara sahip özellik haritaları oluşturulmaktadır. 1x1 evrişim bire bir bağlantılara ait özellikleri oluştururken, ardından gelen 3x3 evrişimle toplam 9 özellik arasındaki bağlantılar, 5x5 evrişimle 25 özellik arasındaki bağlantılar oluşturulmaktadır. 3x3 maksimum havuzlamada, Maksimum havuzlamada, 9 özellik içinden en büyük değer alınarak örnek azaltımı gerçekleştirilmektedir. Tüm bu çıkartılan farklı özellikler filtre birleştirme katmanında birleştirilerek, diğer katmanlara sunulmaktadır. GoogleLeNet'te bu modüllerden 9 tanesi kullanılmaktadır.

InceptionV2 modeli

InceptionV2, InceptionV1'den kısa bir süre sonra çıkmıştır. InceptionV2 çeşitli yenilikler barındırmaktadır, bunlardan biri de yeni modül yapısıdır. InceptionV2 modeline ait Inception modülü Şekil 2.10'da gösterilmektedir.



Şekil 2.10. InceptionV2 modülü

Şekil 2.10'da gösterildiği üzere InceptionV2 modülü InceptionV1 modülüne oldukça benzemektedir. Modülün, önceki modüle göre temel farkı, 5x5 evrişim katmanının, iki adet 3x3 evrişim katmanıyla değiştirilmesi olmaktadır. Bu sayede 5x5=25 matris çarpımı yerine, 2x3x3=18 matris çarpımı yapılmaktadır. Bu sayede, ilgili katmanda %28 daha az matris çarpımı gerçekleştirilmektedir. Bu modelde de işlemleri daha küçük parçalara ayırarak etkinliği arttırma çabaları sürmektedir. 3x3 'lük evrişim hücreleri, 3x1 ve 1x3 şeklinde iki parçaya ayrılarak birleştirilmektedir. 3x3 evrişim işleminde 9 matris çarpımı gerçekleşmekteyken, 3x1' de 3, 1x3'te 3 olmak üzere toplam 6 matris çarpımı gerçekleşmektedir. Bu iki değer birbirine oranlandığında yaklaşık üçte bir daha az matris çarpımı gerçekleştirildiği görülmektedir. Bu işleme faktörize edilmiş evrişim adı verilmektedir.

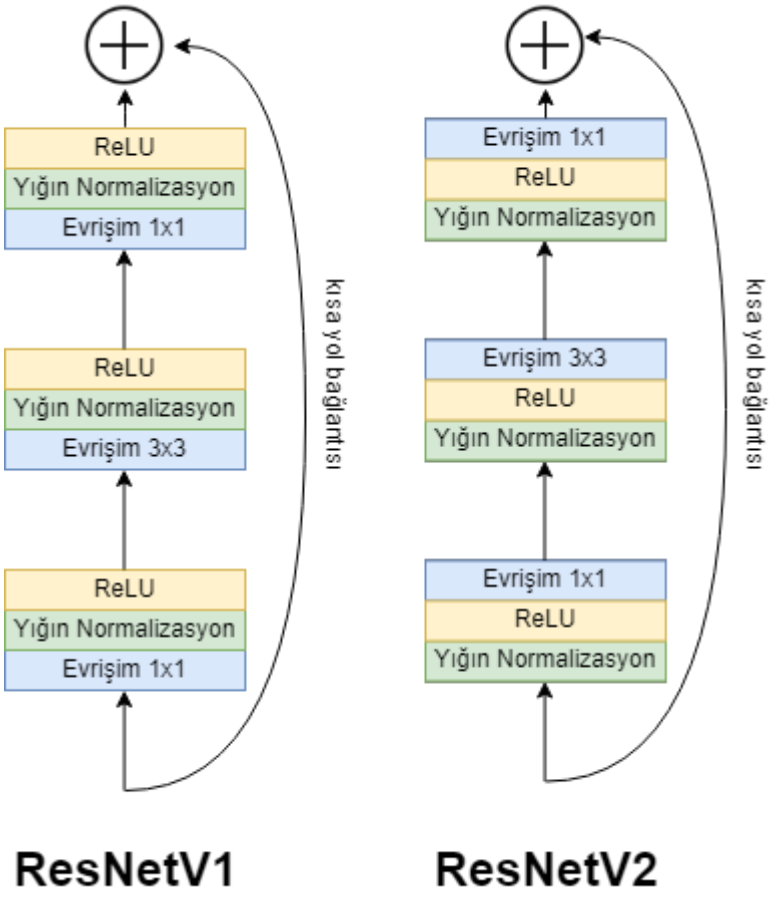
Inception v2'nin getirdiği diğer bir yenilik, yığın normalizasyon (batch normalization) [16] katmanı kullanmaktadır. Bu katman, kendinden önceki katmanın oluşturduğu çıkış değerlerini normalize etmektedir. Bunun temel nedeni derin sinir ağlarındaki ağırlık değerlerinin, ilk oluşturulması sırasında ve ilerleyen eğitim süreçleri boyunca istatistiki dağılımlarının genel model performansı açısından önemli olmasıdır. Çalışmada, Ağırlıkların, ortalaması 0, varyansı 1 olan rastgele dağılıma sahip olduğu durumda, modellerin çok daha iyi performans verdiği bildirilmektedir [16]. Eğitim süreçleri içerisinde bu ağırlık ortalamaları ve varyans değerleri, ağırlık güncellemeleriyle birlikte değişikliğe uğramaktadır. Genel olarak model eğitimleri yığın kesitleri (mini batch) şeklinde olduğundan, eğitim periyodları boyunca oluşan değişim, içsel eş değişken kayması (internal covariate shift) olarak adlandırılmaktadır. Yığın normalizasyon katmanı kendinden sonra gelen katmana standardize edilmiş girdiler sağlayarak bu durumu engellemeye çalışmaktadır. Nitekim bu katman günümüz çalışmalarında en sık kullanılan katmanlardan biri olmakta ve birçok modelde kullanıldığı görülmektedir. Bu katman sahip olduğu düzenleyici etkisiyle bırakma katmanına olan ihtiyacı da oldukça azaltmaktadır. Çalışmada, yazarlar bırakma katmanını bir önceki çalışmadan (v1) çıkartıp yerine yığın normalizasyon katmanını kullanmakta, aşırı uyum soruna yakalanmadan model performansını arttırmaktadırlar.

InceptionV3 modeli

InceptionV3, InceptionV2'deki tüm yenilikleri kapsamakta ve 7x7 faktörize edilmiş evrişim, RMSProp optimizasyon algoritması ve bir çeşit düzenleyici yöntem olan etiket yumuşatma farklılıklarını barındırmaktadır.

2.5.5. ResNet evrişimsel sinir ağı modelleri

İngilizce Residual (Artık) Network (Ağ) kelimelerinin kısaltmasıyla adlandırılan ResNet [17, 18] ağları, ortaya koydukları yaklaşımla, ağ derinliğini 100 ve üzeri katmanlara kadar arttırabilmektedirler. Ağ derinliğinin ve ağdaki ardışık katman sayısının artması yeni sorunlar çıkarmaktadır. Bu sorunlardan biri hesaplama maliyetinin artmasıdır. Hesaplama maliyetindeki artışı azaltmak için Inception modüllerinin ortaya koyduğu yaklaşımlardan yararlanılmaktadır. Ağ derinleştikçe ilk katmanlardaki sinyaller, üst katmanlara doğru zayıflamaya, kaybolmaya başlamaktadır. Benzer şekilde, geri yayılım boyunca, çıkıştaki değerlere göre elde edilen gradyan değerleri kaybolmakta ya da patlamaktadır. Bu değerlerdeki aşırı artışlar ve azalışlar isteyen bir diğer sorun olmaktadır. ResNet modelleri, bu soruna, ağın önceki katmanlarındaki girdi değerlerini, takip eden katmanların girdi değerlerine ekleyerek önceki sinyallerin korunmasını amaçlayan bir yöntemle çözüm bulmaya çalışmaktadır. Bunun için 2 ya da 3 evrişim katmanı sonrasına, mevcut katmanın girdi değerleri eklenerek ilgili katmanın girdi değerleri elde edilmektedir. Bu katmanlardan oluşan yapıya artık blok, önceki katmanlardan gelen bilgiyi sonraki katmana ileten bağlantıya kısa yol bağlantısı denilmektedir. Bu blokların özel bir çeşidi de darboğaz artık bloğu olmaktadır. Sırasıyla 1x1, 3x3, 1x1 evrişim katmanlarının bağlanması ve ilk 1x1 katmanına ait girdi değerlerinin sondaki 1x1 katmanına ait çıkış değerlerine eklenmesiyle elde edilmektedir. ResNet modellerinin, 50,101 ve 152 katmanlı modellerinde darboğaz artık blokları kullanılmaktadır. ResNet modellerinde, evrişim katmanları arasında yığın normalizasyon katmanı kullanılmaktadır. Bu çalışmaların, birbirinden yararlanma konusunda oldukça hızlı olduklarını göstermektedir. Bu tez çalışmasında ResNetV2 modelleri kullanılmakta, ResNetV1 modelleri kullanılmamaktadır. ResNetV1 ve ResNetV2 modellerine ait blok yapıları Şekil 2.11'de gösterilmektedir.



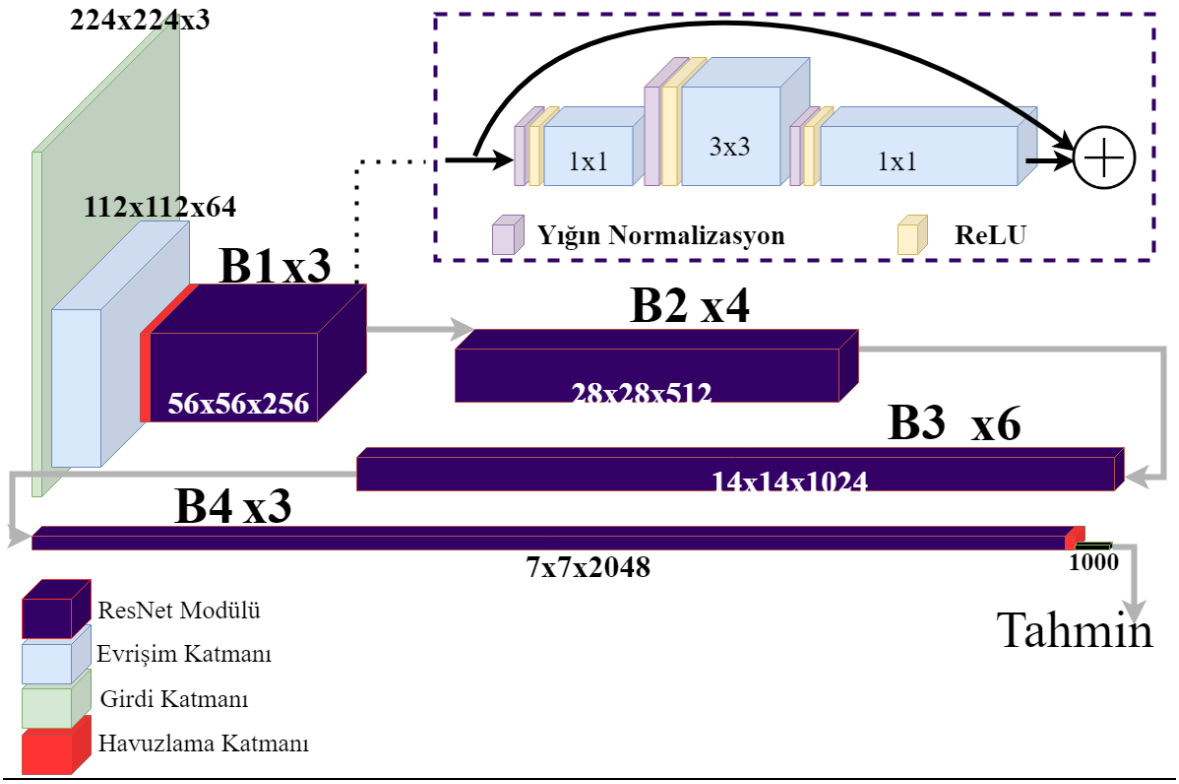
Şekil 2.11. ResNet darboğaz bloğu sürüm farkları

Şekil 2.11'de gösterildiği üzere ResNetV2'de blok yapılarında, yığın normalizayon ve relu aktivasyon katmanı evrişim katmanının önüne alınmaktadır. Bu duruma erken aktivasyon (pre-activation) denilmektedir. Önceki katmandan gelen sinyalin, sonraki katmanlara iletilmesi ok işaretiyle gösterilmektedir. Gelen sinyalle, çıkan sinyal toplanarak birleştirilmektedir, bu durum artı işaretiyle gösterilmektedir.

ResNet mimarileri basitçe dört adet bloktan oluşmaktadır. ResNet blokları birbirlerine ardışık şekilde bağlı ve tekrar eden yapılar çizerler. Katman sayıları arttıkça birinci ve dördüncü bloğun tekrar sayıları değişmemekte, ikinci ve üçüncü bloğun tekrar sayıları değişmektedir. Bir bloktan, diğer bloğa ilk kez geçiliyorsa ilk evrişim katmanında iki kaydırma uygulanır, bu sayede boyut yarıya indirilir. Bloklar arası geçişlerde özellik haritası sayıları da iki katına çıkmaktadır.

ResNet50V2 evrişimsel sinir ağı modeli

Resnet50V2 modeline ait mimari gösterim Şekil 2.12'de sunulmaktadır.

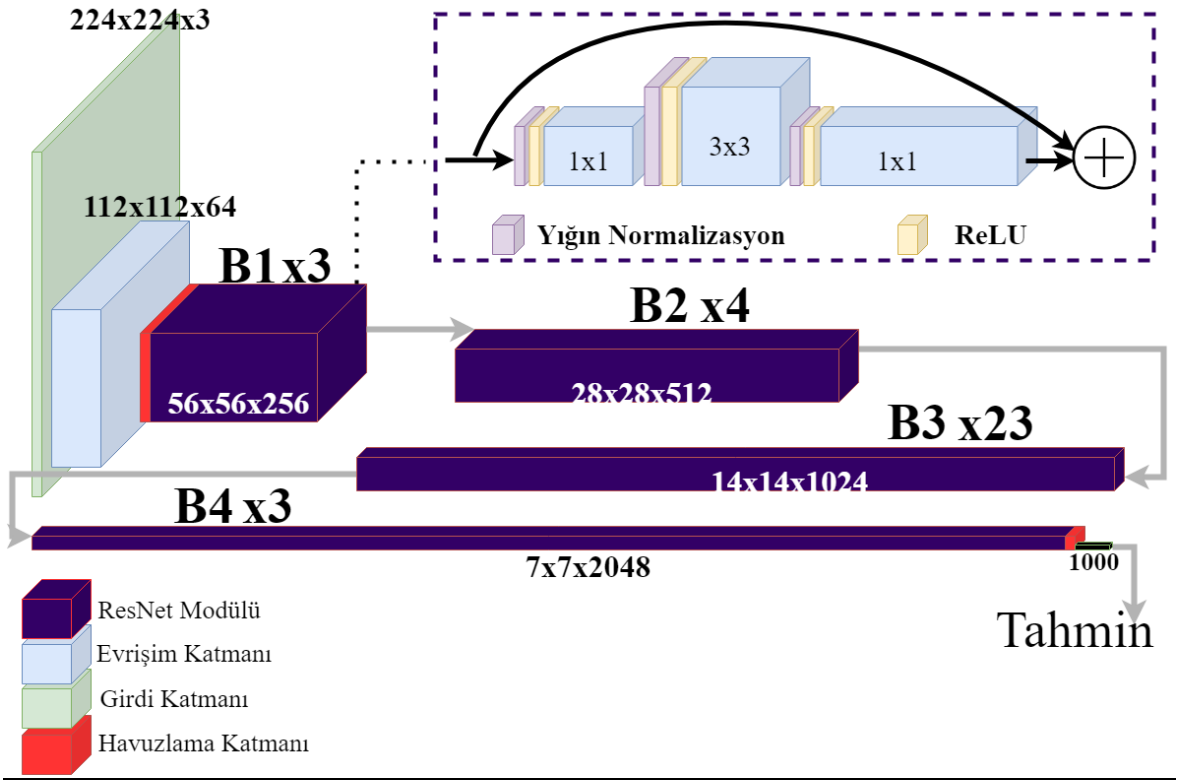


Şekil 2.12. ResNet50V2 modeline ait mimari gösterim

Şekil 2.12'de gösterildiği üzere ResNet50V2 modeli dört blok ve 50 katmandan oluşmaktadır. Bloklar B1, B2, B3, B4 şeklinde ifade edilmektedir. B1 bloğundan 3 adet, B2 bloğundan 4 adet, B3 bloğundan 6 adet ve B4 bloğundan 3 adet olmak üzere toplam 16 adet birbirine bağlı blok bulunmaktadır. Her blokta 3 adet evrişim katmanı olduğundan, toplam 48 adet evrişim katmanı bloklarda bulunmaktadır. Giriş katmanını takip eden 7x7 boyutlarında, 2 kaydırmalı evrişim katmanıyla, çıkışta bulunan 1000 nöronlu tam bağlı katmanla birlikte model de toplam 50 adet katman bulunmaktadır. Bir bloktan diğerine geçerken girdi boyutları yarıya düşmekte, özellik haritası sayısı iki katına çıkmaktadır.

ResNet101V2 evrişimsel sinir ağı modeli

Resnet101V2 modeline ait mimari gösterim Şekil 2.13'te gösterilmektedir.

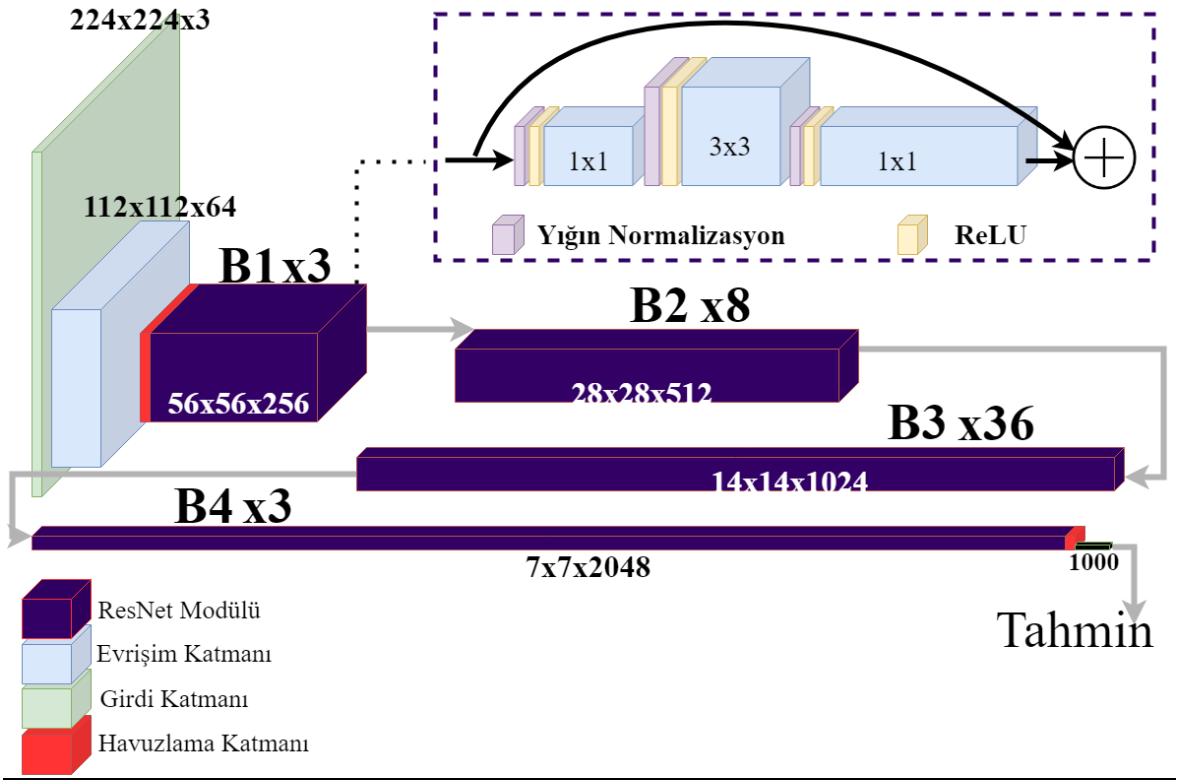


Şekil 2.13. ResNet101V2 modeline ait mimari gösterim

Şekil 2.13'te gösterildiği üzere ResNet101V2 modeli dört blok ve 101 katmandan oluşmaktadır. Bloklar B1, B2, B3, B4 şeklinde ifade edilmektedir. B1 bloğundan 3 adet, B2 bloğundan 4 adet, B3 bloğundan 23 adet ve B4 bloğundan 3 adet olmak üzere toplam 33 adet birbirine bağlı blok bulunmaktadır. Her blokta 3 adet evrişim katmanı olduğundan, toplam 99 adet evrişim katmanı bloklarda bulunmaktadır. Giriş katmanını takip eden 7x7 boyutlarında, 2 kaydırmalı evrişim katmanıyla, çıkışta bulunan 1000 nöronlu tam bağlı katmanla birlikte model de toplam 101 adet katman bulunmaktadır. Bir bloktan diğerine geçerken girdi boyutları yarıya düşmekte, özellik haritası sayısı iki katına çıkmaktadır.

ResNet152V2 evrişimsel sinir ağı modeli

Resnet152V2 modeline ait mimari gösterim Şekil 2.14'te sunulmaktadır.

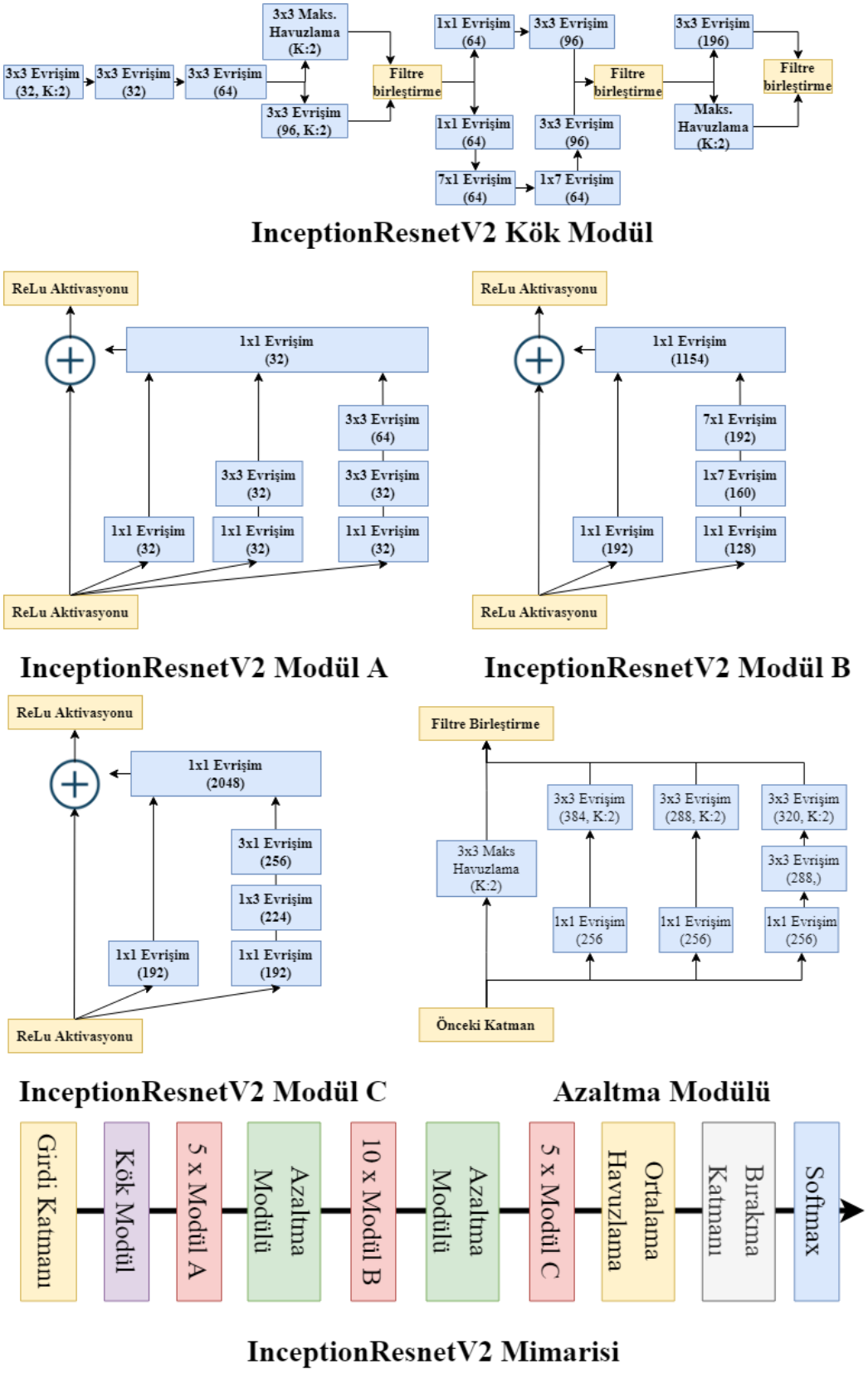


Şekil 2.14. Resnet152V2 modeline ait mimari gösterim

Şekil 2.14'te gösterildiği üzere ResNet152V2 modeli dört blok ve 152 katmandan oluşmaktadır. Bloklar B1, B2, B3, B4 şeklinde ifade edilmektedir. B1 bloğundan 3 adet, B2 bloğundan 8 adet, B3 bloğundan 36 adet ve B4 bloğundan 3 adet olmak üzere toplam 50 adet birbirine bağlı blok bulunmaktadır. Her blokta 3 adet evrişim katmanı olduğundan, toplam 150 adet evrişim katmanı bloklarda bulunmaktadır. Giriş katmanını takip eden 7x7 boyutlarında, 2 kaydırmalı evrişim katmanıyla, çıkışta bulunan 1000 nöronlu tam bağlı katmanla birlikte model de toplam 152 adet katman bulunmaktadır. Bir bloktan diğerine geçerken girdi boyutları yarıya düşmekte, özellik haritası sayısı iki katına çıkmaktadır.

2.5.6. InceptionResNetV2 evrişimsel sinir ağı modeli

InceptionResNetV2 [19], Inception ve ResNet modellerini tasarımsal açıdan birleştirmekte ve Inception modellerinin sunduğu işlem etkinliğiyle, ResNet modellerinin sunduğu sinyallerin daha derin katmanlara aktarılması yaklaşımını birlikte kullanmaktadır. Modele ait mimari gösterim ve mimaride kullanılan InceptionResNetV2 modülleri Şekil 2.15'te sunulmaktadır.

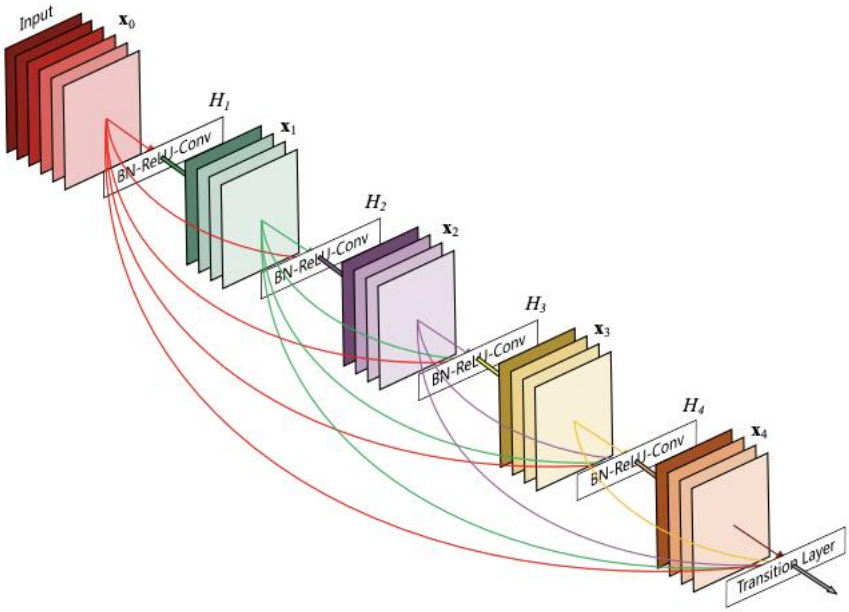


Şekil 2.15. InceptionResNetV2 modeline ait mimari gösterim [19]

Şekil 2.15'te gösterildiği üzere InceptionResnetV2 modeli kök modül, modül A, modül B, modül C ve azaltma modülü olmak üzere 5 adet modüle sahiptir. Mimaride giriş değerlerine sahip girdi katmanından sonra kök modül gelmektedir. Kök modülden sonra A modülü 5 kez, B modülü 10 kez, C modülü 5 kez tekrar etmekte ve bu modüller arasında azaltma modülü bulunmaktadır. Son C modülünden sonra, çıkış değerleri ortalama havuzlama katmanıyla azaltılmakta, bırakma katmanıyla bir kısmı atılmakta ve sınıflandırıcı katmanla sınıflandırılmaktadır. Önceki katmandan gelen sinyaller okla, sinyallerin toplanarak birleştirilmesi işlemi artı sembolüyle gösterilmektedir. Parantez içerisinde filtre sayıları gösterilmektedir.

2.5.7. DenseNet evrişimsel sinir ağı modelleri

DenseNet [20], ResNet mimarisinde sunulan birim sinyallerin ileriki katmanlara aktarılması yaklaşımını ileri götürerek, önceki katmanların ürettiği her sinyalin, sonraki katmanlara aktarılması yaklaşımını getirmektedir. DenseNet modeline ait mimari yaklaşım Şekil 2.16'da gösterilmektedir.



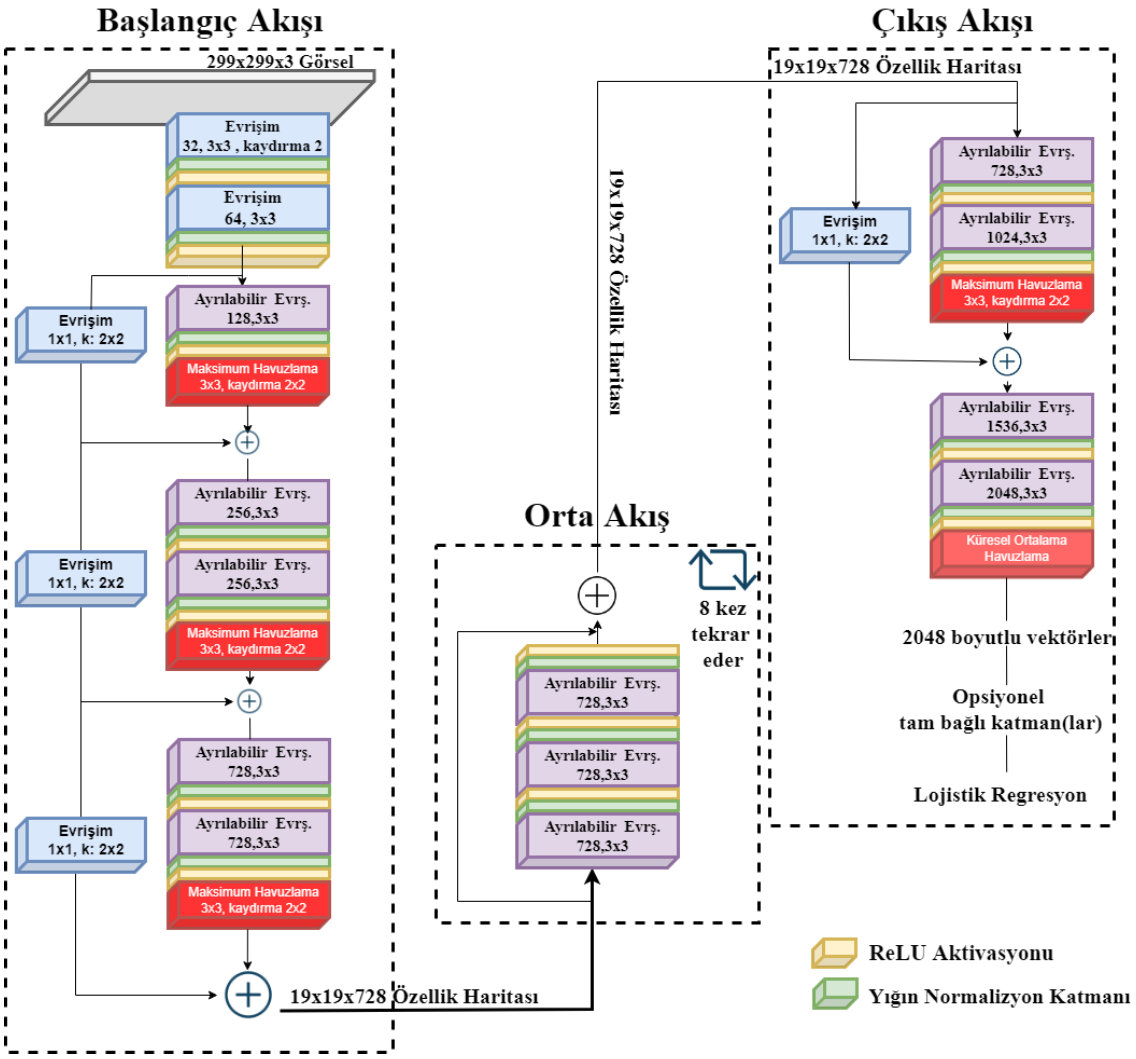
Şekil 2.16. DenseNet modelinde katmanlar arası bağlantılar [20]

Şekil 2.16'da görüldüğü üzere DenseNet ağlarında katmanlar ve önceki katmanlar arasında çok sayıda bağlantı kurulmaktadır. Bağlantılar okla gösterilmektedir. Önceki katmanların

çıktı değerleri, sonraki katmanların girdi değerlerini oluşturmaktadır. Nöronlar arasında çok sayıda bağlantı kurulduğundan, bu yapıya İngilizce yoğun, sık anlamları taşıyan Dense ismi verilmektedir. Bu yoğun bağlantıları barındıran katmanlara, blok ismi verilmektedir. Bu bloklar, birbirlerini takip eden şekilde birleştirilerek DenseNet modelleri oluşturulmaktadır.

2.5.8. Xception evrişimsel sinir ağı modeli

Xception, [21] ayrılabilir evrişim işleminin başarılı ilk uyarlamalarından biri olarak önemli bir model olmaktadır. Temel olarak Inception modellerinin ortaya koyduğu evrişim işlemlerinin küçük parçalara bölünerek elde edilen işlem tasarrufuna ve ayrılabilir evrişim katmanlarının uyarlanmasına ve ResNet mimarilerinin ortaya koyduğu sinyali ileri katmanlara iletme yaklaşımına dayanmaktadır. Xception modeline ait mimari yapı Şekil 2.17'de gösterilmektedir.



Şekil 2.17. Xception modeline ait mimari gösterim

Şekil 2.17'de gösterildiği üzere Xception modelin başlangıç, orta ve çıkış olmak üzere üç çeşit akış bulunmaktadır. Xception, mimari yapılarına "akış" ismini vermektedir. Orta akış katmanı 8 kez tekrar etmektedir, 8 kez art arda bağlanmaktadır şeklinde de ifade edilebilir. Akış içerisindeki her katman bir blok olarak gösterilmektedir. Blokların üzerinde katman adı, filtre sayısı ve filtre büyüklüğü belirtilmektedir. Kaydırma işlemi (k) uygulanıyorsa ilgili blokta belirtilmektedir.

Xception modelinde, derinlemesine ve noktasal olmak üzere iki tür ayrılabilir evrişim katmanı bulunmaktadır. Görsel sınıflama çalışmalarında bir fotoğraf en, boy ve kanal sayısı (kırmızı, yeşil, mavi) olan matrisle ifade edilmektedir. Bir görsel üzerinde evrişim işlemi uygulandığında, bu işlem tüm boyutlarda gerçekleşmektedir. Örnek olarak 224x224 piksel boyutlarında bir renkli fotoğraf düşünüldüğünde, bu fotoğraf 224x224x3

boyutlarında matrisle ifade edilmektedir. Bu fotoğrafa 3x3 boyutlarında, kaydırması 1 olan normal evrişim filtresi uygulandığında, 222x222 boyutlarında bir özellik haritası elde edilmektedir. Bu özellik haritası oluşturulurken (3x3)x(222x222)x3 kadar çarpım işlemi gerçekleşmektedir. Bu işlem sadece tek bir evrişim filtresi içindir ve ilgili katmanda istenen filtre sayısı kadar çarpılması gerekmektedir. İşlem sırasında 100 adet filtre kullanıldığında 100x(3x3)x(222x222)x3=133.066.800 adet çarpım işlemi gerçekleştirilmesi gerekmektedir. Aynı işlem ayrılabilir evrişim katmanı kullanarak derinlemesine ve noktasal olmak üzere iki aşamada gerçekleştirilebilmektedir. Birinci aşama derinlik boyutu ayrılmakta ve 224x224x1 boyutlarında 3 adet matris üzerinde 3x3 evrişim filtresi uygulanmaktadır. Sonuç olarak 222x222x1 boyutlarında 3 adet matris elde edilmektedir. Bu aşamada toplam çarpım sayısı (222x222)x(3x3)x3=35.928.036 kadar olmaktadır. İkinci aşama da 1x1 boyutlarında 100 adet evrişim filtresi 222x222x3 boyutlarındaki matrise uygulanır. Bu aşamada toplam çarpım sayısı 100x(222x222x3)x(1x1)=14.785.200 kadar olmaktadır. Birinci ve ikinci aşama toplandığında 50.713.236 adet çarpım işlemi gerçekleştiği görülecektir. Sonuçta her iki örnekte de elde edilen çıktı 222x222x100 boyutlarında olmasına rağmen normal evrişimdeki çarpım işlemi, ayrılabilir evrişim yöntemindekine oranlandığında (133.066.800/50.713.236), normal evrişim işleminde yaklaşık 2,623 kat fazla çarpım işlemi gerçekleştiği görülmektedir.

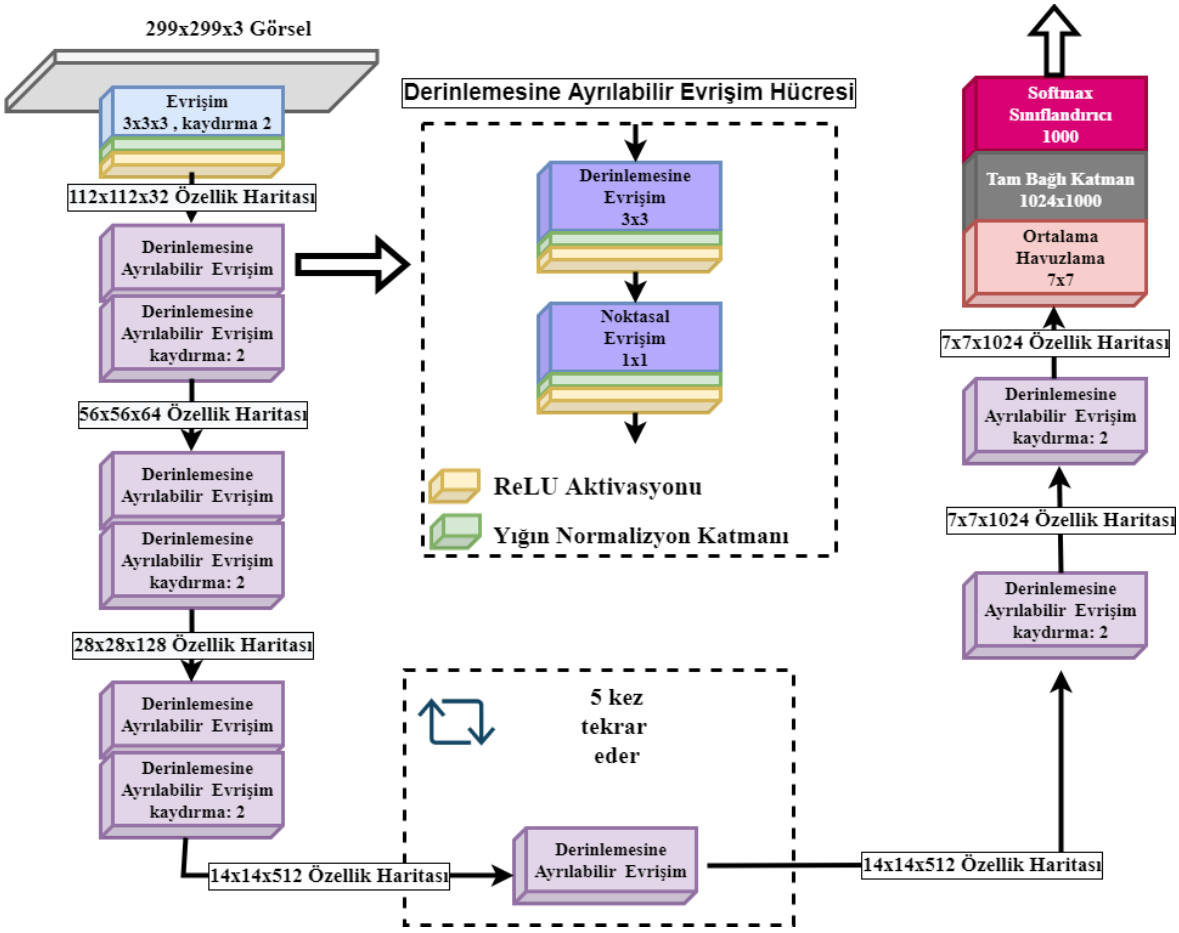
Xception modelinde tam bağlı katmanlar opsiyoneldir ve orijinal çalışmada softmax sınıflandırıcı yerine lojistik regresyonla sınıflandırma gerçekleştirilmektedir. Bu tez çalışmasında, softmax sınıflandırıcı ve tam bağlı katmanlar kullanılmaktadır.

2.5.9. MobileNet evrişimsel sinir ağı modelleri

MobileNet ailesi, mobil cihazlarda da derin öğrenme işlemlerinin gerçekleştirilebilmesi için, küçük ve efektif modeller barındırmaktadır. Ailenin her yeni sürümünde yenilikçi yaklaşımlar denenmekte ve model performansları arttırılmaktadır. İzleyen bölümlerde, MobileNetV1, MobileNetV2 ve MobileNetV3 modelleri ve kullandıkları yenilikler aktarılmaktadır.

MobileNetV1 evrişimsel sinir ağı modeli

MobileNetV1 [22] modeli, MobileNet ailesinin ilk modeli olmaktadır. MobileNetV1'de 1x1 noktasal evrişim işleminin gerçekleştirimine odaklanılmaktadır. Bunun için noktasal çarpım işlemlerini çok daha etkili yapan yazılım kütüphanelerinden yararlanılmakta ve performans artışı sağlanmaktadır. MobileNetV1 modeline ait mimari gösterim Şekil 2.18'de sunulmaktadır.

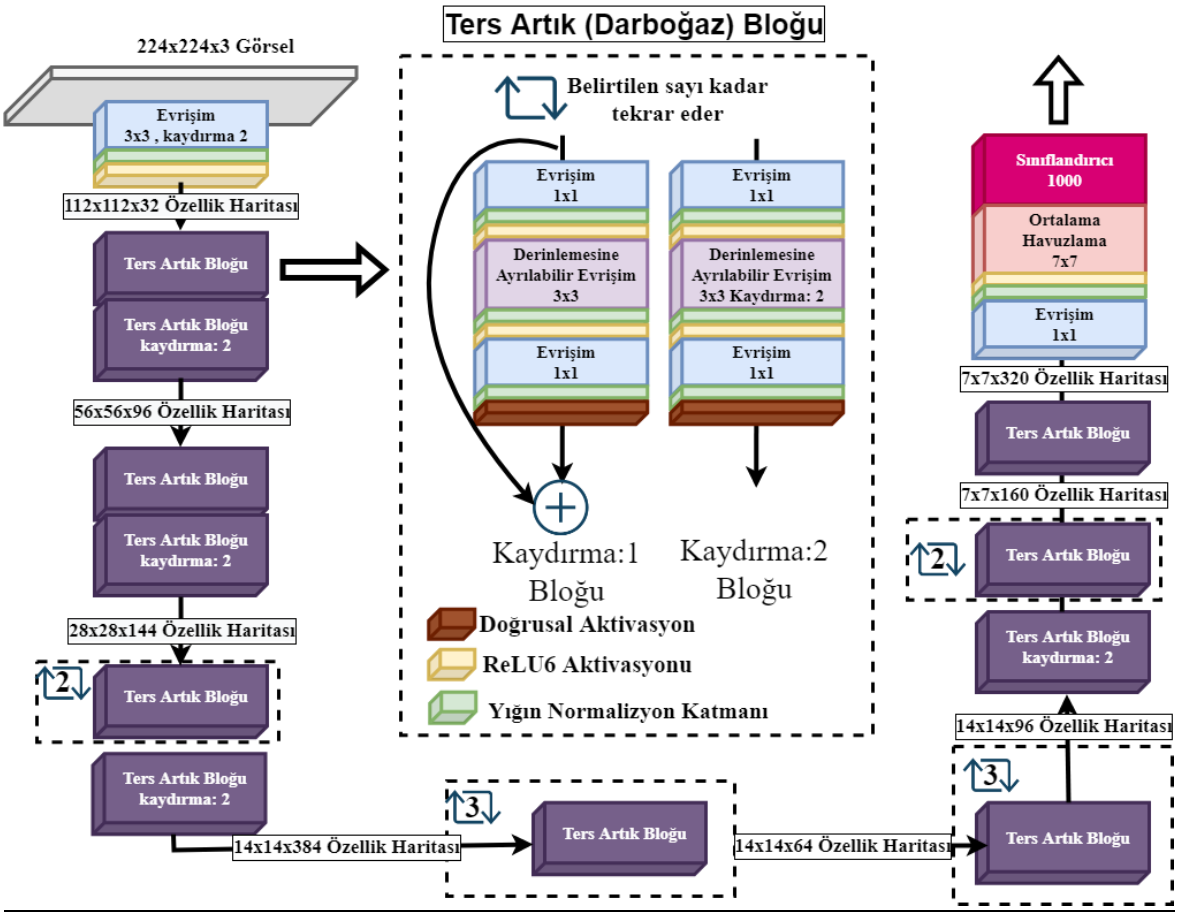


Şekil 2.18. MobileNetV1 modeline ait mimari gösterim

Şekil 2.18'de gösterildiği üzere MobileNetV1 modeli evrişim katmanı, tekrar eden derinlemesine ayrılabilir evrişim hücrelerinden, ortalama havuzlama katmanından, tam bağlı katmandan ve sınıflandırıcıdan oluşmaktadır. Derinlemesine evrişim hücresinde, 3x3 derinlemesine evrişim işleminin ardından, 1x1 noktasal evrişim gerçekleştirilmektedir. Evrişim katmanlarını ReLu aktivasyon katmanı ve yığın normalizasyon katmanları izlemektedir. Kaydırmalı evrişim hücrelerinde, 2 kaydırma işlemi uygulanarak girdi boyutu yarıya indirilmektedir.

MobileNetV2 evrişimsel sinir ağı modeli

MobileNetV2 [23] ters artık bloğu, ReLu6 aktivasyon fonksiyonu olmak üzere yeni yaklaşımları ve yapıları önceki mimariye (MobileNetV1) uygulamaktadır. Modelin sahip olduğu yenilikler bu bölümde açıklanmaktadır. Modele ait mimari yapı ve ters artık darboğaz bloğu Şekil 2.19'da gösterilmektedir



Şekil 2.19. MobileNetV2 modeline ait mimari gösterim

Şekil 2.19'da gösterildiği üzere MobileNetV2 modeli ters artık bloğu denen yapıların tekrarlarından oluşmaktadır. Ters artık bloğun, ResNet mimarisindeki artık blok yaklaşımının tersi şeklinde olduğu görülmektedir. ResNet artık bloklarında, girişteki ve çıkıştaki kanal sayısı ara katmanlardaki kanal sayısından fazlayken, ters artık blok yaklaşımında, blok içerisindeki kanal sayısı azaltılmak yerine arttırılarak tersine çevrilmektedir. Ters artık bloklar içerisinde, kanal sayısı arttırılmaktadır. Arttırma miktarı, genişleme faktörü olarak adlandırılan bir değere göre yapılmaktadır. Bu değer 6 belirlendiği durumda, blok girişinde 32 kanal ve blok çıkışında 64 kanal olduğunda, blok

içinde 32x6=192 kanala çıkartılmaktadır. Sadece doğrusal aktivasyon fonksiyonu barındıran artık bloktan farklı olarak, ters artık blok doğrusal ve doğrusal olmayan aktivasyon fonksiyonları barındırmaktadır. MobileNetV2'de kullanılan ReLU6 aktivasyon fonksiyonu, değerleri 0 ve 6 arasında sınırlamaktadır, bu sayede aktivasyon fonksiyonu en yüksek 6 değerini verebilmektedir. Ters artık bloğun kaydırmalı ve normal olmak üzere iki şekli bulunmaktadır. Normal blokta, girişteki sinyaller üç katman sonraya iletilmektedir. Kaydırma işlemi (k) uygulanan blokta sinyal iletilmemekte ve kaydırma işlemi sonunda girdi boyutu yarıya düşürülmektedir.

MobileNetV3 evrişimsel sinir ağı modeli

2019 yılında sunulan MobileNetV3 modeli mimarisi nöral ağ arama algoritmalarıyla oluşturulmuştur. NasNet modeline benzer arama algoritması kullanılan çalışmada, farklı olarak ödül fonksiyonunu sadece doğruluk yerine, doğruluk ve gecikmenin bir oranı olarak belirlenmektedir. Bu sayede, oluşturulan mimarinin, gecikmeyi de dikkate alması istenmektedir. MobileNet ailesinin işlem kapasiteleri düşük ya da mobil cihazları hedefledikleri düşünüldüğünde bu durum oldukça anlamlı bulunmaktadır. Daha düşük donanımlı cihazlar için kullanılmak üzere tasarlanan MobileNetV3Small ve kapasitesi daha yüksek cihazlar için tasarlanan MobileNetV3Large olarak iki farklı model bulundurmaktadır.

MobileNetV3 modelinde sıkma ve heyecanlandırma (İngilizce squeeze and excitation) olarak adlandırılmakta olan bir yaklaşım, ters darboğaz bloklarıyla birlikte kullanılmaktadır. Sıkma ve heyecanlandırma işleminde özellik haritalarının kanal bazlı tekrar kalibrasyonu hedeflenmektedir. Bunun için girdi değerlerine küresel ortalama havuzlama işlemi uygulanmaktadır. Ardından filtre sayısıyla sıkma katsayısının çarpımı kadar azaltılan sayıda nöron barındıran tam bağlı katman kullanılmakta, bu katmanı takip eden diğer tam bağlı katmandaysa başlangıçtaki filtre sayısı kadar nöron bulunmaktadır. İlk tam bağlı katmanda ReLu aktivasyon fonksiyonu kullanılmaktayken, ikinci tam bağlı katmanda, daha yumuşak değerler elde etmek için Sigmoid aktivasyonu tercih edilmektedir. Sigmoid katmanından elde edilen değerlerin, girdi katmanıyla çarpılmasıyla (Hadamard çarpımı) çıkış değerleri üretilmektedir. MobileNetV3 çalışmasında, Sigmoid fonksiyonu yerine sert Sigmoid fonksiyonu adını verdikleri bir fonksiyon tercih etmektedirler. MobileNetV2'de kullandıkları ReLu6 aktivasyon fonksiyonunun girdi

değerine 3 eklenmesi ve çıkan sonucun 6'ya bölünmesiyle sert Sigmoid aktivasyon fonksiyonu hesaplanmaktadır. Sigmoid yerine sert Sigmoid kullanmalarındaki temel neden hesaplama karmaşıklığının azaltılması olmaktadır. Modelde, katmanlar boyunca aktivasyon fonksiyonu olarak, Swish fonksiyonun bir uyarlaması olan ve sert Swish olarak adlandırdıkları bir aktivasyon fonksiyonu kullanılmaktadır. Swish fonksiyonunda girdi değerleri, girdi değerlerinin Sigmoid fonksiyon değerleriyle çarpılmaktayken, sert Swish aktivasyonunda, Sigmoid yerine sert Sigmoid kullanılmaktadır. Modelin sığ katmanlarında ReLu aktivasyonu, derin katmanlarında sert Swish aktivasyonu kullanılmaktadır.

2.5.10. NASNet evrişimsel sinir ağı modelleri

Son çalışmalarda, etkinliği yüksek küçük mimari yapılarla, bunların ilgili soruna karşı ölçeklenmesinin öneminin arttığı görülmektedir. Yazarlar, bu sorunu gene bir modele çözdürmeyi düşünerek, oluşturdukları ESA üreten modelle, çözülmeye çalışılan soruna, uygun mimariyi oluşturan bir yaklaşım geliştirmişlerdir. İnsan üretimi çalışmalarda belli yapısal parçaların öne çıktığını gören NASNet [25] geliştiricileri, bu yapısal modellerden oluşan bir havuz hazırlayarak bu havuza, NASNet arama uzayı adını vermişlerdir. Derin öğrenme yaklaşımları olan tekrarlayan sinir ağı (İng. recurrent neural network, kısaca TSA) ve pekiştirmeli öğrenme modelini birlikte kullanarak, bu havuzdaki yapılarla, modelin yeni mimari yapılar oluşturmalarını sağlamışlardır. Model çalışmasının yapıldığı yıllarda, genetik algoritmalar gibi farklı yaklaşımlar kullanarak, yeni mimarileri makinalara keşfettirme çalışmaları olmasına karşın, yazarların oluşturdukları havuz ile oldukça büyük olan arama uzayını, daha küçük bir arama uzayına dönüştürdükleri görülmektedir. Bu havuzun içinde bulunan yapı taşları, önceki çalışmalarda öne çıkan küçük ama etkili hücresel yaklaşımlardan oluşmaktadır. Bunlar, birim aktarma, 1x3 sonra 3x1 evrişim, 1x7 sonra 7x1 evrişim, 3x3 genişlemiş evrişim, 3x3 ortalamaya göre havuzlama (average pooling), 3x3 maksimuma göre havuzlama (max pooling), 5x5 maksimuma göre havuzlama, 7x7 ortalamaya göre havuzlama, 1x1 evrişim, 3x3 evrişim, 3x3 derinlemesine ayrılabilir evrişim, 5x5 derinlemesine ayrılabilir evrişim, 7x7 derinlemesine ayrılabilir evrişim gibi yapısal öğelerden oluşmaktadır.

2.5.11. EfficientNet evrişimsel sinir ağı modelleri

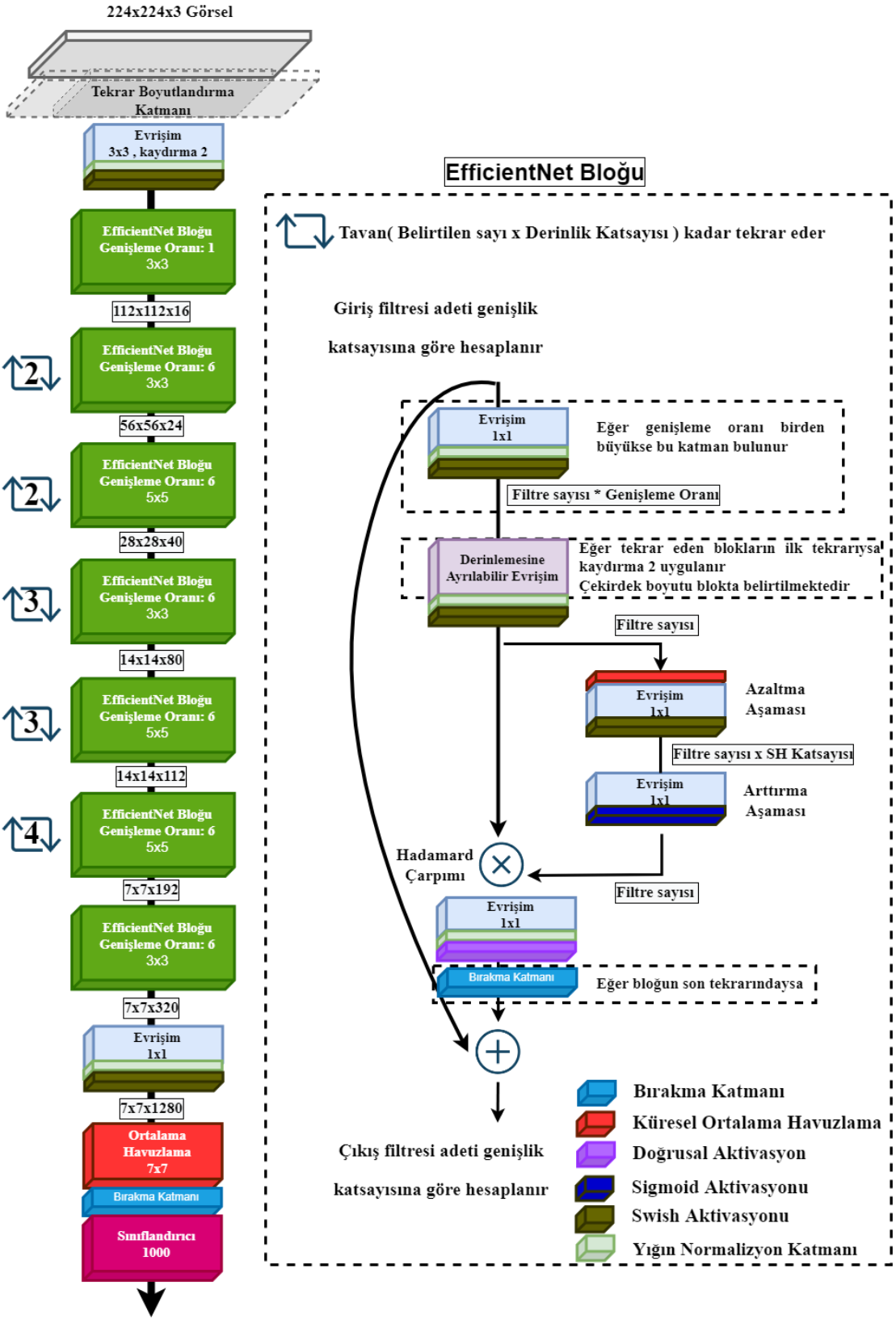
Ölçeklendirme üzerine olan çalışmaların özellikle son yıllarda ortaya konan mimari modellerde arttığı görülmektedir. MobileNet, NASNet gibi mimarilerde ortaya konan modelin ölçeklendirilmesi çalışmalarını, EfficientNet [26] model ailesi de takip etmektedir. EfficientNet, temel olarak derinlik, genişlik ve çözünürlük olmak üzere üç ölçeklendirme parametresi önermektedir. Çizelge 1.1'de modellere ait katsayı değerleri sunulmaktadır.

Çizelge 1.1. EfficientNet modelleri ve katsayı değerleri

Model	Genişlik Katsayısı	Derinlik Katsayısı	Çözünürlük	Bırakma Oranı %
B0	1	1	224	20
B1	1	1,1	240	20
B2	1,1	1,2	260	30
B3	1,2	1,4	300	30
B4	1,4	1,8	380	40
B5	1,6	2,2	456	40
B6	1,8	2,6	528	50
B7	2	3,1	600	50

Çizelge 1.1'de EfficientNet ailesinde bulunan 8 modelin genişlik, derinlik, çözünürlük ve bırakma oranı olmak üzere parametre değerleri sunulmaktadır. Derinlik modelin katman sayısını, genişlik her katmandaki kanal sayısını ve çözünürlükte modelin girdi katmanına sunulan matris boyutlarını ifade etmektedir. Bırakma oranı, bırakma katmanlarında kullanılan oranı ifade etmektedir. Model büyüdükçe bırakma oranı da artmaktadır. Geliştiriciler, bu parametreleri, önerdikleri formülle belli bir orana göre sabitlemişlerdir. Bu üç parametreden biri değiştiğinde, kalan parametrelerde bu oranı verecek şekilde tekrardan ölçeklendirilmektedir. EfficientNetB0 modeli temel model olup genişlik katsayısı 1, derinlik katsayısı 1, çözünürlük 224 ve bırakma oranı 0,20 olarak belirlenmiştir. EfficientNetB1 modelinde, genişlik katsayısı 1, derinlik katsayısı 1,1, çözünürlük 240 ve bırakma oranı 0,20 olarak belirlenmiştir. EfficientNetB2 modelinde, genişlik katsayısı 1,1, derinlik katsayısı 1,2, çözünürlük 260 ve bırakma oranı 0,30 olarak belirlenmiştir. EfficientNetB3 modelinde, genişlik katsayısı 1,2, derinlik katsayısı 1,4, çözünürlük 300 ve bırakma oranı 0,30 olarak belirlenmiştir. EfficientNetB4 modelinde, genişlik katsayısı 1,4, derinlik katsayısı 1,8, çözünürlük 380 ve bırakma oranı 0,40 olarak belirlenmiştir. EfficientNetB5 modelinde, genişlik katsayısı 1,6, derinlik katsayısı 2,2,

çözünürlük 456 ve bırakma oranı 0,40 olarak belirlenmiştir. EfficientNetB6 modelinde, genişlik katsayısı 1,8, derinlik katsayısı 2,6, çözünürlük 528 ve bırakma oranı 0,50 olarak belirlenmiştir. EfficientNetB7 modelinde, genişlik katsayısı 2, derinlik katsayısı 3,1, çözünürlük 600 ve bırakma oranı 0,50 olarak belirlenmiştir. Şekil 2.20'de modellere ait genel mimari gösterimi ve blok yapısı sunulmaktadır.



Şekil 2.20. EfficientNet modellerine ait mimari gösterim

Şekil 2.20’de gösterildiği üzere EfficientNet modelleri tekrar boyutlandırma katmanı, evrişim katmanları, küresel ortalamaya göre havuzlama katmanı, bırakma katmanı, sınıflandırıcı katman ve 7 adet birbirine bağlı tekrar eden EfficientNet bloklarından oluşmaktadır. EfficientNet modellerinde, girdi katmanını takip eden ve çözünürlüğü modelin ayarlarına göre tekrardan boyutlandıran bir katman bulunmaktadır. Bu sayede model girdi değerlerinden arındırılmakta ve katsayı oranları korunabilmektedir. Modellerde, derinlik katsayısına göre tekrar sayıları değişen 7 blok bulunmaktadır. Bu bloklar birbirine bağlıdır ve derinlik katsayısıyla, tekrar sayısının çarpımının bir üst sayıya yuvarlanmasıyla hesaplanan sayı kadar tekrar etmektedir. Bloklarda, girişteki filtre sayısıyla, çıkıştaki filtre sayısı genişlik katsayısı ile hesaplanmaktadır. Blok içerisinde genişleme oranı kadar filtre sayısı arttırılmaktadır. Blok içerisinde sıkıştırma ve heyecan katmanları kullanılmaktadır.

Modelde katmanlar boyunca aktivasyon fonksiyonu olarak Swish aktivasyonu kullanılmaktadır. Sıkıştırma ve heyecan katmanlarında Sigmoid aktivasyonu kullanılmaktadır.

3. İLGİLİ ÇALIŞMALAR

Bu bölümde çalışmalar, gerçekleştirilen çalışmaya benzerlik düzeylerine göre ilgili başlıklar altında aktarılacaktır. İlk başlıklarda, alanla ilgili ilk, kök çalışmalar yer alırken, yapılan çalışmaya en yakın çalışmalar bölümün son başlığında yer alacaktır. İlgili çalışmaların tespitinde Web Of Science ve Scopus akademik veri tabanları kullanılmaktadır.

3.1. Kötücül Yazılım Analizi

Mobil kötücül yazılım analizi, mobil uygulamaları, çeşitli tekniklerle inceleyen, kötücül uygulamaları tespit etmeyi ve bunları sınıflandırmayı amaçlayan bir çalışma alanıdır. Bir kötücül yazılım analisti tarafından manuel ya da çeşitli bilgisayar programları ile otomatik şekilde gerçekleştirilebilir. Manuel analiz, en etkili yöntem olmakta birlikte, analizi gerçekleştiren uzmanın bilgi düzeyine göre başarısı değişmekle birlikte, oldukça zaman alan bir çalışmadır. Bu sebeple, süreci tamamen otomatikleştirecek ya da manuel analiz yapan uzamının, şüpheli alanları tespit etmesine yardımcı olacak yeni yöntemlere ihtiyaç duyulmaktadır. Geliştirilen yöntemler ve kötücül analiz süreci, statik analiz, dinamik analiz ve her ikisinin birlikte kullanıldığı karma analiz başlıklarında ele alınmaktadır.

Derin öğrenme ve makine öğrenmesi çalışmalarında, bu analizler veri setlerindeki normal ve kötücül yazılımlar üzerinde uygulanırlar. Veri hazırlama süreci olarak da ifade edilen bu süreç sonunda, ham uygulama dosyalarının işlenmesi sonucu elde edilen veriler, derin öğrenme modellerinin giriş değerlerini oluşturur. Bu aşamada kullanılan teknikler ve yöntemler model başarısını etkileyecektir.

3.1.1 Statik analiz

Statik analiz, uygulama dosyaları çalıştırılmadan gerçekleştirilen analizdir. Bu analiz çalışmalarından en basiti, uygulama dosyasının, SHA, MD5 gibi bir özetleme (hash) algoritmasıyla, belli bir uzunlukta bir karakter dizisiyle ifade edilmesidir. Bu karakter dizisi her dosya için benzersizdir ve çok hızlı çalışır. Bu yöntem imza tabanlı analiz olarak adlandırılmakta ve virüs tarama uygulamaları tarafından yaygın şekilde kullanılmaktadır.

Bu yöntemin çalışabilmesi için, kötücül dosyanın imza tabanı olarak adlandırılan veri tabanına daha önceden kaydedilmiş olması gerekmektedir. Dosyada yapılacak en ufak değişiklik, dosyanın imzasını da değiştireceğinden bu yöntem tek başına kullanılmamalıdır.

İmza tabanlı analize benzer bir başka statik analiz yöntemiyse, uygulama dosyası içerisinde kötücül yazılımlara ait kod parçalarını bulmaktadır. Düzgün çalışan bir uygulama dosyasına, enjekte edilen kötücül kodlar taranarak tespit edilir. Bu işlemde daha önceden tespit edilen kötücül kod sekanslarını tespit edebilecek bir regex sorgusu hazırlanır. Bu regex sorgusu kötücül kodun imzası görevini görür. Diğer imza bazlı yönteme göre daha etkili bir yöntem olsa da günümüzde geliştirilen karşıt tekniklerle, bu yöntem de etkisiz kalmaktadır [29,30]. Statik analiz çalışmalarını engellemek için, zararlı yazılım geliştiricileri tarafından çeşitli teknikler kullanılmaktadır. Bunlar genel olarak uygulamanın kod dizilimleri ve değişken isimlendirmelerini değiştirmeye dayanan kod karıştırma, uygulama içerisine kriptolanmış şekilde kodlar yerleştirme ve çalışma zamanında çözmeye dayanan kod şifreleme, uygulama içerisine çalışma zamanında internet üzerinden ya da uygulama paketinin içerisinde bulunan dosyadan olacak şekilde kodlar, kütüphaneler yüklemeye dayanan dinamik kod yükleme yöntemlerini kullanabilmektedirler. Bir başka yöntemse, daha çok oltalamaya dayanan ve başta zararlı kod bulundurmayan uygulamada bir açılır uyarıyla (yeni bir güncelleme var) kullanıcının kendi kendine zararlı yazılımı kurmasına dayanan güncelleme saldırıları olmaktadır.

Karşıt analiz tekniklerinin oldukça gelişmesiyle, bu tekniklere karşı dayanıklı yeni statik analiz yöntemlerinin geliştirilmesi gerekmektedir. Derin öğrenme tabanlı çalışmalar bu konuda oldukça ümit vericidir. Derin öğrenme algoritmalarının giriş değerlerini oluşturacak bu veriler, çeşitli statik analiz araçlarıyla, uygulama dosyası çalıştırılmadan elde edilir.

3.1.2. Dinamik analiz

Uygulamaların uygun ortamda çalıştırılması sonucu oluşan dinamik verilerin toplanmasıyla gerçekleştirilen analizdir. Uygulamalar statik analiz atlama tekniklerini uygulayabildiklerinden, daha doğru sonuçlar için dinamik analiz gerçekleştirmek gerekebilmektedir. Uygulamaya ait ağ trafiği, log kayıtları, çalışma zamanında

gerçekleştirdiği sistem ve API çağrıları, olaylar dinamik analiz ile toplanır. Analiz için gerçek cihazlar ya da sanal emülasyon cihazlar kullanılır. Çoğunlukla zararlı yazılımlar belli bir süre ve belli bir girdi işleminden sonra zararlı davranışları sergilemeye programlandığından, sonuçları görmek için beklemek gerekebilir. Zararlı yazılımlar, sanallaştırma ve emülasyon ortamlarını fark edecek şekilde programlanabildiklerinden, gerçek cihazlar üzerinde çalıştırmak daha doğru sonuçlar almayı sağlayabilir. Zararlı yazılım çalıştırıldığı ortam yeterince yalıtılmamışsa, bağlı olduğu cihazlara zarar verebilir. Bu yüzden statik analize göre daha tehlikelidir.

3.1.3. Karma analiz

Karma analiz hem statik hem dinamik analiz ile elde edilen verilerin, birlikte değerlendirilmesidir. Statik analiz süreci sonunda şüpheli noktalar tespit edilip, dinamik analizle bu noktaların çalışırken nasıl davranışlar sergilediği araştırılır. Aynı zaman uygulamanın statik yapısıyla, dinamik yapısının ne kadar uyum sağladığı da bir ölçüt olabilmektedir. Uygulama kurulum sırasında istediği izinleri, çalışma zamanında kullanmıyorsa bu durum karma analiz ile tespit edilebilir. Dinamik analiz ve statik analize göre daha karmaşıktır.

3.1.4. Görsel analiz

Bir sorunun, görsel bir şekilde sunumu üzerinde gerçekleştirilen analizler, görsel analizlerdir. Ses dalgaları, müzik uygulamalarında sinüs dalgaları formunda görselleştirilmektedirler. Benzer şekilde bir osiloskop, elektrik sinyallerine ait verileri, çeşitli şekillerde ekranında gösterebilmekte, bu sayede alan uzmanları ekranda gördükleri verileri yorumlayabilmektedir. Benzer bir analiz kötücül yazılımlar içinde gerçekleştirilebilmektedir. Görsel kötücül analizi, kötücül yazılımlara ait dosyaların bir yazılım aracılığıyla görsellere dönüştürülerek analiz edilmesidir. Genellikle statik analiz olarak gerçekleştirilmektedir. İkili dosya kodu doğrudan kullanılabileceği gibi, dosyadan çeşitli yöntemlerle elde edilen yeni verilerin görselleştirilmesiyle de gerçekleştirilebilir. Uygulanışının diğer yöntemlere göre kolaylığı ve görseller üzerinde, kendi kendine örüntü yakalayan derin öğrenme uygulamalarına olanak vermesi sayesinde, giderek popülerlik kazanmaktadır. Diğer görsel sınıflandırma çalışmalarının, kötücül yazılımlara ait görselleri

de sınıflandırabilmesi ve bunu işletim sistemi bağımsız yapabilmesi sayesinde öğrenim aktarımına (transfer öğrenme) olanak sağlamaktadır.

3.2. Alanla İlgili Öncül Çalışmalar

Bu bölümde, Android kötücül yazılım analizi ve sınıflandırması gerçekleştiren öncül çalışmalar incelenmektedir. İncelenen çalışmalarda, makine öğrenmesi, derin öğrenme içeren çalışmalar ve bu yöntemleri içermeyen çalışmalar değerlendirilmektedir. ESA modellerini içeren çalışmalarda, uygulama dosyalarının görsellere dönüştürülmesiyle görsel analiz gerçekleştiriliyorsa, bu çalışmalar izleyen ilgili bölümde ele alınmaktadır.

Shabtai ve arkadaşlarının 2011 yılında gerçekleştirdikleri Andromaly adlı çalışmada [27], yazarlar makine öğrenmesi temelli bir anomali tespit sistemi sunmuşlardır. Çalışmanın gerçekleştirildiği sırada, Android işletim sistemine ait bir kötücül yazılım bulunmadığını belirten yazarlar, kendi geliştirdikleri dört adet kötücül yazılımı örnek olarak kullanmışlardır. Veri setinde 20 zararsız oyun, 20 zararsız araç ve 4 kötücül olmak üzere toplam 44 adet örnek bulunmaktadır. Çalışmada, makine öğrenmesi algoritmaları olan k-ortalamalar, lojistik regresyon, histogram, karar ağacı, bayes ağları ve saf bayes sınıflandırıcılarını kullanılmıştır ve en başarılı sonuçları, karar ağacı (J48), saf bayes algoritmaları göstermiştir.

Zhao ve arkadaşlarının 2011 yılında gerçekleştirdikleri AntiMalDroid adlı çalışmada [28], Google resmi uygulama marketinden topladıkları ve zararsız etiketledikleri 100 adet uygulamadan ve Geinimi, DroidDream, Plankton olmak üzere 3 aileye ait toplam 90 kötücül yazılım uygulamasından oluşan veri setinde, bir makine öğrenmesi algoritması olan destek vektör makineleri kullanarak ailesel sınıflandırma gerçekleştirmişlerdir. Çalışmada, uygulamaların Intent istekleri ve sistem kaynaklarına erişimleri kayıt altına alınarak, bu kayıtlardan uygulamalara davranış imzaları elde edilmiştir. Bu davranış imzalarına göre model eğitilmiştir. Sunulan model, Geinimi ailesine ait 30 örnekten 28'ini, DroidDream ailesine ait 30 örnekten, 27'sini ve Plankton ailesine ait 30 örnekten 27'sini doğru tespit ederek sırasıyla, %93,3, %90, %90 sınıflandırma doğruluğu göstermiştir.

Zhou ve Jiang tarafından 2012 yılında gerçekleştirilen ve MalGenome adlı veri setinin de oluşturulduğu çalışmada [29], Ağustos 2010 ve Ekim 2011 tarihleri arasında toplanan ve 49 aileye ait 1260 kötücül yazılım toplanmış ve piyasada bulunan popüler anti virüs yazılımlarıyla taranmıştır. Tarama için AVG, Lookout, Norton ve Trend Micro anti virüs araçlarını kullanan yazarlar, sırasıyla %54,7, %79,6, %20,2 ve %76,7 ikili sınıflandırma (tespit) performansı gözlemlemişlerdir. Yazarlar, sonuçların cesaret verici olmadığını belirtmişlerdir [29].

Arp ve arkadaşlarının 2014 yılında gerçekleştirdikleri ve Drebin adlı veri setinin de oluşturulduğu çalışmada [30], yazarlar, doğrudan cep telefonu üzerinde çalışabilen, statik analiz gerçekleştiren makine öğrenmesi temelli bir model geliştirmişlerdir. Veri setinde, Ağustos 2010 ile Ekim 2012 tarihleri arasında toplanan, 5560 kötücül yazılımla 123.453 zararsız yazılım bulunmaktadır. Veri setindeki kötücül örnekler, MalGenome veri setindeki tüm örnekleri de içermektedir. Yazarlar uygulamalara ait donanım bileşenleri, istenen izinler, uygulama bileşenleri, filtrelenmiş intentler, kısıtlanmış API çağrıları, kullanılan izinler, şüpheli API çağrıları, ağ adresleri şeklindeki statik özellikleri kullanarak bir özellik vektörü oluşturmuşlardır. Oluşturulan özellik vektörleriyle, destek vektör makinesini eğiten yazarlar, kendi oluşturdukları veri setinde %93,90, MalGenome veri setinde, %95,90 tespit doğruluğu gözlemlemişlerdir. Yapılan çalışmada, yazarlar ismini belirtmedikleri 10 adet anti virüs yazılımıyla kendi modellerini kıyaslamış ve bir anti virüs yazılımı hariç diğer 9 yazılımdan daha iyi tespit gerçekleştirdiklerini belirtmişlerdir [30].

Derin öğrenme yöntemleri kullanarak Android kötücül yazılımları tespit etmeye çalışan ilk çalışmalar 2014 yılında başlamıştır. Yuan ve arkadaşlarının 2014 yılında yayımladıkları Droid-Sec adlı çalışmada [31], yazarlar derin öğrenme yaklaşımlarının, makine öğrenmesi yaklaşımlarından daha etkili olabileceğini göstermek istemişlerdir. Statik analizle, API istekleri, izinlerden ve dinamik analizle, dinamik davranışlardan olmak üzere toplam 202 özellik çıkartılmıştır. Çalışmada kullanılan veri seti, Contagio mobile veri setinden 250 kötücül, Google Play Store'dan 250 normal uygulama olmak üzere toplam 500 uygulamadan oluşmaktadır. Kötücül ve normal uygulamalar eşit olacak şekilde 300 uygulama eğitim seti, 200 uygulama test seti olarak seçilmiştir. Çalışmada, derin öğrenme yaklaşımı olan derin inanç ağları kullanılarak, model %96,5 doğrulukla kötücül yazılımları tespit edebilmiştir. Aynı çalışmada, makine öğrenmesi algoritmaları olan, destek vektör

makinesi %80, saf bayes %79, lojistik regresyon %78 ve çok katmanlı algılayıcı %79,5 doğrulukla sınıflandırabilmiştir.

Yuan ve arkadaşlarının 2016 yılında gerçekleştirdikleri çalışmada [32], yazarlar, 2014 yılındaki çalışmaya [31] benzer bir yol izlemişlerdir. Farklı olarak daha büyük bir veri seti ve daha az özellik kullanmışlardır. Benzer şekilde statik analizle izinler ve API istekleri ve dinamik analizle, dinamik davranışlardan oluşan 192 adet özellik çıkartmışlardır. Veri seti olarak, Contagio ve Genome Project veri setlerinin birleşiminden oluşan 1760 kötücül ve Google Play Store'dan indirilen 20 000 normal uygulama kullanmışlardır. Kullanılan derin inanç ağı algoritmasıyla model, %96,76 doğrulukla kötücül yazılımları tespit etmiştir.

Hou ve arkadaşlarının 2016 yılında gerçekleştirdikleri DroidDelver adlı çalışmada [33], yazarlar, uygulamanın smali dosyasından API isteklerini çıkartıp, daha sonra benzer blokları oluşturan API isteklerini bloklar halinde gruplandırmıştır. API çağrı blokları adını verdikleri bu gruplamaları, modelin kullandığı özellikleri oluşturmaktadır. Veri seti olarak, Comodo Cloud Security Center'dan elde edilen, 2500 kötücül ve 2500 normal toplam 5000 uygulama kullanılmıştır. Kullanılan derin inanç ağı algoritmasıyla model, %96,66 doğrulukla kötücül yazılımları tespit etmiştir.

Hou ve arkadaşlarının 2016 yılında gerçekleştirdikleri Deep4MalDroid adlı çalışmada [34], yazarlar, dinamik analiz ve İstiflenmiş Oto-kodlayıcı (SAE) yaklaşımıyla çalışan bir sınıflandırıcı geliştirmişlerdir. Yazarlara göre, Linux sistem çağrıları, uygulamaların API çağrılarına göre tespit açısından daha güvenilirdir. API çağrıları sürümden, sürüme değişiklik göstermekte, kararlı saldırganlar tarafından aynı davranışı yerine getirecek yeni kütüphaneler hazırlanabilmektedir. Fakat Linux sistem çağrıları çoğunlukla değişmediğinden daha homojen sonuçlar verebileceği yazarlar tarafından düşünülmüştür. Genymotion emulasyon yazılımında çalıştırılan uygulamaların oluşturduğu Linux çekirdeğine ait sistem çağrıları toplanmıştır. Toplanan sistem çağrılarından bir çizge oluşturulmuştur. Elde edilen çizgeye ait derece, ağırlık gibi özelliklerden, modele ait özellikler çıkartılmıştır. Veri seti olarak, Comodo Cloud Security Center'dan alınan 1500 kötücül, 1500 normal toplam 3000 adet uygulama kullanılmıştır. Veri seti açık erişime sahip değildir. Veri setinin yapısı ve dağılımı belirsizdir. İstiflenmiş Oto-kodlayıcı (SAE) kullanan model, veri setindeki uygulamaları, %93,68 doğrulukla sınıflandırmıştır.

McLaughlin ve arkadaşlarının 2017 yılında gerçekleştirdikleri çalışmada [35], statik analiz ve ESA modelinin uygulanmasına dayanmaktadır. Uygulama dosyasından statik analizle, önce dex, sonra smali dosyasi elde edilmiş, son olarak smali dosyasından opcode sekansları operandlar dahil edilmeden çıkartılmıştır. Elde edilen sekanslar One Hot Encoder yöntemiyle, 1 ve 0'lardan oluşan bir matrise dönüştürülmüştür. Elde edilen matris ESA modelinin girişini oluşturmaktadır. Çalışmada, 1260 kötücül, 863 normal uygulamadan oluşan MalGenome veri seti, 2475 kötücül, 3627 normal uygulamadan oluşan McAfee laboratuvarından alınan veri seti, 9902 kötücül, 9268 normal uygulamadan oluşan gene McAfee laboratuvarından alınan veri seti olmak üzere, yazarların, küçük, büyük, çok büyük olarak sınıflandırdıkları, 3 adet veri seti kullanılmıştır. Oluşturan model küçük, büyük ve çok büyük veri setleri üstünde çalıştırılıp sırasıyla %98, %80, %87 doğrulukla kötücül yazılımları sınıflandırabilmiştir. Modelin küçük veri setinde diğer veri setlerine göre yüksek doğruluk elde etmesini yazarlar, modelin ezberleme yapması olarak değerlendirmişlerdir. Bu çalışma, veri setinin model başarımını göstermede önemli bir etken olduğunu göstermektedir. Küçük veri setleriyle çalışmak yanıltıcı değerler alınmasına neden olabilmektedir. Çalışma, ilk ESA kullanan yaklaşımlarından biridir.

Karbab ve arkadaşlarının 2018 yılında gerçekleştirdikleri MalDozer adlı, statik analiz ve ESA modeline dayanan çalışmada [36], uygulamanın dex dosyasından elde edilen ham API çağrı sekansları kullanılmıştır. Doğal dil işlemeyi amaçlayan başka bir çalışmadan etkilendiklerini ve benzer bir tekniği uyguladıklarını söyleyen yazarlar, diğer çalışmadaki verilen metinden anlam çıkartılması gibi, uygulama kodlarından kötücül anlamı çıkartmaya çalışmışlar. Uygulamanın classes.dex dosyasından elde edilen API çağrıları, word2vec denen yöntemle one hot encoder yönteminden farklı olarak semantik ilişkilerin korunduğu bir vektöre dönüştürülmekte ve ESA modelinin girdi değerlerini oluşturmaktadır. Yazarlar, word2vec yönteminin, one hot encoder yöntemine göre hem performans açısından hem ölçeklenebilirlik açısından oldukça üstün olduğunu belirtmektedirler. Veri seti olarak, MalGenome, Drebin veri setlerinden ve kendi oluşturdukları MalDozer isimli veri setinden oluşan sırasıyla 9, 20, 32 aileye ait 1258, 5555, 20.089 kötücül örneği ve 37.627 normal olmak üzere tüm veri setinde 33.066 kötücül ve 37.627 normal uygulama, toplam 70.693 uygulama bulunmaktadır. Yazarlar malgenome, drebin, maldozer ve toplam veri set olmak üzere her birinde modeli çalıştırmış ve sırasıyla %99,84, %99,21, %98,18, %96,29 F1 skoru elde etmişlerdir. Yazarlar sınıflandırma başarısının ölçümünde temel olarak F1 skorunu kullanmışlardır.

3.3. Görsel Kötücül Analizi Gerçekleştiren Çalışmalar

Görsel kötücül analizi ve sınıflandırılması çalışmalarında bilinen ilk örnek, Nataraj ve arkadaşları tarafından 2011 yılında gerçekleştirilen ve Windows kötücüllerinin sınıflandırılması üzerine MalImg adlı 25 aileye ait 9458 adet örnek bulunduran veri setinin oluşturulduğu çalışmadır [37]. Çalışmada, kötücül yazılımlar gri tonlamalı görsellere dönüştürülerek, makine öğrenmesi algoritmalarıyla ailesel sınıflandırılmıştır.

Benzer şekilde, makine öğrenmesi kullanan ve Android kötücül yazılımlarının, görsellere dönüştürülerek görsel kötücül yazılım analizini gerçekleştiren ilk çalışmalar 2016 yılında ortaya çıkmaya başlamıştır. Kumar ve arkadaşları tarafından 2016 yılında gerçekleştirilen çalışmada [38], uygulamalara ait APK dosyaları ikili formatta okunarak kötücül görsellere dönüştürülmüştür. Görsellere dönüştürmede, RGB (Kırmızı, yeşil, mavi), CMYK (camgöbeği, galibarda, sarı, siyah), gri tonlamalı ve HSL (renk, doygunluk ve açıklık) görsel formatları kullanılmıştır. Çalışmada, kullanılan veri seti 108 zararsız ve 138 kötücül örnekten oluşmaktadır. İkili sınıflandırma gerçekleştirilen çalışmada, en iyi sonucu bir makine öğrenmesi yaklaşımı olan rastgele orman (İngilizce random forest) gri tonlamalı görseller üzerinde %91 doğrulukla almıştır.

Yang ve Wen tarafından 2017 yılında sunulan çalışmada [39], APK dosyasını bütün olarak kullanmak yerine, içerisindeki dosyaları kullanmaya ve bu dosyaların 8 bit okunarak birleştirilmesinden elde edilen görseller üzerinde, makine öğrenmesi kullanılarak görsel analiz gerçekleştirilmiştir. Android uygulama dosyası sıkıştırılmış formatta bir dosyadır ve uygun bir araçla içerisindeki dosyalar çıkartılabilir. Uygulama dosyasının içindekileri uygun bir araçla çıkartan yazarlar, classes.dex, AndroidManifest.xml, resources.arsc, CERT.RSA dosyalarını, önce 8 bit işaretsiz integer vektörlere dönüştürmüş, sonra bu vektörleri birleştirerek iki boyutlu bir matris oluşturmuş, son olarak bu matristen gri tonlamalı görseller elde etmişlerdir. Yazarlar, doğrudan APK kullanmaya göre bu yöntemin daha fazla görsel ayırıcı içerdiğini belirtmektedirler [39]. Çalışmada, Drebin veri setinden alınan 640 kötücül ve Baidu uygulama marketinden alınan 640 zararsız uygulama olmak üzere toplam 1280 adet örnekten oluşan bir veri seti kullanmışlardır. Rastgele ağaç algoritmasıyla, ağaç sayısı 200 olmak üzere, veri setindeki görseller üzerinde, model %95,42 doğrulukla tespit gerçekleştirmiştir.

Chen ve arkadaşlarının 2018 yılında sundukları ve yöntem olarak bir makine öğrenme algoritması olan XGBoost kullanan çalışmada [40], APK dosyalarından çıkartılan classes.dex dosyaları 8 bit şeklinde okunarak, gri tonlamalı görsellere dönüştürülerek sınıflandırılmışlardır. Önceki çalışmalardan farklı olarak APK dosyasının tamamı ya da içerisindeki dosyalardan oluşturulan bir görsel yerine, ilgili çalışmada sadece classes.dex dosyası kullanılmıştır.

Android işletim sistemine ait kötücül yazılımların derin öğrenme yöntemleriyle görsel analizine dayanan ilk çalışmalar 2018 yılında başlamıştır. Ding ve arkadaşları tarafından 2018 yılında gerçekleştirilen çalışmada [41], Android uygulama dosyalarından çıkartılan classes.dex dosyaları, 8 bit okunarak gri tonlamalı görsellere dönüştürülmüş ve üretilen görseller evrişimsel sinir ağları kullanılarak ikili sınıflandırılmıştır. Yazarlar, AlexNet benzeri ve kendi ürettikleri oldukça sade bir ağ kullanmışlardır. Yazarlar çalışmada amaçlarının en iyi modeli üretmek değil, ESA modellerinin ilgili sorun karşısındaki fizibilitesini araştırmak olduklarını belirtmişlerdir.

Hsien-De Huang ve Kao tarafından 2018 yılında gerçekleştirilen R2-D2 [42] adlı diğer bir çalışmada, Ocak 2017 ile Ağustos 2017 arasında toplanan ve yaklaşık 2 milyon zararlı ve zararsız uygulamadan elde edilen clasess.dex dosyaları, RGB kanallı, renkli görsellere dönüştürülmüştür. Görsellerin renklendirilmesi için yazarlar, piksel değerine göre renk atayan bir haritalama metodu kullanmışlardır. Çalışmada yazarlar, o zamanki popüler modeller olan AlexNet, VGG, GoogleNet, Inception-V3 ESA modellerini denemişler, en iyi sonuca Inception-V3'le erişmişlerdir.

Dex dosyalarından, uygulamaları temsil eden görseller oluşturmak yaygın bir yaklaşım olsa da farklı yaklaşımlarda denenmektedir. Bu yaklaşımlardan biri de tüm uygulama dosyasını, başka bir işlem uygulamadan ikili okuyarak görsel dosyalara dönüştürmektir. Bu yaklaşımda temel motivasyon, kötücüllere ait olan izlerin sadece dex dosyasında değil, tüm uygulama dosyasında olabileceği ve geliştirilen model yardımıyla bu izlerin yakalanabileceğidir. Al-Fawa'reh ve arkadaşlarının 2020 yılında gerçekleştirdikleri çalışmada [43], yazarlar, doğrudan APK dosyalarını gri tonlamalı 50x50 piksel boyutunda görsellere dönüştürerek kötücül tespiti gerçekleştirmişlerdir. Çalışmada, CICMalDroid2020 [1] veri setinden yararlanılmıştır. Dengeli ve dengesiz olmak üzere iki adet veri seti oluşturmuşlardır. Dengelenmiş veri setinde 10878 adet zararsız, 10878 adet

kötücül yazılım ve dengelenmemiş veri setinde, 578 zararsız, 10878 kötücül yazılım bulunmaktadır. Kendi sundukları ESA modelini, DenseNet121, DenseNet169, EfficientNetB7, InceptionV3, MobileNet, MobileNetV2, ResNet50, Vgg16, Vgg19, Xception modelleriyle karşılaştıran yazarlar, kendi modellerinde, dengeli veri setinde %74 ikili sınıflandırma doğruluğu ve dengesiz veri setinde %95,9 ikili sınıflandırma doğruluğu gözlemlemişlerdir.

Darwaish ve Naït-Abdesselam tarafından 2020 yılında sunulan çalışmada [44], APK dosyalarından geliştirdikleri yöntemle renkli RGB kanallı görseller oluşturarak, ESA modeliyle ailesel sınıflandırılması gerçekleştirilmiştir. Çalışmada, APK dosyasından elde edilen, AndroidManifest.xml dosyası ve classes.dex dosyası görsellerin oluşturulmasında kullanılmıştır. Androguard tersine mühendislik aracıyla, AndroidManifest dosyasından elde edilen izinler, aktiviterler, servisler, alıcılar ve intentler gibi statik özellikler görsellerin yeşil kanalını, Classes.dex dosyasından elde edilen API çağrıları ve benzersiz opkodlar görsellerin kırmızı kanalını, hem AndroidManifest hem de Classes.dex dosyasından elde edilen şüpheli izinler, uygulama bileşenleri, şüpheli API çağrıları ve korumalı karakter katarları görsellerin mavi kanalını oluşturacak şekilde haritalandırılmıştır. Oluşturulan görseller en yakın komşuluk interpolasyonu kullanılarak 64x64 piksel olacak şekilde yeniden boyutlandırılmıştır. Farklı tarih aralıklarında ve örnek büyüklüklerinde dört adet veri seti oluşturan yazarlar. 2009-2011 tarihleri arasındaki 4000 zararsız, 4000 kötücül içeren veri setinde %98,21, 2012-2014 tarihleri arasındaki 12 000 zararsız, 12 000 kötücül içeren veri setinde %98,39, 2015-2016 tarihleri arasındaki 30 000 zararsız, 30 000 kötücül içeren veri setinde %95,73, 2017-2020 tarihleri arasındaki 11 000 zararsız, 11 000 kötücül içeren veri setinde %98,77 olmak üzere kendi sundukları ESA modelinde dört farklı tespit doğruluğu raporlamışlardır. Ailesel sınıflandırmada, 6 aile ve aile başına 600 örnek olursa %91,21, 6 aile ve aile başına 2300 örnek olursa %92,4, 10 aile ve aile başına 175 örnek olursa %96, 10 aile ve aile başına 4100 örnek olursa %88,91, 5 aile ve aile başına 4100 örnek olursa %92,3 ailesel sınıflandırma doğruluğu elde edildiği raporlanmıştır. Çalışmada sunulan sonuçlardan, aile başına örnek sayısı aynı kalmak koşuluyla, aile sayısı arttıkça ailesel sınıflandırma başarımının düştüğü, örnek sayısının arttıkça tespit performansının azaldığı çıkarılmaktadır.

Hemalatha ve arkadaşları 2021 yılında gerçekleştirdikleri çalışmada [45], dört adet dense bloğundan oluşan, DenseNet temelli bir model sunmuşlardır. Kötücül yazılımlardan, gri tonlamalı görseller oluşturarak, görselleri 64x64 olacak şekilde tekrar boyutlandırmışlardır.

Modeli, Windows kötücül yazılımlarından oluşan MalImg, BIG 2015, MaleVis, Malicia adlı dört farklı veri setinde eğiterek, sırasıyla %98,23, %98,46, %98,21 ve %89,48 doğrulukla örnekleri sınıflandırmışlardır.

3.4. Görsel Kötücül Yazılım Analizinde Birleşim Kullanan Çalışmalar

Bu bölümde, birden fazla yaklaşımdan oluşturulan birleşim ile Android yazılımlarından elde edilen görselleri sınıflandıran ve görsel kötücül analizi gerçekleştiren çalışmalar incelenecektir. Bu çalışmalardaki model birleşimleri, sadece makine öğrenmesi algoritmalarından oluşabildiği gibi, derin öğrenme ve makine öğrenmesi birleşimleri veya sadece derin öğrenme modellerinin birleşimleri şeklinde de olabilmektedir.

Huang ve arkadaşları tarafından 2018 yılında sunulan MixDroid adlı çalışmada [46], yazarlar, derin öğrenme yöntemi olan ESA yaklaşımıyla, makine öğrenmesi yaklaşımları olan, K en yakın komşuluk (KNN), Lojistik Regresyon (LR), destek vektör makinesi (DVM), rastgele orman (RO) yaklaşımlarını, ağırlıklı oylama yöntemiyle birleştirmişlerdir. Çalışmada, 10 000 zararsız ve 10 000 kötücül yazılımdan oluşan bir veri seti kullanılmıştır. Android uygulama dosyasından elde edilen classes.dex dosyasını gri tonlamalı görsellere dönüştürerek, VGG16 modelini bu görsellerle eğitmişlerdir. İzinler ve bytekod dosyasından çıkartılan özelliklerin birleşiminden oluşturulan özellik vektörleriyle, KNN, LR, DVM, RO modellerini eğitmişlerdir. VGG16 modeli %93,9 doğrulukla, KNN modeli %88,1 doğrulukla, LR modeli %92,6 doğrulukla, DVM modeli %91,4 doğrulukla, RO modeli %92,4 doğrulukla veri setindeki örnekleri ikili sınıflandırmıştır. Bu modellerin, ağırlıklı oylama yöntemiyle elde edilen birleşim %98,1 tespit doğruluğuna erişmiştir. Oylamada, modellerin sahip olduğu ağırlıkların tespiti için ızgara araması (İngilizce grid search) kullanılmıştır.

Vasan ve arkadaşları tarafından 2020 yılında yayımlanan IMCEC adlı çalışmada [47], ince ayar öğrenim aktarması ile eğitilen ResNet50, VGG16 ESA modelleriyle makine öğrenmesi modeli olan vektör destek makinasından oluşan birleşik model sunmuşlardır. ResNet ve VGG16 modellerini hem sınıflandırıcı olarak hem de üç adet vektör destek makinesinin eğitiminde kullandıkları görsellere ait özellikleri elde etmek için kullanmışlardır. Yazarlar, MalImg veri setini %70 eğitim, %30 test olmak üzere iki parçaya bölümlendirmişlerdir. Çalışmada, oluşturdukları birleşim modeliyle Windows

kötücül yazılımlarından oluşan görsel veri setini %99,50 doğrulukla ailesel sınıflandırmışlardır.

Narayanan ve Davuluru tarafından 2020, Nisan ayında yayınlanan çalışmada [48], yazarlar, tekrarlayan sinir ağlarıyla, VGG16, ResNet, AlexNet ve basit ESA modellerinden oluşan bir birleşim sunmuşlardır. Kaggle adlı platform üzerinde paylaşılan ve dokuz aileye ait Windows kötücül yazılımlarına ait olan BIG 2015 adlı veri setini kullanan yazarlar, veri setini %72 eğitim, %8 geçerleme ve %20 test şeklinde bölümlendirmişlerdir. Veri setinden oluşturdukları görsel kötücül veri setini, sundukları birleşimle, %99,8 doğrulukla sınıflandırmışlardır.

Ren ve arkadaşları tarafından 2020 yılında sunulan çalışmada [49], evrişimsel sinir ağları ve tekrarlayan sinir ağlarını birleştiren modeller kullanarak, classes.dex dosyasından elde edilen gri tonlamalı görseller üzerinde ikili sınıflandırma gerçekleştirmişlerdir. Uygulamalara ait APK dosyalarından elde edilen classes.dex dosyaları ikili şekilde okunarak bir vektöre aktarılmıştır, sabit girdi boyutlarına sahip olan modellere sunabilmek adına, oluşturulan vektörler belirlenen girdi boyutuna göre en yakın komşu interpolasyonu tekrar örnekleme algoritması kullanılarak yeniden boyutlandırılmıştır. Vektörün girdi boyutundan büyük olduğu durumlarda ortalama havuzlama katmanı kullanılarak, örnek azaltımı gerçekleştirilmiştir. Çalışmada, DexCNN olarak adlandırdıkları sade ESA modeli ve DexCRNN olarak adlandırdıkları birleşik modeller, Google uygulama marketinden elde edilen 8000 zararsız ve VirusShare sitesinden elde edilen 8000 kötücül uygulamanın işlenmesiyle oluşturulan görsel veri setiyle eğitilmiştir. Çalışmada, sade ESA modeli olan DexCNN, %93,6 doğrulukla, ESA ve TSA birleşimi olan, iki yönlü uzun kısa dönem hafıza yaklaşımını kullanan DexCRNN_BiLSTM, %95,8 doğrulukla kötücül tespiti gerçekleştirmiştir.

3.5. Görsel Kötücül Yazılım Analizinde Modellerin Performans Karşılaştırması

ImageNet yarışmasını kazanan ve teknolojide gelinen son nokta (state of the art, sota) olarak adlandırılan modellerin toplu bir şekilde başarımlarını araştıran çalışmaların sayıca az olduğu görülmektedir. Bu çalışmalar tez çalışmasının sunduğu temellere, en yakın ve ilgili çalışmalar olmaktadırlar.

El-Shafai ve arkadaşları tarafından 2021 yılında yayımlanan çalışmada [50], Windows kötücül yazılımlarına ait görsellerden oluşan MalImg adlı veri setinde, 8 son teknoloji model ince ayar öğrenim aktarımı yöntemi kullanılarak ailesel sınıflandırma yapmaları için eğitilmiştir. Çalışmada en iyi model VGG16 olarak bulunmuş ve model Windows işletim sistemine ait kötücül yazılımları %99,97 doğrulukla sınıflandırmıştır. İlgili çalışma, ince-ayar öğrenim aktarımını, birden fazla model ve aynı veri setinde uygulayarak karşılaştırmasıyla benzemekle birlikte, Windows kötücül yazılımlarını ailesel sınıflandırmasında kullanıldığı ve birleşik model performansları araştırılmadığı için gerçekleştirilen tez çalışmasından ayrılmaktadır.

Yadav ve arkadaşları tarafından 2022 yılında yayımlanan çalışmada [51], 26 adet son teknoloji model karşılaştırılmış, R2-D2 adlı çalışmada üretilen RGB görsel veri setinden alınan örneklerle oluşturulan ve 5986 örnekten oluşan veri setinde modellerin ikili sınıflandırma (tespit) performansları araştırılmıştır. En başarılı model olan EfficientNet-B4 veri setindeki örnekleri %95,7 başarımla sınıflandırmıştır. Çalışmada modeller üzerinde ince ayar gerçekleştirilmediği görülmektedir. Yazarlar son olarak, modellerin çıkış katmanı olan softmax sınıflandırıcı yerine, makine öğrenmesine ait modeller koyarak birleşim performansını araştırsa da elde ettikleri performansın daha düşük olduğunu bildirmişlerdir. Çalışma gerek yöntem gerek boyut açısından ve çözmeye çalıştığı sorun gibi birçok açıdan farklı olsa da doğal olarak bu çalışmayla, tezde savunulan görüşlerde benzerlikler bulunmaktadır. Tez bir süreci kapsadığından sonuç aşamasına gelene kadar yeni çalışmalar ortaya çıkabilmektedir. Nitekim, 2022 yılındaki çalışma çıkmadan önce, bu tezin kapsamında ara sonuçlar ve görüşler, 2021 yılının Aralık ayında Düzce Üniversitesi Bilim ve Teknoloji Dergi ’sinde, ICAIAME2021 özel sayısında yayımlamış bulunmaktadır.

Her ne kadar bu çalışmalar gerçekleştirilen tez çalışmasıyla benzerlik gösterseler de gerçekleştirilen çalışmayla, kategorik sınıflandırma ve sınıflandırılan örneklerin ait olduğu işletim sistemi açısından farklılıklar göstermektedir. En büyük farklılıksa gerçekleştirilen çalışmalarda, elde edilen modellerin birleşimlerinden elde edilecek yeni modellere ait performansların yeterince araştırılmamış olmalarıdır.

3.6. Kategorik Kötücül Yazılım Sınıflandırması Gerçekleştiren Çalışmalar

İncelenen çalışmaların tespit ya da ailesel sınıflandırma şeklinde olduğu görülmektedir. Android kötücül yazılım analizinde, tespit ya da ailesel sınıflandırma çalışmalarından farklı, yeni bir yaklaşımda kötücül yazılımların kategorik sınıflandırılmasıdır. Android uygulamalarının kategorik sınıflandırmalarını içeren çalışmalar bulunmaktayken, Android sistemini hedef alan kötücül yazılımların kategorik sınıflandırma çalışmaları son yıllarda görülmeye başlanmıştır. Bu durumun sebeplerinden biri uygulamaların kategorik sınıflandırmalarını içeren açık veri seti eksikliği olabilir. Nitekim, kategorik ve açık erişime sahip veri setlerinin artması sonucunda, kategorik sınıflandırma içeren çalışmalarda artış gözlenmiştir.

Lashkari ve arkadaşları tarafından 2018 yılında yayınlanan CICAndMal2017 [52] adlı çalışma ve Taheri ve arkadaşları tarafından 2019 yılında yayınlanan CICInvesAndMal2019 [53] adlı çalışma açık erişime sahip kategorik sınıflandırma içeren veri setlerinin ilk örneklerindendir. Çalışma, iki parça halinde yayımlanmıştır, veri setinde, 5065 adet zararsız örnek ve reklamcı, fidyeci, ürkütücü (İngilizce scareware) ve SMS olmak üzere dört kategoriye ait 426 adet kötücül yazılım bulunmaktadır. Kötücül yazılımlar 42 adet aileden toplanmışlardır. Veri setinde uygulamalara ait hem statik hem dinamik özellikler sunulmaktadır. 2018 yılında sunulan çalışmada [52], makine öğrenmesi algoritmaları olan, RO, K en yakın komşuluk, KA modelleri kullanılmış ve sırasıyla %49,90, %49,50, %47,80 kesinlikle kötücül yazılımların kategorik sınıflandırması gerçekleştirilmiştir. Çalışmada değerlendirme metriği olarak kesinlik ve geri çağırma raporlanmıştır. 2019 yılında sunulan çalışmada [53], statik analiz özellikleriyle, dinamik analiz özelliklerini de dahil eden yazarlar, RO algoritmasıyla, %83,30 kesinlikle kategorik sınıflandırma gerçekleştirmişlerdir.

Mahdavifar ve arkadaşları tarafından 2020 yılında gerçekleştirilen ve CICMaldroid2020 veri setinin de oluşturulduğu çalışmada [1], Android uygulamaları reklamcı, bankacı, SMS, riskli, zararsız olmak üzere beş kategoride sınıflandırılmıştır. Toplam 17341 uygulamadan oluşan veri seti üzerinde, dinamik analiz yöntemini uygulayan araştırmacılar, 13077 adet uygulamayı başarıyla analiz edebilmişlerdir. Diğer uygulamaların, süre dolması, bozuk uygulama dosyası, bellek hataları gibi çeşitli hatalar sebebiyle incelenemediği, yazarlar tarafından belirtilmiştir. Elde edilen analiz sonuçlarından %12'si

yüklenirken hata verdiğini belirten yazarlar, tüm bu hatalardan sonra 1253 reklamcı, 2100 bankacı, 3904 SMS, 2546 riskli, 1795 zararsız olmak üzere toplam 11598 adet uygulamaya ait analizden oluşan bir veri seti oluşturmuşlardır. Yazarlar, uygulama dosyalarından, intent, izinler, servisler, farklı dosya türlerine ait frekans sayıları, kod şaşırtma olayları ve hassas API çağrıları gibi statik özellikleri ve sistem çağrıları, bağlayıcı çağrılar (İngilizce binder calls), kompozit davranışlar ve ağ trafiğine ait paket kayıtları gibi dinamik özellikleri çıkartmışlardır. Elde edilen özellikler üstünde normalizasyon uygulayan yazarlar, elde ettikleri vektörü model eğitiminde kullanmışlardır. Sundukları sahte etiket derin sinir ağı modeliyle, yarı denetimli bir eğitim gerçekleştiren yazarlar, toplam 11598 örnekten oluşan %10 etiketli veri setini %96,7 doğrulukla, tamamı etiketli veri setini %97,6 doğrulukla kategorik olarak sınıflandırmışlardır.

Imtiaz ve arkadaşları tarafından 2021 yılında gerçekleştirilen DeepAMD adlı çalışmada [54], yazarlar, CICAndMal2017 ve CICInvesAndMal2019 veri setlerini kullanarak, örnekleri statik ve dinamik analiz kullanarak sınıflandırmışlardır. Sunulan model, statik ve dinamik olmak üzere iki katmandan oluşmaktadır. Statik analize ait özellikleri kullandıklarında zararsız ve kötücül olmak üzere iki sınıfa göre sınıflandırırken, statik analizde kötücül sınıflandırılan örnekleri, dinamik analize ait özellikleri kullandıklarında, reklamcı, ürkütücü, SMS, fidyeci olmak üzere dört kategoriye göre sınıflandırmışlardır. Benzer şekilde dinamik analiz aşamasında, statik analizde kötücül çıkan uygulamalara ailesel sınıflandırmada gerçekleştirmişlerdir. Sundukları derin sinir ağı modelini, veri setinde eğiten yazarlar, statik katmanda, %93,4 ikili sınıflandırma, %92,5 kategorik sınıflandırma, %90 ailesel sınıflandırma doğruluğu, dinamik katmanda, %80,3 kategorik sınıflandırma, %55,7 ailesel sınıflandırma doğruluğu gözlemlemişlerdir.

Mahdavifar ve arkadaşları tarafından 2022 yılında gerçekleştirilen çalışmada [55], CICMalDroid2020 [1] veri seti üzerinde, statik ve dinamik özellikleri kullanarak karma analiz gerçekleştiren yazarlar, sahte etiket istiflenmiş (yığılmış) otomatik kodlayıcı (İngilizce stacked autoencoder) yaklaşımıyla yarı denetimli öğrenme kullanan bir model sunmuşlardır. Çalışmada, 1253 reklamcı, 2100 bankacı, 3904 SMS, 2546 riskli, 1795 zararsız olmak üzere toplam 11598 adet uygulama analiz edilmiş ve çalışmada belirtildiği üzere, karma analiz gerçekleştiren modelin eğitilmesi için gerekli olan statik ve dinamik özelliklerin çıkartılması, özellik çıkartımı ve ön işlemler dahil yaklaşık 24 saat sürmüştür. Sunulan model, %1 etiketlenmiş örnekler üzerinde, %95,19, %10 etiketlenmiş örnekler

üzerinde, %98,28, tamamı etiketlenmiş örnekler üzerinde %98,52 kategorik sınıflandırma doğruluğu göstermiştir.

Bu bölümde incelenen çalışmalar, kategorik sınıflandırma açısından benzerlik göstermektedir. İncelenen çalışmaların, dinamik ve statik özellikleri birlikte kullandığı bu nedenle dinamik ya da karma analiz gerçekleştirdikleri görülmektedir. Tez çalışmasında, bir statik analiz yöntemi olan görsel kötücül analizi gerçekleştirilmektedir. Statik analiz, dinamik ve karma analize göre daha az işlem içerdiğinden dolayı daha hızlı gerçekleşebilmektedir. İncelenen çalışmalarda, görsel kötücül analizi ile Android kötücül yazılımlarının kategorik sınıflandırmasını yapan çalışma bulunamadığından, tez çalışmanın bu alandaki ilk çalışmalardan olduğu düşünülmektedir.

4. YARARLANILAN ARAÇLAR ve VERİ SETİ

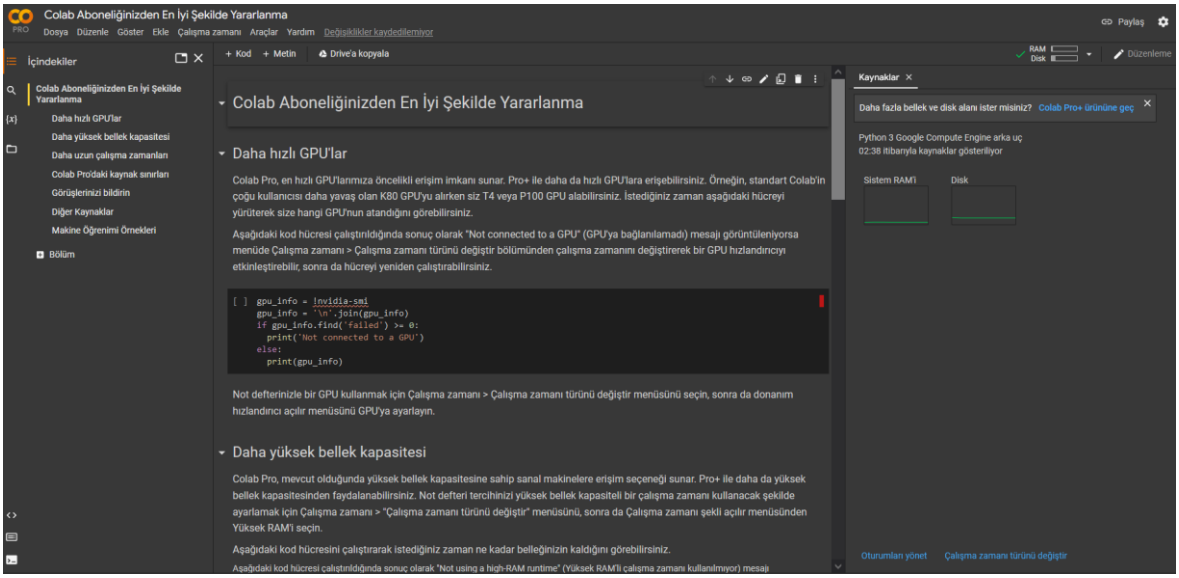
Bu bölümde, yararlanılan araçlar, veri setinin belirlenmesi, uygulama dosyasından görsel oluşturma yöntemleri, veri setinin hazırlanması ve uygulanan önişlemler açıklanmaktadır.

4.1. Yararlanılan Araçlar

Bu bölümde, çalışma boyunca kullanılan araçlar açıklanmakta ve kullanım örnekleri verilmektedir. Çalışmada, Google Colab geliştirme ortamı, Tensorflow v2.6.0 çerçevesi, Androguard v3.5.5 tersine mühendislik aracı, Numpy v1.19.5 veri işleme ve matematik kütüphanesi, Python Imaging Library (kısaca PIL) görsel işleme kütüphanesi, MatPlotLib ve Seaborn görselleştirme kütüphaneleri kullanılmaktadır.

4.1.1. Google Colab

Google Colab, web tarayıcısı üzerinde çalışan bir geliştirme ortamıdır. Colab sayfaları (Şekil 4.1) üzerinde Python kodları çalıştırılabilmektedir. Colab, özellikle veri bilimi çalışmalarında kullanılmak üzere geliştirildiğinden, içerisinde birçok yazılım kütüphanesi yüklü olarak gelmektedir. Kullanılacak kütüphanenin olmadığı durumda, linux komutları yardımıyla başka kütüphaneler de yüklenebilmektedir. Uygulama kodları Colab'ın sunucularında çalışmaktadır. Colab yazılan kodların çalıştırılması için ücretli ve ücretsiz olmak üzere çeşitli donanım seçenekleri de sunmaktadır. Ücretsiz planda, Colab tarafından açıklanmayan bir kullanım süresi bulunmaktadır. Kullanım süresi dolduğunda sunucu kapanmakta ve sunucuda bulunan tüm dosyalar silinmektedir. Bu sebeple Colab üzerinde çalışırken, dosyaları ve üretilen modelleri güvenli bir ortama aktarmak ve sık sık kayıt almak gerekmektedir.



Şekil 4.1. Google Colab aracına ait ekran görüntüsü

4.1.2. Tensorflow ve Keras

Tensorflow, Google çalışanları tarafından geliştirilen açık kaynak bir uygulama çerçevesidir. Makine öğrenmesi ve derin öğrenme çalışmalarında sıklıkla tercih edilmektedir. Tensorflow hem akademik çalışmalara hem de endüstriyel çalışmalara uygundur. Tensorflow'un internet sitesinde belirtiği üzere, Intel, Coca Cola, Qualcomm, Spotify, Twitter, Google, Airbus, Airbnb gibi birçok firma Tensorflow'u çeşitli uygulamalarında kullanmaktadır. Tensorflow, sistem belliğini, grafik belliğini optimum şekilde kullanacak veri setleri oluşturabilmektedir. Bu veri setlerini ve verileri modele sunmadan önce, ön işleme kütüphanesinde bulunan fonksiyonlar yardımıyla işleyebilmektedir. Tensorflow yardımıyla modeller oluşturulabilmektedir. Tensorflow 2 ve üstü sürümlerde Keras aracı entegre olmuştur, model oluştururken Keras aracı kullanılmaktadır. Keras, aynı zamanda Xception modelinin de geliştiricisi olan François Chollet tarafından geliştirilmektedir. Keras, derin modeller oluştururken tekrar eden ve karmaşık hale gelen kodları, daha basit ve kullanıcı dostu bir hale getirmektedir. Kullanıcı sadece katmanları komutlarla birbirine bağlamakta ve arka planda gerçekleşen birçok karmaşık süreçten kendini arındırmaktadır. Tensorflow 2 ve üzeri sürümlerde artık varsayılan katman kütüphanesi olarak Keras kullanılmaktadır. Google Colab ortamında Tensorflow ve Keras kütüphaneleri hazır bulunduğundan kurulum gerektirmemektedir.

```
import tensorflow as tf
from tensorflow import keras
from keras import layers

x = layers.Input(shape=(224,224,3))

cov = layers.Conv2D(32,(32,32),(2,2))(x)
cov = layers.BatchNormalization()(cov)
cov = layers.Activation('relu')(cov)

top = layers.GlobalAvgPool2D()(cov)
top = layers.Dropout(0.5)(top)

top = layers.Dense(units=32,activation='relu')(top)
top = layers.Dropout(0.5)(top)

prediction_layer = tf.keras.layers.Dense(units=5,activation='softmax')
outputs = prediction_layer(top)
model = tf.keras.Model(x, outputs)
```

Şekil 4.2. Tensorflow aracında model oluşturma

Şekil 4.2'de Tensorflow ve Keras araçları kullanılarak oluşturulan basit bir ESA modeline ait kodlar gösterilmektedir. Öncelikle Tensorflow kütüphanesi yazılıma dahil edilmektedir. Okuma kolaylığı olması adına Tensorflow kütüphanesinden keras kütüphanesi ve keras kütüphanesinden layers kütüphanesi yazılımı dahil edilmektedir. Layers kütüphanesi modeli oluşturmak için gerekli katmanların olduğu kütüphanedir. İlgili kütüphaneden girdi katmanı olan "Input" çağrılmakta ve modele sunulacak girdi değerleri parametre olarak belirtilmektedir. İlgili kütüphaneden görsellerle uğraşıldığı için ve görseller 2 boyutlu olduğundan, "Conv2D" 2 boyutlu evrişim katmanı çağrılmakta, ilk parametre oluşturulacak filtre sayısını, ikinci parametre evrişim çekirdeğinin boyutlarını, üçüncü parametre kaydırma miktarını göstermektedir. İki boyutlu evrişim katmanının girişine giriş katmanı bağlanmaktadır. Çıkışına yığın normalizasyon katmanı "BatchNormalization" çağrılarak eklenmektedir. Yığın normalizasyonun ardından ReLu aktivasyonu için "Activation("relu")" katmanı kullanılmaktadır. Kürüsel ortalama havuzlama işlemi için "GlobalAvgPool2D" katmanının girişi, aktivasyon fonksiyonun çıkışına bağlanmaktadır. "Dropout" katmanı yardımıyla, bırakma işlemi gerçekleştirilmekte, sunulan parametre değeri bırakma oranını göstermekte ve 0 ile 1 arasında bir kayan noktalı sayı seçilmektedir. Örnekteki 0,5 değeri, eğitim sırasında ağdaki değerlerin yarısının rastgele bırakılacağını

göstermektedir. Bırakma katmanı, küresel ortalama katmanına bağlanmaktadır. “Dense” katmanı kullanarak ve parametre değerlerini vererek, 32 nörondan oluşan ve ReLu aktivasyonu kullanan bir tam bağlı katman elde edilmektedir. Dense katmanının girişi, önceki bırakma katmanına bağlanmakta, çıkışına da 0,5 bırakma oranına sahip diğer bir bırakma katmanı bağlanmaktadır. Tahmin ve sınıflandırma işlemini gerçekleştirmesi için bir tahmin katmanına ihtiyaç bulunmaktadır. Bunun için sınıf sayısı kadar nöron içeren bir yapay sinir katmanı, uygun aktivasyon fonksiyonuyla oluşturulmakta ve önceki bırakma katmanına girişi bağlanmaktadır. Tüm oluşturulan yapı Tensorflow/Keras kütüphanesinden “Model” fonksiyonu yardımıyla, girdi ve çıktı katmanları parametre olarak verilerek bir “Model” nesnesine dönüştürülmektedir. Model nesnesi üzerinde “summary()” metodu çalıştırıldığında, tüm yapılan bağlantıların ve eğitilebilir parametre sayısının bulundu özet çıktığı elde edilmektedir. Özet çıktısı Şekil 4.3’te gösterilmektedir.

```
model.summary()

Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_8 (InputLayer)        [(None, 224, 224, 3)]     0

 conv2d_22 (Conv2D)          (None, 97, 97, 32)        98336

 batch_normalization_21 (Bat  (None, 97, 97, 32)       128
 chNormalization)

 activation_21 (Activation)  (None, 97, 97, 32)        0

 global_average_pooling2d_6   (None, 32)               0
 (GlobalAveragePooling2D)

 dropout_19 (Dropout)        (None, 32)                0

 dense_19 (Dense)            (None, 32)                1056

 dropout_20 (Dropout)        (None, 32)                0

 dense_20 (Dense)            (None, 5)                 165

=================================================================
Total params: 99,685
Trainable params: 99,621
Non-trainable params: 64
```

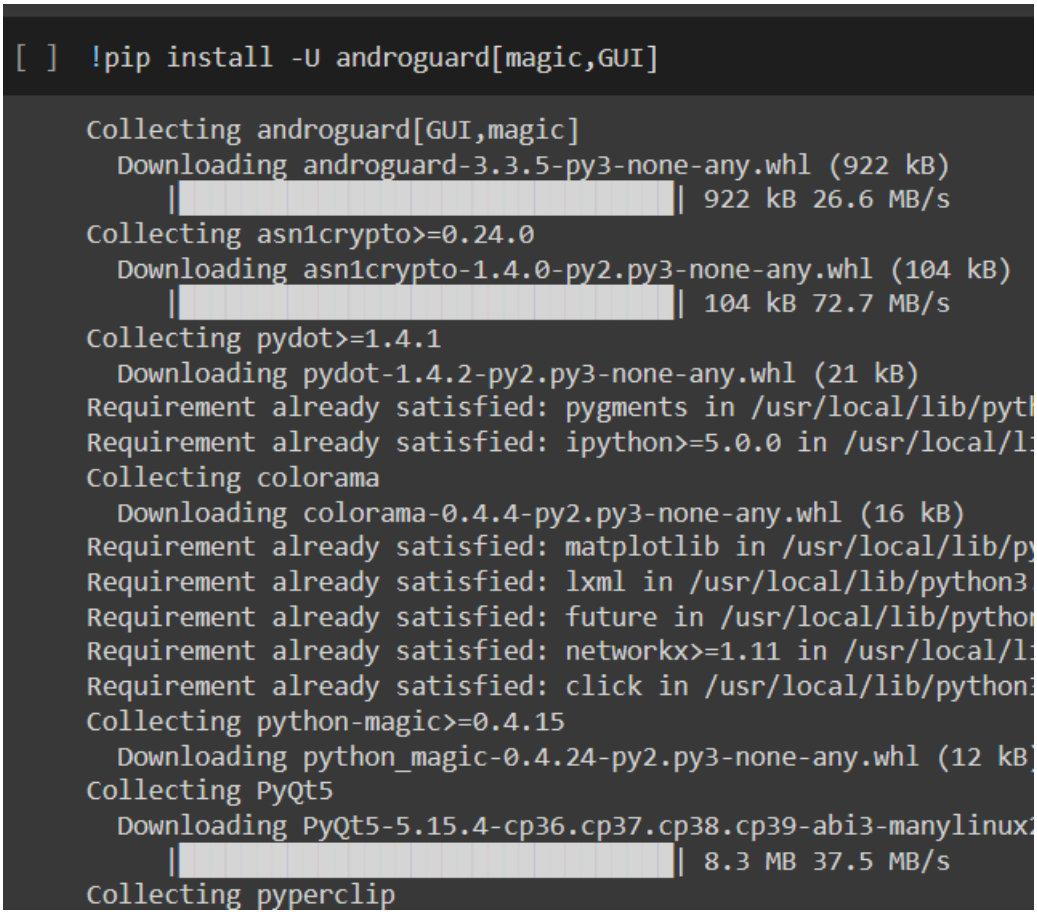
Şekil 4.3. Örnek modele ait özet bilgileri

Şekil 4.3’te gösterildiği üzere oluşturulan örnek model 99.621 adeti eğitilebilir, 64 adeti eğitilemez parametre olmak üzere toplam 99.685 parametreden oluşmaktadır. Eğitilebilir

parametreler, model eğitimde değerleri güncellenen parametrelerdir. Evrişim katmanında 32x32 büyüklüğünde çekirdek ve 32 adet filtre kullanılmakta ve 32x32x32 işleminden 98.336 eğitilebilir parametre oluşmaktadır. Küresel ortalama katmanında her özellik matrisinin ortalaması alınmakta ve tek bir ortalama değerle ifade edilmektedir. Bu sayede kanal sayısı kadar bir vektör oluşmaktadır. Yapay sinir ağında 32 adet nöron bulunduğundan ve önceki katmanda 32 elemanı bulunan bir vektör sunulduğundan 32x32 adet eğitilebilir parametre oluşmaktadır. Nöron sayısı kadar bias değeri bulunduğundan ve bias değerleri de eğitilebilir parametre olduklarından toplam 1056 adet eğitilebilir parametre bulunmaktadır. Sınıflandırıcı katmanda da benzer hesapla 165 adet eğitilebilir parametre bulunmaktadır.

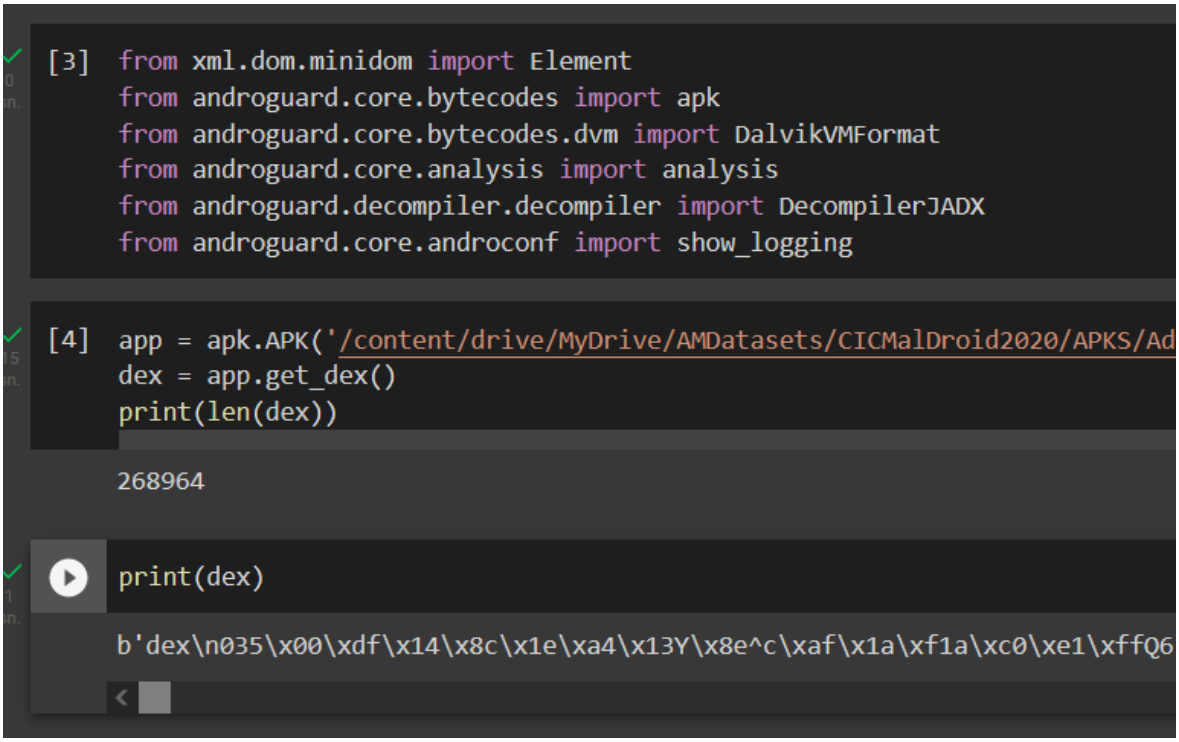
4.1.3. Androguard

Androguard, Android uygulamalarının incelenmesinde kullanılan bir tersine mühendislik aracıdır. Araç, uygulama dosyasından, dex, odex dosyalarını çıkartabilmektedir. Dex, odex dosyalarını derlenmemiş hallerine dönüştürebilmektedir. AndroidManifest dosyasını çıkartabilmektedir. Uygulama dosyasından elde edilen AndroidManifest dosyası ikili formatta olduğundan, bu dosyayı okunabilir hale dönüştürebilmektedir. Uygulamanın istediği izinleri, aktiviteleri, sertifika bilgilerini raporlayabilmektedir. Androguard, Colab ortamında bulunmamaktadır ve kurulması gerekmektedir. Kurulum işlemine ait ekran görüntüsü Şekil 4.4’te verilmektedir.



Şekil 4.4. Androguard kurulumu

Şekil 4.4'te Androguard kurulumu için gerekli kod komutu gösterilmektedir. Google Colab ortamında, "!pip install -U androguard[magic,GUI]" komutunun çalıştırılmasıyla kurulabilmektedir. Colab ortamında Linux komutları çalıştırılırken başına "!" işareti konması gerekmektedir. Androguard aracı kurulduktan sonra, Android uygulama paketleri üzerinden tersine mühendislik işlemleri gerçekleştirilebilmektedir. Androguard aracının kullanımına ait bir örnek Şekil 4.5'te gösterilmektedir.



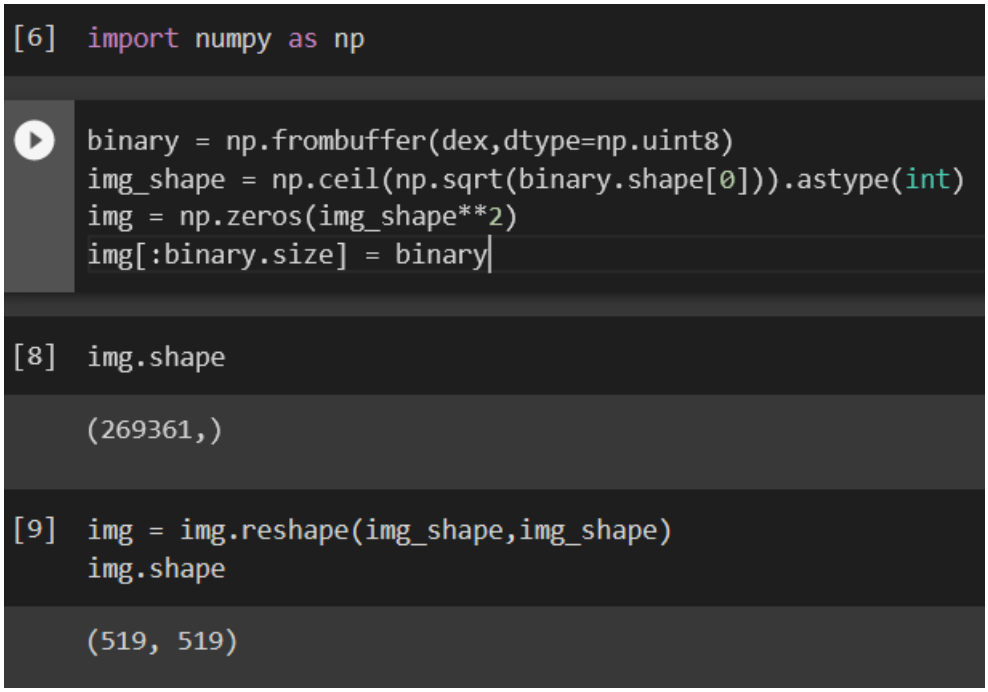
Şekil 4.5. Androguard kullanarak dex dosyası çıkartma

Şekil 4.5'te gösterildiği üzere Androguard aracıyla APK dosyalarından classes.dex dosyası elde edilmektedir. Androguard yazılımına ait kütüphaneler "androguard.core.bytecodes" kütüphanesi "apk" olarak adlandırılıp içeri aktarılmaktadır. İlgili kütüphanenin APK metodu çağrılarak, incelenecek dosyanın dosya yolu metoda girdi parametresi olarak sunulmaktadır. Metodun döndürdüğü yeni nesne üzerinde, "get_dex()" metodu çalıştırılarak, APK dosyasından classes.dex dosyası elde edilmektedir. Dex dosyasını "print()" fonksiyonuyla yazdırıldığında bayt kodu olarak okunabilmektir, "len()" fonksiyonuyla dosya boyutu öğrenilebilmektedir.

4.1.4. Numpy

Numpy, Python diline ait oldukça gelişmiş bir veri işleme, matematik kütüphanesidir. Özellikle matris ve dizi işlemlerinde kullanışlı ve performanslı çözümler sunmaktadır. Numpy ile, matris işlemleri, dizi işlemleri, veri işlemleri, ikili işlemler, katar işlemleri, rasgele sayı üretimi, istatistiki işlemler ve benzeri birçok işlem gerçekleştirilebilmektedir. Numpy adıyla, 2006 yılından beri geliştirilmekte ve desteklenmektedir. Numpy aracı

Google Colab ortamında kendiliğinden bulunduğundan, kurulum işlemi gerekmemektedir. Numpy aracının kullanımına ait örnek Şekil 4.6'da gösterilmektedir.

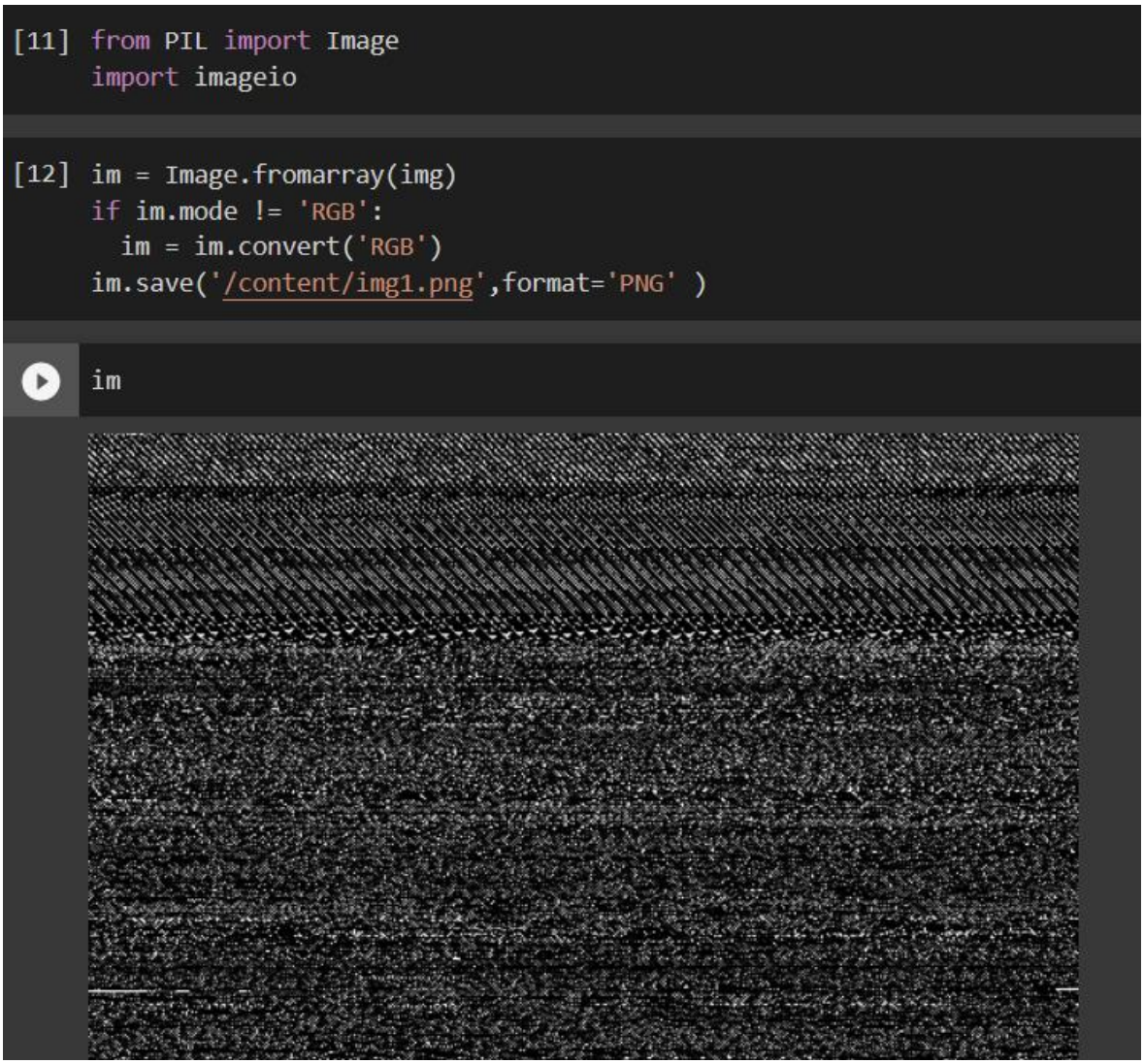


Şekil 4.6. Numpy aracı kullanarak dosyaları ikili okuma

Şekil 4.6'da gösterildiği üzere Numpy aracıyla dosyalar ikili okunabilmekte ve matrisler oluşturulabilmektedir. Numpy kütüphanesi "np" olarak yazılımı eklendikten sonra, "np.frombuffer(dex, dtype=np.uint8)" komutuyla uygulama yazılımlarından elde edilen dex dosyası ikili formatta okunmaktadır. İkili formatta okunan verinin, "np.sqrt()" metoduyla karekökü alınmakta ve "np.ceil()" metoduyla bir üst sayıya yuvarlamaktadır. Matris boyutlarını belirtmek için "astype(int)" metoduyla, tam sayıya dönüştürülmektedir. Bu değer kare şeklinde olan görselin bir eksenin boyutunu vermektedir. Numpy aracında "np.zeros()" metodu kullanılarak, bayt kod dosyasının uzunluğunda ve tüm değerleri sıfır olan boş bir vektör oluşturulmaktadır. İkili formatta okunan veriler, bu oluşturulan vektöre kopyalanmaktadır. Numpy aracının "reshape()" metodu, vektör üzerinde çağrılarak, kare matrisi temsil eden eksen değerleri parametre olarak sunulmaktadır. İlgili metotla vektör şeklinde olan ikili formattaki veri, matris formatına dönüştürülmektedir.

4.1.5. Python Imaging Library

PIL, görseller üzerinde işlem yapabilmeyi sağlayan, açık kaynak kodlu bir Python kütüphanesidir. Kütüphane, oldukça geniş dosya formatı desteği sunmaktadır. Bir görseli formatlarken, dosya formatı, standardı ve varsa formatlama da ihtiyaç olan diğer parametreleri (sıkıştırma oranı, sıkıştırma algoritması vb.) desteklemektedir. Görseller üzerinde, geometrik dönüşümler, renk dönüşümleri, kesme, birleştirme işlemleri yapılabilmekte ve dosyaları görsele dönüştürme gerçekleştirilebilmektedir. İkili dosya formatından, arşiv formatından dosya okuyabilmekte ve dosyalar üzerinde toplu işlemler gerçekleştirebilmektedir. PIL, Google Colab ortamında bulunduğundan kurulum gerektirmemektedir. Şekil 4.7'de PIL kullanımı gösterilmektedir.



Şekil 4.7. Python Imaging Library kullanarak ikili veriden görsel oluşturma

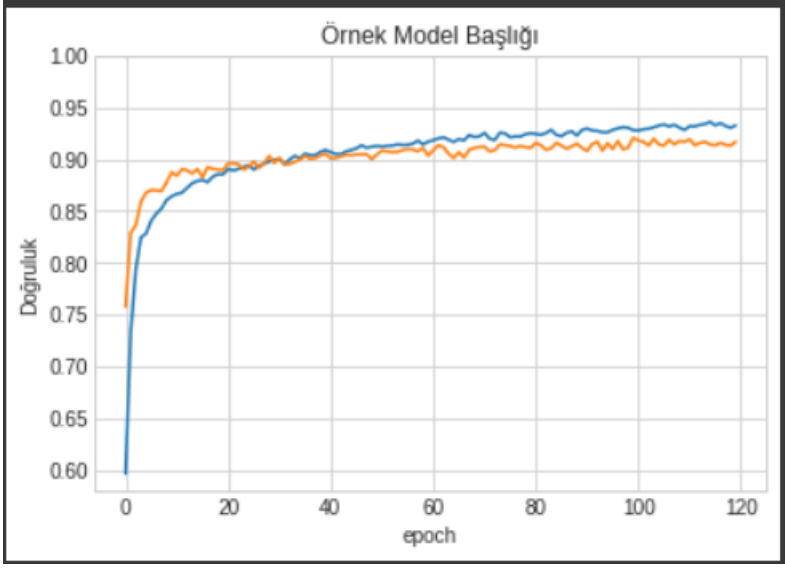
Şekil 4.7'de gösterildiği üzere PIL aracını kullanarak ikili dosyalardan görseller oluşturulmaktadır. Bayt kod olarak elde edilen uygulama dosyalarının görsellere dönüştürülmesi için PIL kullanılmaktadır. PIL kütüphanesi ve imageio kütüphaneleri yazılıma dahil edilmektedir. PIL kütüphanesinden "fromarray" metodu çağrılmakta ve parametre olarak her elemanında, 0 ile 255 arasında, 8 bit verilere sahip olan ve Numpy aracıyla oluşturulan matris metoda parametre olarak sunulmaktadır. İlgili metot sayesinde matris artık tek kanallı görsel olarak bellekte tutulmaktadır. ESA modelleri 3 kanal kullandığından, bellekteki görsel "convert()" metoduyla kırmızı, yeşil ve mavi kanallı görsele dönüştürülmektedir. Bu dönüştürülme işleminde tüm kanallara aynı matris kopyalandığından, sonuç olarak gri tonlamalı bir görsel oluşmaktadır. PIL aracının "save()" metodu yardımıyla, dosya yolu ve dosya türü belirtildikten sonra, bellekteki görsel, bir sabit alana kaydedilmektedir.

4.1.6. MatPlotLib ve Seaborn

MatPlotLib ve Seaborn verilerin görselleştirmelerinde kullanılan Python kütüphaneleridir. Seaborn kütüphanesi MatPlotLib kütüphanesi temel alınarak oluşturulmuştur. Bu sebeple, her iki kütüphane de büyük oranda birbirine bütünleşik ve birlikte çalışabilmektedir. Kullanıcıların hangi kütüphaneyi tercih edecekleri, kullanıcıların görsel beğenilerine ve tercihlerine göre şekillenmektedir. Bu tez çalışmasında, modellere ait karmaşa matrisleri Seaborn aracıyla, modellere ait eğitim grafikleri MatPlotLib aracıyla oluşturulmaktadır. Her iki kütüphane de Numpy aracıyla ve Numpy veri yapılarıyla uyumlu çalışabilmektedir. Seaborn ve MatPlotLib kütüphaneleri, Colab ortamında hazır bulunduğundan kurulum işlemi gerektirmemektedir. Şekil 4.8'de MatPlotLib ile grafik oluşturma gösterilmektedir.

```
import matplotlib.pyplot as plt
import numpy as np

init_history_path = '/content/drive/MyDrive/ICAIAME2021/Model/Xception/Hi
history= np.load(init_history_path,allow_pickle=True).item()
epochs= len(history['accuracy'])
acc = history['accuracy']
val_acc = history['val_accuracy']
plt.title('Örnek Model Başlığı')
plt.style.use('seaborn-whitegrid')
plt.plot(acc, label='Eğitim Doğruluğu')
plt.plot(val_acc, label='Geçerleme Doğruluğu')
plt.ylabel('Doğruluk')
plt.xlabel('epoch')
plt.ylim([min(plt.ylim()),1])
plt.savefig(('/content/model.png'),bbox_inches='tight',transparent=True)
```



Şekil 4.8. MatPlotLib kullanarak model grafiği oluşturma

Şekil 4.8'de MatPloLib aracıyla model grafiği oluşturmak için kullanılan kod ve çıktısı verilmektedir. Grafik oluşturmak için MatPlotLib aracına ait olan pyplot kütüphanesi "plt" olarak yazılıma dahil edilmektedir. Grafiklerle çalışırken Numpy kütüphanesini de eklemek oldukça yardımcı olmaktadır. Modellere ait eğitim verileri önceden kaydedildiğinden, Numpy aracının "load" metoduyla bu veriler Numpy formatında yüklenmektedir. Bu verilerden epoch sayısı, eğitim doğruluğu ve geçerleme doğruluğu bilgileri alınmaktadır. Pyplot aracının "title()" metoduyla grafiğin başlığı yazılmaktadır. Önceden tanımlı grafik stillerinden birini kullanmak için "style.use()" metoduna,

kullanılacak stil parametre olarak verilmektedir. Grafiği göstermek için “plot” metodu kullanılmakta, parametre olarak veriler ve etiket ismi belirtilmektedir. Eksenleri belirtmek için, yatay eksende “xlabel()”, dikey eksende “ylabel()” metotları kullanılmakta ve eksen isimleri parametre olarak verilmektedir. Eksendeki değerleri belirli sınırlar içerisinde göstermek istendiğinde, “ylim()” metodu kullanılmakta, parametre olarak eksen sınırları belirtilmektedir. Oluşturulan grafiği kaydetmek için “savefig()” metodu kullanılmakta, parametre olarak dosya yolu verilmesi gerekmektedir. Şekil 4.9‘da Seaborn aracıyla ısı haritası oluşturma işlemi gösterilmektedir.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()
data = np.random.rand(10, 12)
plt.figure(figsize=(9, 7))
sns.heatmap(data, vmin=0, vmax=1,annot=True, fmt='.2f' )
plt.show()
```

Şekil 4.9. Seaborn kullanarak ısı haritası oluşturma

Şekil 4.9'da gösterildiği üzere Seaborn kütüphanesi kullanılarak ısı haritaları oluşturulmaktadır. Örnek olması açısından Numpy aracıyla 10x12 boyutlarında 0 ile 1 arasında rasgele sayılardan oluşan bir matris oluşturulmaktadır. Seaborn kütüphanesi "sns" olarak yazılıma dahil edilmektedir. Isı haritasında bir stil kullanılmak istendiğinde, "set()" metodundan yararlanılmaktadır. Seaborn ile MatPlotLib kütüphanelerinin ortak çalışabildiği örnekte görülebilmektedir ve "plt.figure()" metoduna, parametre olarak istenen şekil boyutları verilmektedir. Isı haritasını oluşturmak için "heatmap()" metodu çağrılmaktadır ve gösterilecek veri, ısı haritasının minimum ve maksimum değerleri, değerlerin şekil üzerinde gösterilmesi durumu (annot=True), gösterilecek değerlerin nasıl biçimlendirileceği (fmt=".2f") gibi parametreler verilmektedir.

4.2. Veri Setinin Belirlenmesi

Derin öğrenme modellerinin kendi kendine özellik çıkarımı yapabilmesi için veri seti seçimi, çalışmanın başarısı açısından önemli hususlardan olmaktadır. Android ekosistemi oldukça hızlı gelişmekte, yeni sürüm işletim sistemleri ve bu sistemlere uygun uygulama yazılımları ve kötücül yazılımlar üretilmektedir. Bu hızlı gelişim, veri setlerinde güncel tehditleri içermediğinden eskimesine sebep olmaktadır. Çalışmalarda güncel kötücül yazılımları içeren ve yeterince örnek bulunduran veri setlerini tercih etmek gerekmektedir. Veri seti seçiminde diğer önemli bir hususta, veri setinin erişilebilirliği olmaktadır. Android veri setlerinden bazılarına artık destek verilmemekte ve paylaşılmamaktadır. Veri setinde, görsellere dönüştürülebilmek için işlenmemiş Android uygulama paketlerinin bulunması gerekmektedir.

Bu tez çalışmasında, erişim kolaylığı, örnek çokluğu, örneklerin güncelliğinden ve işlenmemiş örnekleri barındırmasından ötürü CICMalDroid2020 [1] adlı veri seti kullanılmaktadır. Çizelge 4.1'de CICMalDroid2020 veri setine ait örnek dağılımları gösterilmektedir.

Çizelge 4.1. CICMaldroid veri seti örnek dağılımı

Bankacı	1 515
Riskli	4 362
Reklamcı	2 506
SMS	4 822
Zararsız	4 042
TOPLAM	17 247

Çizelge 4.1'de örnek kategorisi ve ilgili kategorideki örnek sayısı gösterilmektedir. Veri seti Aralık 2017 ile Aralık 2018 tarihleri arasında toplanan 17.341 adet örnek barındırmakta olup örnekler zararsız yazılımlar, reklam yazılımları, riskli yazılımlar, sahte banka yazılımları, kısa mesaj yazılımları olarak 5 kategoride sınıflandırılmaktadır. Zararsız yazılımlar, kullanıcı cihazlarına herhangi bir zarar vermesi beklenmeyen yazılımları belirtmektedir. Sahte banka yazılımları (Bankacılık), banka ara yüzlerini taklit ederek kullanıcıların bankacılık hesaplarına ulaşmayı hedefleyen yazılımları belirtmektedir. Reklam yazılımları (Reklamcı), kullanıcı cihazlarında izinsiz ve istemsiz şekilde engellenemez reklamlar gösteren ve kullanıcı hassas bilgilerini izni dışında alan, kullanan yazılımları belirtmektedir. Kısa mesaj yazılımları (SMS), kullanıcıların telefonlarından hedeflenen ücretli kısa mesaj servislerine üye ederek gelir elde etmeyi ya da kısa mesajları izinsiz alarak uzaktaki sunuculara göndermeyi amaçlayan yazılımları belirtmektedir. Riskli yazılımlar (Riskli), özünde belli bir davranış göstermeyen fakat geliştiricisinin uzaktaki komut sunucusunda vereceği talimatlar doğrultusunda fidyeci yazılım, truva atı gibi davranabilen, bunun için kullanıcıları, cihazlarına bu zararlıları yüklemeye yönlendirebilen yazılımları belirtmektedir.

Veri seti kategori temelli bir sınıflandırma içermektedir. Bu durum çeşitli avantaj ve dezavantajları beraberinde getirmektedir. Bir zararlının DroidKungFu ailesine ait olduğunu sınıflandırmak, birçok kullanıcı açısından anlam ifade etmezken, bunun bir truva atı olduğunu belirtmek daha fazla fikir vermektedir. Benzer şekilde aile temelli veri setlerinde 10 aile varsa, sınıflandırma modeli yeni bir örneği bu 10 aileden birine dahil etmeye çalışmaktadır. Kötücül yazılım kategorileriyse aileler kadar sık değişmemektedir. Kategorik sınıflandırma, yeni örneklerin daha iyi sınıflandırılması için fırsatlar sunmaktadır. Fakat veri setinde aileler belirtilmediğinde, veri setindeki örneklerin ailesel dağılımları, veri setinin dengesi hakkında bilgi sahibi olunamamaktadır. Veri setinin

bölümlendirmesi, eğitilmesi aşamasında bir örnek yazılımdan çok fazla gelmesi o örneğin ezberlenmesine, diğer örneklerin daha az öğrenilmesine ve sonuç olarak daha başarısız bir sınıflandırma performansına neden olabilme, sınıflandırıcının genellenebilirliğini azaltabilme dezavantajını taşımaktadır.

4.3. Uygulama Dosyalarından Görsel Oluşturma Yöntemleri

Uygulama dosyalarından görseller elde edilmesinde kullanılan yaklaşımlar olan Android uygulama dosyasından ve classes.dex dosyasından olmak üzere iki yöntem bu bölümde anlatılmaktadır.

4.3.1. Uygulama dosyasının görsele dönüştürülmesi (Apk2Image)

Uygulama dosyası görsele dönüştürülürken izlenmekte olan aşamalar;

1. Uygulama dosyasının boyutu hesaplanır. Bu boyutun karekökü alınarak bir üst sayıya yuvarlanır.
2. Numpy adlı araçla, yuvarlanan sayı eksen boyutlarını belirtmek üzere boş bir matris oluştururlur.
3. Numpy aracıyla uygulama dosyası 8 bit olarak okunur ve bu değer matrisin hücrelerine sırayla kaydedilir.
4. Python Image Library adlı araç yardımıyla ilgili matris gri tonlamalı görsele dönüştürülür. Bu dönüştürme işleminde, aynı matris verisi her üç kanala (Kırmızı, yeşil, mavi) kopyalanır.
5. Python Image Library adlı araçla görsel, çalışmada kullanılacak boyutlara göre tekrardan boyutlandırırlır.

4.3.2. Dex dosyasının görsele dönüştürülmesi (Dex2Image)

Dex dosyası görsele dönüştürülürken izlenmekte olan aşamalar;

1. Androguard adlı araç yardımıyla uygulama dosyasından, classes.dex dosyası çıkartılır. Birden fazla dex dosyası varsa, sadece ilk dex dosyası alınır.
2. Dex dosyasının boyutu hesaplanır. Bu boyutun karekökü alınarak bir üst sayıya yuvarlanır.
3. Numpy adlı araçla, yuvarlanan sayı eksen boyutlarını belirtmek üzere boş bir matris oluştururlur.

4. Numpy aracıyla uygulama dosyası 8 bit olarak okunur ve bu değer matrisin hücrelerine sırayla kaydedilir.
5. Python Image Library adlı araç yardımıyla ilgili matris gri tonlamalı görsele dönüştürülür. Bu dönüştürme işleminde, aynı matris verisi her üç kanala (Kırmızı, yeşil, mavi) kopyalanır.
6. Python Image Library adlı araçla görsel, çalışmada kullanılacak boyutlara göre tekrardan boyutlandırırlır.

4.4. Veri Setinin Hazırlanması ve Ön İşlemler

Seçilen veri setinde, 17.341 adet Android uygulaması bulunduğu belirtilmesine rağmen açık paylaşıma sunulan dosyalarda 1515 adet reklamcı, 2506 adet bankacı, 4362 adet riskli, 4822 adet SMS, 4042 adet zararsız olmak üzere toplam 17247 adet uygulama bulunmakta, sabit disk üzerinde yaklaşık 70,6GB yer kaplamaktadır. Bu örneklerin modeller tarafından kullanılabilmesi için ön işlemlerden geçmesi gerekmektedir. Literatürde kötücül görsellerin elde edilmesinde, ham dosyadan görsele (apk2image) [38, 43] ve bayt kodu dosyasından görsele (dex2image) [40-42] olmak üzere iki yaklaşım işlem kolaylığı ve azlığı sebebiyle ön plana çıkmaktadır. Çalışman ileri aşamasına geçmeden önce bu iki yaklaşımdan hangisinin daha iyi sonuç verebileceği araştırılmaktadır. Her iki yaklaşımda kullanılarak, aynı uygulama dosyasına ait iki adet görsel ve bu görsellerden oluşan iki adet veri seti oluşturulmaktadır.

Dex dosyalarından görseller elde edilmesi, Bölüm 4.3.2'de anlatıldığı şekilde gerçekleştirilmektedir. İşlemlere dair ayrıntılarsa bu bölümde aktarılmaktadır. İşlemler sırasında, bir den fazla dex dosyası bulunduran uygulama sayısı 854 adet olup, 739 adeti zararsız uygulamalarda bulunduğundan çalışma sonuçlarını etkilemeyeceği düşünülmektedir. Benzer şekilde, çıkartma işleminde 464 adet uygulama paketi sıkıştırma yapısı bozuk olduğu için açılamamış ve veri setinden çıkartılmaktadır. Geriye kalan 16.783 adet classes.dex dosyasinin toplam büyüklüğü, veri setinin işlenmemiş haline göre yaklaşık 6 kat daha küçüktür ve yaklaşık 12GB olmaktadır. Classes.dex dosyaları elde edildikten sonra, her bir dex dosyası NumPy adlı araçla byte olarak okunarak elde edilen toplam byte sayısının karekökü alınıp, çıkan değer bir üst sayıya tamamlanmaktadır. Dosya 98750 Byte olduğunda bu değerin karekökü yaklaşık 314,14, elde edilen değerse 315 olmaktadır. Bu değer kare şeklinde matris boyutumuzun eksenlerinin ölçüsünü (315x315) ifade etmektedir. Çıkan değerin 224'ten küçük olduğu durumda, en küçük eksen değeri 224

atanmakta, matris doldurulduktan sonra geri kalan boşluklara 0 değeri atanmaktadır. Tez çalışmasında kullanılan önceden eğitilmiş modellerin çoğunda 224x224 matris değerleri sorunsuz çalışabildiğinden, bu değer en küçük eksen değeri olarak belirlenmektedir. Tez çalışmasında kullanılan modeller 3 kanallı görseller istemektedirler. Python Image Library adlı araçla bu matrisler gri tonlamalı kırmızı, yeşil, mavi olmak üzere 3 kanallı jpeg formatında görsellere dönüştürülmektedir. Görsellerin gri tonlamalı olmasının nedeni her üç kanala aynı matris değerlerinin verilmesidir. Bu aşamada görsellerin oluşturulması tamamlanmış bulunmaktadır.

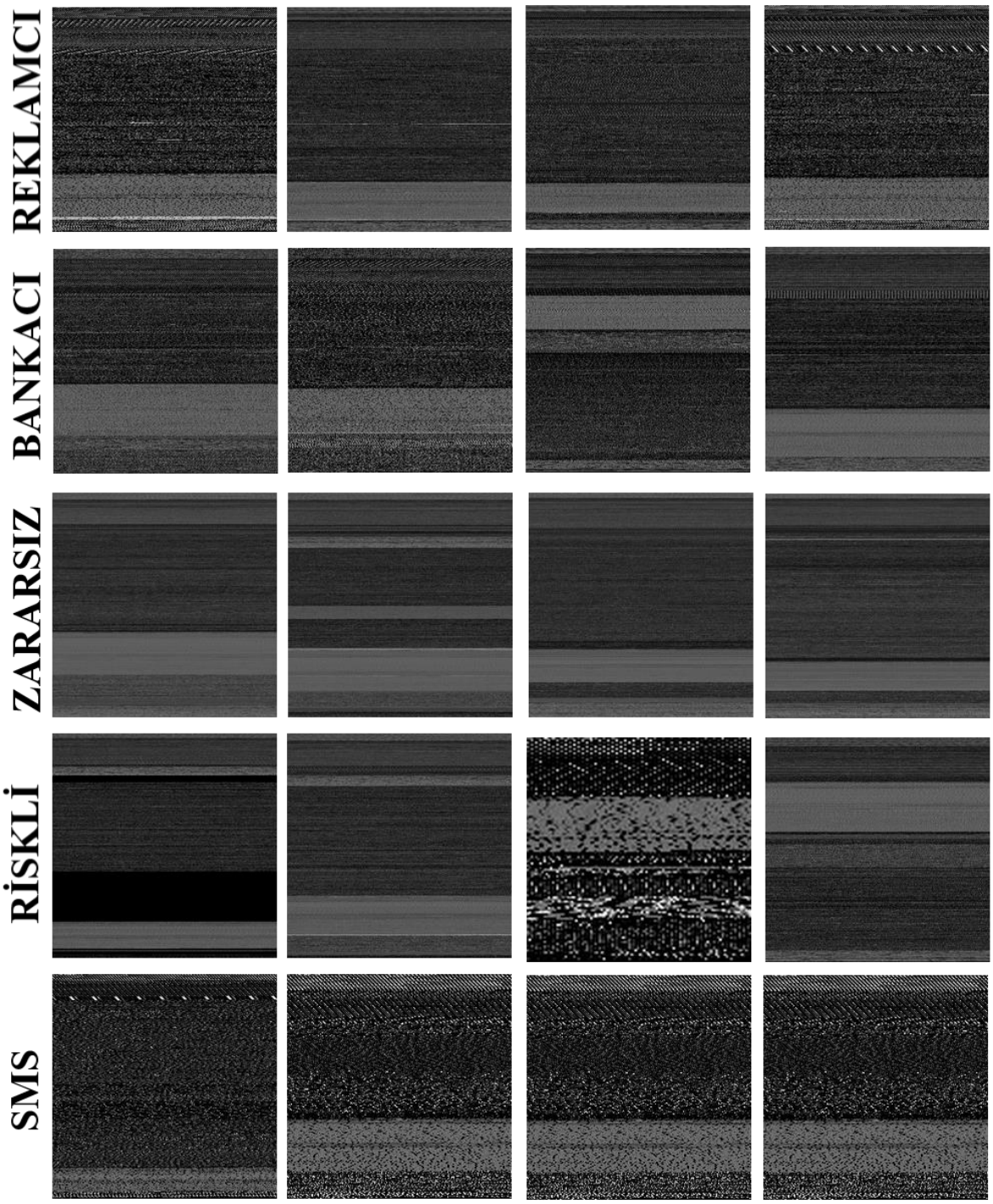
Oluşturulan görseller kare formatında ve dex dosyasının boyutuna göre çeşitli ebatlarda bulunmaktadırlar. Birçok modelin varsayılan değeri 224x224 piksel olduğundan ve daha büyük boyutlarda yapılan denemelerde ciddi süre artışına rağmen modelin performansında aynı ölçüde artış sağlanamadığından, bu farklı boyuttaki görseller 224x224 boyutlarına göre tekrardan boyutlandırılmaktadır. Boyutlandırma algoritması olarak, diğer algoritmalara göre (KNN, bilinear, kubik v.b.) görece daha başarılı sonuçlar veren Lanczos algoritması tercih edilmekte ve sıkıştırma kalitesi olarak %90 kullanılmaktadır. Bu aşamada görsel veri seti tamamlanmakta ve model eğitimine hazır hale gelmektedir.

Apk2Image elde etme işlemi de benzer şekilde yapılmaktadır. Tek farkı dex dosyası yerine Apk dosyası üzerinde ilgili işlemler gerçekleştirilmektedir. Dex dosyasını elde etme işlemi sistem kaynaklarına fazladan yük getirmektedir. Bu sebeple, işlem sırasında karşılaşılan ekstra yükün, performansa katkısını araştırmak gerekmektedir. Hem apk2image yöntemiyle hem dex2image yöntemiyle iki adet veri seti oluşturukmaktadır. Çizelge 4.2'de veri setlerine ait örneklerin kategorilere ve veri seti türüne göre dağılımları gösterilmektedir.

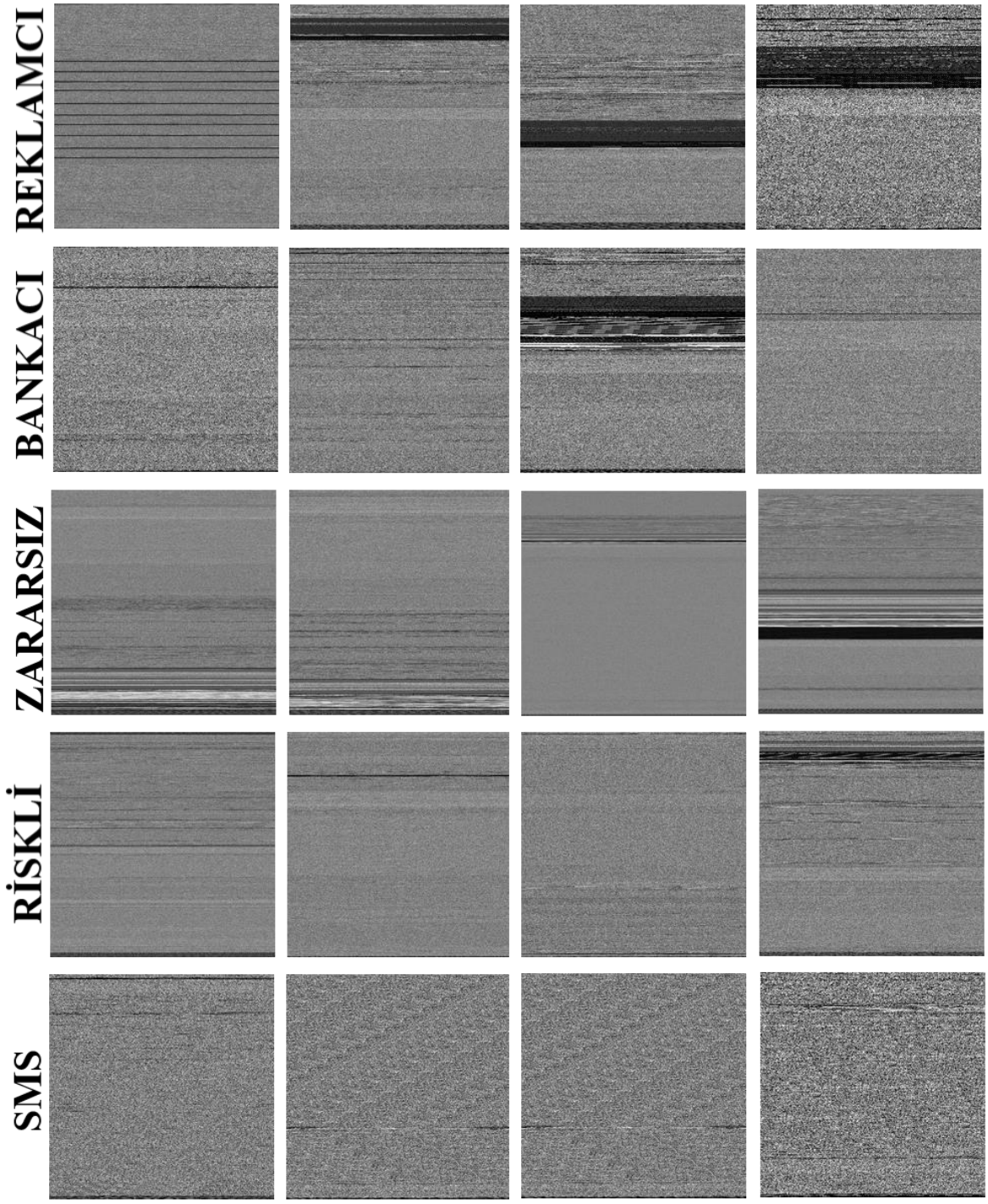
Çizelge 4.2. Veri setlerine ait örnek dağılımları

Kategori	Eğitim Veri Seti	Geçerleme Veri Seti	Test Veri Seti	Toplam
Reklamcı	1 211	242	61	1 514
Bankacı	2 003	400	101	2 504
Riskli	3 124	624	157	3 905
SMS	3 856	772	193	4 821
Zararsız	3 231	646	162	4 039
Toplam	13 425	2 684	674	16 783

Çizelge 4.2, kategori, eğitim veri seti, geçerleme veri seti, test veri seti başlıklarından oluşmaktadır. Kategori ilgili yazılımın ait olduğu kategoriyi belirtmektedir. Eğitim veri seti, modellerin eğitimleri sırasında kullanılmakta, modeller sadece bu veri setindeki örnekleri görebilmektedir. Geçerleme veri seti, modelin test veri setindeki örnekleri ezberlemesi halinde, görmediği örnekler karşısında göstereceği performansın araştırıldığı veri seti olmaktadır. Modele ait hiper-parametrelerin belirlenmesinde ve katmanların oluşturulmasında geçerleme veri seti kullanılmaktadır. Test veri seti, geçerleme veri setini kullanan geliştiricinin etkisini ortadan kaldırmak için kullanılmaktadır. Geliştiriciler bu veri setini en son aşamada, modele dair sonuçlara karar verirken kullanırlar, önceki aşamalarda görmemeleri gerekmektedir. Veri setinin oluşturulmasında, elde edilen görseller, her bir kategoriden, NumPy aracıyla, rasgele seçilen %20'si geçerleme veri seti ve geçerleme veri setinden rasgele seçilen %20'si test veri seti olarak bölümlendirilmektedir. Rastgele seçilen örneğin her iki görseli de ilgili veri setine aktarılmakta ve bu sayede veri setlerinin birbirinin aynısı olması sağlanmaktadır. Bu işlemlerden sonra eğitim, geçerleme ve test veri setinin örnek sayısının, toplam örnek sayısına oranı sırasıyla %80, %16, %4 olarak gerçekleşmektedir. Veri setlerine ait örnek görseller, aynı uygulamaların her iki yöntemle elde edilen görselleri olmak üzere Şekil 4.10 ve Şekil 4.11'de gösterilmektedir.



Şekil 4.10. Dex dosyalarından elde edilen görsel örnekleri



Şekil 4.11. Apk dosyasından elde edilen görsel örnekleri

Ön işlemler sonucunda elde edilen veri setlerinden hangisinin çalışmanın devamında kullanılacağına karar vermek için, beş farklı modelden yararlanılmaktadır. Bunun sebebi sonuçların model etkisinden kaynaklanmamasını sağlamaktır. Derin öğrenme çalışmaları rassal süreçler içerdiğinden, bu rassallığın sonuçlara etki etmesi, aynı deneyin, farklı modellerle yapılmasıyla engellenmektedir. Modellerin önceden sahip oldukları ağırlıklarının sonuçlara etki etmemesi için, model ağırlıkları sıfırlanarak tüm modeller sıfırdan aynı şartlarda eğitilmektedir. Modeller sıfırdan eğitildiğinden dolayı, eğitim süreci daha kısa olan modeller tercih edilmektedir. Elde edilen sonuçlar Çizelge 4.3'te gösterilmektedir.

Çizelge 4.3. Apk2Image ve Dex2Image ön işleme yöntemlerinin karşılaştırması

Model	APK Veri Seti Geçerleme Doğruluğu %	DEX Veri Seti Geçerleme Doğruluğu %
MobileNetV3Large	87,425	91,010
MobileNetV3Small	87,843	92,593
MobileNetV2	87,724	91,069
DenseNet121	91,517	93,220
ResNet50V2	87,873	91,846

Çizelge 4.3'te MobileNetV3Large, MobileNetV3Small, MobileNetV2, DenseNet121 ve ResNet50V2 modellerinin, iki yöntemle elde edilen geçerleme veri setleri üzerindeki kategorik sınıflandırma doğrulukları gösterilmektedir. Dex2Image yöntemi, tüm modellerde Apk2Image yönteminden ortalama %5,11 daha iyi sonuç vermektedir.

Karşılaştırmalar sonucunda, tüm modellere uygulanacak ve tez çalışmasında kullanılacak veri seti, dex (bytekod) dosyalarından elde edilen olarak belirlenmektedir. Tez çalışmasının devamında veri seti denildiğinde bu veri seti ifade edilmektedir. Veri setinin tüm işlerden sonra toplam boyutu yaklaşık 994MB olmaktadır. Bu boyutun veri setinin işlenmemiş haline göre yaklaşık 70 kat daha küçük olduğu ve bu sayede paylaşımının daha kolay olacağı görülebilmektedir. Windows ortamındaki zararlılar için görsel veri seti (MalImg) bulunmaktayken Android ortamındaki zararlılar için yaygın paylaşılan ve güncel bir görsel veri seti bulunamamaktadır. Bu noktadan sonra elde edilen veri seti DroidMalImg olarak adlandırılmakta ve ilgili bağlantıdan [2] açık erişime sunulmaktadır.

5. GELİŞTİRİLEN YÖNTEM

Bu bölümde, modellerin eğitilmeleri için sunulan ve model eğitimlerine toplu bir yaklaşım getiren, geliştirilen yığın ince ayar öğrenim aktarımı yöntemi, temel hiper-parametre değerlerinin belirlenmesi, öğrenim aktarımı aşaması ve ince ayar aşaması açıklanmaktadır.

5.1. Yığın İnce Ayar Öğrenim Aktarımı Yöntemi

Son yıllarda, benzer alanlarda önceden eğitilmiş modellerin sayısı gittikçe artmaktadır. Önceden eğitilen modellerin paylaşımının, yeni araç ve yazılım çerçeveleriyle kolaylaşması, araştırmacıların modelleri paylaşmaktaki açıklığı ve benzeri nedenler sayesinde model havuzlarında bulunan model sayısı her geçen yıl artmaktadır. ImageNet adlı yarışmada yüksek başarı göstererek, derin öğrenme ve derin sinir ağlarının popülerliği arttıran AlexNet'ten günümüze 10 yıl geçmesine rağmen, sadece TensorFlow/Keras uygulama kütüphanesinde, 33 adet önceden eğitilmiş model bulunmaktadır. Tez çalışmasına başlandığında aynı kütüphanede 26 adet model bulunmakta olduğundan, yaklaşık bir yıl içerisinde 7 yeni modelin daha bu kütüphaneye dahil olduğu görülmektedir. Tüm bu modeller, ImageNet veri setinde eğitilmiş ağırlıklarıyla, kabul edilebilir kolaylıkta kullanılabilir şekilde bulunmaktadır. Yıllar içerisinde bu sayısının artacağını, model paylaşımının ve geliştirmesinin kolaylaşacağını öngörmek yanlış olmayacaktır. Bu artan model sayısı, araştırmalarında, geliştirmelerinde önceden ince ayar öğrenim aktarımı yöntemini kullanmak isteyen araştırmacılara ve geliştiricilere yeni tasarım tercihleri sunmaktadır. Bunlardan birisi en başarılı modelleri, daha uzun süre hiper-parametre optimizasyonuna sokmak ve az sayıda ya da tek modelle yüksek başarı sağlamaktır. Bu tez çalışmasında geliştirilen ve uygulanan yöntemse, birçok modelin yığın, toplu bir şekilde ince ayar öğrenimi olmaktadır. Bu sayede, modellerin ilgili sorun karşısında sergileyeceği performans öğrenilebilmekte, model tercihinde sezgisel yaklaşıma göre daha gerçekçi bir yaklaşım ortaya konulmaktadır.

Toplu ince ayar öğrenim aktarımının temel farkı, tek bir modele uzun süreler hiper-parametre optimizasyonu yapmak yerine, aynı sürede birçok modeli eğitmek ve bunların birleşimleriyle daha iyi sonuçlara ulaşmak hedeflenmektedir. Bu yöntemde temel hiper-

parametre değerleri belirlenerek tüm hazır modeller bu hiper parametre değerlerine göre eğitilmektedirler.

Model birleşimi kullanmak istemeyen ve tek bir model kullanmak isteyen araştırmacılar içinde, geliştirilen yöntem, çözülmesi hedeflenen sorun ve mevcut modeller arasındaki bağlantıyı görmede genel bir bakış sağlamaktadır. Mevcut veri setinin boyutuna göre aynı modeller kullanılmasına rağmen farklı sonuçlar alınmaktadır. İlgili çalışmalar başlığında aktarılan çalışmalardan anlaşılabileceği üzere, veri seti boyutu değiştikçe araştırmacıların elde ettiği sonuçlarında değiştiği görülmektedir. Yöntemin kullanılması sayesinde, modellerin ilgili sorun karşısındaki başarımları hakkında, öngörü elde eden araştırmacılar, modelleri birleştirmek yerine ilgili sorun karşısında daha başarılı model üzerine yoğunlaşabilirler. Tamamen sezgisel yaklaşımla kullanılacak modeli belirlemek yerine, geliştirilen yöntemle, modele karar vermek daha başarılı sonuçlar alınmasını sağlayacaktır.

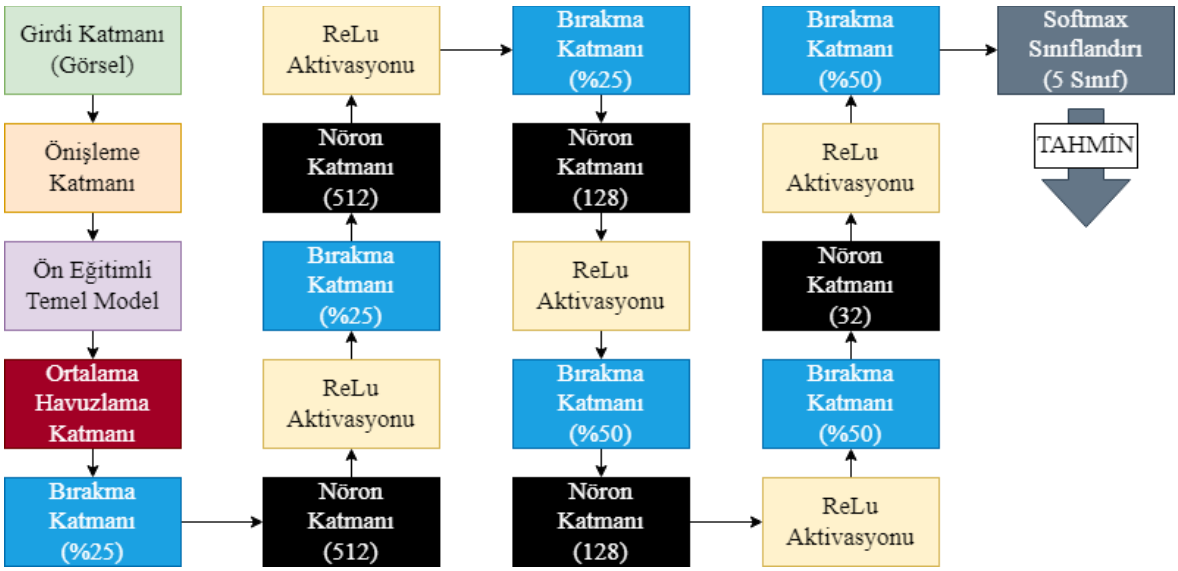
Geliştirilen yöntem, temel hiper parametre değerlerinin belirlenmesi ve eğitim aşamalarından oluşmaktadır. Eğitim aşaması da öğrenim aktarımı aşaması ve ince ayar aşaması olmak üzere iki aşamada gerçekleşmektedir.

5.2. Temel Hiper Parametre Değerlerinin Belirlenmesi

Temel parametre değerlerini belirlemek için öncelikle bir model seçilmesi gerekmektedir. Seçilen ve üzerinde öğrenim aktarması uygulanan model (Xception, ResNet gibi), genellikle temel model olarak adlandırılmaktadır. Görselleri modellere sunmadan önce, modele göre ön işlemlerden geçmesi gerekmektedir, Tensorflow kütüphanesi içinde barındırdığı ön işleme katmanıyla, görsellere ait girdi değerlerini modellerin istediği formata getirmektedir. Tüm girdiler, Tensorflow kütüphanesinde bulunan, modellere ait ön işleme katmanlarıyla işlenmektedir. Tez çalışmasındaki birçok model, 224x224 piksel boyutundaki görsellerle çalışabilirken, NASNetLarge modeli 331x331 piksel boyutundaki görselleri kabul etmektedir. Bunun için ilgili görseller otomatik olarak tekrardan boyutlandırılmaktadır. Benzer şekilde yapılarında tekrar boyutlandırma katmanları bulunan EfficientNet modelleri de görselleri kendiliğinden tekrar boyutlandırmaktadırlar.

Çalışmada, 25 modelden biri rastgele seçilmektedir. Seçilen model olan Xception modelinde temel model parametreleri ve eklenecek tam bağlı katman yapısı

belirlenmektedir. Bu katmanın belirlenirken Xception modeline aşırı uymamasına dikkat edilmektedir. Bunun sebebi, bu katmanların tüm modellere kullanılacak olmasındandır. Oldukça az sayıda deneme, yanılma yapılarak kullanılacak eklentiye ve diğer hiper parametre değerlerine karar verilmektedir. Eklentinin kullanılmadığı durumda, ince ayar aşamasında modeller çok eğitim veri setini çok hızlı ezberlemekte ve geçerleme veri setinde kötü performans sergilemektedir. Bu ek katman, düzenleyici ve sınıflandırıcı eklenti olacak belirtilecektir. Genel model mimarisi ve tüm modellere eklenen bu düzenleyici ve sınıflandırıcı katmanlar Şekil 5.1'de gösterilmektedir.



Şekil 5.1. Genel model mimarisi ve eklenen düzenleyici ve sınıflandırıcı katmanlar

Şekil 5.1'de gösterildiği üzere modeller, girdi katmanı, ön işleme katmanı, temel model, küresel ortalamaya göre havuzlama katmanı, yapay sinir ağı katmanları, bırakma katmanları ve sınıflandırıcı katmandan oluşmaktadır. Girdi katmanı görsellerin sunulduğu katmandır. Ön işleme katmanında girdiler modele uygun hale getirilmektedir. Temel modelin üst katmanı modelden çıkartılmakta, iki boyutlu küresel ortalama (GlobalAverage2D) katmanı, bırakma (Dropout) katmanı, sırasıyla 512, 512, 128, 128, 32 adet olacak şekilde nöron katmanları ve bu katmanların çıkışına bırakma katmanları yerleştirilmektedir. Son olarak, 5 adet sınıf olduğundan dolayı sınıf sayısı kadar nöron çıkış katmanı olarak yerleştirilmektedir. Çıkış katmanının aktivasyon fonksiyonu, softmax olarak belirlenmektedir. Nöron katmanlarının aktivasyon fonksiyonu ReLu, bırakma katmanlarının bırakma oranı çıkış katmanından geriye, sırasıyla 0,50; 0,50; 0,50; 0,25; 0,25; 0,25 olarak belirlenmektedir. Kayıp fonksiyonu olarak seyrek kategorik çapraz

entropi, model optimizasyon fonksiyonu olarak Nesterov Adam fonksiyonu kullanılmaktadır. Öğrenme oranı parametresi öğrenim aktarımı aşamasında 0,001, ince ayar aşamasında 0,0001 olarak belirlenmektedir. Bu belirlenen katmanlar ve parametreleri diğer tüm modellere de aynı şekilde uygulanmaktadır.

5.3. Öğrenim Aktarımı Aşaması

Öğrenim aktarımı, bir modelin bir problemde elde ettiği tecrübelerin, benzer başka bir probleme uyarlanmasıdır [57,58]. Buradaki problem aslında görsellerin sınıflandırılması olduğu için, temel modellerin ImageNet adlı görsel sınıflandırma yarışmasında elde ettiği ağırlıklar, model katmanları eğitime kapatılarak korunmaktadır. Öğrenim aktarımı aşamasında sadece modelin çıkış katmanına eklenen katmanların eğitimlerine izin verilmektedir. Model eğitiminde, örnek yığın büyüklüğü 32 olacak şekilde toplam 999 epocha kadar eğitim şansı tanınmaktadır. Öğrenim sırasında, geçerleme doğruluğu baz alınarak en iyi doğruluk değerinden sonra 20 epoch olacak şekilde bir beklemeyle en ufak ilerleme sağlamayan (minimum delta 0) modeller erken durdurma uygulanarak durdurulmaktadır. Durdurulan ya da eğitimini tamamlayan modeller, ağırlıklar korunarak ince ayar aşamasına geçirilmektedir.

5.4. İnce Ayar Aşaması

İnce ayar aşamasında, modelin öğrenim aktarımı aşamasında elde ettiği ağırlık değerleri korunarak, önceden eğitime kapatılan temel model katmanlarının tekrardan eğitime açılması ve bu sayede daha yüksek bir doğruluk değerine ulaşılması amaçlanmaktadır. Eğitime açılan katman sayısı, toplam eğitilen parametre sayısı Çizelge 5.1’de gösterilmektedir.

Çizelge 5.1. İnce ayar uygulanacak katman sayıları ve parametre bilgileri

Model	Temel Model Toplam Katman Sayısı	Temel Model Eğitilen Katman Sayısı	Temel Model Toplam Parametre	Tüm Model Toplam Parametre	İnce Ayar Eğitilen Toplam Parametre
Xception	132	20	20 861 480	22 259 693	8 724 589
VGG16	19	3	14 714 688	15 326 469	5 331 397
VGG19	22	4	20 024 384	20 636 165	7 691 205
ResNet50V2	190	29	23 564 800	24 963 013	13 482 437
ResNet101V2	377	57	42 626 560	44 024 773	18 603 461
ResNet152V2	564	85	58 331 648	59 729 861	21 105 605
InceptionV3	311	47	21 802 784	23 200 997	7 719 557
InceptionResNetV2	780	117	54 336 736	55 472 805	19 508 325
MobileNet	86	13	3 228 864	4 102 789	2 467 781
MobileNetV2	154	24	2 257 984	3 262 981	2 364 997
MobileNetV3Small	235	36	1 529 968	2 403 893	1 815 317
MobileNetV3Large	269	41	4 226 432	5 231 429	3 341 877
DenseNet121	427	65	7 037 504	7 911 429	2 220 933
DenseNet169	595	90	12 642 880	13 844 485	3 958 853
DenseNet201	707	107	18 321 984	19 654 661	5 138 117
NASNetMobile	769	116	4 269 716	5 160 025	2 486 453
NASNetLarge	1039	156	84 916 818	87 330 839	34 893 797
EfficientNetB0	237	36	4 049 571	5 054 568	2 943 909
EfficientNetB1	339	51	6 575 239	7 580 236	4 507 509
EfficientNetB2	339	51	7 768 569	8 839 102	5 251 653
EfficientNetB3	384	58	10 783 535	11 919 604	6 386 739
EfficientNetB4	474	72	17 673 823	18 940 964	9 533 573
EfficientNetB5	576	87	28 513 527	29 911 740	15 916 981
EfficientNetB6	666	100	40 960 143	42 489 428	21 778 723
EfficientNetB7	813	122	64 097 687	65 758 044	33 864 069

Çizelge 5.1'de temel model olarak adlandırılan 25 adet modele ait katman sayısı, eğitilen katman sayısı, toplam parametre sayısı ve düzenleyici, sınıflandırıcı ek katmanlarında dahil olduğu tüm modele ait toplam parametre sayısı ve ince ayar aşamasında eğitilen parametre sayısı sunulmaktadır. Çizelge 5.1'de belirtilen katman sayısı Tensorflow aracı tarafından raporlanan yazılımsal katmanlardır. Normalde eğitilemeyen katmanlar, katman olarak sayılmamaktayken Tensorflow aracı bu katmanları da katman olarak görmektedir. Bundan dolayı VGG16 teorik olarak 16 katman kabul edilirken, Çizelge 5.1'de 19 katman olarak gözükmektedir. Eğitime açılan katman sayısı çıkış katmanından geriye doğru temel modelde kaç adet katmanın eğitime açıldığı göstermektedir. Katmanların eğitime açılmasında konusunda genel görüş, ilk katmanların ancak temel özellikleri bildiği, çıkışa

yakın katmanlarınsa daha karmaşık özellikleri öğrendiği yönündedir. Geriye yayılım aşamasında, ağırlık güncellemelerini çıkış katmanından giriş katmanına doğru gerçekleştirmektedir. Bu iki sebepten dolayı çıkışa yakın katmanların eğitime açılması tercih edilmektedir. Modeller toplu olarak eğitildiği için eğitime açılacak katmanların nasıl belirleneceği sorunu ortaya çıkmaktadır. Temel modelin son 10 katmanı şeklinde sabit bir rakam belirlendiğinde, kimi modeller 20 katmanken kimi modeller 200 katmandan fazla olmaktadır. Bu noktada eğitime açılacak katman sayısının belirlenmesinde, katman sayısının yüzdesini almak adil bir yaklaşım olacağı düşünüldüğünden tercih edilmektedir. Eğitime açılacak katman sayısı yüzdesi, yönteme özgü bir hiper-parametre değeri olarak ortaya çıkmaktadır. Bu değer çalışmada %15 olarak belirlenmekte ve temel modelin toplam katman sayısının %15'i alınarak bir üst tam sayıya tamamlanıp, çıkıştan geriye belirlenen katman sayısı kadar temel model katmanı eğitime tekrar açılmaktadır. 132 katmanı olan Xception modelin açılan katman sayısı bu hesaplamaya göre 20 katman, 813 katmana sahip EfficientNetB7 modelinde 122 katman olarak gerçekleşmektedir. Çizelge 5.1'de ince ayar toplam eğitilen parametre sütununda, modele eklenen düzenleyici ve sınıflandırıcı katmanlar ve eğitime açılan katmanlar dahil olmak üzere bu aşamada eğitilen toplam parametre sayısı gösterilmektedir. Eğitim ilk aşamayla aynı şekilde aynı epoch sayısı, örnek büyüklüğü ve erken durdurma yaklaşımı uygulanarak gerçekleştirilmektedir.

6. MODELLERİN EĞİTİLMESİ ve DEĞERLENDİRİLMESİ

Bu bölümde, modellerin eğitilmesi ve değerlendirilmesi için gerçekleştirilen aşamalar, deney ortamı, karmaşa matrisi ve model değerlendirme metrikleri ve tüm modellere ait bulgular açıklanmaktadır.

6.1. Deney Ortamı

Tüm deneyler Google Colab ortamında gerçekleştirilmektedir. Google Colab, bulut ortamında Jupyter Notebook benzeri geliştirme aracıyla etkileşimli yazılım geliştirmeye izin veren bir Google hizmetidir. Colab, donanım kaynaklarının dağıtımını uygunluk durumuna göre gerçekleştirmektedir. Deneyler, çoğunlukla Nvidia V100-sxm2 16GB vram'li grafik işlem birimi, 51GB sistem belleği ve 8 çekirdek sanal işlemcisi olan deney ortamında gerçekleştirilmektedir. Deney çıktılarının ve sürelerin kıyaslanabilmesi için deney ortamında AI-Benchmark [56] adlı kıyaslama yazılımı çalıştırılmış ilgili araçla toplam 34196 skoru alınmaktadır, bu değer aracın kendi sıralamasında aynı grafik işlem birimi için 35086 şeklindedir. Bu nedenle sürelerin fiziksel deney ortamına yakın olacağı düşünülmektedir.

6.2. Model Değerlendirme Metrikleri ve Karmaşa Matrisi

Modellerin değerlendirilmesinde, makine öğrenmesi alanında model değerlendirmelerinde yaygın kullanılan temel model değerlendirme metriklerinden ve oranlanmış karmaşa matrislerinden faydalanılmaktadır.

Çalışmanın ana motivasyonu ve çözmeye çalıştığı problem olan, kötücül yazılımların kategorik sınıflandırılmasında, seyrek kategorik doğruluk sınıflandırma metriğinden yararlanılmaktadır. Seyrek kategorik doğruluk metriği, çıkış değerlerinde tam sayılı değerler kullanılarak hesaplandığından, one-hot kodlanmış değerlere göre hesaplanan kategorik doğruluktan ayrılmaktadır. Bu metrik Tensorflow/Keras yardımıyla model eğitimi sırasında hesaplanmakta ve raporlanmaktadır. Tensorflow/Keras günümüzde en yaygın kullanılan makine öğrenimi ve derin öğrenme çerçevelerinden olduğundan

raporlanan değerlerde hata olmaması beklenmektedir. Nitekim, elle yapılan kontrollerde, raporlanan değerlerin doğruluğu kontrol edilmektedir.

Seyrek kategorik doğruluk metriği bize modelin kategorik sınıflandırma performansı hakkında önemli bilgiler sunmaktadır, fakat çalışmalarda modellerin ikili sınıflandırma (tespit) performansları da önemli olabilmektedir. Kategorik sınıflandırmaya göre eğitilen modellerden, ikili sınıflandırma metrikleri doğrudan hesaplanamamaktadır. Bunun hesaplanması için ilgili problem ikili sınıflandırmaya, bir kategoriye karşı diğer kategoriler birleştirilerek dönüştürülebilmektedir. Bu şekilde her bir kategori için ikili sınıflandırma metrikleri elde edilebilmektedir. Kötücül yazılım çalışmalarında özel bir neden olmadıkça kullanılan ikili sınıflandırma, kötücül yazılımla, zararsız (iyicil) yazılımın birbirinden ayrılmasıdır. Bu sebeple, veri setinde bulunan zararsız örnek kategorisi ve diğerleri (bankacı, riskli, SMS, reklamcı) şeklinde bir karşılaştırmaya gidilmektedir. Bu noktada modeller tekrardan eğitilmemekte, sadece kategorik eğitilen modeller kullanılmaktadır. Kötücül ve zararsız örnek sayıları her bir kategoride farklı olduğundan, veri seti dağılımının dengesiz bir dağılıma sahip olduğu, elde edilen değerlerin bu şekilde hesaplandığı belirtilmelidir. Bu şekilde hesaplanan metrikler okuyuculara, ilgililere fazladan bilgi sunmak amacıyla hesaplanmakta olup, modellerin doğrudan ikili sınıflandırma amaçlayan bir şekilde eğitilmesiyle sonuçların farklılaşabileceği göz önünde bulundurulmalıdır.

İkili sınıflandırma metrikleri hesaplanırken, genellikle karmaşa matrisi ve bu karmaşa matrisinden elde edilen değerlerle diğer metriklerin hesaplanması şeklinde bir yol izlenir. Çizelge 6.1‘de karmaşa matrisinde kullanılan ifadeler gösterilmektedir.

Çizelge 6.1. Karmaşa matrisi örneği

		Tahmini sınıf	
		+	-
Gerçek sınıf	+	**GP** Gerçek Pozitif	**HN** Hatalı Negatif Tip II hata
	-	**HP** Hatalı Pozitif Tip I Hata	**GN** Gerçek Negatif

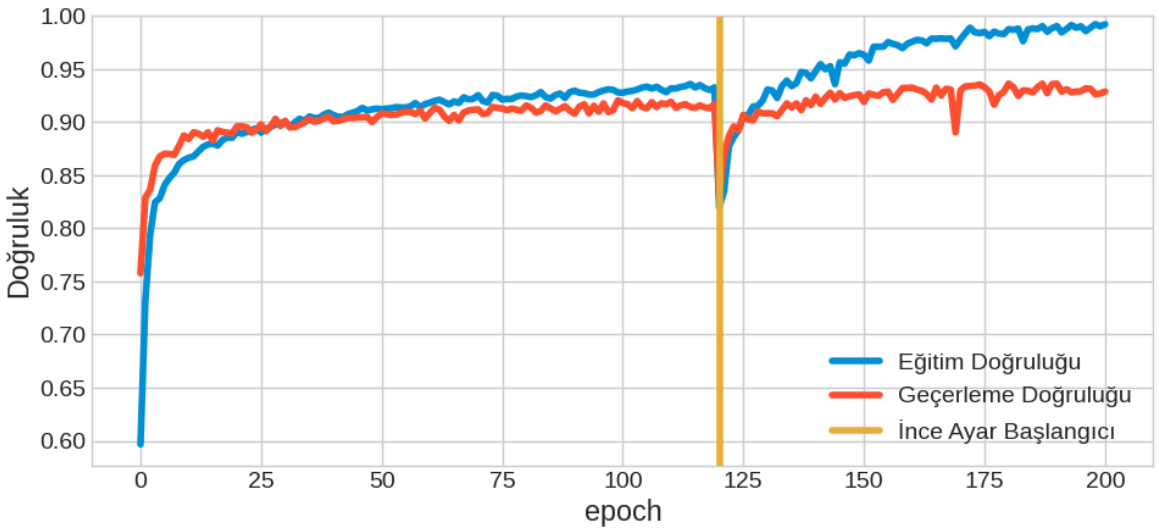
Çizelge 6.1, tahmini sınıf ve gerçek sınıf olarak iki bölümden oluşmaktadır. Gerçek sınıf, temel gerçeği, tahmini sınıf modelin gerçekleştirdiği tahmini göstermektedir. Çizelgedeki artı ve eksi işaretler sırasıyla pozitif ve negatifi ifade etmektedir. Pozitif değer, kötücül (zararlı) çıktığı durumu ve negatif değer, zararsız çıktığı durumu göstermektedir. Gerçek pozitif, model zararlı tahminlediği ve temel gerçeğin zararlı çıktığı durumdaki örnek sayısını, gerçek negatif, model zararsız tahminlediği ve temel gerçeğin zararsız çıktığı durumdaki örnek sayısını göstermektedir. Modelin zararlı tahminlediği fakat temel gerçeğin zararsız çıktığı durum hatalı pozitif olarak adlandırılmakta ve birinci tip hata olarak değerlendirilmektedir. Modelin zararsız tahminlediği fakat temel gerçeğin zararlı çıktığı durum hatalı negatif olarak adlandırılmakta ve ikinci tip hata olarak değerlendirilmektedir. Kötücül sınıflandırma problemi için birinci tip hatalar, ikinci tip hatalara göre daha tercih edilebilir durmaktadır. Bir başka deyişle ikinci tip hataların düşük olması daha iyidir. Zararlı tahminlenen yazılımın, zararsız çıkması bu yazılımın yüklü olduğu cihazda sorun oluşturmamaktayken, tersi durumda maddi ve manevi sorunlar oluşmaktadır. Çizelge 6.2'de karmaşa matrisinden faydalanarak elde edilen, F1-skoru, doğruluk, kesinlik, geri çağırma, gerçek negatif oranı, hatalı negatif oranı, hatalı pozitif oranı metriklerinin hesaplama formülleri ve metriğin açıklaması birlikte verilmektedir.

Çizelge 6.2. Model değerlendirme metrikleri, formülleri ve açıklamaları

Metrik	Hesaplama Formülü	Metriğin Açıklaması
Doğruluk	$\frac{GP + GN}{GP + GN + HP + HN}$	Modelin genel performansı hakkında bilgi verir.
F1-Skoru	$\frac{2 * GP}{2 * GP + HP + HN}$	İkili sınıflandırma problemlerinde kullanılmasıyla daha iyi değerlendirme şansı verir. Dengesiz sınıflarda kullanılması faydalı olmaktadır.
Kesinlik	$\frac{GP}{GP + HP}$	Pozitif tahminlerin ne kadar doğrulukla gerçekleştiğini gösterir
Geri Çağırma (Gerçek Pozitif Oranı)	$\frac{GP}{GP + HN}$	Gerçek pozitif tahminlerin gerçek pozitif örneklere oranını gösterir
Özgüllük (Gerçek Negatif Oranı)	$\frac{GN}{GN + HP}$	Gerçek negatif tahminlerin gerçek negatif örneklere oranını gösterir
Hatalı Negatif Oranı (Tip I Hata Oranı)	$\frac{HN}{HN + GN}$	Tip I hata oranını gösterir
Hatalı Pozitif Oranı (Tip II Hata Oranı)	$\frac{HP}{HP + GP}$	Tip II hata oranını gösterir

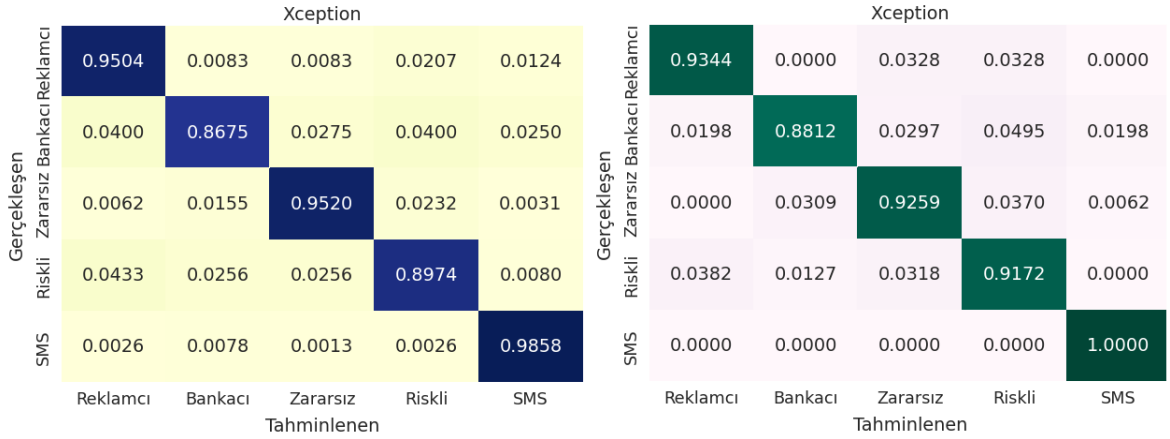
6.3. Xception Modeli Eğitim Sonuçları

Xception modeli, yığın ince ayar öğrenim aktarımında hiper-parametre değerlerinin belirlenmesinde rasgele seçilen modeldir. Hiper-parametre değerlerinin belirlenmesi bu modele göre gerçekleştirildiğinden, yöntemden diğer modellere göre daha başarılı sonuç almaktadır. Modele ait eğitim grafiği Şekil 6.1'de sunulmaktadır.



Şekil 6.1. Xception modeline ait eğitim grafiği

Şekil 6.1'de gösterildiği üzere öğrenim aktarımı aşamasının sonlarına doğru düz bir eğitim grafiği çizmeye başlamakta, turuncu çizgiyle gösterilen ince ayar aşamasından sonra eğitime açılan yeni parametrelerden dolayı önce performans kaybı yaşmanta daha sonra performans artışı gözlemlenmektedir. Modelin, geçerleme veri seti üzerinde, ince ayar aşamasından önceki kategorik sınıflandırma doğruluğu %92,06 iken, ince ayar aşamasından sonra model eğitimi %93,62 doğruluk değerine ulaşarak tamamlamaktadır. Model öğrenim aktarımı aşamasını, her bir epoch 25 saniye sürmek üzere toplam 120 epochta, ince ayar aşamasını her bir epoch 30 saniye sürmek üzere toplam 81 epochta tamamlamaktadır. İnce ayar yöntemi uygulanarak yaklaşık %1,7 performans artışı sağlanmaktadır. Modele ait oranlanmış karmaşa matrisi Şekil 6.2'de sunulmaktadır.



Şekil 6.2. Xception modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

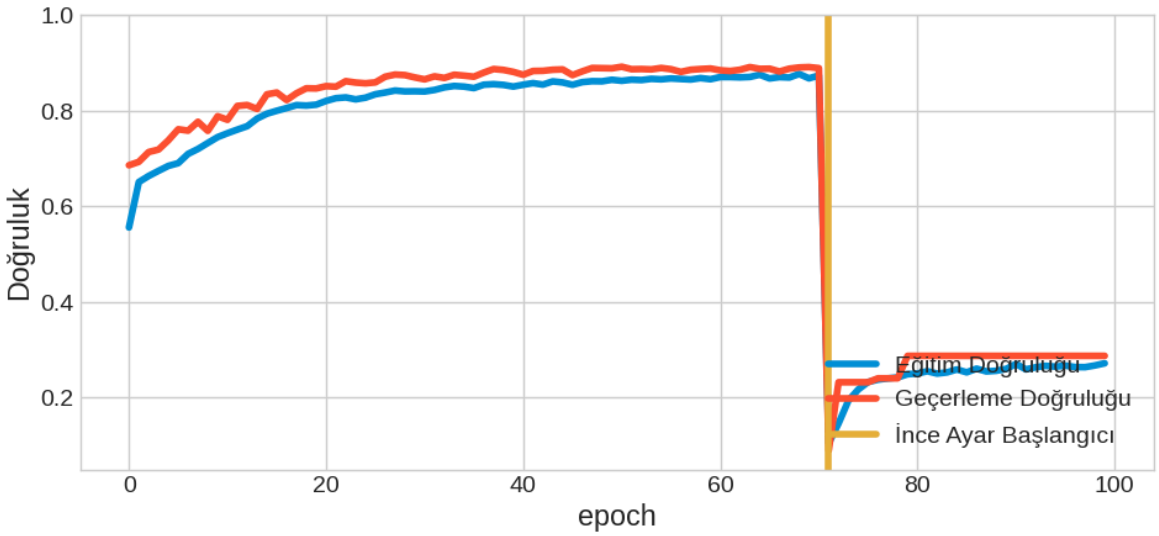
Şekil 6.2'de yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri gösterilmektedir. Modelin geçerleme ve test veri setlerindeki karmaşa matrisi incelendiğinde, model test veri seti üzerinde, bankacı, riskli ve SMS kategorilerindeki örnekleri sırasıyla %88,12, %91,72, %100 doğrulukla daha iyi tahmin ederken, reklamcı ve zararsız kategorilerindeki örnekleri sırasıyla %93,44, %92,59 doğrulukla daha kötü tahmin ettiği görülmektedir.

6.4. VGG Modelleri Eğitim Sonuçları

Bu bölümde VGG model ailesine ait değerlendirme sonuçları, öğrenim aktarımı aşaması, ince ayar sınıflandırma doğrulukları, eğitim aşamaları, karmaşa matrisleri sunulmaktadır.

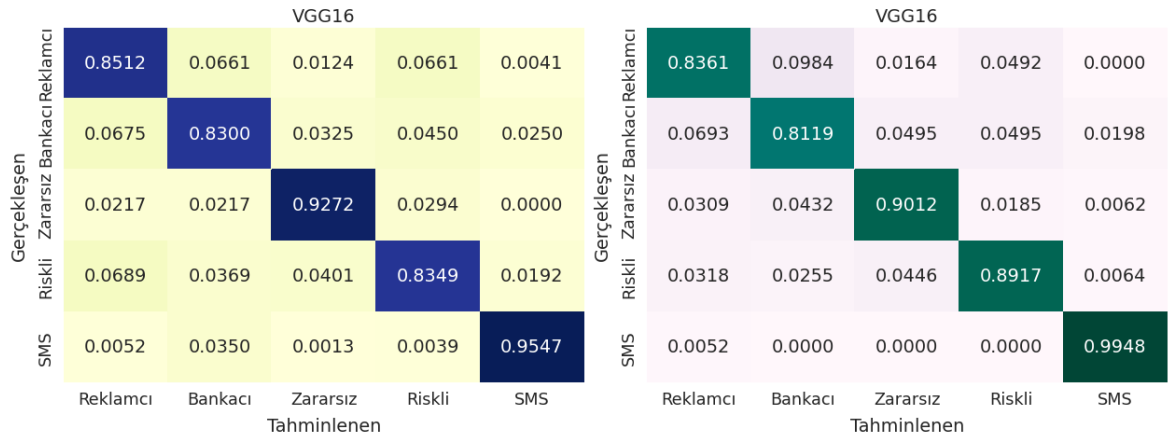
6.4.1. VGG16 modeli sonuçları

VGG model ailesinin 16 katmandan oluşan üyesi olan VGG16 önerilen yöntemden olumsuz etkilenen modellerden biridir. Modele ait eğitim grafiği Şekil 6.3'te gösterilmektedir.



Şekil 6.3. VGG16 modeline ait eğitim grafiği

Şekil 6.3'te gösterildiği üzere model, öğrenim aktarımı aşamasında %89,23 kategorik sınıflandırma doğruluğuna ulaşmakta, ince ayar aşamasında eğitime açılan parametrelerle birlikte daha iyi doğruluk değerine ulaşamadan yerel minimum değerde takılı kalmaktadır. Modellerin, elde ettikleri en iyi doğruluk değerine göre ağırlıkları korunduğu için, VGG16 bu yıkıcı etkiden, etkilenmemekte fakat performans arttırıcı etkiden de faydalanamamaktadır. VGG16 modeline ait karmaşa matrisi Şekil 6.4'te verilmektedir.

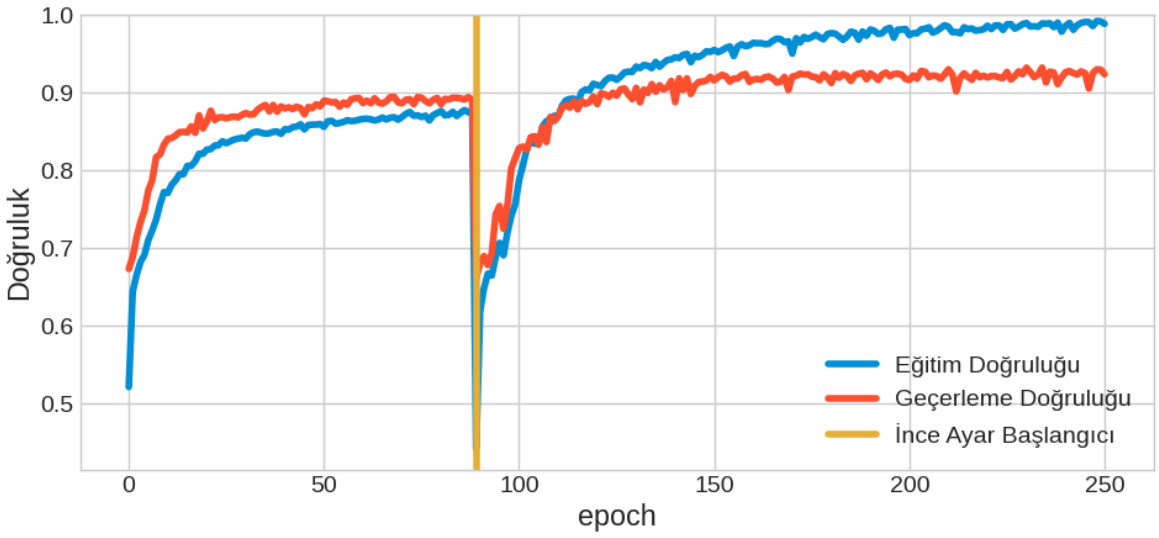


Şekil 6. 4. VGG16 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.4'te yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri gösterilmektedir. Model, test veri setine ait karmaşa matrisinde sadece riskli ve SMS kategorilerinde sınıflandırma doğruluğunu sırasıyla %91,72, %100 olacak şekilde attırabilmekte, reklamcı, bankacı, zararsız kategorilerinde sırasıyla %83,61, %81,19, %90,12 doğruluk göstererek daha başarısız olmaktadır.

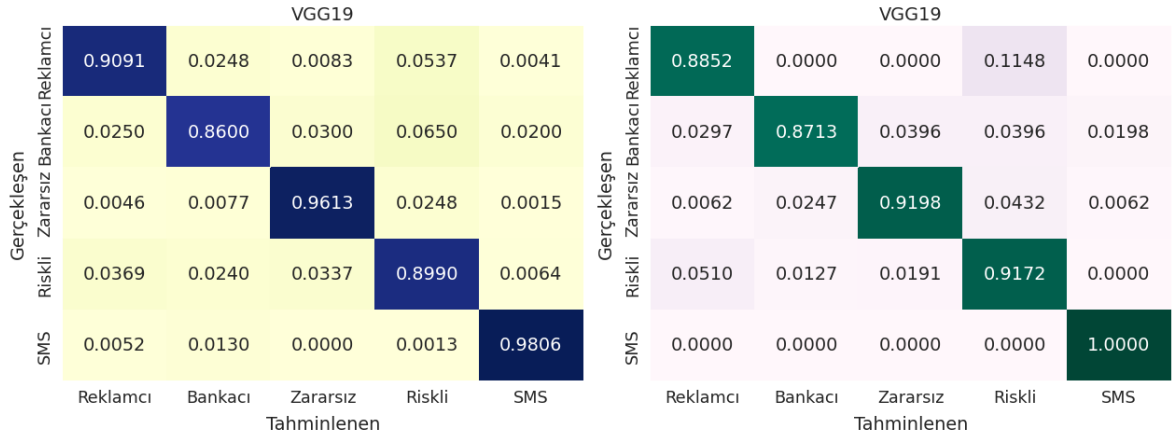
6.4.2. VGG19 modeli sonuçları

VGG ailesinin 19 katmanlı modeli olan VGG19, daha az katmanlı model olan VGG16'ya göre daha iyi bir performans sergilemektedir. Modele ait eğitim grafiği Şekil 6.5'te sunulmaktadır.



Şekil 6.5. VGG19 modeline ait eğitim grafiği

Şekil 6.5'te gösterildiği üzere VGG19 öğrenim aktarımı aşamasında, geçerleme veri seti üzerinde elde edilen %89,49'luk sınıflandırma doğruluğunu değerine ulaşmaktadır. İnce ayar ile %93,25 sınıflandırma doğruluğa çıkartmaktadır. Bu yaklaşık %4,2'lik bir performans artışıdır. VGG19 ve VGG16 modelleri aynı yöntemle eğitildiğinden ve farklı değişkenin katman sayısı olmasından ötürü, aradaki farkın katman sayısından kaynaklandığı düşünülmektedir. Öğrenim aktarımı aşamasını her bir epoch 25 saniye olmak üzere toplam 89 epochta tamamlayan model, ince ayar aşamasını her bir epoch 28 saniye olmak üzere 162 epochta tamamlamaktadır. Modele ait karmaşa matrisleri Şekil 6.6'da gösterilmektedir.



Şekil 6. 6. VGG19 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

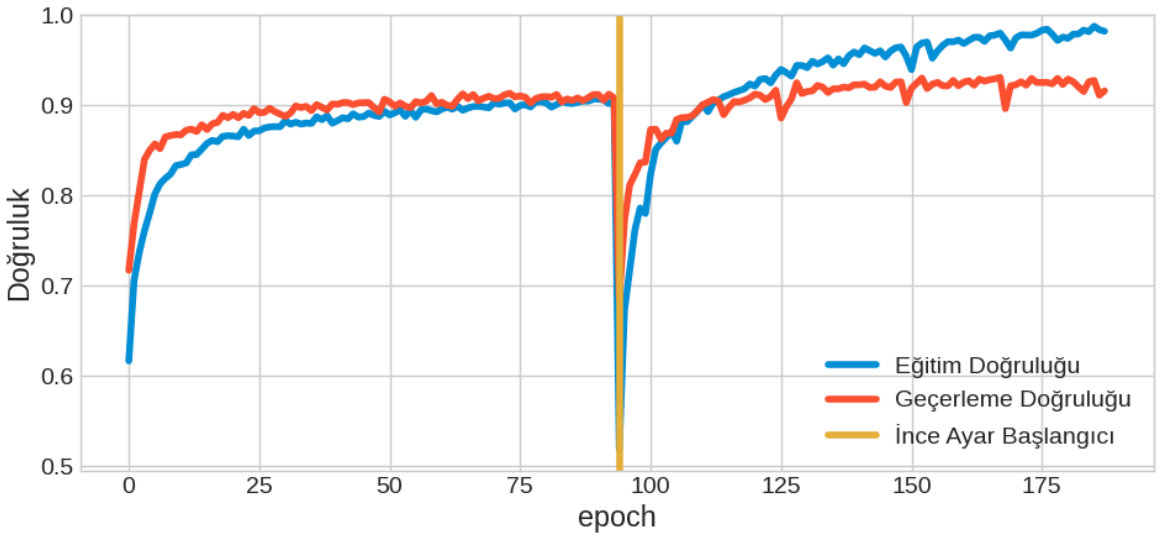
Şekil 6.6'da gösterilen karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri verilmektedir. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin bankacı, riskli ve SMS kategorilerinde sırasıyla %87,13, %91,72, %100 doğrulukla daha iyi, zararsız, reklamcı kategorilerinde sırasıyla %91,98, %88,52 doğrulukla daha kötü performans sergilediği görülmektedir.

6.5. ResNetV2 Modelleri Eğitim Sonuçları

Bu bölümde ResNetV2 model ailesine ait değerlendirme sonuçları, öğrenim aktarımı aşaması, ince ayar sınıflandırma doğrulukları, eğitim aşamaları, karmaşa matrisleri sunulmaktadır.

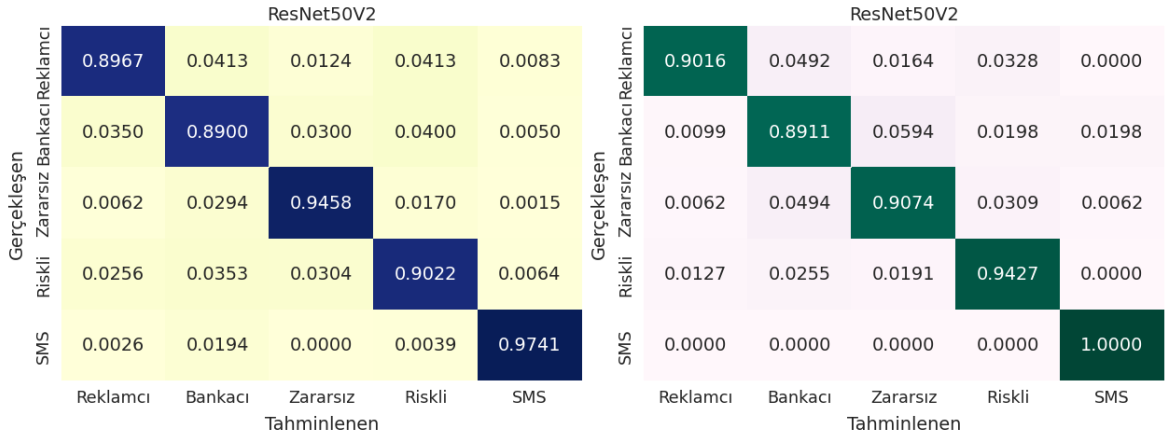
6.5.1. ResNet50V2 modeli sonuçları

ResNet model ailesinin 50 katmanlı modeli olan ResNet50V2, yöntemden olumlu etkilenen modellerden biridir. Modele ait eğitim grafiği Şekil 6.7'de gösterilmektedir.



Şekil 6.7. ResNet50V2 modeline ait eğitim grafiği

Şekil 6.7'de gösterildiği üzere ResNet50V2 modeli, öğrenim aktarımı aşamasında %91,35'lik kategorik sınıflandırma doğruluğu elde eden model, ince ayarla birlikte %93,10'luk kategorik sınıflandırma doğruluğuna erişmektedir. İlk aşamada her bir epoch 19 saniye olmak üzere toplam 94 epoch eğitilen model, ikinci aşamada her bir epoch 24 saniye olmak üzere toplam 94 epoch daha eğitim görmektedir. Modele ait karmaşa matrisleri Şekil 6.8'de sunulmaktadır.

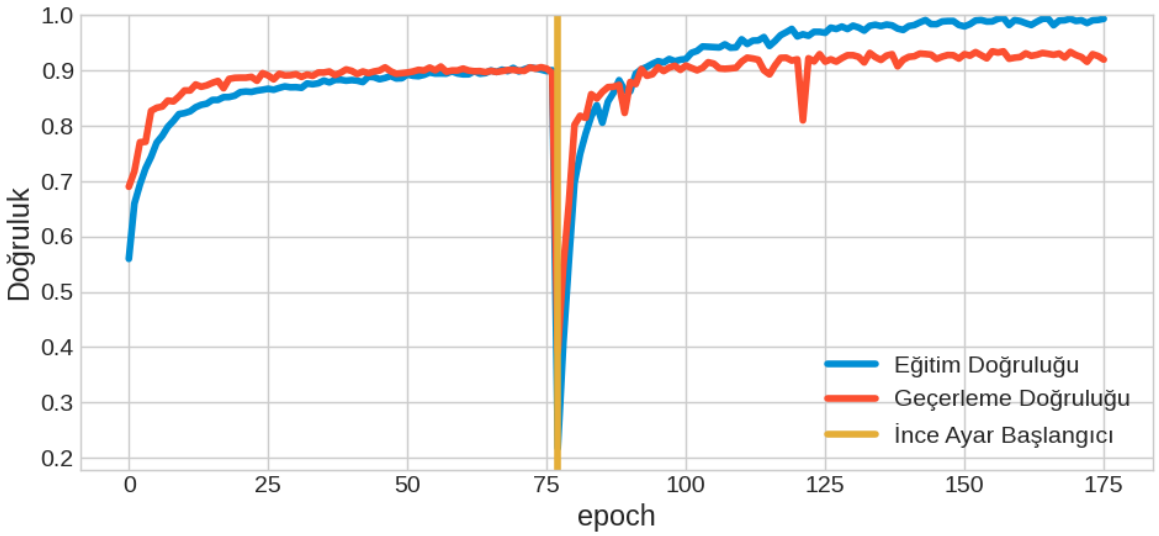


Şekil 6.8. ResNet50V2 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.8'de gösterilen karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri verilmektedir. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin kötücül yazılımlara ait olan reklamcı, bankacı, riskli, SMS kategorilerinde sırasıyla %90,16, %89,11, %94,27, %100 doğrulukla daha iyi performans sergilediği fakat zararsız yazılımlara ait olan kategoride %90,74 doğrulukla daha kötü performans sergilediği görülmektedir.

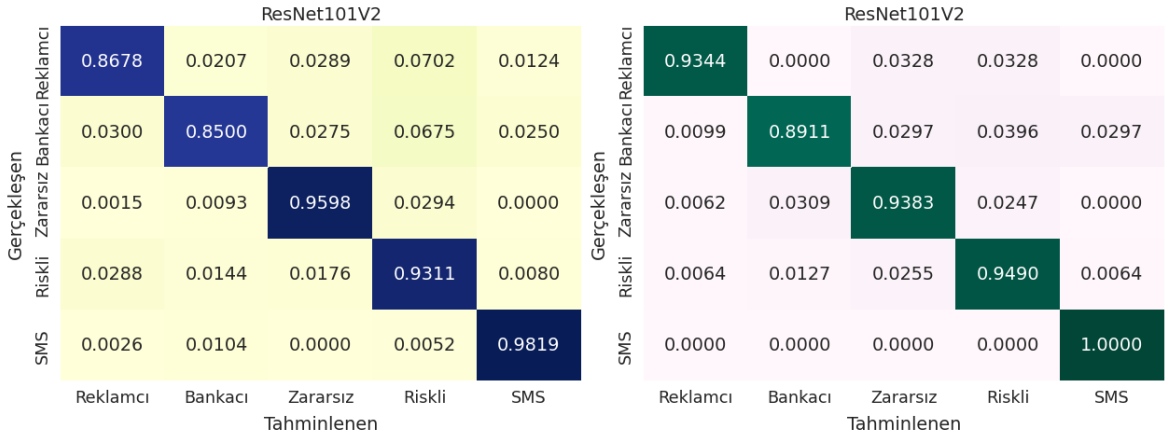
6.5.2. ResNet101V2 modeli sonuçları

ResNet ailesine ait 101 katmanlı ResNetV2 modeli, yöntemden olumlu etkilenen modellerden biridir. Modele ait eğitim grafiği Şekil 6.9'da sunulmaktadır.



Şekil 6.9. ResNet101V2 modeline ait eğitim grafiği

Şekil 6.9'da gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %90,72 kategorik sınıflandırma doğruluğuna erişmekte, ince ayar aşamasından sonra %93,47 kategorik sınıflandırma doğruluğuna erişmektedir. Model öğrenim aktarımı aşamasında, her bir epoch 32 saniye olmak üzere toplam 77 epoch kadar eğitilirken, ince ayar aşamasında her bir epoch 40 saniye olmak üzere toplam 99 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.10'da sunulmaktadır.

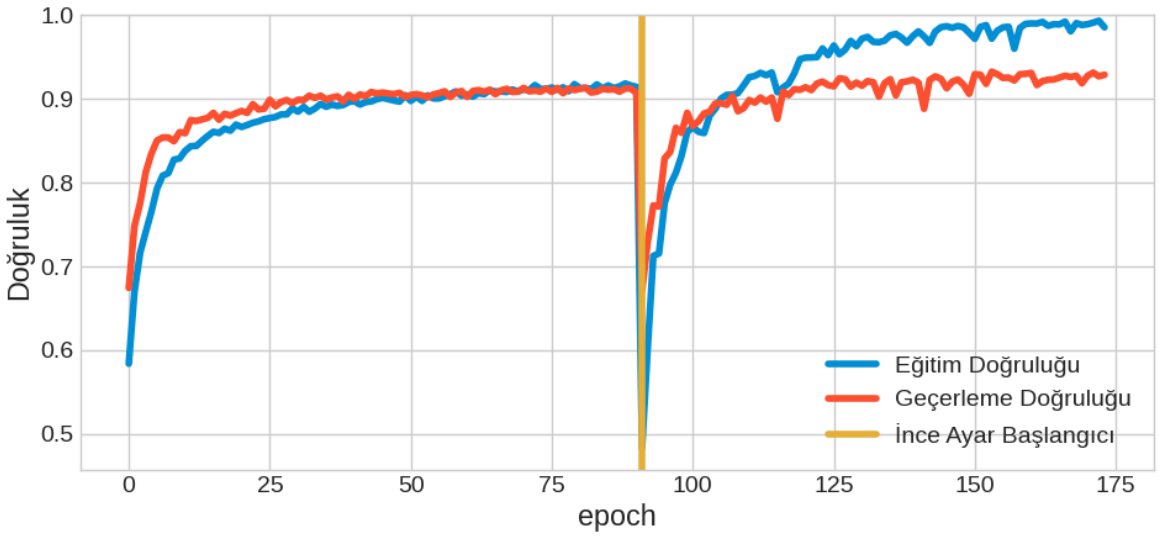


Şekil 6.10. ResNet101V2 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.10'da gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri verilmektedir. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, zararlı yazılımlara ait reklamcı, bankacı, riskli, SMS kategorinde sırasıyla %93,44, %89,11, %94,90, %100 doğrulukla performans artışı sağlayan modelin, zararsız yazılımlara ait kategoride %93,83 doğrulukla performans kaybı yaşadığı görülmektedir.

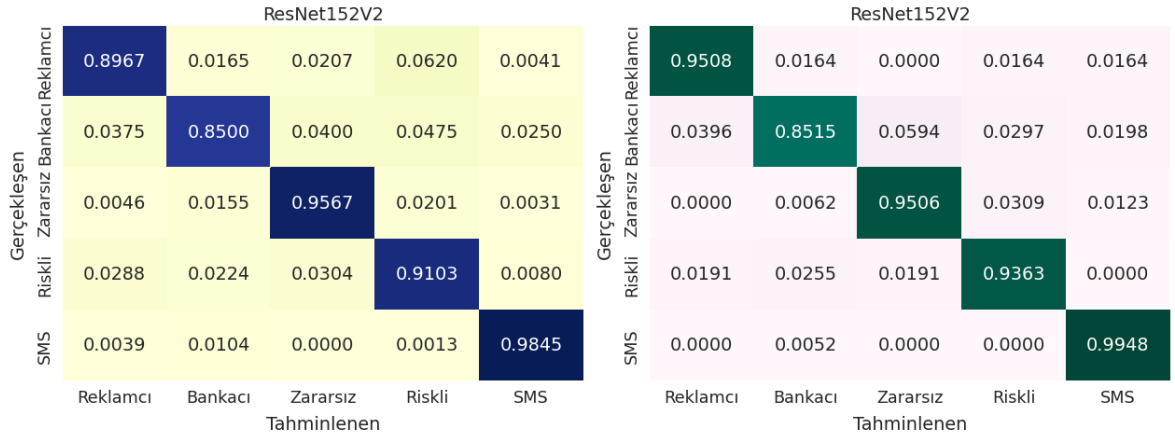
6.5.3. ResNet152V2 modeli sonuçları

ResNet ailesinin, gerçekleştirilen deneylerde, 152 katmanla en fazla katmana sahip olan üyesi ResNet152V2 modeli yöntemden olumlu sonuç alan modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.11'de gösterilmektedir.



Şekil 6.11. ResNet152V2 modeline ait eğitim grafiği

Şekil 6.11'de gösterildiği üzere model öğrenim aktarımı aşamasında %91,31 kategorik sınıflandırma doğruluğuna, ince ayar aşamasıyla %93,25 kategorik sınıflandırma doğruluğuna ulaşmaktadır. Öğrenim aktarımı aşamasında her bir epoch 44 saniye olmak üzere toplam 91 epoch eğitilen model, ince ayar aşamasında her bir epoch 57 saniye olmak üzere toplam 83 epoch daha eğitilmektedir. Modele ait karmaşa Şekil 6.12'de gösterilmektedir.

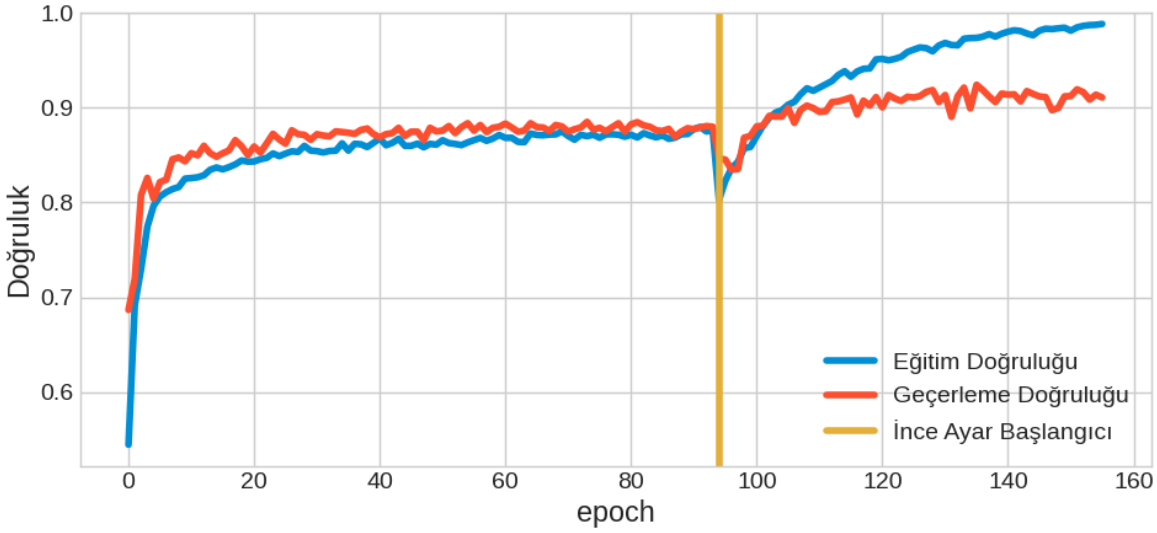


Şekil 6.12. ResNet152V2 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.12'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin kötücül yazılımlara ait reklamcı, bankacı, riskli, SMS kategorilerinde sınıflandırma doğruluğu sırasıyla %95,08, %85,15, %93,63, %99,48 olacak şekilde artmaktayken, zararsız yazılımlara ait kategoride sınıflandırma doğruluğunda %95,06 olarak az da olsa kayıp yaşanmaktadır.

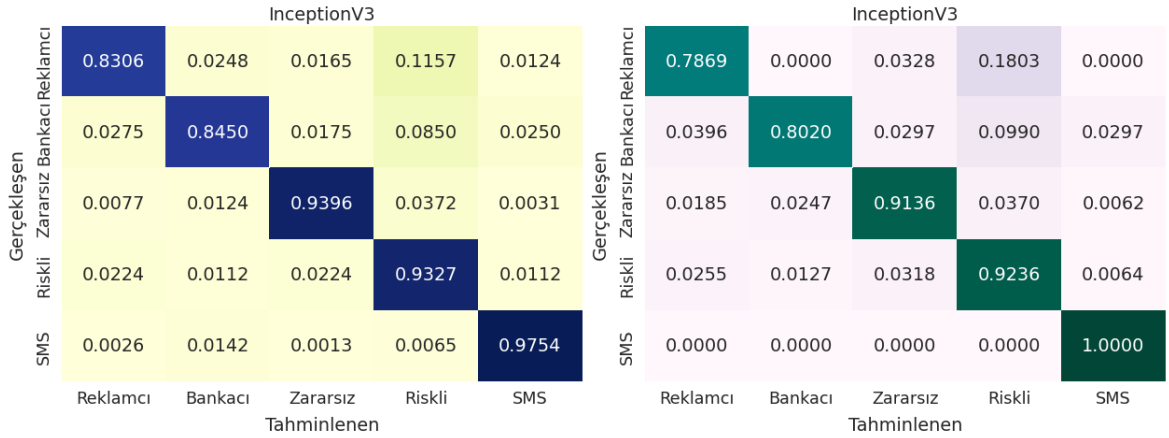
6.6. InceptionV3 Modeli Eğitim Sonuçları

Inception ailesinin son modellerinden biri olan InceptionV3, yöntemden olumlu etkilenen modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.13'te gösterilmektedir.



Şekil 6.13. InceptionV3 modeline ait eğitim grafiği

Şekil 6.13'te gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında, %88,52 kategorik sınıflandırma doğruluğu sergilemekte, ince ayar aşamasıyla birlikte %92,43 kategorik sınıflandırma doğruluğuna ulaşmaktadır. Model öğrenim aktarımı aşamasında her epoch 18 saniye olmak üzere toplam 94 epoch kadar eğitilirken, ince ayar aşamasında her epoch 21 saniye olmak üzere toplam 62 epoch daha eğitim görmektedir. Şekil 6.14'te modele ait karmaşa matrisleri sunulmaktadır.

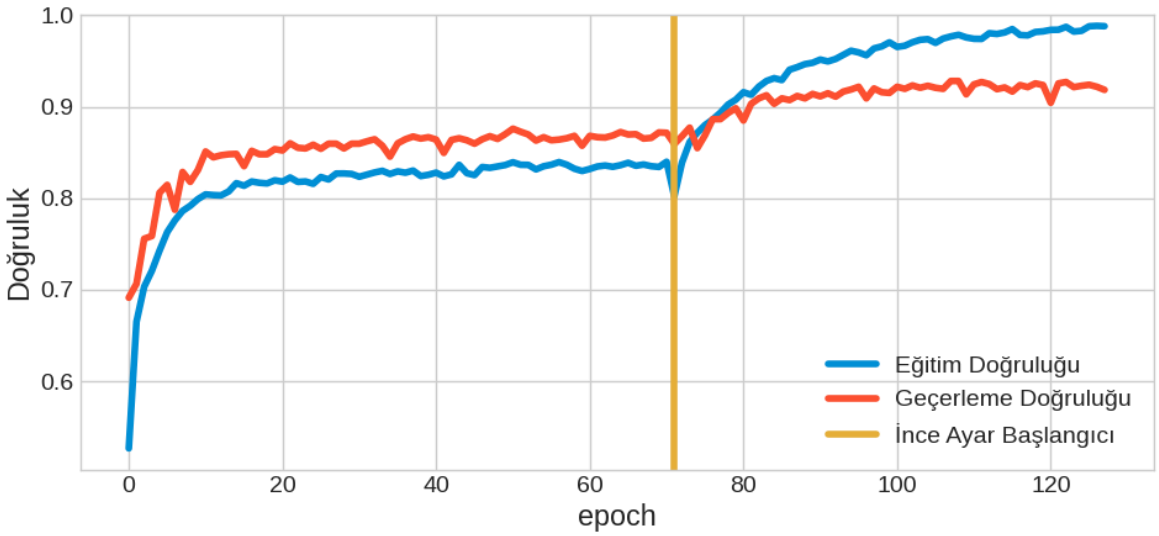


Şekil 6.14. InceptionV3 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.14'te gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin reklamcı, bankacı, zararsız, riskli kategorilerinde sırasıyla %78,69, %80,20, %91,36, %93,36 doğruluk göstererek daha kötü performans sergilediği, sadece SMS kötücüllerine ait kategoride %100 doğruluk göstererek daha iyi performans sergilediği görülmektedir.

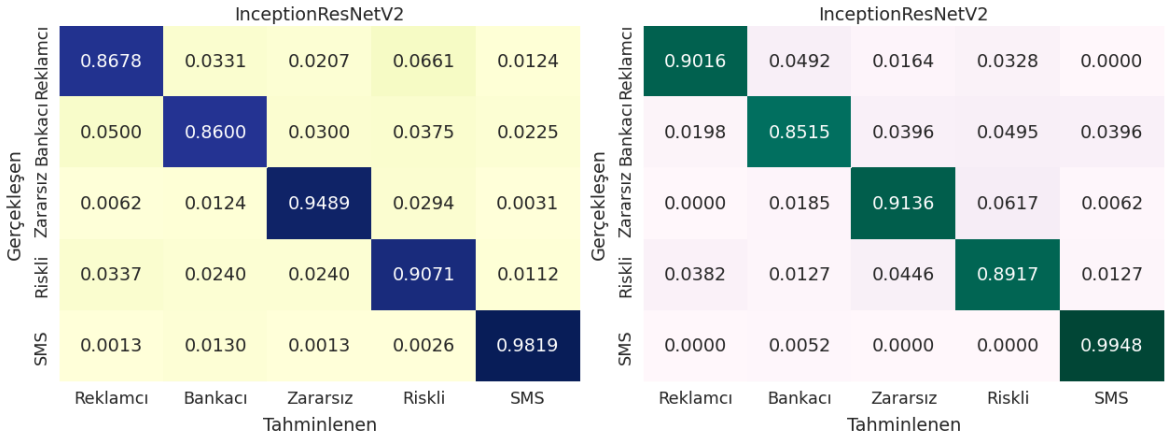
6.7. InceptionResNetV2 Modeli Eğitim Sonuçları

Inception ve ResNet yaklaşımlarının bir birleşimi olan InceptionResNetV2 modeli, yöntemden olumlu etkilenen modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.15'te sunulmaktadır.



Şekil 6.15. InceptionResNetV2 modeline ait eğitim grafiği

Şekil 6.15'te gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %87,59 kategorik sınıflandırma doğruluğu göstermekte, ince ayar aşamasıyla birlikte %92,80 kategorik sınıflandırma doğruluğuna ulaşmaktadır. Öğrenim aktarımı aşamasında, her bir epoch 39 saniye olmak üzere toplam 71 epoch eğitilen model, ince ayar aşamasında her bir epoch 48 saniye olmak üzere toplam 57 epoch kadar daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.16'te gösterilmektedir.



Şekil 6.16. InceptionResNetV2 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

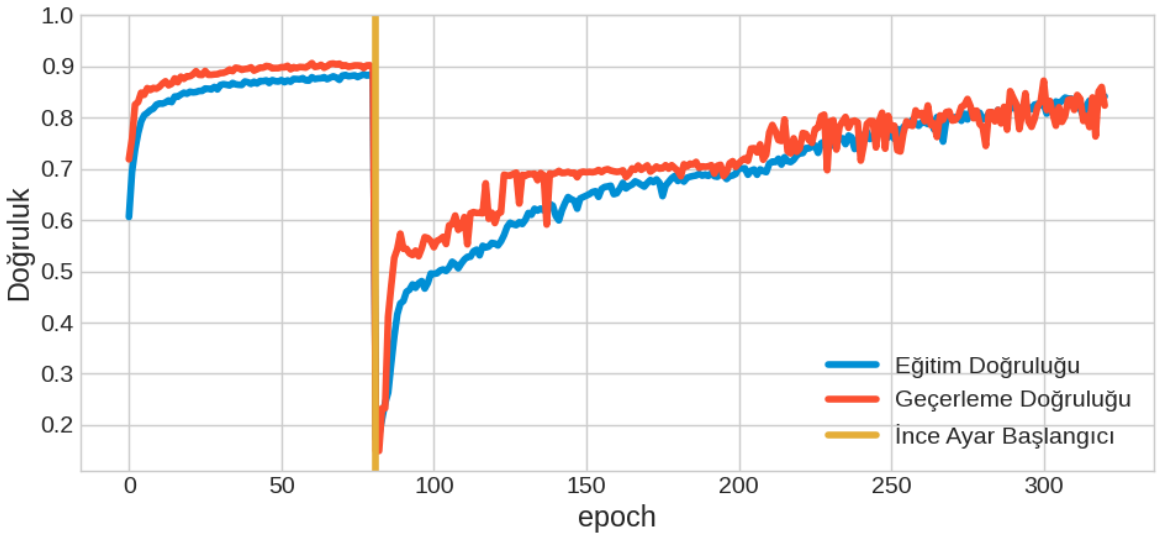
Şekil 6.16'da gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin reklamcı ve SMS kategorilerinde sırasıyla %90,16, %99,48 doğruluk göstererek daha iyi sınıflandırma yaptığı, bankacı, zararsız, riskli kategorilerde sırasıyla %85,15, %91,36, %89,17 doğruluk göstererek daha kötü sınıflandırma yaptığı görülmektedir.

6.8. MobileNet Modelleri Eğitim Sonuçları

Bu bölümde MobileNet model ailesine ait değerlendirme sonuçları, öğrenim aktarımı aşaması, ince ayar sınıflandırma doğrulukları, eğitim aşamaları, karmaşa matrisleri sunulmaktadır.

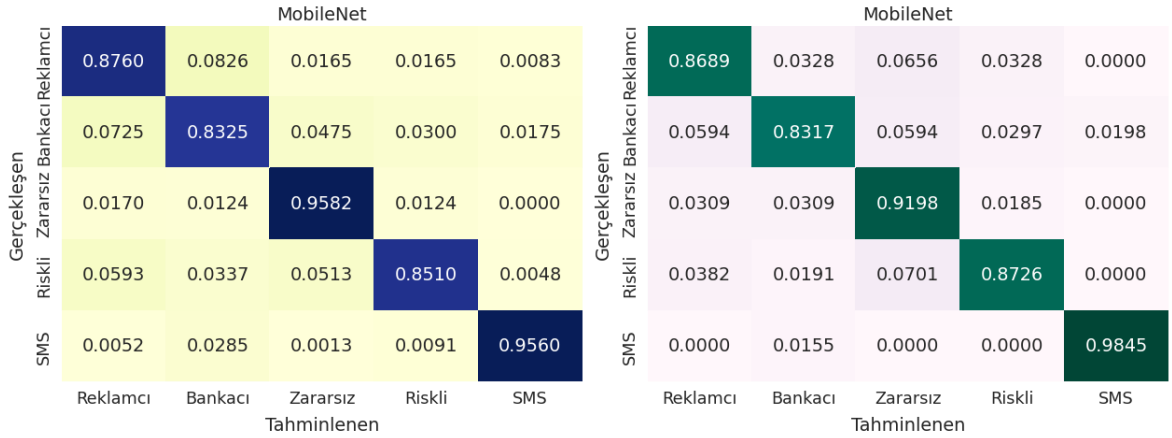
6.8.1. MobileNet modeli sonuçları

MobileNet modeli, sunulan yöntemden faydalanamayan modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.17'de gösterilmektedir.



Şekil 6.17. MobileNet modeline ait eğitim grafiği

Şekil 6.17'de gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %90,64'lük kategorik sınıflandırma doğruluğu etmekte, ince ayar aşamasında yeterince süre verilmesine rağmen yerel bir değerde takılarak daha fazla eğitilememektedir. Model, öğrenim aktarımı aşamasında her bir epoch 10 saniye olmak üzere toplam 81 epoch eğitilirken, ince ayar aşamasında her bir epoch 11 saniye olmak üzere toplam 240 epoch kadar daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.18'de sunulmaktadır.

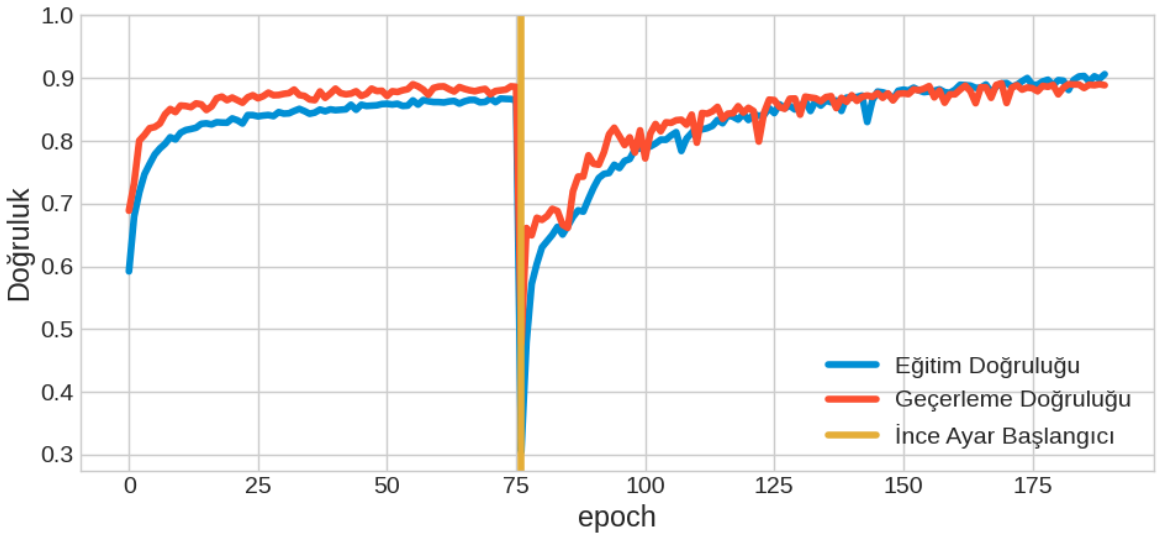


Şekil 6.18. MobileNet modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.18'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin reklamcı, bankacı ve zararsız kategorilerinde sırasıyla %86,89, %83,17, %91,98 doğruluk göstererek daha hatalı sınıflandırma gerçekleştirirken, riskli ve SMS kategorilerinde sırasıyla %87,26, %98,45 doğruluk göstererek daha doğru sınıflandırma gerçekleştirdiği gözlemlenmektedir.

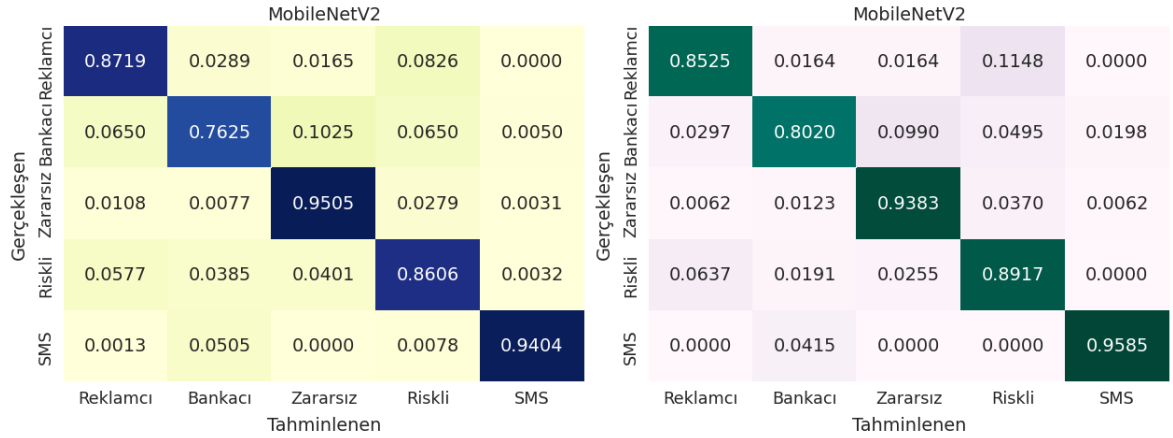
6.8.2. MobileNetV2 modeli sonuçları

MobileNetV2 modeli, yöntemden faydalanamayan modellerdendir. Modele ait eğitim grafiği Şekil 6.19'da sunulmaktadır.



Şekil 6.19. MobileNetV2 modeline ait eğitim grafiği

Şekil 6.19'da gösterildiği üzere model MobileNetV1 modeliyle benzer bir eğitim grafiği çizmektedir. Geçerleme veri seti üzerinde, model, öğrenim aktarımı aşamasında %89 kategorik sınıflandırma doğruluğu göstermekteyken, ince ayar aşamasında yeterince süre verilmesine rağmen daha fazla eğitilememektedir. Model, öğrenim aktarımı aşamasında her bir epoch 11 saniye olmak üzere toplam 76 epoch kadar eğitilirken, ince ayar aşamasında her bir epoch 13 saniye olmak üzere 114 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.20'de sunulmaktadır.

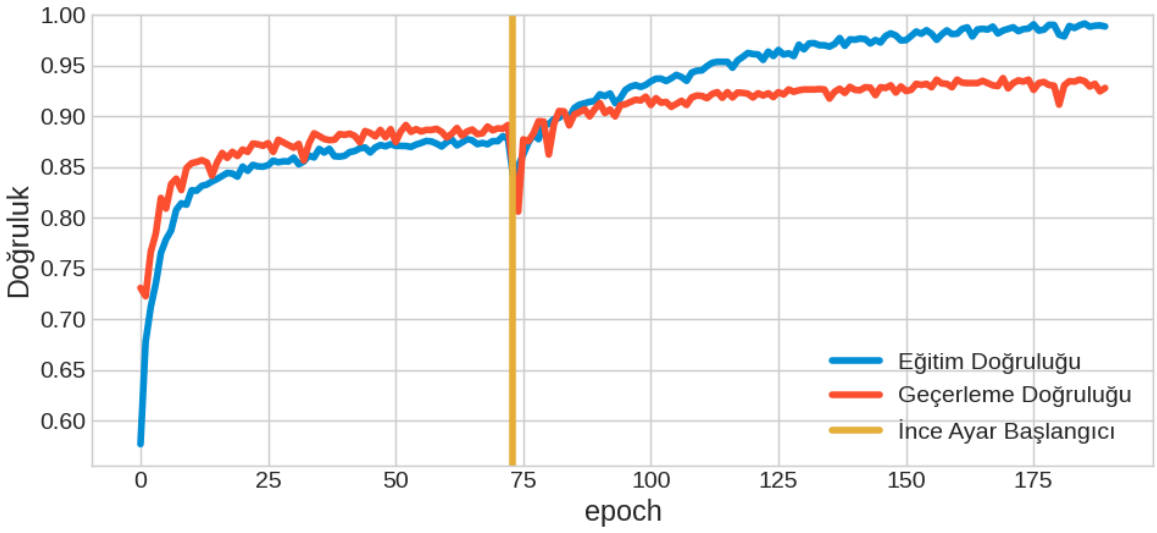


Şekil 6.20. MobileNetV2 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.20'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin reklamcı, zararsız kategorilerinde sırasıyla %85,25, %93,83 doğruluk göstererek daha hatalı sınıflandırdığı, bankacı, riskli, SMS kategorilerinde sırasıyla %80,20, %89,17, %95,85 doğruluk göstererek daha doğru sınıflandırdığı görülmektedir.

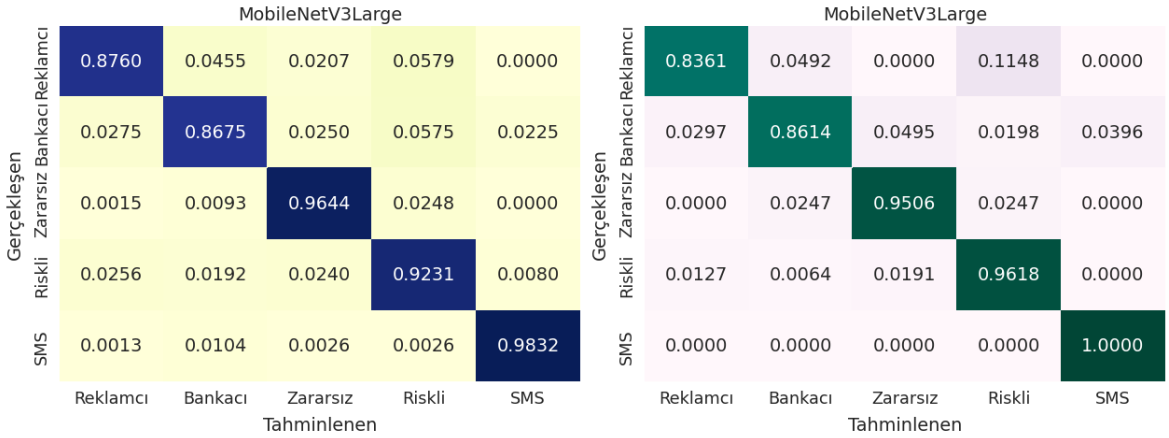
6.8.3. MobileNetV3Large modeli sonuçları

MobileNetV3Large modeli, yöntemden olumlu etkilenen modellerdendir. Modele ait eğitim grafiği Şekil 6.21'de gösterilmektedir.



Şekil 6.21. MobileNetV3Large modeline ait eğitim grafiği

Şekil 6.21'de gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %89,15 kategorik sınıflandırma doğruluğu elde etmekteyken, ince ayar aşamasında %93,77 kategorik sınıflandırma doğruluğuna çıkmaktadır. Model, öğrenim aktarımı aşamasında her bir epoch 10 saniye olmak üzere toplam 73 epoch eğitilmekte, ince ayar aşamasında her epoch 11 saniye olmak üzere toplam 117 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.22'de sunulmaktadır.

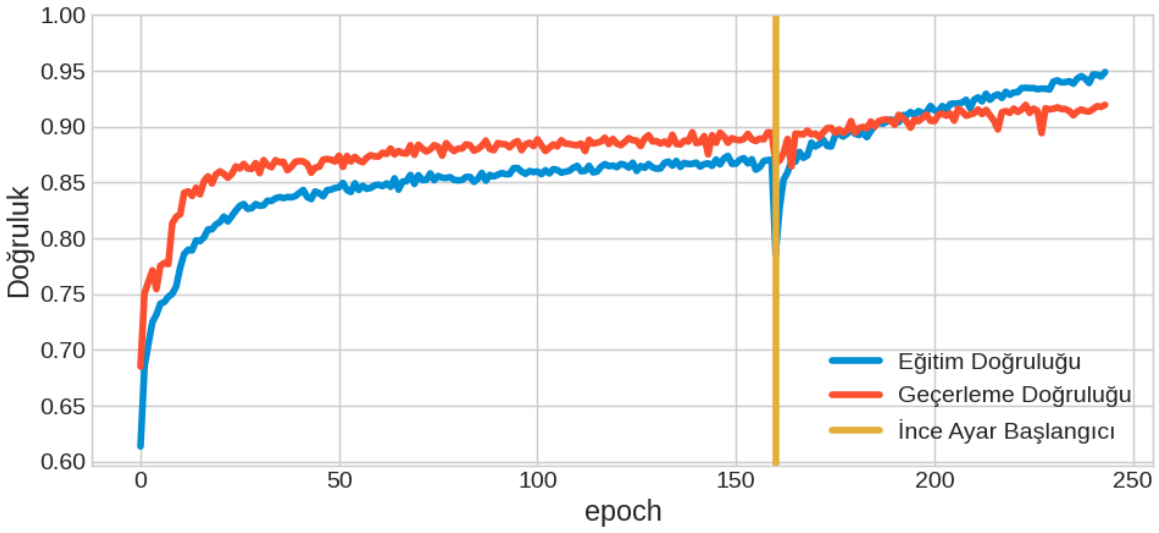


Şekil 6.22. MobileNetV3Large modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.22'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin, reklamcı, bankacı, zararsız kategorilerinde sırasıyla %83,61, %86,14, %95,06 doğruluk göstererek daha hatalı, riskli ve SMS kategorilerinde sırasıyla %96,18, %100 doğruluk göstererek daha doğru sınıflandırdığı gözlenmektedir.

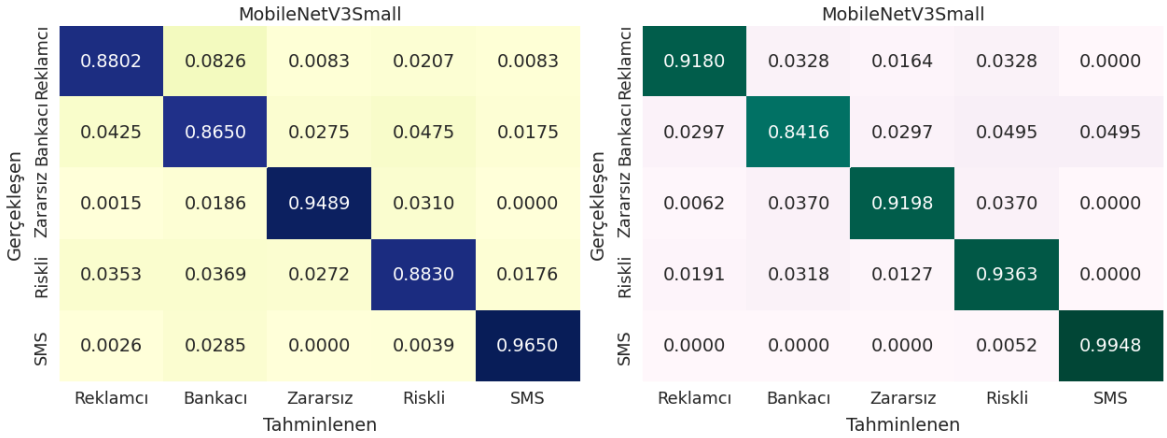
6.8.4. MobileNetV3Small modeli sonuçları

MobileNetV3Small modeli, yöntemden olumlu etkilenen modellerdendir. Modele ait eğitim grafiği Şekil 6.23'te sunulmaktadır.



Şekil 6.23. MobileNetV3Small modeline ait eğitim sonuçları

Şekil 6.23'te gösterildiği üzere model öğrenim aktarımı aşamasında %89,49 kategorik sınıflandırma doğruluğu göstermekte, ince ayar aşamasından sonra, %91,95 kategorik sınıflandırma doğruluğuna erişmektedir. Model, öğrenim aktarımı aşamasında her epoch 8 saniye olmak üzere toplam 160 epoch eğitilmekte, ince ayar aşamasında, her epoch 10 saniye olmak üzere toplam 84 epoch daha eğitim görmektedir. Modele ait karmaşa matrisleri Şekil 6.24'te sunulmaktadır.



Şekil 6.24. MobileNetV3Small modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

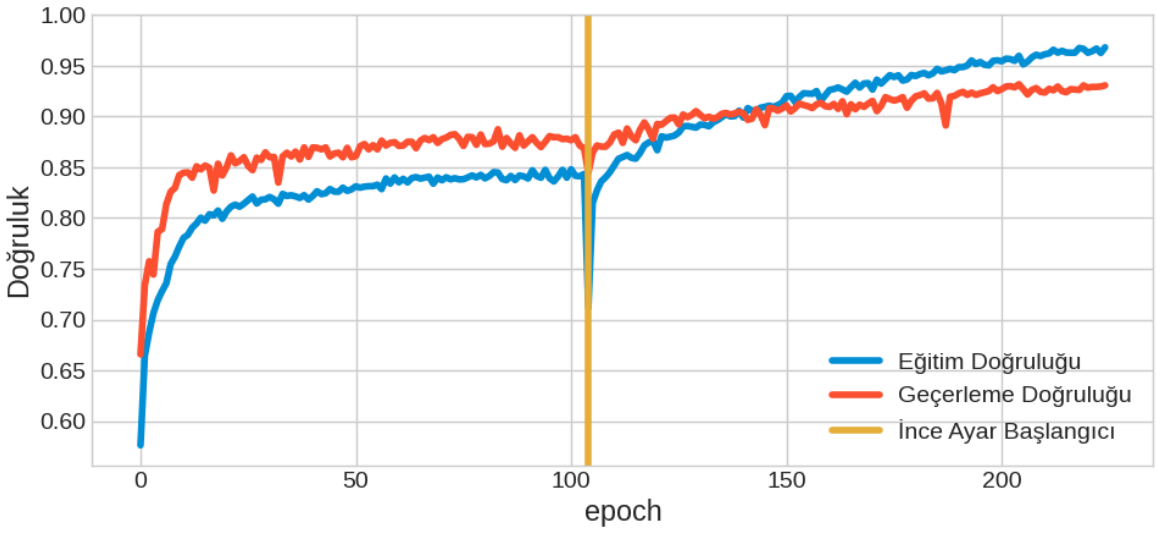
Şekil 6.24'te gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model reklamcı, riskli, SMS kategorilerinde sırasıyla %91,80, %93,63, %99,48 doğruluk göstererek daha doğru sınıflandırmakta, bankacı, zararsız kategorilerinde sırasıyla %84,16, %91,98 doğruluk göstererek daha hatalı sınıflandırmaktadır.

6.9. DenseNet Modelleri Eğitim Sonuçları

Bu bölümde DenseNet model ailesine ait değerlendirme sonuçları, öğrenim aktarımı aşaması, ince ayar sınıflandırma doğrulukları, eğitim aşamaları, karmaşa matrisleri sunulmaktadır.

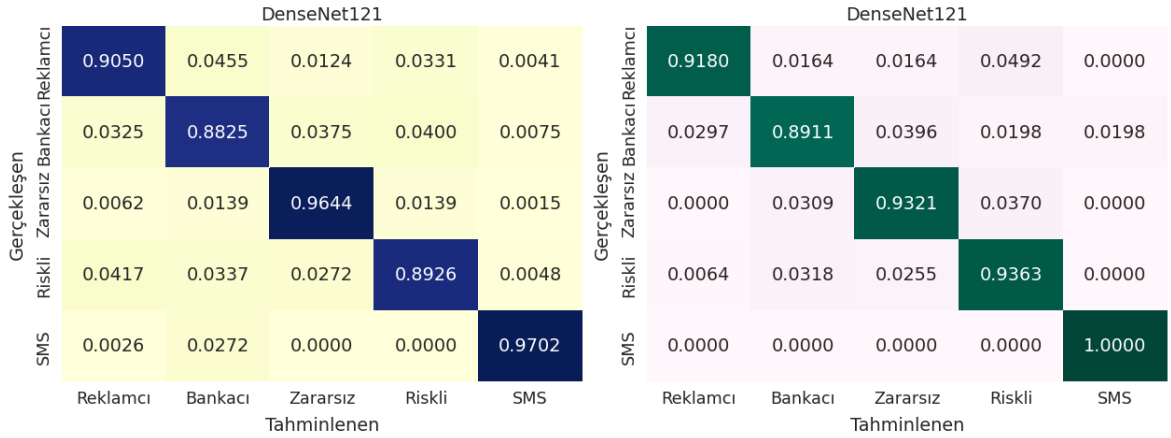
6.9.1. DenseNet121 modeli sonuçları

DenseNet121 yöntemden olumlu faydalanan modellerdendir. Modele ait eğitim grafiği Şekil 6.25'te gösterilmektedir.



Şekil 6.25. DenseNet121 modeline ait eğitim grafiği

Şekil 6.25'te gösterildiği üzere model öğrenim aktarımı aşamasında %88,74 kategorik sınıflandırma doğruluğu göstermekte, ince ayar aşamasından sonra %93,18 kategorik sınıflandırma doğruluğuna erişmektedir. Model, öğrenim aktarımı aşamasında her epoch 21 saniye olmak üzere toplam 104 epoch eğitilmekte, ince ayar aşamasında her epoch 27 saniye olmak üzere toplam 121 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.26'da verilmektedir.

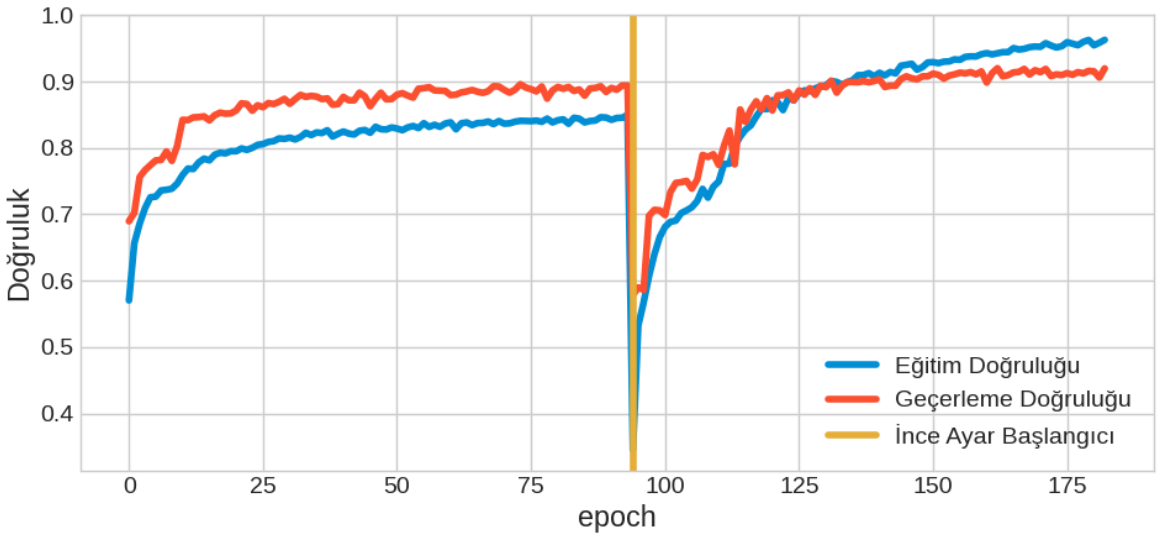


Şekil 6.26. DenseNet121 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.24'te gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model reklamcı, bankacı, riskli ve SMS kategorilerinde sırasıyla %91,80, %89,11, %93,63, %100 doğruluk göstererek daha doğru, zararsız kategorisinde %93,21 doğruluk göstererek daha hatalı sınıflandırdığı görülmektedir.

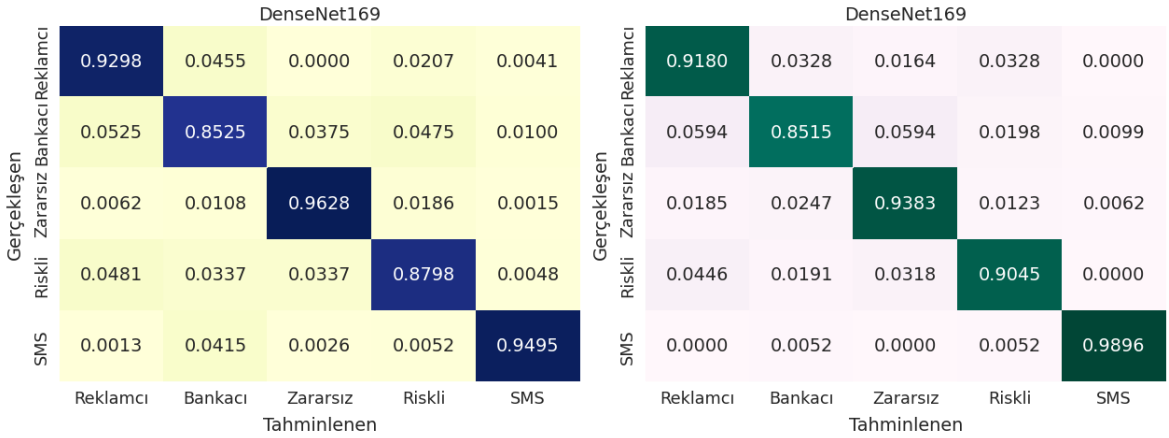
6.9.2. DenseNet169 modeli sonuçları

DenseNet169, yöntemden olumlu etkilenen modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.27'de gösterilmektedir.



Şekil 6.27. DenseNet169 modeline ait eğitim grafiği

Şekil 6.27'de gösterildiği üzere model geçerleme veri setindeki örnekleri, öğrenim aktarımı aşamasında %89,60 doğrulukla, ince ayar aşamasından sonra %92,02 doğrulukla kategorik olarak sınıflandırmaktadır. Öğrenim aktarımı aşamasında, her epoch 27 saniye olmak üzere toplam 94 epoch eğitilen model, ince ayar aşamasında her epoch 34 saniye olmak üzere toplam 89 epoch daha eğitilmektedir. Modele ait karmaşa matrisi Şekil 6.28'de gösterilmektedir.

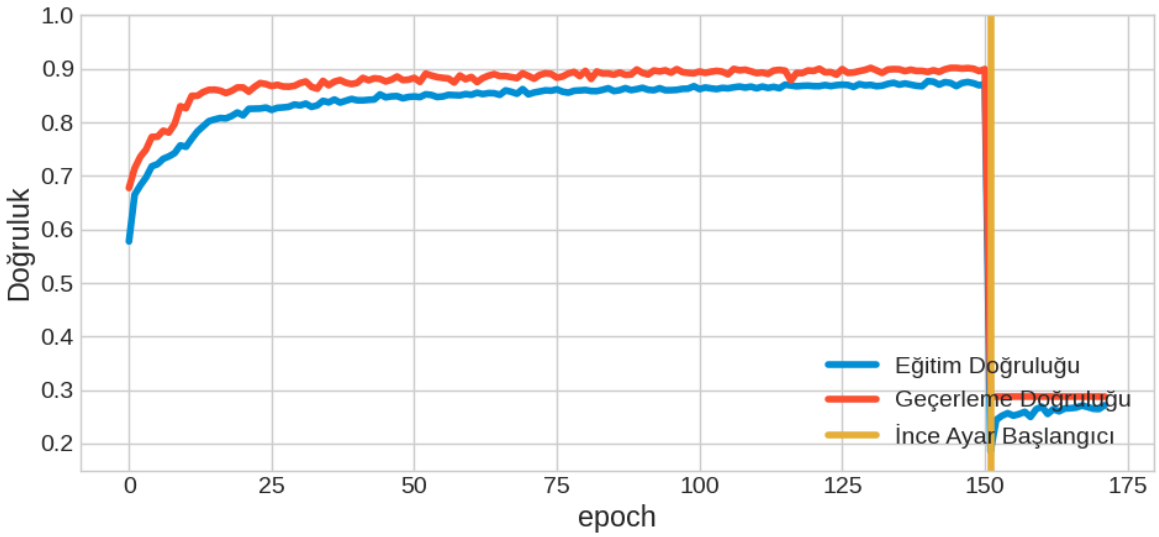


Şekil 6.28. DenseNet169 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.28'te gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin reklamcı, bankacı, zararsız kategorilerinde sırasıyla %91,80, %85,15, %93,83 doğruluk göstererek daha hatalı, riskli ve SMS kategorilerinde %90,45, %98,98 doğruluk göstererek daha doğru sınıflandırdığı görülmektedir.

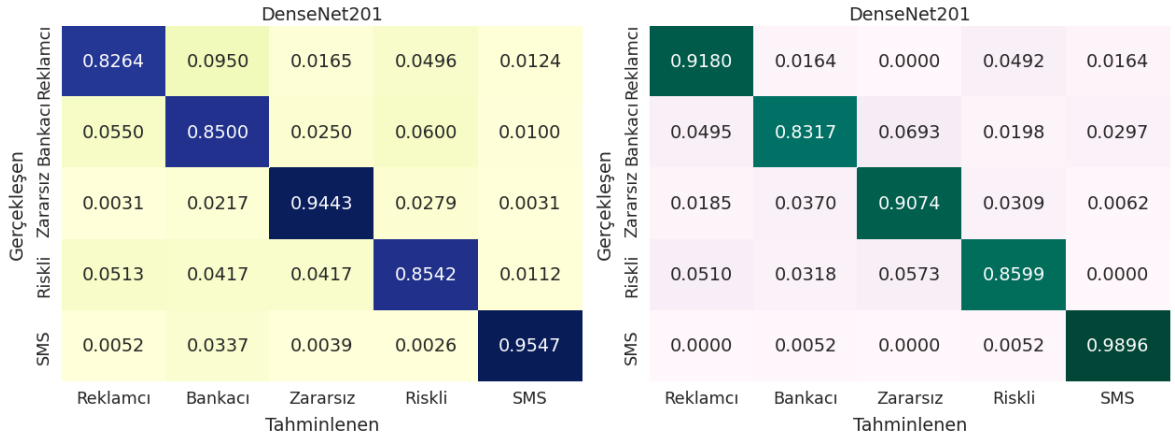
6.9.3. DenseNet201 modeli sonuçları

DenseNet201 modeli yöntemden faydalanamayan modellerdendir. Modele ait eğitim grafiği Şekil 6.29'da gösterilmektedir.



Şekil 6.29. DenseNet201 modeline ait eğitim grafiği

Şekil 6.29'da gösterildiği üzere model geçerleme veri setinde, öğrenim aktarımı aşamasında örnekleri %90,16 doğrulukla kategorik olarak sınıflandırmakta, ince ayar aşamasında yerel bir değerde takılmaktadır. Öğrenim aktarımı aşamasında, her epoch 34 saniye olmak üzere toplam 151 epoch eğitilen model, ince ayar aşamasında her epoch 43 saniye olmak üzere toplam 21 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.30'da sunulmaktadır.



Şekil 6.30. DenseNet201 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

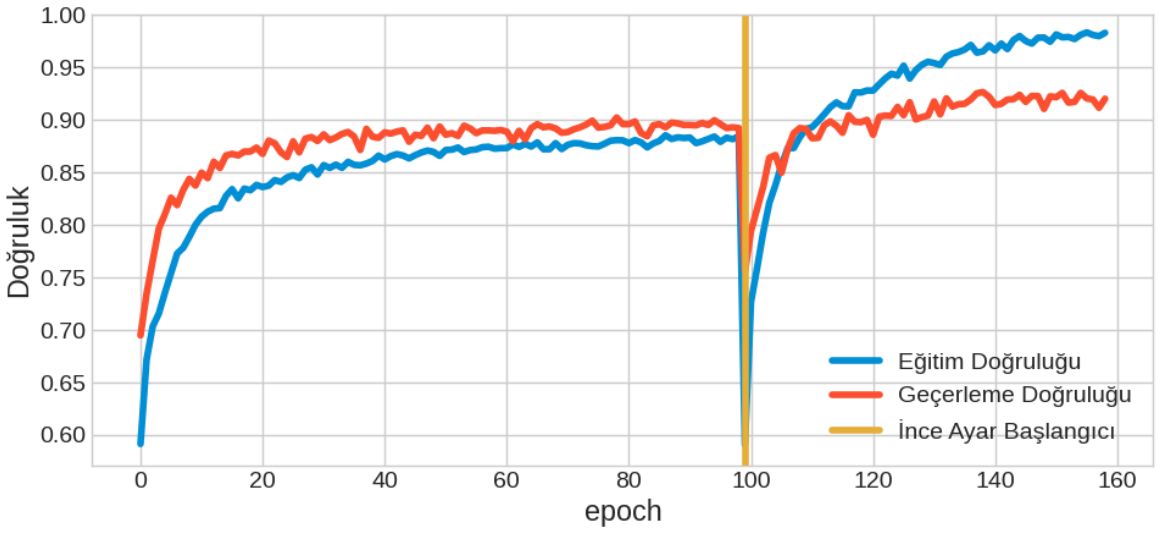
Şekil 6.30 gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model bankacı, zararsız kategorilerinde sırasıyla %83,17, %90,74 doğruluk göstererek daha hatalı sınıflandırırken, reklamcı, riskli ve SMS kategorilerinde sırasıyla %91,90, %85,99, %98,96 doğruluk göstererek daha doğru sınıflandırma gerçekleştirmektedir.

6.10. NASNet Modelleri Eğitim Sonuçları

Bu bölümde NASNet model ailesine ait değerlendirme sonuçları, öğrenim aktarımı aşaması, ince ayar sınıflandırma doğrulukları, eğitim aşamaları, karmaşa matrisleri sunulmaktadır.

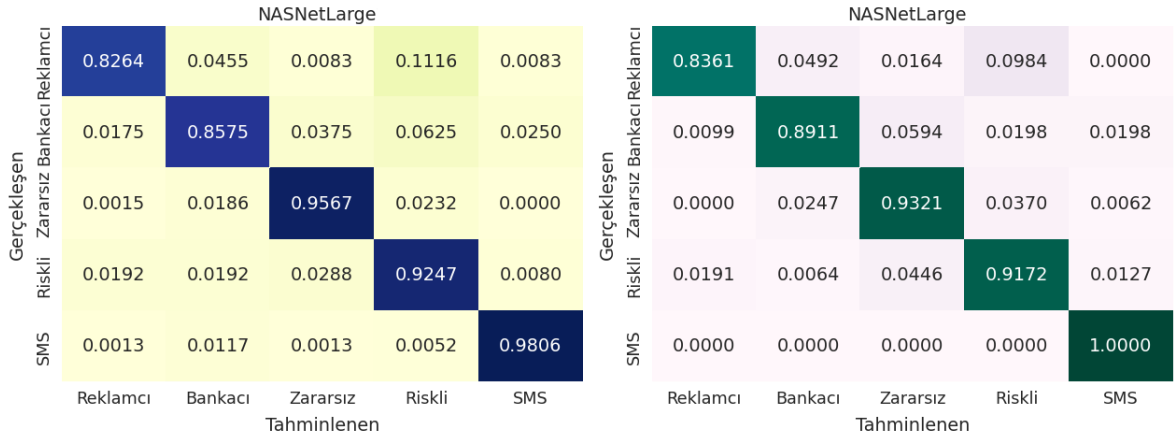
6.10.1. NASNetLarge modeli sonuçları

NASNetLarge modeli, yöntemden olumlu etkilenmektedir. Modele ait eğitim grafiği Şekil 6.31'de gösterilmektedir.



Şekil 6.31. NASNetLarge modeline ait eğitim grafiği

Şekil 6.31'de gösterildiği üzere model öğrenim aktarımı aşamasında geçerleme veri setindeki örnekleri %90,23 doğrulukla, ince ayar aşamasından sonra %92,95 doğrulukla kategorik olarak sınıflandırmaktadır. Öğrenim aktarımı aşamasında her epoch 143 saniye olmak üzere toplam 99 epoch eğitilmekte, ince ayar aşamasında her epoch 177 saniye olmak üzere 60 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.32'de gösterilmektedir.

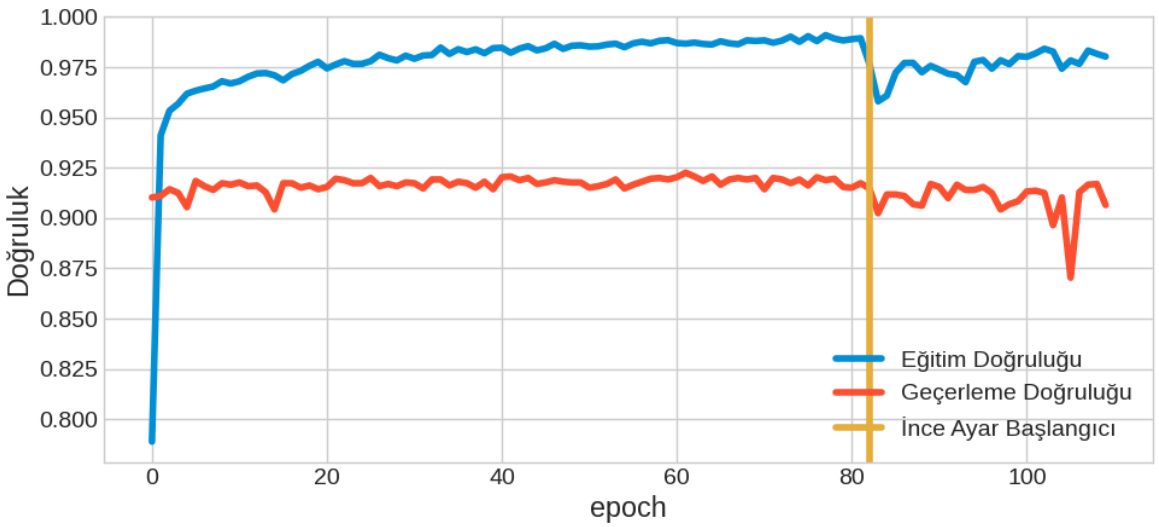


Şekil 6.32. NASNetLarge modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.32'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin örnekleri zararsız ve riskli kategorilerinde sırasıyla %93,21, %91,72 doğruluk göstererek daha hatalı, SMS, bankacı ve reklamcı kategorilerinde sırasıyla %100, %89,11, %83,61 doğruluk göstererek daha doğru sınıflandırdığı görülmektedir.

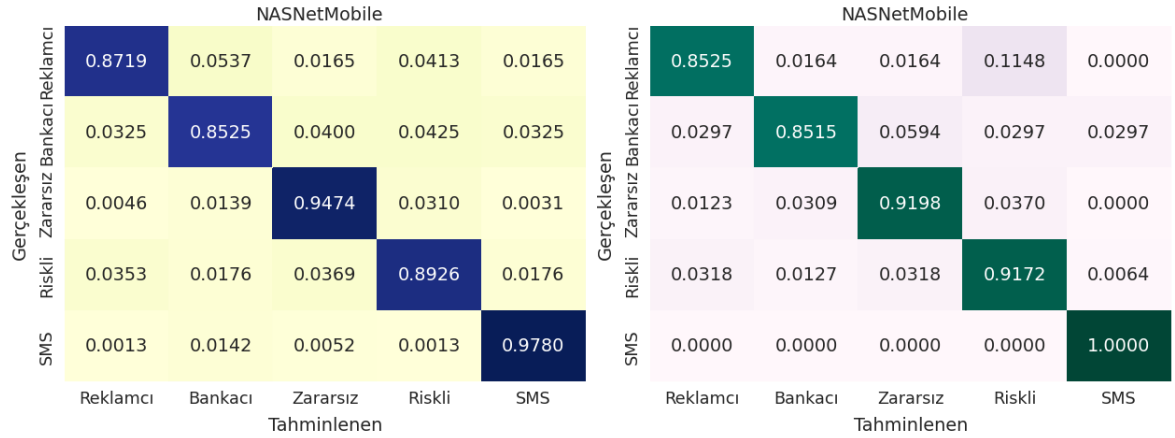
6.10.2. NASNetMobile modeli sonuçları

NASNetMobile yöntemden faydalanamayan modellerdendir. Modele ait eğitim grafiği Şekil 6.33'te gösterilmektedir.



Şekil 6.33. NASNetMobile modeline ait eğitim grafiği

Şekil 6.33'te gösterildiği üzere model öğrenim aktarımı aşamasında %92,25 doğrulukla örnekleri kategorik olarak sınıflandırmakta, ince ayar aşamasında daha fazla eğitilememektedir. Öğrenim aktarımı aşamasında her epoch 20 saniye olmak üzere toplam 82 epoch eğitilen model, ince ayar aşamasında her epoch 29 saniye olmak üzere toplam 32 epoch daha eğitim sürdürülmektedir. Modele ait karmaşa matrisleri Şekil 6.34'te sunulmaktadır.



Şekil 6.34. NASNetMobile modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

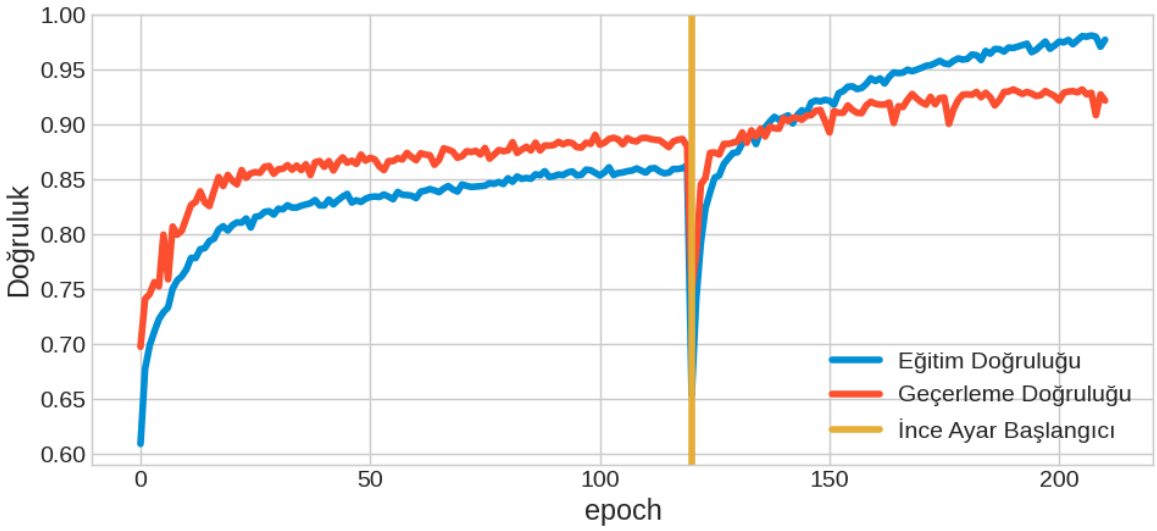
Şekil 6.34'te gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model riskli ve SMS kategorilerinde sırasıyla %91,72, %100 doğruluk göstererek daha doğru, reklamcı, bankacı, zararsız kategorilerinde sırasıyla %85,25, %85,15, %91,98 doğruluk göstererek daha hatalı örnek sınıflandırması gerçekleştirmektedir.

6.11. EfficientNet Modelleri Eğitim Sonuçları

Bu bölümde EfficientNet model ailesine ait değerlendirme sonuçları, öğrenim aktarımı, ince ayar sınıflandırma doğrulukları, eğitim grafiği ve karmaşa matrisleri sunulmaktadır.

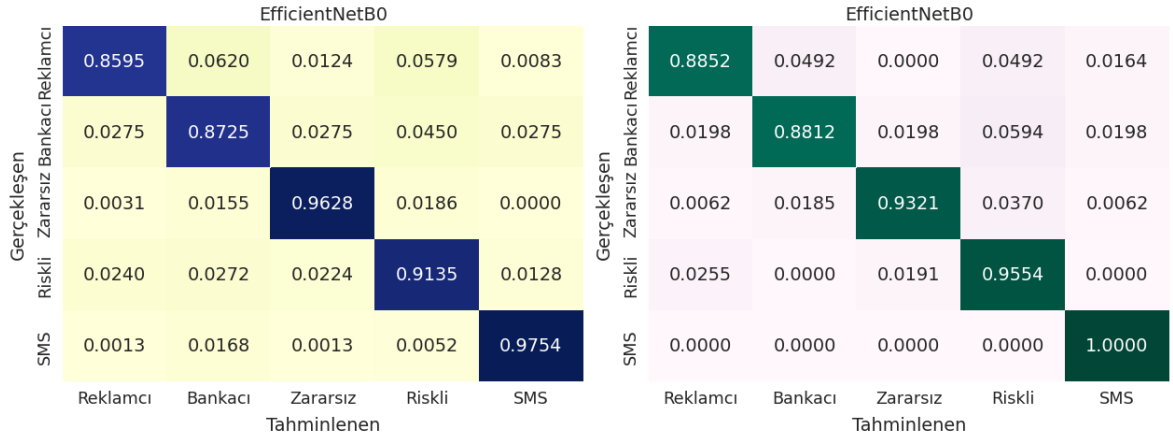
6.11.1. EfficientNetB0 modeli sonuçları

EfficientNet model ailesinin en küçük modeli olan EfficientNetB0, yöntemden olumlu etkilenen modellerden biri olmaktadır. Modele ait eğitim grafiği Şekil 6.35'te gösterilmektedir.



Şekil 6.35. EfficientNetB0 modeline ait eğitim grafiği

Şekil 6.35'te gösterildiği üzere model öğrenim aktarımı aşamasında, %89,08 kategorik sınıflandırma doğruluğuna erişmekte, ince ayar aşamasında %93,21 kategorik sınıflandırma doğruluğuna ulaşmaktadır. Öğrenim aktarımı aşamasında, her bir epoch 15 saniye olmak üzere toplam 120 epoch kadar eğitilen model, ince ayar aşamasında her bir epoch 18 saniye olmak üzere toplam 91 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.36'da sunulmaktadır.

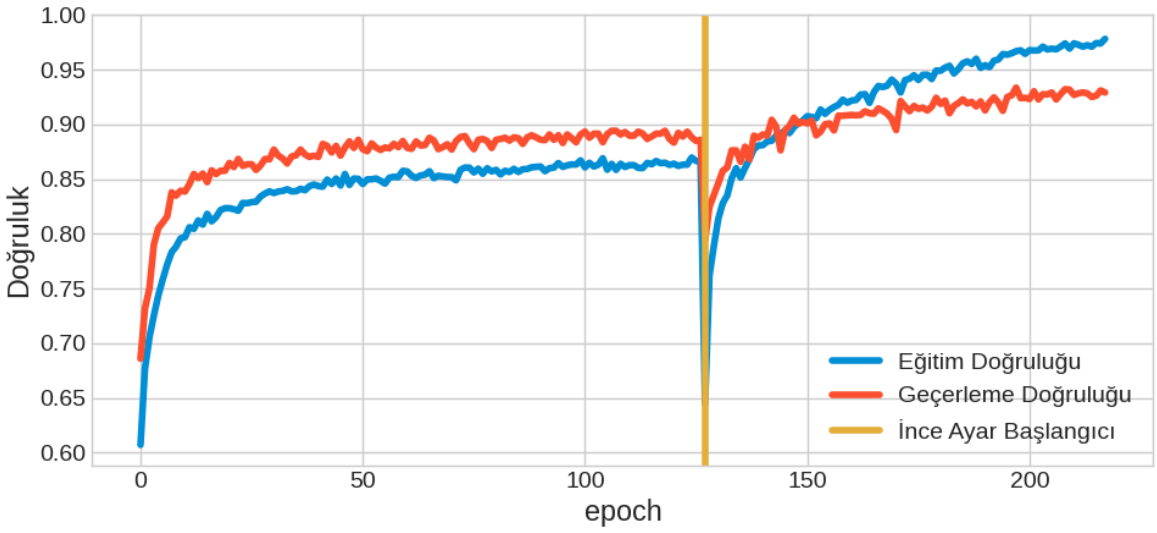


Şekil 6.36. EfficientNetB0 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.36'da gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model reklamcı, bankacı, riskli ve SMS kategorilerinde sırasıyla %88,52, %88,12, %95,54, %100 doğruluk göstererek daha iyi sınıflandırma gerçekleştirmekte fakat zararsız kategorisinde %93,21 doğruluk göstererek daha hatalı sınıflandırma gerçekleştirmektedir.

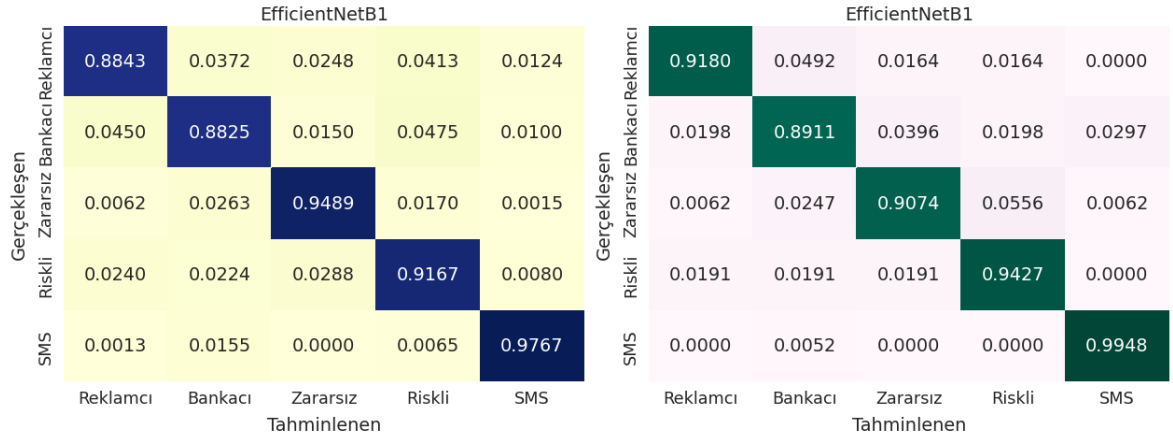
6.11.2. EfficientNetB1 modeli sonuçları

EfficientNetB1 yöntemden olumlu sonuç etkilenen modellerdendir. Modele ait eğitim grafiği Şekil 6.37'de sunulmaktadır.



Şekil 6.37. EfficientNetB1 modeline ait eğitim grafiği

Şekil 6.37'de gösterildiği üzere geçerleme veri setinde, model öğrenim aktarımı aşamasında, %89,41 kategorik sınıflandırma doğruluğu gösterirken, ince ayar aşamasında %93,36 kategorik sınıflandırma doğruluğuna erişmektedir. Öğrenim aktarımı aşamasında her bir epoch 19 saniye olmak üzere toplam 127 epoch eğitilen model, ince ayar aşamasında her epoch 24 saniye olmak üzere toplam 91 epoch daha eğitim görmektedir. Modele ait karmaşa matrisi Şekil 6.38'de gösterilmektedir.

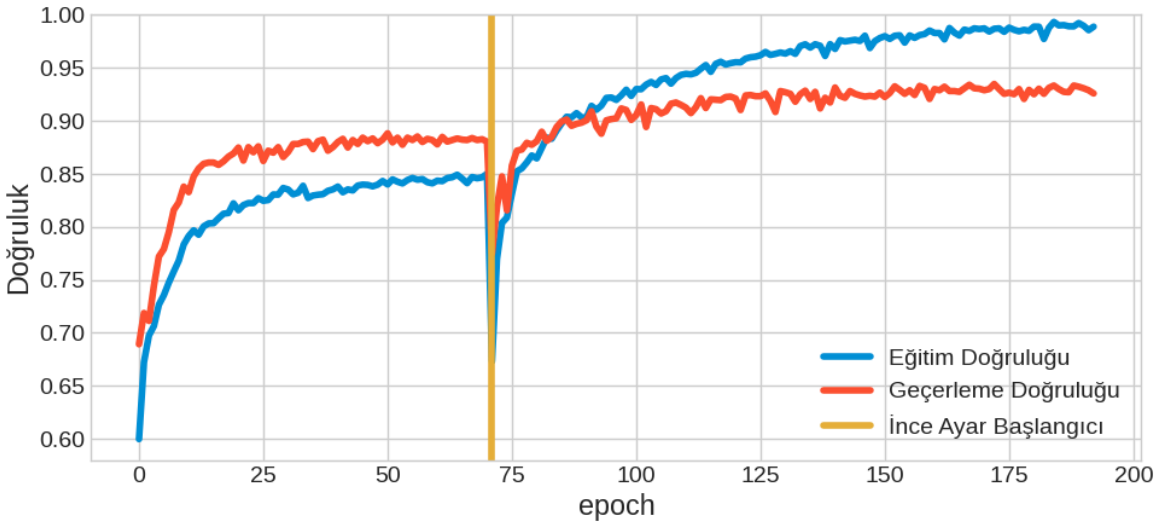


Şekil 6.38. EfficientNetB1 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.38'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model reklamcı, bankacı, riskli, SMS kategorilerinde sırasıyla %91,80, %89,11, %94,27, %99,48 doğruluk göstererek daha iyi sınıflandırma yapmaktayken, zararsızlara ait kategoride %90,74 doğruluk göstererek daha hatalı sınıflandırdığı görülmektedir.

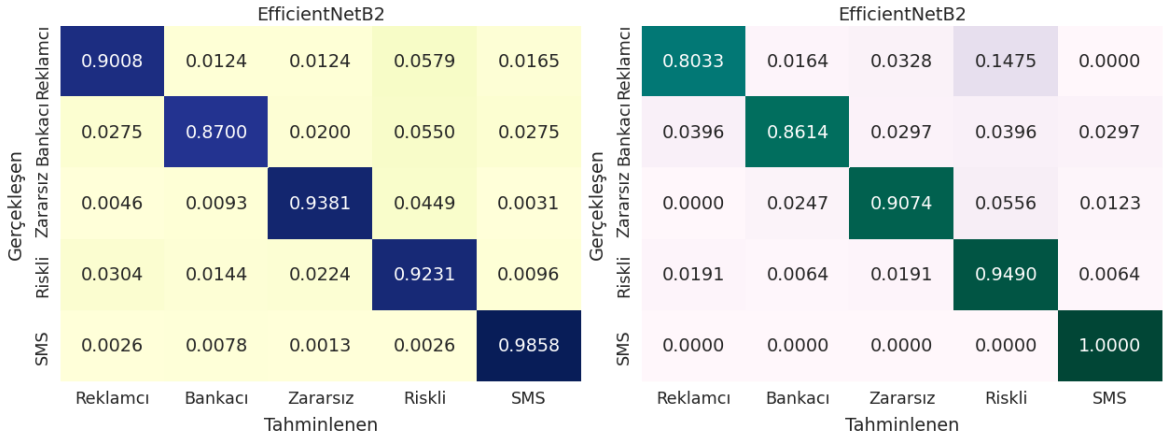
6.11.3. EfficientNetB2 modeli sonuçları

EfficientNetB2 modeli, yöntemden olumlu etkilenen modellerdendir. Modele ait grafik Şekil 6.39'da sunulmaktadır.



Şekil 6.39. EfficientNetB2 modeline ait eğitim grafiği

Şekil 6.39'da gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %88,82 kategorik sınıflandırma doğruluğu göstermekte, ince ayar aşamasında %93,47 kategorik sınıflandırma doğruluğuna erişmektedir. Öğrenim aktarımı aşamasında, her bir epoch 20 saniye olmak üzere toplam 71 epoch eğitilen model, ince ayar aşamasında her bir epoch 25 saniye olmak üzere toplam 122 epoch daha eğitim görmektedir. Modele ait karmaşa matrisleri Şekil 6.40'ta sunulmaktadır.

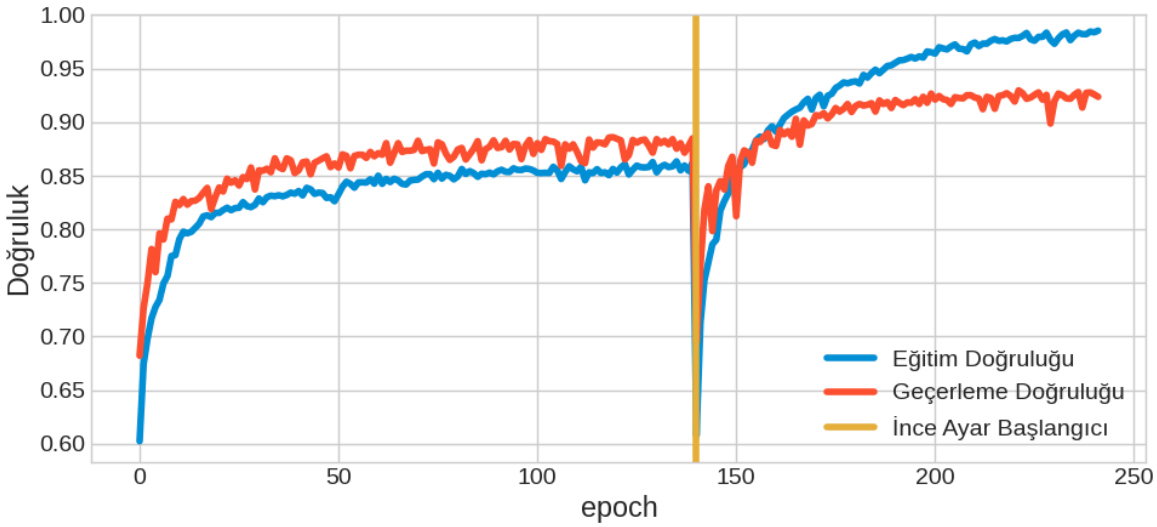


Şekil 6.40. EfficientNetB2 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.40'ta gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model riskli ve SMS kategorilerinde sırasıyla %94,90, %100 doğruluk göstererek daha iyi sınıflandırma yaparken, reklamcı, bankacı, zararsız kategorilerinde sırasıyla %80,33, %86,14, %90,74 doğruluk göstererek daha hatalı sınıflandırma yapmaktadır.

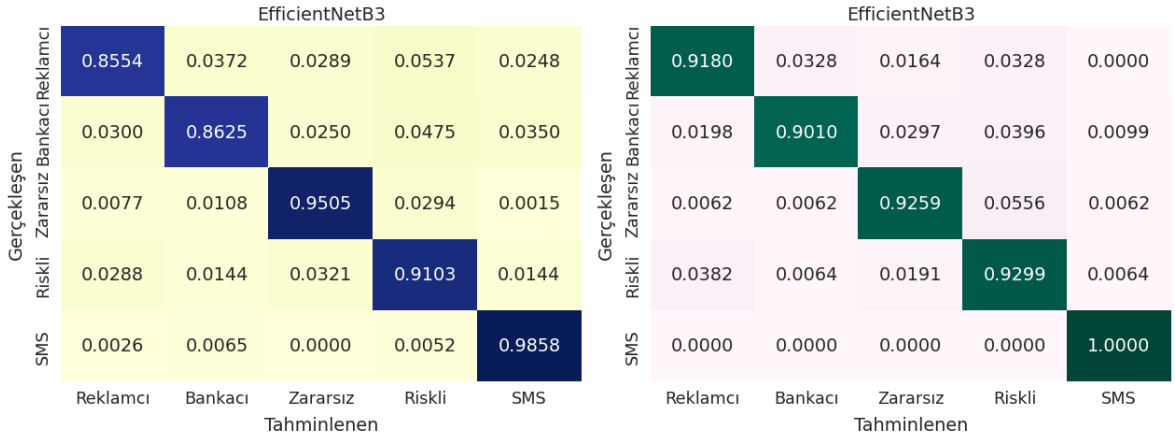
6.11.4. EfficientNetB3 modeli sonuçları

EfficientNetB3 modeli yöntemden olumlu etkilenmektedir. Modele ait eğitim grafiği Şekil 6.41'de gösterilmektedir.



Şekil 6.41. EfficientNetB3 modeline ait eğitim grafiği

Şekil 6.41'de gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %89,59 kategorik sınıflandırma doğruluğu göstermekte, ince ayar aşamasında %92,95 kategorik sınıflandırma doğruluğu elde ederek sınıflandırma performansını attırmaktadır. Öğrenim aktarımı aşamasında, her epoch 24 saniye olmak üzere toplam 140 epoch kadar eğitilen model, ince ayar aşamasında her epoch 30 saniye olmak üzere toplam 102 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.42'de sunulmaktadır.

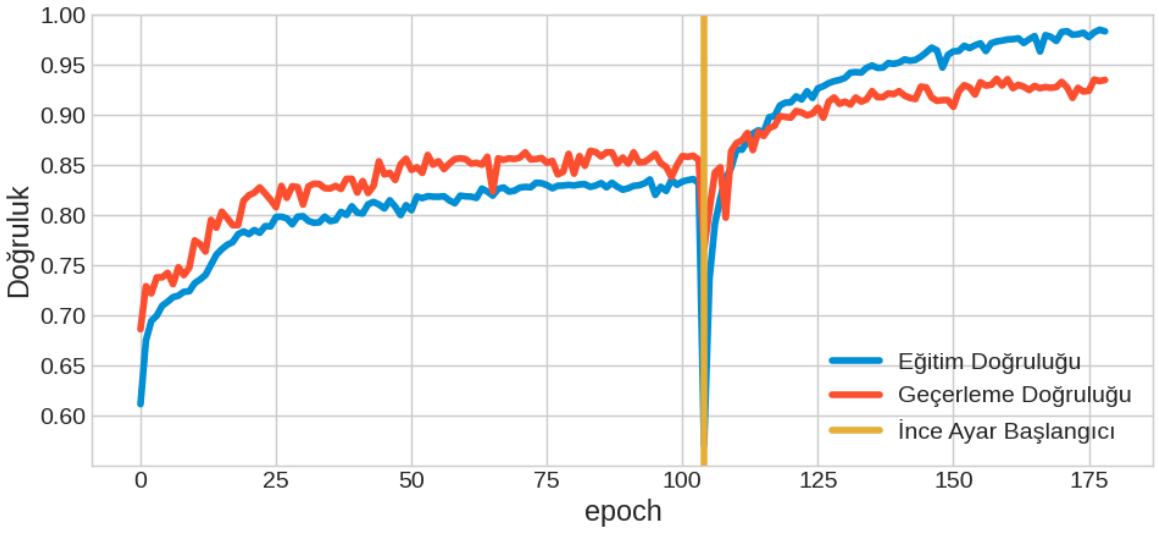


Şekil 6.42. EfficientNetB3 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.42'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model kötücül yazılımlara ait reklamcı, bankacı, riskli, SMS kategorilerinde sırasıyla %91,80, %90,10, %92,99, %100 doğruluk göstererek daha iyi sınıflandırmakta, zararsız yazılımlara ait kategoride %92,59 doğrulukla daha hatalı sınıflandırma gerçekleştirmektedir.

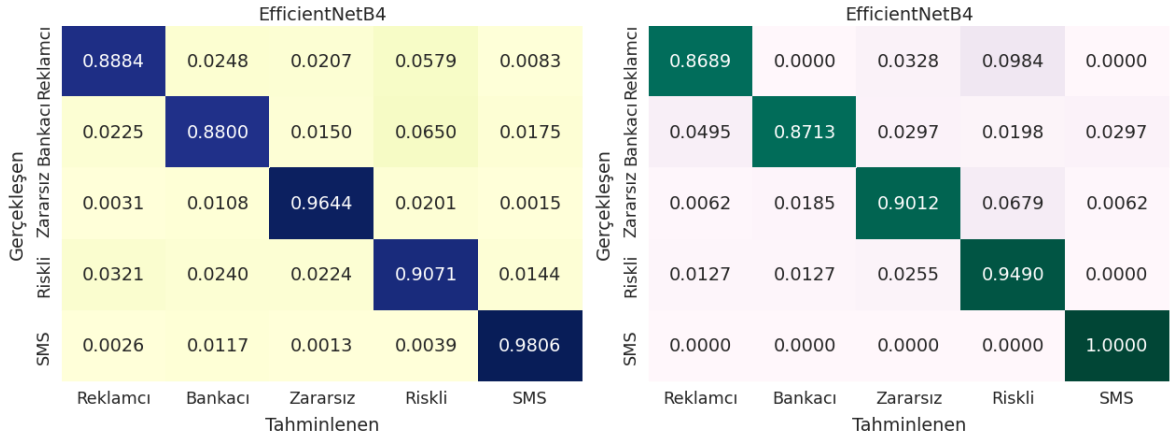
6.11.5. EfficientNetB4 modeli sonuçları

EfficientNetB4 modeli yöntemden olumlu etkilenen modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.43'te gösterilmektedir.



Şekil 6.43. EfficientNetB4 modeline ait eğitim grafiği

Şekil 6.43'te gösterildiği üzere model geçerleme veri setindeki örnekleri, öğrenim aktarımı aşamasında %86,43 doğrulukla, ince ayar aşamasından sonra, %93,62 doğrulukla kategorik olarak sınıflandırmaktadır. Öğrenim aktarımı aşamasında her epoch 31 saniye olmak üzere toplam 104 epoch kadar eğitilen model, ince ayar aşamasında her epoch 39 saniye olmak üzere toplam 75 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.44'te sunulmaktadır.

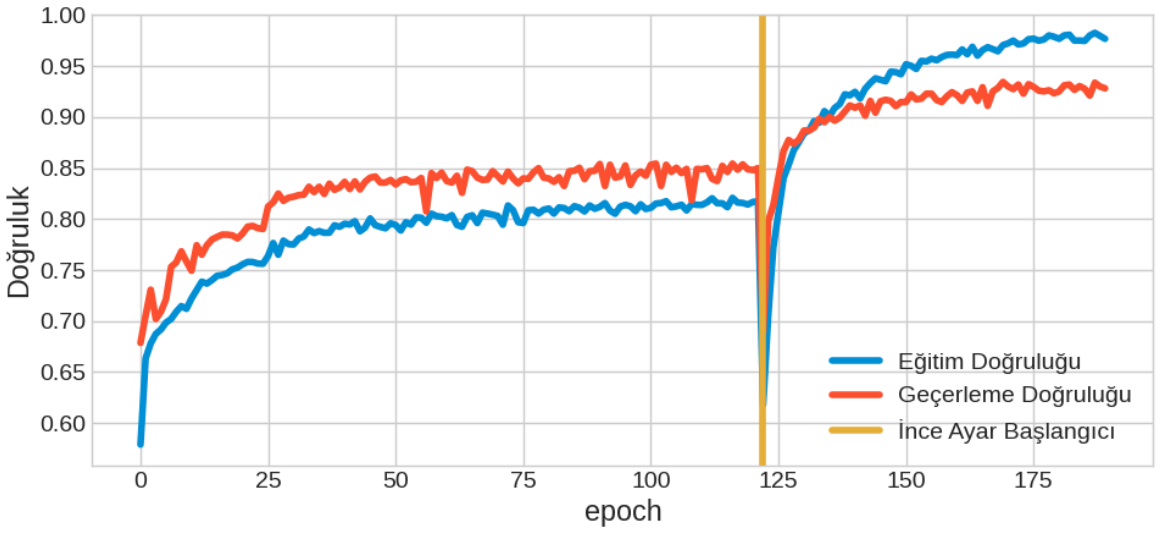


Şekil 6.44. EfficientNetB4 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.44'te gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model riskli ve SMS kategorilerinde sırasıyla %94,90, %100 doğruluk göstererek daha hatasız, reklamcı, bankacı ve zararsız kategorilerinde sırasıyla %86,69, %87,13, %90,12 doğruluk göstererek daha hatalı örnek sınıflandırması gerçekleştirmektedir.

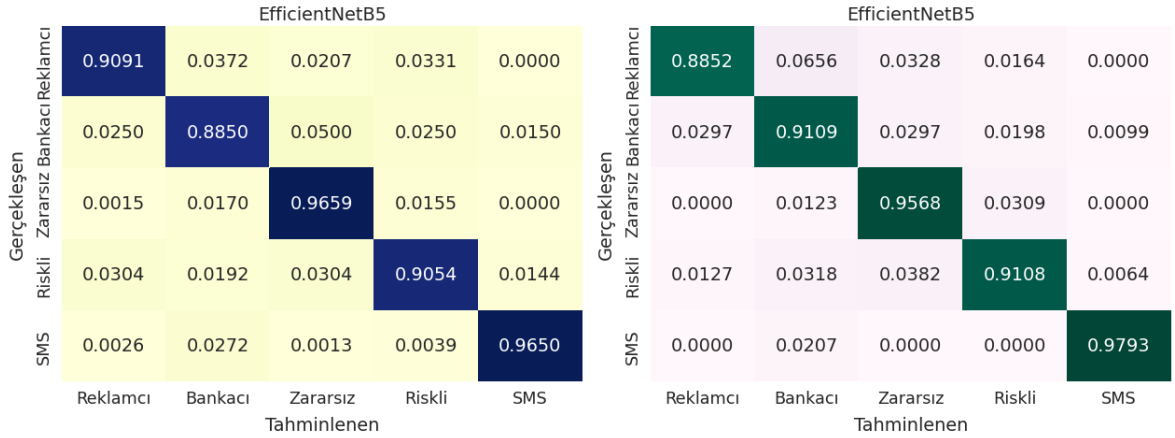
6.11.6. EfficientNetB5 modeli sonuçları

EfficientNetB5, modeli yöntemden en fazla fayda sağlayan modellerden olmaktadır. Modele ait eğitim grafiği Şekil 6.45'te gösterilmektedir.



Şekil 6.45. EfficientNetB5 modeline ait eğitim grafiği

Şekil 6.45'te gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında %85,43 gibi görece kötü bir kategorik sınıflandırma performansı sergilemekte, ince ayar aşamasıyla birlikte %93,44 doğrulukla kategorik sınıflandırma gerçekleştirmektedir. Model öğrenim aktarımı aşamasında, her bir epoch 42 saniye olmak üzere toplam 122 epoch eğitilirken, öğrenim aktarımı aşamasında her bir epoch 53 saniye olmak üzere toplam 68 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.46'da sunulmaktadır.

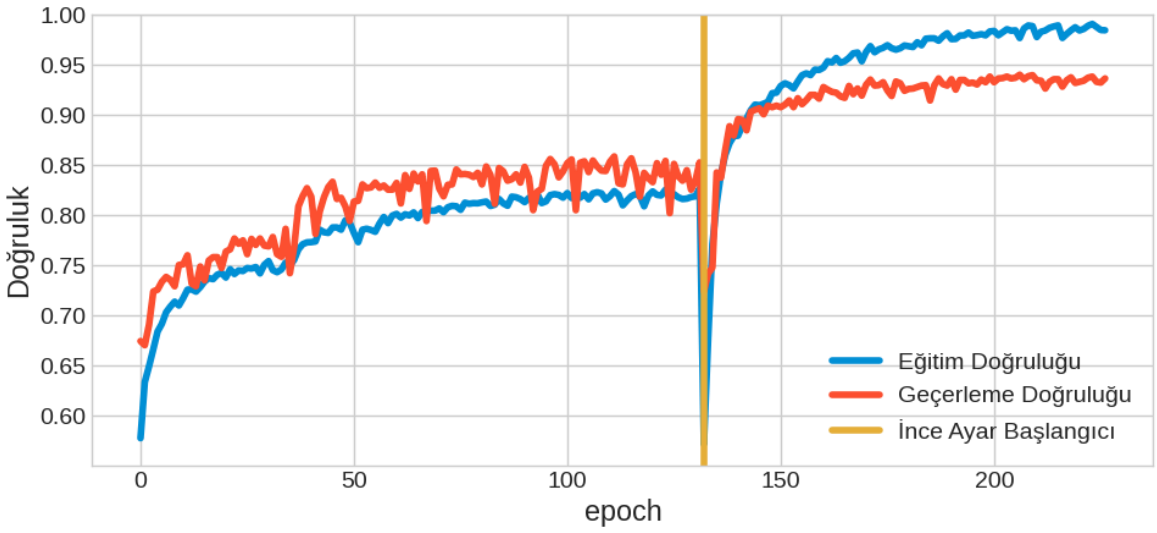


Şekil 6.46. EfficientNetB5 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.46'da gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, modelin reklamcı, zararsız kategorilerinde sırasıyla %88,52, %95,68 doğruluk göstererek daha hatalı sınıflandırdığı, bankacı, riskli, SMS kategorilerinde sırasıyla %91,09, %91,08, %97,93 doğruluk göstererek daha hatasız sınıflandırdığı görülmektedir.

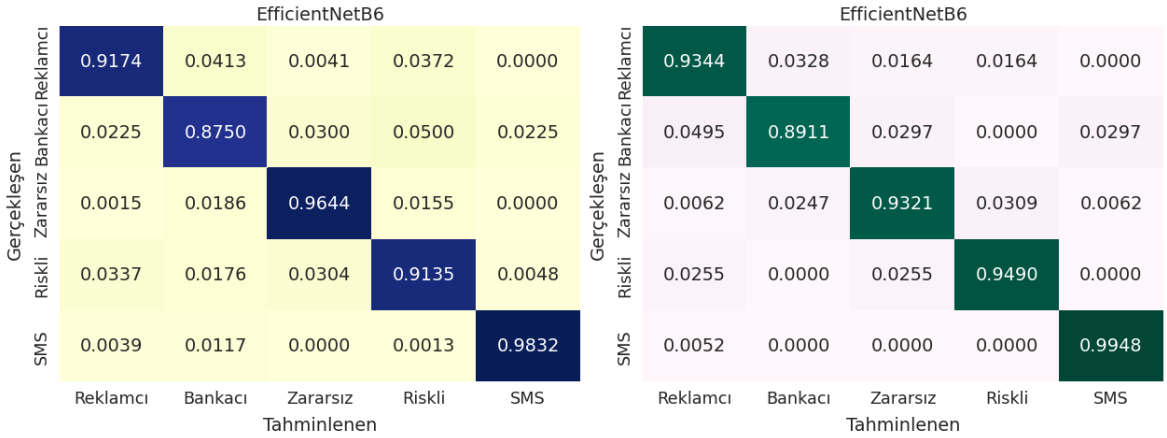
6.11.7. EfficientNetB6 modeli sonuçları

EfficientNetB6 modeli, yöntemden olumlu etkilenmektedir. Modele ait eğitim grafiği Şekil 6.47'de gösterilmektedir.



Şekil 6.47. EfficientNetB6 modeline ait eğitim grafiği

Şekil 6.47'de gösterildiği üzere model geçerleme veri setinde, öğrenim aktarımı aşamasında, %85,87 görece düşük bir kategorik sınıflandırma doğruluğu sergilemekte, ince ayar aşamasında %94,03 kategorik sınıflandırma doğruluğuna erişerek yöntemden en olumlu etkilenen modellerden olmaktadır. Öğrenim aktarımı aşamasında, her bir epoch 52 saniye olmak üzere toplam 132 epoch eğitilen model, ince ayar aşamasında her bir epoch 66 saniye olmak üzere toplam 95 epoch daha eğitilmektedir. Modele ait karmaşa matrisleri Şekil 6.48'de sunulmaktadır.

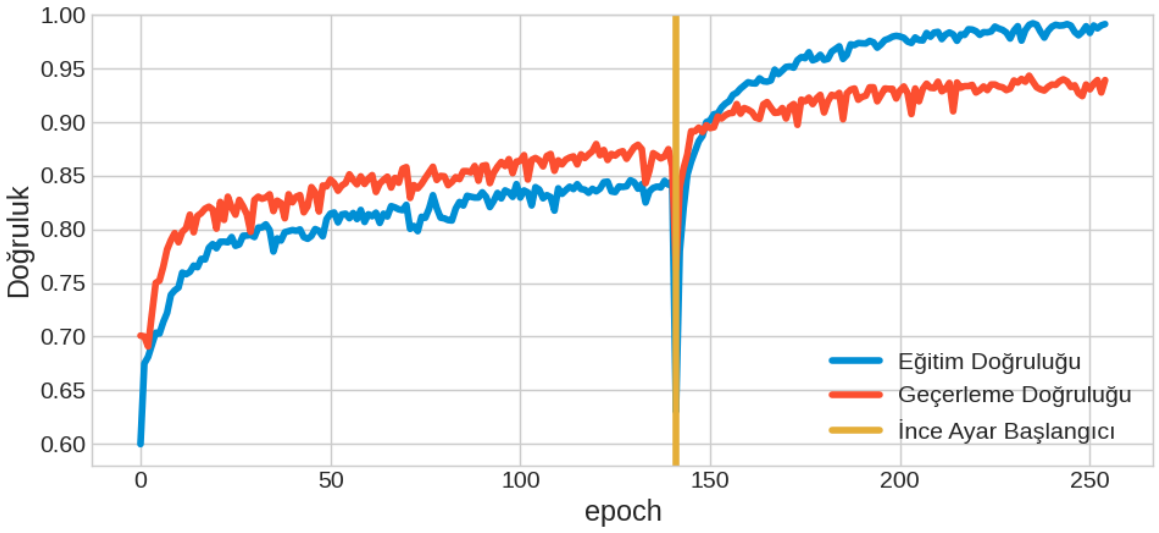


Şekil 6.48. EfficientNetB6 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.48'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model kötücül yazılımlara ait reklamcı, bankacı, riskli, SMS kategorilerinde sırasıyla %93,44, %89,11, %94,90, %99,48 doğruluk göstererek daha hatasız sınıflandırmakta, zararsız yazılımlara ait kategoride %93,21 doğruluk göstererek daha hatalı sınıflandırmaktadır.

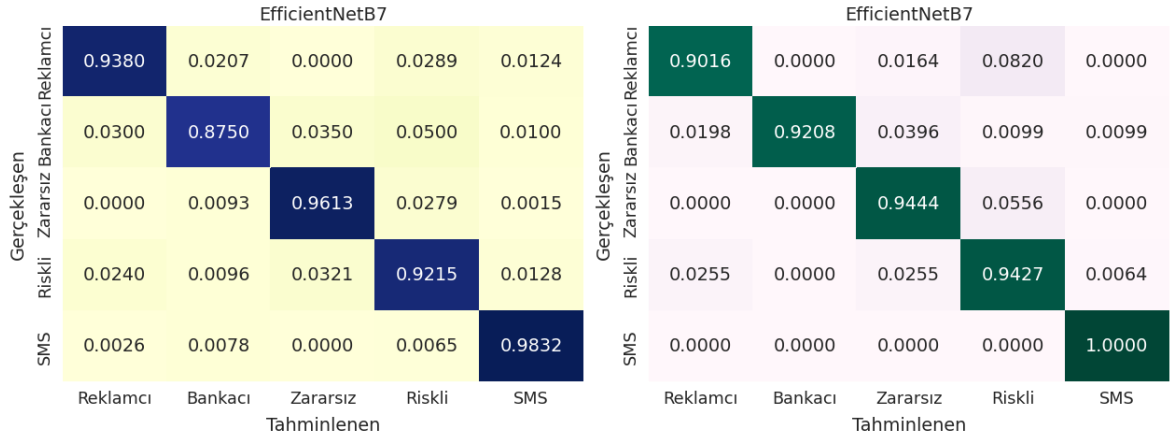
6.11.8. EfficientNetB7 modeli sonuçları

EfficientNet ailesinin en fazla katmana sahip modeli olan EfficientNetB7, yöntemden olumlu etkilenmektedir. Modele ait eğitim grafiği Şekil 6.49'da sunulmaktadır.



Şekil 6.49. EfficientNetB7 modeline ait eğitim grafiği

Şekil 6.49'da gösterildiği üzere model geçerleme veri seti üzerinde, öğrenim aktarımı aşamasında, örnekleri %87,96 doğrulukla kategorik sınıflandırırken, ince ayar aşamasında, örnekleri %94,33 doğrulukla kategorik olarak sınıflandırmaktadır. Yöntemden oldukça olumlu etkilendiği görülen model, aynı zamanda geçerleme veri seti üzerinde kategorik sınıflandırmada en başarılı model olmaktadır. Öğrenim aktarımı aşamasında, her epoch 68 saniye olmak üzere toplam 141 epoch eğitilen model, ince ayar aşamasında her epoch 88 saniye olmak üzere toplam 114 epoch daha eğitim görmektedir. Modele ait karmaşa matrisleri Şekil 6.50'de sunulmaktadır.



Şekil 6.50. EfficientNetB7 modeline ait karmaşa matrisi (sol geçerleme, sağ test veri seti)

Şekil 6.50'de gösterildiği üzere karmaşa matrislerinde yatay eksende tahminlenen sınıflar, dikey eksende gerçek sınıflar ve satır ve sütunların kesiştiği hücrelerde sınıflara ait doğruluk değerleri bulunmaktadır. Şekilde sol tarafta bulunan matris geçerleme veri setine göre hesaplanan karmaşa matrisini, sağ tarafta bulunan matris test veri setine göre hesaplanan karmaşa matrisini göstermektedir. Geçerleme veri setinde elde edilen sınıf doğruluklarıyla, test veri setindeki sınıf doğrulukları karşılaştırıldığında, model reklamcı ve zararsız yazılım kategorilerinde sırasıyla %90,16, %94,44 doğruluk göstererek daha hatalı sınıflandırdığı, bankacı, riskli ve SMS kategorilerinde sırasıyla %92,08, %94,27, %100 doğruluk göstererek daha hatasız sınıflandırdığı görülmektedir.

6.12. Karşılaştırma Sonuçları

Önceki bölümde sunulan bilgiler, bu bölümde toplu bir şekilde çizelgelerde gösterilmektedir. Modellerin sınıflandırma performansları ve eğitim süresileri Çizelge 6.3'te gösterilmektedir.

Çizelge 6.3. Modellerin kategorik sınıflandırma performansları ve eğitim süreleri

Model	Öğrenim Aktarımı Geçerleme Doğruluğu (Geçerleme) *(%)*	İnce Ayar Geçerleme Doğruluğu (Geçerleme) *(%)*	Test Veri Seti Doğruluğu *(%)*	Öğretim Aktarımı Toplam Eğitim Süresi *(sn)*	İnce Ayar Toplam Eğitim Süresi *(sn)*	Toplam Eğitim Süresi *(sn)*
EfficientNetB7	87,96	94,33	95,25	9588	10032	19620
EfficientNetB6	85,87	94,03	94,81	6864	6270	13134
MobileNetV3Large	89,15	93,77	94,36	730	1287	2017
Xception	92,06	93,62	93,92	3000	2430	5430
EfficientNetB4	86,43	93,62	93,32	3224	2925	6149
ResNet101V2	90,72	93,47	95,10	2464	3960	6424
EfficientNetB2	88,82	93,47	92,73	1420	3050	4470
EfficientNetB5	85,43	93,44	93,92	5124	3604	8728
EfficientNetB1	89,41	93,36	93,92	2413	2184	4597
ResNet152V2	91,31	93,25	94,51	4004	4731	8735
VGG19	89,49	93,25	93,18	2225	4536	6761
EfficientNetB0	89,08	93,21	94,51	1800	1638	3438
DenseNet121	88,74	93,18	94,51	2184	3267	5451
ResNet50V2	91,35	93,10	93,92	1786	2256	4042
NASNetLarge	90,23	92,95	93,32	14157	10620	24777
EfficientNetB3	88,59	92,95	94,36	3360	3060	6420
InceptionResNetV2	87,59	92,80	92,14	2769	2736	5505
InceptionV3	88,52	92,43	91,25	1692	1302	2994
NASNetMobile	90,64	92,25	92,58	2580	1798	4378
DenseNet169	89,60	92,02	93,03	2538	3026	5564
MobileNetV3Small	89,49	91,95	93,32	1280	840	2120
MobileNet	90,64	90,64	90,95	810	2640	3450
DenseNet201	90,16	90,16	90,95	5134	903	6037
VGG16	89,23	89,23	90,65	1491	667	2158
MobileNetV2	89,00	89,15	90,50	836	1482	2318

Çizelge 6.3'te modellere ait öğrenim aktarımı geçerleme doğruluğunu, ince ayar geçerleme doğruluğunu, test veri seti doğruluğunu ve öğrenim aktarımı toplam eğitim süresiyle ince ayar toplam eğitim süresi bilgilerini verilmektedir. Öğrenim aktarımı geçerleme doğruluğu, modelin öğrenim aktarımı aşamasında, geçerleme veri seti üzerinde gösterdiği kategorik

sınıflandırma doğruluğunu belirtmektedir. İnce ayar geçerleme doğruluğu, modelin ince ayar aşamasından sonra geçerleme veri seti üzerinde gösterdiği kategorik sınıflandırma doğruluğunu göstermektedir. Test veri seti doğruluğu eğitilmiş modelin, test veri setinde elde ettiği kategorik sınıflandırma doğruluğunu göstermektedir. Öğrenim aktarımı toplam eğitim süresi, modelin öğrenim aktarımında eğitildiği toplam süreyi saniye cinsinden ifade etmektedir. İnce ayar toplam eğitim süresi, modelin ince ayar aşamasında eğitildiği toplam süresi saniye cinsinden ifade etmektedir. Süreler epoch sayısıyla, her bir epochta geçen sürenin çarpımıyla hesaplanmaktadır. Toplam süre, her iki aşamadaki sürelerin toplamı olmakta ve modelin toplam eğitim süresini göstermektedir. EfficientNetB7 modeli geçerleme veri setinde %94,33 doğruluğa, test veri setinde %95,25 doğruluğa ulaşarak hem geçerleme veri seti doğruluğunda hem de test veri seti doğruluğunda en yüksek kategorik sınıflama performansını sırasıyla, göstermektedir. Dikkat çeken diğer model olan MobileNetV3Large, geçerleme veri setinde %93,77, test veri setinde %94,36 olmak üzere yüksek kategorik doğruluğa erişmesinin yanı sıra, verdiği performansa göre, toplam 2017 saniye olan oldukça düşük bir eğitim süresine sahiptir ve eğitim süresi en düşük model olmaktadır. Modeller geçerleme ve test veri setinde genel olarak benzer kategorik sınıflandırma doğruluğu gösterirken, geçerleme veri setine göre %93,47 doğrulukla altıncı sırada yer alan ResNet101V2 modelinin, test veri setinde %95,10 doğruklukla ikinci sıraya yükseldiği görülmektedir. Test veri seti birleşimlerin karşılaştırılmasında kullanılacağından, modeller ince ayar geçerleme doğruluğuna göre en iyiden en kötüye sıralanmaktadır.

Modellerin geçerleme veri seti üzerinde yapılan değerlendirmeye göre hesaplanan F1-skoru, doğruluk, kesinlik, geri çağırma, gerçek negatif oranı, hatalı negatif oranı, hatalı pozitif oranı Çizelge 6.4'te gösterilmektedir.

Çizelge 6.4. Geçerleme veri seti üzerinde alınan tespit performansları

Model	F1 Skoru (%)	Doğruluk (%)	Kesinlik (%)	Geri Çağırma (%)	Gerçek Negatif Oranı (%)	Hatalı Negatif Oranı (%)	Hatalı Pozitif Oranı (%)
EfficientNetB7	98,54	97,80	98,76	98,33	96,13	1,66	3,86
EfficientNetB6	98,64	97,95	98,86	98,42	96,43	1,57	3,56
MobileNetV3Large	98,64	97,95	98,86	98,42	96,43	1,57	3,56
Xception	98,50	97,72	98,47	98,52	95,20	1,47	4,79
EfficientNetB4	98,79	98,17	98,86	98,72	96,43	1,27	3,56
ResNet101V2	98,64	97,95	98,72	98,57	95,97	1,42	4,02
EfficientNetB2	98,38	97,54	98,05	98,72	93,80	1,27	6,19
EfficientNetB5	98,34	97,50	98,90	97,79	96,59	2,20	3,40
EfficientNetB1	98,45	97,65	98,38	98,52	94,89	1,47	5,10
ResNet152V2	98,32	97,46	98,61	98,03	95,66	1,96	4,33
VGG19	98,52	97,76	98,76	98,28	96,13	1,71	3,86
EfficientNetB0	98,69	98,02	98,81	98,57	96,28	1,42	3,71
DenseNet121	98,57	97,83	98,86	98,28	96,43	1,71	3,56
ResNet50V2	98,30	97,42	98,28	98,33	94,58	1,66	5,41
NASNetLarge	98,42	97,61	98,62	98,23	95,66	1,76	4,33
EfficientNetB3	98,30	97,42	98,42	98,18	95,04	1,81	4,95
InceptionResNetV2	98,38	97,54	98,38	98,38	94,89	1,61	5,10
InceptionV3	98,41	97,57	98,09	98,72	93,96	1,27	6,03
NASNetMobile	98,00	96,98	98,32	97,69	94,73	2,30	5,26
DenseNet169	98,47	97,69	98,81	98,13	96,28	1,86	3,71
MobileNetV3Small	98,45	97,65	98,38	98,52	94,89	1,47	5,10
MobileNet	97,94	96,90	98,65	97,25	95,82	2,74	4,17
DenseNet201	98,05	97,05	98,22	97,89	94,42	2,10	5,57
VGG16	97,81	96,68	97,69	97,93	92,72	2,06	7,27
MobileNetV2	97,47	96,19	98,40	96,56	95,04	3,43	4,95

Çizelge 6.4'te modellerin geçerleme veri seti üzerinde yapılan değerlendirmeye göre hesaplanan F1-skoru, doğruluk, kesinlik, geri çağırma, gerçek negatif oranı, hatalı negatif oranı, hatalı pozitif oranı gösterilmektedir. Bu metrikler Bölüm 6.2'de aktarılan model değerlendirme metriklerine göre hesaplanmaktadır. Geçerleme veri setinde, en yüksek F1-skoru %98,79, doğruluk %98,17 olarak gerçekleşmekte ve EfficientNetB4 modeli elde etmektedir. Dengesiz veri setlerinde F1-skorunu baz almak daha doğru bir yaklaşım olmaktadır. EfficientNetB4 modeli, %98,79 F1-skoruyla geçerleme veri seti sonuçlarına

göre en başarılı model olmaktadır. EfficientNetB4 modeli hatalı negatif oranı en düşük modellerden olmaktadır. Zararsız dediği örneklerin sadece %1,27'si, zararlı çıkmaktadır.

Çizelge 6.5. Test veri seti üzerinde alınan tespit performansları

Model	F1 Skoru (%)	Doğruluk (%)	Kesinlik (%)	Geri Çağırma (%)	Gerçek Negatif Oranı (%)	Hatalı Negatif Oranı (%)	Hatalı Pozitif Oranı (%)
EfficientNetB7	98,24	97,32	98,24	98,24	94,44	1,75	5,55
EfficientNetB6	98,14	97,18	97,86	98,43	93,20	1,56	6,79
MobileNetV3Large	98,43	97,62	98,43	98,43	95,06	1,56	4,93
Xception	97,85	96,73	97,66	98,04	92,59	1,95	7,40
EfficientNetB4	97,57	96,29	96,91	98,24	90,12	1,75	9,87
ResNet101V2	98,14	97,18	98,05	98,24	93,82	1,75	6,17
EfficientNetB2	97,76	96,58	97,10	98,43	90,74	1,56	9,25
EfficientNetB5	98,23	97,32	98,62	97,85	95,67	2,14	4,32
EfficientNetB1	97,76	96,58	97,10	98,43	90,74	1,56	9,25
ResNet152V2	98,33	97,47	98,43	98,24	95,06	1,75	4,93
VGG19	98,05	97,03	97,49	98,63	91,97	1,36	8,02
EfficientNetB0	98,44	97,62	97,87	99,02	93,20	0,97	6,79
DenseNet121	98,05	97,03	97,85	98,24	93,20	1,75	6,79
ResNet50V2	97,57	96,29	97,09	98,04	90,74	1,95	9,25
NASNetLarge	97,55	96,29	97,83	97,26	93,20	2,73	6,79
EfficientNetB3	98,15	97,18	97,67	98,63	92,59	1,36	7,40
InceptionResNetV2	97,46	96,14	97,27	97,65	91,35	2,34	8,64
InceptionV3	97,66	96,43	97,28	98,04	91,35	1,95	8,64
NASNetMobile	97,56	96,29	97,46	97,65	91,97	2,34	8,02
DenseNet169	97,84	96,73	98,03	97,65	93,82	2,34	6,17
MobileNetV3Small	98,15	97,18	97,49	98,82	91,97	1,17	8,02
MobileNet	96,65	94,95	97,42	95,89	91,97	4,10	8,02
DenseNet201	96,96	95,40	97,06	96,87	90,74	3,12	9,25
VGG16	97,17	95,69	96,89	97,46	90,12	2,53	9,87
MobileNetV2	97,54	96,29	98,02	97,07	93,82	2,92	6,17

Çizelge 6.5'te modellerin test veri seti üzerinde yapılan değerlendirmeye göre hesaplanan F1-skoru, doğruluk, kesinlik, geri çağırma, gerçek negatif oranı, hatalı negatif oranı, hatalı pozitif oranı gösterilmektedir. Bu değerler Bölüm 6.2'de gösterildiği şekilde hesaplanmaktadır. Test veri setinde, MobileNetV3Large ve EfficientNetB0 modelleri öne çıkmaktadır. MobileNetV3Large ile EfficientNetB0'ın oldukça yakın performans sergilemektedir. İki modelde aynı doğruluk değerini %97,62 alarak elde etmişken, EfficientNetB0 modeli daha yüksek F1-skoru (%98,44) ve daha düşük hatalı negatif oranına (%0,97) sahiptir. İki model birbirine yakınsa, ikinci tip hata oranı daha düşük olan

modeli tercih etmek daha doğru bir seçenek olmaktadır. EfficientNetB0 modelinin ikinci tip hata değeri (%0,97), MobileNetV3Large modeline göre (%1,56) düşük olduğu için EfficientNetB0 modeli test veri seti sonuçlarına göre daha başarılı bulunmaktadır.

Her bir epochtaki yığın büyüklüğü 32 olacak şekilde, modellerden hem test hem de geçerleme veri setindeki bütün örnekleri değerlendirmeleri istediğinde bu işlemi gerçekleştirirken harcadıkları toplam süreler ve her adım için harcanan süreler Çizelge 6.5'te gösterilmektedir.

Çizelge 6.6. Modellerin test ve geçerleme veri setlerini yorumlama süreleri

Model	Geçerleme Toplam Süre (2684 Örnek)	Geçerleme Adım Süresi	Test Toplam Süre (674 Örnek)	Test Adım Süresi
	sn	*ms*	sn	ms
EfficientNetB7	29	236	5	234
EfficientNetB6	21	177	4	177
MobileNetV3Large	4	38	1	41
Xception	8	87	2	87
EfficientNetB4	13	106	2	107
ResNet101V2	11	112	3	119
EfficientNetB2	9	75	2	77
EfficientNetB5	19	169	3	150
EfficientNetB1	11	90	2	69
ResNet152V2	16	163	4	159
VGG19	12	134	3	129
EfficientNetB0	11	108	2	108
DenseNet121	11	101	2	102
ResNet50V2	6	67	1	63
NASNetLarge	53	570	12	551
EfficientNetB3	10	84	2	84
InceptionResNetV2	14	126	3	137
InceptionV3	7	62	1	67
NASNetMobile	11	83	2	83
DenseNet169	12	106	2	108
MobileNetV3Small	5	45	1	33
MobileNet	4	36	1	39
DenseNet201	12	112	3	112
VGG16	8	91	2	87
MobileNetV2	6	58	1	63

Çizelge 6.6'da 25 modele ait geçerleme veri setini yorumlama süresi, test veri setini yorumlama süresi ve adım süreleri gösterilmektedir. Geçerleme ve test veri setlerinde

bulunan örnek sayıları parantez içerisinde belirtilmektedir. Adım süreleri, yığın miktarı kadar verinin grafik belleğine alınmasına kadar geçen donanımsal gecikmeleri göstermektedir. Belirtilen süreler, Tensorflow aracı tarafından raporlanan sürelerdir. MobileNetV3Large modeli 2684 adet örneği 4 saniyede değerlendirmektedir. Toplam süre örnek sayısına oranlandığında, örnek başına, yaklaşık 1,5ms kadar bir sürede sınıflandırma gerçekleştirdiği görülmektedir. Bu dinamik ve hibrit analiz düşünüldüğünde oldukça kısa bir süre kalmaktadır. En başarılı kategorik sınıflandırma doğruluğuna sahip olan model, EfficientNetB7, 2684 örneği 29 saniyede sınıflandırmaktadır. Örnek başına harcanan süre 10,8ms olarak görülmektedir. EfficientNetB7, MobileNetV3Large modeline göre örnek başına 7,25 kat daha fazla süre harcarken, geçerleme veri setine ait sınıflandırma doğruluğunda yaklaşık %0,6 ve test veri setine ait sınıflandırma doğruluğunda yaklaşık %0,9 daha iyi performans gösterebilmektedir. Modellerin eğitimi için harcanan süre maliyet olarak düşünülebilmektedir. Model eğitimi için donanım sağlayan firmalar, donanımın çalıştığı süre başına belirli bir ücretlendirme yapmaktadır. Harcanan zaman maliyet ve kategorik sınıflandırma doğruluğu performans olarak düşünülürse, bir performans/maliyet oranı hesaplanabilmektedir. Gerek eğitim süresi gerek gösterdiği performans düşünüldüğünde, sınırlı kaynaklar ve yetersiz donanım şartlarında en uygun model MobileNetV3Large olarak gözükmektedir.

7. MODELLERİN BİRLEŞTİLMESİ ve BİRLEŞİMLERİN DEĞERLENDİRİLMESİ

Bu bölüm modellerin birleştirilmesi ve birleşimlerin değerlendirilmesi bölümlerini içermektedir. Modellerin birleştirilmesi bölümünde, modellerin birleştirilmesi için kullanılan yöntem ve model birleşimleri açıklanmaktadır. Birleşimlerin değerlendirilmesi bölümünde, oluşturulan birleşimlere ait temel model değerlendirme metrikleri sunulmakta ve bu metriklere göre birleşimlerin değerlendirilmesi gerçekleştirilmektedir.

7.1. Modellerin Birleştirilmesi ve Birleşik Öğrenme

Birleşik öğrenme ya da öğrenimlerin birleştirilmesi, eğitilen birden çok modelin aynı sorun üzerinde birleştirilerek kullanılmasıdır. Birleşik öğrenme gerçekleştirmek için literatürde çeşitli yaklaşımlar bulunmaktadır. Modellerin ürettikleri sonuçları öznitelik olarak kabul edip, bunları birleştirerek yeni bir öznitelik haritası üretip, bu harita üzerinde makine öğrenmesi, derin öğrenme gibi yöntemlerle yeni bir sınıflandırıcı geliştirmek bu çeşitli yöntemlere örnek olarak verilebilmektedir. Fakat bu yöntemler kendi kısıtlarını da beraberinde getirmektedir. Bu tez çalışmasında, birleşik öğrenme gerçekleştirmek için oylamaya dayanan bir yöntem kullanılmaktadır. Oylama temelli yöntemler ilk olarak 1996 yılında yayınlanan çalışmada [59] ortaya konmuştur. Yöntem, ESA modellerinin birleştirilmesinde de kullanılmaktadır [60]. Oyların değerlendirilme yaklaşımına göre daha sonraları, sert oylama ve yumuşak oylama şeklinde literatürde adlandırılmaktadır. Sert oylamada en çok oyu alan etiket sonuç olarak raporlanırken, yumuşak oylamada oy sonuçları olan olasılık dağılımları toplanarak, ortalaması alınır ve bu sonuca göre en yüksek değere sahip etiket raporlanmaktadır. Kullanılan oyları, oy verenlere göre değerlendirilip, bir ağırlık değeriyle çarpılıyorsa ağırlıklı değerlendirme, her oy eşitse ağırlıksız değerlendirme gerçekleştirilmektedir. Tez çalışmasında, birleşimler için kullanılan yöntem ağırlıksız sert oylama olmakla birlikte, çalışmanın devamında, ifade kolaylığı açısından demokratik konsensüs ya da konsensüs adlandırılması tercih edilmektedir. Üretilen modeller demokratik konsensüs yaklaşımı kullanılarak birleştirilmektedir.

Konsensüs bir sorun hakkında tarafların görüşlerinin sorulduğu ve yapılan oylama sonucunda bir karar verildiği yapılardır. Benzer şekilde, konsensüsü oluşturan modellerin, her birine, üzerinde karar verilecek yazılıma ait görsel sunulmaktadır. Konsensüs içerisindeki tüm modeller tahminde bulunduktan sonra, en çok tahmine sahip olan kategori değeri, konsensüs sonucu olarak raporlanmaktadır. Oylamada eşitlik olması durumunda, ilk oy kullanan konsensüs üyesinin verdiği karar kabul edilmektedir. Demokratik konsensüs denmesinin temel sebebi oy verenlerin, oy ağırlıklarının eşit olmasıdır. Konsensüs aşamasında verilen kararlara göre yeni bir öğrenme gerçekleşmemektedir. Tez çalışmasında, konsensüs yaklaşımının tercih edilme sebebi, konsensüs birleşimlerinin kolaylıkla oluşturulabilmesi ve değiştirilebilmesidir. Konsensüs yerine makine öğrenmesi, derin öğrenme tabanlı başka bir yaklaşım kullanıldığında, her yeni konsensüste modelin tekrar eğitilmesi gerekmektedir. Derin sinir ağı yaklaşımında, girdi sayısı model kurulurken belirlenmekte ve sonradan değiştirilmesi oldukça zordur, değişiklik yapıldığında ağırlıklar değişeceğinden ağın tekrardan eğitilmesi gerekmektedir. Konsensüs yaklaşımı tez çalışmasının sunduğu ve kurduğu pratiklerle de uyumlu görülmektedir. Örneğin ileriki yıllarda, yeni son teknoloji modellerin üretildiği bir durumda, başka hiçbir değişiklik yapmadan, sadece bu modelleri eğiterek konsensüs birleşimlerine kolaylıkla eklenebileceği bilinmektedir.

Konsensüs oluşturmak için, yığın ince ayar öğrenim aktarımı yöntemiyle eğitilen modeller, geçerleme veri seti kategorik sınıflandırma doğruluğu temel alarak en iyiden, en kötüye olacak şekilde sıralanmaktadır. Geçerleme doğruluklarının kullanılmasındaki sebep, konsensüs modellerinin de hala test edilmesi gereken modeller olmasından kaynaklanmaktadır. Çizelge 7.1'de model birleşimleri sunulmaktadır.

Çizelge 7.1. Model birleşimleri

En3	EfficientNetB7 + EfficientNetB6 + MobileNetV3Large
En4	En3 + Xception
En5	En4 + EfficientNetB4
En6	En5 + ResNet101V2
En7	En6 + EfficientNetB2
En8	En7 + EfficientNetB5
En9	En8 + EfficientNetB1
En10	En9 + ResNet152V2
En11	En10 + VGG19
En12	En11 + EfficientNetB0
En13	En12 + DenseNet121
En14	En13 + ResNet50V2
En15	En14 + NASNetLarge
En16	En15 + EfficientNetB3
En17	En16 + InceptionResNetV2
En18	En17 + InceptionV3
En19	En18 + NASNetMobile
En20	En19 + DenseNet169
En21	En20 + MobileNetV3Small
En22	En21 + MobileNet
En23	En22 + DenseNet201
En24	En23 + VGG16
En25	En24 + MobileNetV2

Çizelge 7.1'de En3'ten En25'e kadar toplan 23 adet birleşimin yapısı gösterilmektedir. En3, EfficientNetB7, EfficientNetB6 ve MobileNetV3Large modellerinin birleşiminden oluşmaktadır. En3 birleşimine, Xception modelinin dahil edilmesiyle, En4 birleşimi oluşturulmaktadır. Birleşim oluşturulmaları, yeni modeller dahil edilerek En25 kadar sürmektedir. En25, tüm modellerin bulunduğu birleşimdir. Analiz edilecek uygulamalar, her bir model tarafından değerlendirilip, sonuçlar birleştirilerek, uygulama sayısı kadar satırı ve model sayısı kadar sütuna sahip olan bir matris elde edilmektedir. Her bir satır için en çok tekrar eden değer konsensüs sonucu olarak bildirilmektedir. Sonuçlar eşit çıktığı durumda, en başta bulunan yani en iyi modelin yanıtı sonuç olarak bildirilmektedir. Bu

sebeple konsensüste en az üç model bulunmak zorundadır, aksi durumlarda sadece en iyi modelin dediği olmaktadır.

7.2. Birleşim Modellerinin Değerlendirilmesi

Birleşimlerin değerlendirmesinde modellerin değerlendirilmesiyle aynı metrikler kullanılmaktadır. Bu metrikler model değerlendirme bölümünde açıklanmaktadır. Birleşim modellerinin değerlendirmesinde kategorik sınıflandırma doğruluğu ve tespit doğruluğu olmak üzere iki bölümde değerlendirilmektedir.

7.2.1. Kategorik sınıflandırma performansları

Birleşimlere ait kategorik sınıflandırma doğrulukları Çizelge 7.2'de gösterilmektedir.

Çizelge 7.2. Birleşim modellerinin kategorik sınıflandırma performansları

Birleşim Modeli	Geçerleme Veri Seti Kategorik Sınıflandırma Doğruluğu %	Test Veri Seti Kategorik Sınıflandırma Doğruluğu %	Geçerleme Veri Seti Toplam Tahmin Süresi (2684 Örnek) sn	Test Veri Seti Toplam Tahmin Süresi (674 Örnek) sn	Örnek Başına Tahmin Süresi ms
En3	94,74	95,54	42,756	10,763	15,9
En4	95,11	95,25	51,474	12,767	19,1
En5	95,15	95,25	59,853	14,424	22,1
En6	95,23	95,40	66,652	16,857	24,8
En7	95,34	95,40	73,857	20,632	28,1
En8	95,49	95,25	96,481	26,678	36,6
En9	95,56	95,54	109,848	29,180	41,4
En10	95,41	95,84	125,221	25,835	44,9
En11	95,56	95,84	111,432	28,232	41,5
En12	95,56	96,14	120,610	30,539	45
En13	95,49	96,14	117,370	29,136	43,6
En14	95,60	95,99	116,768	30,272	43,7
En15	95,45	96,14	164,346	42,167	61,4
En16	95,30	95,99	170,394	43,846	63,7
En17	95,23	95,99	182,374	50,550	69,3
En18	95,23	95,99	202,310	52,207	75,7
En19	95,30	95,84	208,849	54,166	78,3
En20	95,23	95,69	220,704	57,228	82,7
En21	95,23	95,84	229,057	57,093	85,2
En22	95,30	95,69	224,836	57,075	83,9
En23	95,11	95,69	233,756	58,894	87,1
En24	94,85	95,54	241,006	61,560	90,1
En25	94,70	95,54	246,000	62,471	91,8

Çizelge 7.2'de 23 adet birleşime ait geçerleme veri seti, test veri seti kategorik sınıflandırma doğrulukları ve bu işlemler sırasında harcadıkları süreler gösterilmektedir. Geçerleme ve test veri setinde bulunan örnek sayısı parantez içerisinde gösterilmektedir. Geçerleme veri setinde en yüksek doğruluk değerini En14 birleşimi elde ederken, test veri setinde En12, En13 ve En15 modelleri %96,14 doğrulukla en yüksek kategorik sınıflandırma doğruluğu göstermektedir. Birleşimler, test veri setinde benzer değerler

gösterdiklerinden dolayı, geçerleme veri setindeki performansları da değerlendirilmelidir. En12 modeli, En13 ve En15 modellerine göre geçerleme veri setinde %95,56 doğrulukla daha iyi kategorik sınıflandırma performansı sergilemektedir. En12 modeli, birleşiminde daha az sayıda model barındırdığından daha hızlı kategorik sınıflandırma gerçekleştirecektir. Örnek başına tahmin süresi geçerleme ve test veri setindeki toplam sürenin, geçerleme ve test veri setindeki toplam örnek sayısına bölünmesiyle hesaplanmaktadır. En12 modelinin bir örneği tahmin etme süresi 45 milisaniye olarak gerçekleşmektedir. Çizelge 7.2'de sürelerde model sayısı artmasına rağmen sürede azalmalar görülmektedir, bunun sebebi, deney ortamı olan Google Colab'ın sistem kaynaklarında yaşanan dalgalanmalar olabileceği düşünülmektedir. En15 modelinden sonra birleşimlerin daha iyiye gitmediği görülmektedir. En15 modeliyle birlikte birleşime dahil olan NASNetLarge modelinin örnek başına tahmin süresini %40,5 arttırdığı görülmektedir.

7.2.2. Kötücül tespit performansları

Bölüm 6.2'de açıklanan yaklaşımla, birleşimlere ait geçerleme veri setindeki tespit performansları hesaplanmaktadır. Hesaplamalar sonucunda elde edilen değerler Çizelge 7.3'te gösterilmektedir.

Çizelge 7.3. Birleşim modellerinin geçerleme veri seti tespit performansı

Birleşim Modeli	F1-Skoru %	Doğruluk %	Kesinlik %	Geri Çağırma %	Gerçek Negatif Oranı %	Hatalı Negatif Oranı %	Hatalı Pozitif Oranı %
En3	98,71	98,06	99,06	98,38	97,05	1,61	2,94
En4	98,67	97,98	98,91	98,42	96,59	1,57	3,40
En5	98,89	98,32	99,11	98,67	97,21	1,32	2,78
En6	98,94	98,39	99,11	98,77	97,21	1,22	2,78
En7	98,94	98,39	99,01	98,87	96,90	1,12	3,09
En8	98,89	98,32	99,11	98,67	97,21	1,32	2,78
En9	98,91	98,36	99,06	98,77	97,05	1,22	2,94
En10	98,91	98,36	99,11	98,72	97,21	1,27	2,78
En11	98,89	98,32	99,11	98,67	97,21	1,32	2,78
En12	98,99	98,47	99,25	98,72	97,67	1,27	2,32
En13	98,99	98,47	99,25	98,72	97,67	1,27	2,32
En14	98,99	98,47	99,25	98,72	97,67	1,27	2,32
En15	98,94	98,39	99,25	98,62	97,67	1,37	2,32
En16	98,84	98,24	99,16	98,52	97,36	1,47	2,63
En17	98,86	98,28	99,16	98,57	97,36	1,42	2,63
En18	98,82	98,21	99,01	98,62	96,90	1,37	3,09
En19	98,86	98,28	99,11	98,62	97,21	1,37	2,78
En20	98,81	98,21	99,11	98,52	97,21	1,47	2,78
En21	98,84	98,24	99,16	98,52	97,36	1,47	2,63
En22	98,86	98,28	99,16	98,57	97,36	1,42	2,63
En23	98,79	98,17	99,15	98,42	97,36	1,57	2,63
En24	98,76	98,13	99,11	98,42	97,21	1,57	2,78
En25	98,74	98,09	99,01	98,47	96,90	1,52	3,09

Çizelge 7.3'te 23 adet birleşime ait ve geçerleme veri seti üzerinde hesaplanan F1-skoru, doğruluk, kesinlik, geri çağırma, gerçek negatif oranı, hatalı negatif oranı, hatalı pozitif oranı gösterilmektedir. Bu değerler Bölüm 6.2'de gösterildiği şekilde hesaplanmaktadır. En12, En13 ve En14 birleşimleri geçerleme veri seti üzerinde aynı performansı göstermektedirler. En7 modeliyse en düşük hatalı negatif oranını (%1,12) gösteren model olmaktadır. Pozitif örneklerin kötücül yazılımlar olduğu düşünüldüğünde, bu durum, modelin zararsız işaretlediği örneklerde daha az hata yaptığını göstermektedir.

Birleşim modellerinin test veri setinde gösterdikleri tespit performansları Bölüm 6.2'de gösterildiği şekilde hesaplanmakta ve Çizelge 7.4'te gösterilmektedir.

Çizelge 7.4. Birleşim modellerinin test veri seti tespit performansları

Birleşim Modeli	F1-Skoru %	Doğruluk %	Kesinlik %	Geri Çağırma %	Gerçek Negatif Oranı %	Hatalı Negatif Oranı %	Hatalı Pozitif Oranı %
En3	98,53	97,77	98,44	98,63	95,06	1,36	4,93
En4	98,33	97,47	98,43	98,24	95,06	1,75	4,93
En5	98,14	97,18	98,05	98,24	93,82	1,75	6,17
En6	98,24	97,32	98,05	98,43	93,82	1,56	6,17
En7	98,24	97,32	98,05	98,43	93,82	1,56	6,17
En8	98,24	97,32	98,05	98,43	93,82	1,56	6,17
En9	98,24	97,32	98,05	98,43	93,82	1,56	6,17
En10	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En11	98,43	97,62	98,43	98,43	95,06	1,56	4,93
En12	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En13	98,44	97,62	98,24	98,63	94,44	1,36	5,55
En14	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En15	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En16	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En17	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En18	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En19	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En20	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En21	98,34	97,47	98,24	98,43	94,44	1,56	5,55
En22	98,24	97,32	98,24	98,24	94,44	1,75	5,55
En23	98,24	97,32	98,24	98,24	94,44	1,75	5,55
En24	98,24	97,32	98,24	98,24	94,44	1,75	5,55
En25	98,24	97,32	98,24	98,24	94,44	1,75	5,55

Çizelge 7.4'ye 23 adet birleşime ait ve test veri seti üzerinde hesaplanan F1-skoru, doğruluk, kesinlik, geri çağırma, gerçek negatif oranı, hatalı negatif oranı, hatalı pozitif oranı gösterilmektedir. Test veri setindeki performanslar incelendiğinde, En3 modeli %98,53 F1 skoru, %97,77 doğruluk, %98,44 kesinlik, %98,63 geri çağırma, %95,06 gerçek negatif oranı, %1,36 hatalı negatif oranı, %4,93 hatalı pozitif oranıyla En3 tüm metriklerde diğer modellere göre daha iyi bir performans göstermektedir. En3'ten sonra en iyi performansı %98,44 F1 skoru, %1,36 hatalı negatif oranıyla En13 modeli göstermektedir. En13 birleşiminden sonra, birleşime yeni katılan üyelerin, birleşimi daha

iyiye götürmediği, sonuçların tekrar etmeye başladığı ve performansın düştüğü görülmektedir.

8. TARTIŞMA, SONUÇ ve ÖNERİLER

Android işletim sistemini kullanan cihaz sayısı her geçen gün artmakta, cihazların güvenlik güncelleştirmesi destekleri, her Android versiyonu için yaklaşık 3 yıl sürmekte, fakat ilgili versiyon güvenlik güncelleştirmesi alsa bile, cihaz üreticileri tarafından bu güncelleştirmeler sunulmamakta ya da geç sunulmaktadır. Tüm bu güvenlik güncellemelerinden mahrum cihazlar, saldırganlar tarafından kazançlı hedefler olarak görülmektedir. Tez çalışmasında belirtildiği üzere, bu güvenlik açığı bulunduran cihazlar, siber uzayda potansiyel bir tehdit oluşturmaktadır. Bu tehditlere karşı, makine öğrenmesi ve derin öğrenme kullanan çalışmalar yaygınlaşmaktadır. Kendi kendine özellik çıkartabilme yeteneğiyle derin öğrenme uygulamaları, özellikle, daha önce karşılaşılmamış, bilinmeyen yöntem ve teknikleri içeren sıfırıncı gün saldırıları için, imza tabanlı yöntemlere göre daha uygun çözümler olabildiği görülmektedir [29,30].

İncelenen çalışmalarda, derin öğrenme tabanlı görsel analiz yaklaşımlarının mevcut sorunu çözmede başarılı sonuçlar verdiği görülmektedir. Bir statik analiz çeşidi olan görsel analizin, dinamik ve hibrit analize göre oldukça kısa sürdüğü görülmektedir. Dinamik ve statik analiz gerçekleştiren ve aynı veri setini kullanan bir çalışmada, dinamik ve statik özelliklerin toplanması ve ön işlemler için harcanan toplam süre, 11598 örnek için yaklaşık 24 saat olarak belirtilmiştir [55]. Gerçekleştirilen tez çalışmasında, 16783 örneğe sahip görsel veri setinin hazırlanması için gereken süre, yaklaşık 1 saat sürmektedir. Bu sürenin, optimize edilmemiş kodlama ile elde edildiğini de belirtmek gerekmektedir. Benzer şekilde kullanılan donanımda bu sürenin artmasına ya da azalmasına neden olabileceği göz önünde bulundurulmalıdır. Tez çalışmasında kullanılan modellerin eğitim süreleri göreli olarak diğer yaklaşımlara göre fazla gözükmektedir. Fakat uzun dönemli düşünüldüğünde, bu süre için harcanan maliyet daha düşük kalması beklenmektedir. Her yeni örnekte dinamik ve statik özelliklerin tekrar çıkartılması gerektiği ve bu işlemin görsel analiz yöntemine göre oldukça fazla süre aldığı göz önünde bulundurulmalıdır.

Statik görsel analizin diğer bir avantajı da dinamik analizi engelleyen yöntemlerden ve hatalardan büyük oranda etkilenmemesidir. Saldırganlar, dinamik analiz araçlarını bildiğinden, ilgili araçlarda hata verip çalışmamasına neden olabilecek zafiyetleri sömürebilmektedir. Dinamik araçta hata veren kötücül uygulama, gerçek bir cihaza

bulaştığında hatasız çalışabilmektedir. Bunu engellemenin en iyi yolu gerçek cihazlarda dinamik analiz gerçekleştirmektir fakat bu hem teorikte hem pratikte oldukça maliyetlidir. Gerçekleştirilen tez çalışmasıyla aynı veri setini kullanan dinamik çalışmalar, 17341 adet örnek bulunan veri setinden, hatalardan sonra 11598 adet örneği değerlendirebilirken [1,55], tez çalışmasında, hatalardan sonra 16783 adet örnek değerlendirilebilmektedir. Bu durumda, hataların nedeni, bilinçli olup olmadığı belli olmamakla beraber, statik görsel analizin daha az hatayla karşılaştığı ve daha çok örneği değerlendirebildiği, bozulmalardan daha az etkilendiği görülmektedir.

Güncel bir veri seti olan CICMalDroid2020 [1] veri setinden, DroidMalImg [2] görsel veri seti oluşturularak açık erişime sunulmaktadır. Bu veri seti, Android uygulamalarına ait kötücül yazılımlar barındıran ve açık erişimle sunulan ilk görsel kategorik veri seti olma özelliğini taşımaktadır. Veri seti oluşturulurken, literatürde karşılaşılan, apk2image [38, 43] ve dex2image [40-42] olmak üzere iki görsel oluşturma yöntemi birbiriyle karşılaştırılmaktadır. Dex dosyalarından görsel elde etmenin, APK dosyalarını doğrudan kullanmaya göre, modellerde ortalama olarak %5,11 daha yüksek sınıflandırma doğruluğu gösterdiği gözlemlenmektedir.

Çalışmada, 25 adet popüler, son teknoloji evrişimsel sinir ağı modeli kullanılmaktadır. Bu sayı benzer çalışmalara göre oldukça yüksek bir sayıdır. Karşılaştırılan model sayısı olarak en yakın çalışma olan ve 26 adet model kullanan çalışmada [51], gerçekleştirilen tez çalışmasından farklı olarak, ResNet model ailesinin ilk mimarisi olan, ResNet50, ResNet101 ve ResNet152 modellerinin de kullanıldığı görülmektedir. Gerçekleştirilen çalışmada, ResNet ailesinin yeni mimarileri olan ResNetV2 modelleri tercih edilmekte ve eski ResNet mimarileri yer almamaktadır. Gerçekleştirilen tez çalışmasında, MobileNetV3Small ve MobileNetV3Large modelleri yer alırken, benzer çalışmada [51] bu modeller yer almamaktadır. Veri seti büyüklüğü açısından değerlendirildiğinde, benzer çalışmadaki veri setinde 5986 adet örnek bulunurken [51], gerçekleştirilen tez çalışmasındaki veri setinde 16783 adet örnek bulunmaktadır. Fonksiyonellik açısından değerlendirildiğinde, benzer çalışma kötücül tespiti gerçekleştirirken [51], tez çalışmasında kategorik sınıflandırma gerçekleştirilmektedir.

Son teknoloji modellerin karşılaştırılması için geliştirilen toplu ince ayar öğrenim aktarımı yöntemi sunulmaktadır. Geliştirilen yöntem sayesinde, modellerin ilgili veri setlerindeki

performansları hakkında genel bir öngörü elde edilebilmektedir. Bu sonuçlar, benzer boyut ve türde veri seti kullanacak araştırmacılar için ön fikir vermektedir. Sunulan yöntemi kullanarak araştırmacılar kendi veri setlerindeki model başarımlarını gözlemleyebilir, elde ettikleri bulgulara göre model tercihlerini güncelleyebilirler. Sunulan yöntem, 25 modelden, 21 tanesinde kategorik sınıflandırma doğruluğunda performans artışı sağlarken, 4 model yöntemin olumlu etkisinden yararlanamamaktadır. Her model için yöntemin etkisi farklı olmakla birlikte, tüm modellerde elde edilen değişimin ortalaması alındığında, geçerleme veri setindeki kategorik sınıflandırma doğruluğunda, ince ayar sonrası, ortalama olarak yaklaşık %3,9 artış gözlemlenmektedir. Modelin, test ve geçerleme veri setlerinde elde ettiği kategorik sınıflandırma doğruluğunun ortalaması alınıp, öğrenim aşamasındaki kategorik sınıflandırma doğruluğuyla oranlandığında, EfficientNet modellerinin, %7,02 ortalamayla, uygulanan yöntemde diğer modellere göre daha başarılı sonuçlar aldığı gözlemlenmektedir. EfficientNetB6 modeli, ortalamada %9,95 performans artışıyla yöntemden en olumlu etkilenen model olmaktadır. Temel parametre değerlerinin ve düzenleyici katmanın belirlendiği model olan Xception modelinde, %1,85 ortalama performans artışı gerçekleşmektedir. Bu durum, yöntemin, üstünde özel olarak çalışılmayan modellerde bile performans artışı yakalayarak, başarılı sonuçlar verebildiğini göstermektedir.

Eğitimler sonucunda, en yüksek kategorik değerlendirme doğruluğunu elde eden EfficientNetB7 modeli, geçerleme veri setindeki örnekleri %94,33, test veri setindeki örnekleri %95,25 doğrulukla sınıflandırmaktadır. En yüksek maliyet/performans oranını MobileNetV3Large modelinin sağladığı, bir örneği yaklaşık 1,5ms gibi oldukça düşük bir sürede ve test veri setinde %94,36 kategorik sınıflandırma doğruluğuyla gerçekleştirdiği gözlemlenmektedir. Özellikle düşük donanım kapasitesine sahip cihazlar için, MobileNetV3Large uygun ve başarılı bir model olmaktadır. ResNet101V2 modeli geçerleme veri setinde altıncı sırada yer alırken, test veri setinde ikinci sıraya yükselmektedir. Benzer çalışmada da ResNet101V2 modeli, ikinci veri setinde performansını arttırıp birinci sıraya yükseldiği raporlanmıştır [51]. Her iki çalışmada da modelin görmediği örnekler karşısında daha iyi performans sergilemesinin, modelin daha iyi genelleme yapabilmesinden kaynaklanması olası olabilir, fakat kesin bir sonuca varmadan önce daha çok veri ve test gerekmektedir.

Kötücül yazılımların bir kategoriye, zararsız yazılımların diğer bir kategoriye dahil edilmesiyle hesaplanan tespit performansında, geçerleme veri setinde, EfficientNetB4 modeli %98,17 doğruluk ve %98,79 F1 skoru elde etmektedir. Benzer çalışmada da EfficientNetB4 modeli %95,7 doğrulukla en iyi tespit performansını gösteren model olmuştur [51]. Bağımsız çalışmaların sonuçları birbirini destekler niteliktedir ve bu durum, model sayısı ve veri seti farklılığı düşünüldüğünde şans faktörünün düşük olması beklenmektedir. Test veri setinde, en iyi kötücül tespit performansını EfficientNetB0 modeli %97,62 doğruluk ve %98,44 F1 skoruyla göstermektedir.

Çalışmanın sonunda, eğitilen modellerin birleşimlerinden, birleşik modeller oluşturularak, performansları araştırılmakta, oluşturulan En12 birleşimiyle geçerleme veri setinde %95,56 doğruluğa, test veri setinde %96,14 doğruluğa erişilmektedir. En3 birleşimi test veri seti üzerinde %97,77 doğruluk ve %98,53 F1 skoruyla en başarılı tespit performansını sergilemektedir. En13 modeli, En12 modeli kadar başarılı kategorik sınıflandırma doğruluğu ya da En3 modeli kadar başarılı tespit doğruluğu gösteremese de her iki modelin arasında bir performans sergilemektedir. Hem tespit hem de kategorik sınıflandırmada her iki veri setinde de benzer performans sergilediğinden, çalışmada bir birleşimi birinci ilan etmek gerektiği durumda, En13 birleşimi bu unvanı hak etmektedir. Tez çalışmasında, birleşimdeki model sayısının artmasının bir noktadan sonra birleşimi tekrara ya da daha kötü sonuca götürdüğü gözlemlenmektedir.

Sonuç olarak, modellerin ilgili sorun üzerinde doyuma ulaştığı noktada, ince ayar yöntemi ve devamında birleşim yöntemi, model performanslarını arttırmada işe yarayan uygulamalar olduğu görülmektedir. Evrişimsel sinir ağı modellerinin, saldırganlar tarafından sürekli geliştirilen, tespiti güçleştirilen kötücül zararlılarını tespitinde ve sınıflandırılmasında, statik analiz bir yöntem olmasına rağmen yeni veri setlerinde bile oldukça başarılı sonuçlar sunabildiği görülmektedir. Derin öğrenme modellerinin paylaşımının artmasıyla birlikte, oluşan model havuzundaki model sayısının gün geçtikte artacağı öngörülmekte, modellerin toplu bir şekilde eğitilebilmesi yeni bir sorun doğurmaktadır, bu noktada geliştiriciler, karar vericiler için toplu ince ayar öğrenim aktarımı yöntemi sunulmaktadır. Sunulan yöntemden dört model faydalanamazken, geri kalan modeller önceki performanslarından daha iyi değerler elde etmektedirler. Eğitilen modellerden oluşturulan birleşim modeliyle sonuçlar daha da arttırılabilmektedir.

KAYNAKLAR

1. Mahdavifar, S., Kadir, A. F. A., Fatemi, R., Alhadidi, D., and Ghorbani, A. A. (2020, 17-24 August). *Dynamic android malware category classification using semi-supervised deep learning*. In The 18th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC), Calgary, Canada, 515-522.

2. İnternet: DroidMalImg Veri Seti Web: https://drive.google.com/drive/folders/1b70zhVMEnlfv2UC_56Q9PzwXUhSXw58u?usp=sharing, Son Erişim Tarihi: 31.05.2022

3. İnternet: First SMS trojan detected for smartphones running Android. Web: https://www.kaspersky.com/about/press-releases/2010_first-sms-trojan-detected-for-smartphones-running-android, Son Erişim Tarihi: 31.05.2022.

4. İnternet: State of Malware. MalwareBytes. Web: https://go.malwarebytes.com/rs/805-USG-300/images/MWB_StateOfMalwareReport2021.pdf, Son Erişim Tarihi: 31.05.2022.

5. İnternet: There are over 3 billion active Android devices. The Verge. Web: https://www.theverge.com/2021/5/18/22440813/android-devices-active-number-smartphones-google-2021, Son Erişim Tarihi: 31.05.2022

6. İnternet: Google transparency report. Web: https://transparencyreport.google.com/, Son Erişim Tarihi: 31.05.2022.

7. İnternet: Mobile Android version share Worldwide 2018-2021. Statista. Web: https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/, Son Erişim Tarihi: 05.09.2021.

8. İnternet: Cambridge Analytica and Facebook: The scandal and the fallout so far. The New York Times. Web: https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html, Son Erişim Tarihi: 31.05.2022.

9. İnternet: How a cyber attack transformed Estonia. BBC News. Web: https://www.bbc.com/news/39655415, Son Erişim Tarihi: 31.05.2022.

10. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L. (2009, 20-25 June). *Imagenet: A large-scale hierarchical image database*. In 2009 IEEE conference on computer vision and pattern recognition, Miami, Florida, 248-255.

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

12. Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks*. Advances in neural information processing systems, Lake Tahoe, Nevada, USA, 1106-1114.

13. Simonyan, K., and Zisserman, A. (2014, 7-9 May). *Very deep convolutional networks for large-scale image recognition*. International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 1-14.

14. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015, 7-12 June). *Going deeper with convolutions*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA, USA, 1-9.

15. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016, 27-30 June). *Rethinking the inception architecture for computer vision*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 2818-2826.

16. Ioffe, S., and Szegedy, C. (2015, 6-11 July). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. In International conference on machine learning, Lille, France, 448-456.

17. He, K., Zhang, X., Ren, S., and Sun, J. (2016, 27-30 June). *Deep residual learning for image recognition*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, 770-778.

18. He, K., Zhang, X., Ren, S., and Sun, J. (2016, 8-16 October). *Identity mappings in deep residual networks*. In European conference on computer vision, Amsterdam, The Netherlands, 630-645.

19. Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017, 4-9 February). *Inception-v4, inception-resnet and the impact of residual connections on learning*. In Thirty-first AAAI conference on artificial intelligence, San Francisco, California, USA, 4278-4284.

20. Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017, 21-26 July). *Densely connected convolutional networks*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, United States, 4700-4708.

21. Chollet, F. (2017, 21-26 July). *Xception: Deep learning with depthwise separable convolutions*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, United States, 1251-1258.

22. İnternet: Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. Web: https://arxiv.org/abs/1704.04861, Son Erişim Tarihi: 31.05.2022.

23. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. C. (2018, 18-23 June). *Mobilenetv2: Inverted residuals and linear bottlenecks*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, UT, USA, 4510-4520.

24. Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. (2019, 27-30 October). *Searching for mobilenetv3*. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea (South), 1314-1324.

25. Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018, 18-23 June). *Learning transferable architectures for scalable image recognition.* In Proceedings of the IEEE conference on computer vision and pattern recognitioni, Salt Lake City, UT, USA, 8697-8710.

26. Tan, M., and Le, Q. (2019, 9-15 June). *Efficientnet: Rethinking model scaling for convolutional neural networks*. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, 6105-6114.

27. Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). "Andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, *38*(1), 161-190.

28. Zhao, M., Ge, F., Zhang, T., and Yuan, Z. (2011, 28-31 October). *AntiMalDroid: An efficient SVM-based malware detection framework for android*. In International conference on information computing and applications, Qinhuangdao, China, 158-166.

29. Zhou, Y., and Jiang, X. (2012, 24-25 May). *Dissecting android malware: Characterization and evolution*. In 2012 IEEE symposium on security and privacy, San Francisco, CA USA, 95-109.

30. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., and Siemens, C. E. R. T. (2014, February). *Drebin: Effective and explainable detection of android malware in your pocket*. In Network and Distributed System Security Symposium, San Diego, California, USA, 23-26.

31. Yuan, Z., Lu, Y., Wang, Z., and Xue, Y. (2014, 17-22 August). *Droid-Sec: Deep Learning in Android Malware Detection.* In Proceedings of the 2014 ACM Conference on SIGCOMM, Chicago, Illinois, USA, 371-372,

32. Yuan, Z., Lu, Y., and Xue, Y. (2016). Droiddetector: android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, 21(1), 114-123.

33. Hou, S., Saas, A., Ye, Y., and Chen, L. (2016, 3-5 June). *Droiddelver: An android malware detection system using deep belief network based on api call blocks.* In International conference on web-age information management, Nanchang, China, 54-66.

34. Hou, S., Saas, A., Chen, L., and Ye, Y. (2016, 13-16 October). *Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs.* In 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops, Omaha, NE, USA, 104-111.

35. McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupe, A., and Joon Ahn, G. (2017, 22-24 March). *Deep android malware detection*. In Proceedings of the seventh ACM on conference on data and application security and privacy, Scottsdale, USA, 301-308.

36. Karbab, E. B., Debbabi, M., Derhab, A., and Mouheb, D. (2018). MalDozer: Automatic framework for android malware detection using deep learning. *Digital Investigation*, 24, 48-59.

37. Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. (2011, 20 July). *Malware images: visualization and automatic classification*. In Proceedings of the 8th international symposium on visualization for cyber security, Pittsburgh Pennsylvania USA, 1-7.

38. Kumar, A., Sagar, K. P., Kuppusamy, K. S., and Aghila, G. (2016, 7-8 January). *Machine learning based malware classification for Android applications using multimodal image representations*. In 2016 10th international conference on intelligent systems and control, Coimbatore, Tamilnadu, India, 1-6).

39. Yang, M., and Wen, Q. (2017, 28-30 April). *Detecting android malware by applying classification techniques on images patterns*. In 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis, Chengdu, China, 344-347.

40. Chen, H., Du, R., Liu, Z., and Xu, H. (2018, 14-16 December). *Android Malware Classification using XGBoost based on Images Patterns*. 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference, Chongqing, China, 1358-1362.

41. Ding, Y., Wu, R., and Xue, F. (2018, 25-30 June). *Detecting android malware using bytecode image*. In International Conference on Cognitive Computing, Seattle, WA, USA, 164-169.

42. Hsien-De Huang, T., and Kao, H. Y. (2018, 10-13 December). *R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections*. In 2018 IEEE International Conference on Big Data, Seattle, WA, USA, 2633-2642.

43. Al-Fawa'reh, M., Saif, A., Jafar, M. T., and Elhassan, A. (2020, 8-10 December). *Malware detection by eating a whole APK.* In 2020 15th International Conference for Internet Technology and Secured Transactions, London, United Kingdom, 1-7.

44. Darwaish, A., and Naït-Abdesselam, F. (2020, 7-11 December). *Rgb-based android malware detection and classification using convolutional neural network*. In 2020 IEEE Global Communications Conference, Taipei, Taiwan, 1-6.

45. Hemalatha, J., Roseline, S., Geetha, S., Kadry, S., and Damaševičius, R. (2021). An Efficient DenseNet-Based Deep Learning Model for Malware Detection. *Entropy*, 23(3), 344.

46. Huang, W., Hou, E., Zheng, L., and Feng, W. (2018, May). *MixDroid: a multi-features and multi-classifiers bagging system for Android malware detection.* In AIP conference proceedings, 1967(1), 020015.

47. Vasan, D., Alazab, M., Wassan, S., Safaei, B., and Zheng, Q. (2020). Image-Based malware classification using ensemble of CNN architectures (IMCEC). *Computers & Security*, 92, 101748.

48. Narayanan, B. N., and Davuluru, V. S. P. (2020). Ensemble Malware Classification System Using Deep Neural Networks. *Electronics*, 9(5), 721.

49. Ren, Z., Wu, H., Ning, Q., Hussain, I., and Chen, B. (2020). End-to-end malware detection for android IoT devices using deep learning. *Ad Hoc Networks*, 101, 102098.

50. El-Shafai, W., Almomani, I., and AlKhayer, A. (2021). Visualized Malware Multi-Classification Framework Using Fine-Tuned CNN-Based Transfer Learning Models. *Applied Sciences*, 11(14), 6446.

51. Yadav, P., Menon, N., Ravi, V., Vishvanathan, S., and Pham, T. D. (2022). EfficientNet convolutional neural networks-based Android malware detection. *Computers & Security*, 115, 102622.

52. Lashkari, A. H., Kadir, A. F. A., Taheri, L., and Ghorbani, A. A. (2018, 22-25 October). *Toward developing a systematic approach to generate benchmark android malware datasets and classification*. In 2018 International Carnahan Conference on Security Technology, Montreal, Quebec, Canada, 1-7.

53. Taheri, L., Kadir, A. F. A., and Lashkari, A. H. (2019, 1-3 October). *Extensible android malware detection and family classification using network-flows and API-calls.* In 2019 International Carnahan Conference on Security Technology, Chennai, India, 1-8.

54. Imtiaz, S. I., ur Rehman, S., Javed, A. R., Jalil, Z., Liu, X., and Alnumay, W. S. (2021). DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network. *Future Generation computer systems*, 115, 844-856.

55. Mahdavifar, S., Alhadidi, D., and Ghorbani, A. (2022). Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder. *Journal of Network and Systems Management*, 30(1), 1-34.

56. Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L., and Van Gool, L. (2019, October). *Ai benchmark: All about deep learning on smartphones in 2019*. In 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea (South), 3617-3635.

57. Pan, S. J., and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345-1359.

58. Long, J., Shelhamer, E., and Darrell, T. (2015). *Fully convolutional networks for semantic segmentation*. In Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, MA, USA, 3431-3440.

59. Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2), 123-140.

60. Oloko-Oba, M., and Viriri, S. (2021). Ensemble of EfficientNets for the Diagnosis of Tuberculosis. *Computational Intelligence and Neuroscience*, 15(1), 1-12.

GAZİ GELECEKTİR...