Journal of the Faculty of Engineering and Architecture of Gazi University 37:4 (2022) 1823-1838

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Using network traffic analysis deep learning based Android malware detection

Anıl Utku*
Department of Computer Engineering, Faculty of Engineering, Munzur University, 62000, Tunceli, Turkey

Highlights:

- An automatic malware detection system for Android
- Comparative analysis for Android malware detection
- Android malware detection using network traffic analysis

Keywords:

- Android malware detection
- Deep learning
- Machine learning
- Network traffic analysis

Article Info:
Research Article
Received: 14.05.2021
Accepted: 06.11.2021

DOI:
10.17341/gazimmfd.937374

Correspondence:
Author: Anıl Utku
e-mail: anilutku@munzur.edu.tr
phone: +90 428 213 1794

Graphical/Tabular Abstract

Mobile devices, which are becoming more and more widespread today, have turned into hand-held computers thanks to the multimedia communication and applications. Today, the multimedia applications have been supported by traditional mobile phones. Users can use their mobile devices for many purposes such as internet access, online banking, social networks, file sharing and entertainment. The ability to perform transactions such as financial transactions, online shopping and sensitive data transfers on mobile devices with increased functionality makes mobile devices the target of attackers. In this study, a deep learning based malware detection system has been developed based on the interactions of mobile applications on the network. The developed LSTM-based deep learning model has been analyzed comparatively with NB, RF, SVM, MLP and CNN using accuracy, precision, recall and F-1 metrics. The experimental results showed that the developed LSTM-based deep learning model is more successful in malware detection than others with 95% accuracy.

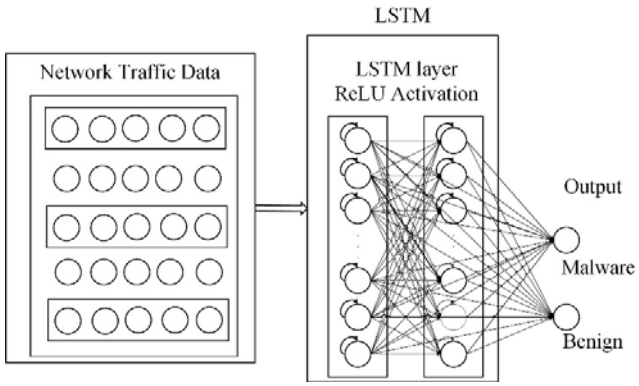


Figure A. Proposed LSTM model for Android malware classification

Purpose: In this study, an automatic detection system for Android malware is proposed. To illustrate the results of the proposed LSTM based model, a comprehensive and comparative experimental study has been carried out.

Theory and Methods:
With the developed LSTM-based deep learning model, it is aimed to detect Android malware by using network traffic data. In addition, the proposed model compared with the existing deep learning algorithms (i.e., MLP and CNN) and conventional machine learning algorithms (i.e., NB, RF and SVM).

Results:
In experimental studies, the LSTM has shown a more successful classification performance in detecting malware compared to other models.

Conclusion:
Experimental results showed that the accuracy values of LSTM, CNN, ML SVM, RF and NB are 0.950, 0.911, 0.738, 0.562, 0.912 and 0.446, respectively. The developed LSTM-based model achieved more successful results for each of the accuracy, precision and recall and f-1 metrics compared to other models.

Journal of the Faculty of Engineering and Architecture of Gazi University 37:4 (2022) 1823-1838

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Ağ trafiği analizi ile derin öğrenme tabanlı Android kötücül yazılım tespiti

Anıl Utku*
Munzur Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 62000 Tunceli, Türkiye

Ö N E Ç I K A N L A R

- Android için otomatik kötücül yazılım tespit sistemi
- Android kötücül yazılım tespiti için karşılaştırmalı analiz
- Ağ trafiği analizi kullanarak derin öğrenme tabanlı Android kötücül yazılım tespiti

Makale Bilgileri
Araştırma Makalesi
Geliş: 14.05.2021
Kabul: 06.11.2021

DOI:
10.17341/gazimmfd.937374

Anahtar Kelimeler:
Android kötücül yazılım tespiti,
derin öğrenme,
makine öğrenmesi,
ağ trafik analizi

ÖZ

Günümüzde giderek yaygınlaşan mobil cihazlar, gelişen multimedya iletişimi ve uygulamaları sayesinde bilgisayarların sağladığı çoğu özelliği kullanıcılarına sunmaktadır. Günümüzde, gelişmiş multimedya uygulamaları geleneksel cep telefonları tarafından desteklenmektedir. Mobil cihazlar artan işlevsellikleri ile birlikte zenginleştirilmiş internet deneyimi, finansal işlemler, sosyal medya platformlarına erişim, paylaşım, müzik ve video gibi birçok amaç için kullanılmaktadır. Bankacılık ve alışveriş gibi hassas kişisel veri aktarımlarının yapıldığı işlemleri, mobil cihazlar üzerinden gerçekleştirme, mobil cihazları saldırganların hedefi haline getirmektedir. Bu çalışmada, mobil uygulamaların ağ üzerindeki etkileşimlerine dayalı olarak derin öğrenme tabanlı bir kötü amaçlı yazılım tespit sistemi geliştirilmiştir. Geliştirilen LSTM tabanlı derin öğrenme modeli, doğruluk, kesinlik, duyarlılık ve F-1 puanı metrikleri kullanılarak NB, RF, SVM, MLP, CNN, RNN ve GRU ile karşılaştırmalı olarak analiz edilmiştir. Deneysel sonuçlar, geliştirilen LSTM tabanlı derin öğrenme modelinin kötücül yazılım tespitinde %95 doğruluk oranı ile karşılaştırılan modellere göre daha başarılı olduğunu göstermiştir.

Using Network traffic analysis deep learning based android malware detection

H I G H L I G H T S

- An automatic malware detection system for Android
- Comparative analysis for Android malware detection
- Deep learning based Android malware detection using network traffic analysis

Article Info
Research Article
Received: 14.05.2021
Accepted: 06.11.2021

DOI:
10.17341/gazimmfd.937374

Keywords:
Android malware detection,
deep learning,
machine learning,
network traffic analysis

ABSTRACT

Mobile devices, which are becoming more and more widespread today, have turned into hand-held computers thanks to the multimedia communication and applications. Today, the multimedia applications have been supported by traditional mobile phones. With their increasing functionality, mobile devices are used for many purposes such as enriched internet experience, financial transactions, access to social media platforms, sharing, music and video. Performing transactions where sensitive personal data transfers such as banking and shopping are carried out on mobile devices make mobile devices the target of attackers. In this study, a deep learning based malware detection system has been developed based on the interactions of mobile applications on the network. The developed LSTM-based deep learning model has been analysed comparatively with NB, RF, SVM, MLP, CNN, RNN and GRU using accuracy, precision, recall and F-1 metrics. The experimental results showed that the developed LSTM-based deep learning model is more successful in malware detection than others with 95% accuracy.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *anilutku@munzur.edu.tr / Tel: +90 428 213 1794

1. GİRİŞ (INTRODUCTION)

Kablosuz iletişim teknolojilerinin gelişimi ve nesnelerin interneti kavramının pazarda yerini almasıyla birlikte akıllı telefonların üretiminde ve gelişiminde de bir artış yaşanmıştır. Akıllı telefonlar ve yeni nesil mobil cihazlar, zaman ve mekândan bağımsız olarak kişisel bilgisayarlar tarafından sağlanan hizmetlerin çoğunu sağladıkları için giderek yaygınlaşmaktadır [1]. Mobil cihazlar, kullanıcıların hayatlarını kolaylaştırmakta, deneyimlerini zenginleştirmekte, sürekli ve etkili bir şekilde veri aktarımını sağlamaktadır [2]. Ancak, mobil cihaz satışlarındaki aralıksız büyüme ve mobil cihazların günlük hayatımızdaki giderek artan etkisi, mobil cihazlara yönelik saldırıların da artmasına neden olmuştur [3].

Android, Google tarafından sunulan desteğin sonucunda Android, iOS, Blackberry ve Windows gibi diğer mobil işletim sistemlerine göre daha popüler bir mobil işletim sistemi haline gelmiştir [4]. Android işletim sistemi, açık kaynak kodlu olmasının sunduğu birçok avantaja sahip olmasına rağmen, izin tabanlı bir güvenlik mekanizmasına sahip olduğu için bazı güvenlik sorunları bulunmaktadır [5]. Android işletim sistemi için resmi uygulama marketleri ve üçüncü parti uygulamalar geliştiren çok sayıda geliştirici bulunmaktadır. Geliştirilen uygulamaların detaylı incelemeleri ve güvenlik testleri yapılmadan uygulama havuzlarına yüklenmesi ciddi güvenlik sorunlarını ortaya çıkaracaktır. Android, hem geliştiriciler açısından hem de işletim sisteminin neden olduğu güvenlik sorunları nedeniyle kötücül yazılım geliştiricilerin ana hedefidir.

Günümüzde eğitim, ulaşım, çevrimiçi alışveriş, sosyal medya, gözetim, iş, bankacılık, üretkenlik ve sağlık yönetimi gibi alanlar için geliştirilmiş birçok mobil uygulama bulunmaktadır [6]. Mobil iletişim teknolojilerinin gelişimi, kullanıcı davranışlarında da değişikliğe sebep olmuştur. Mobil cihazlar bilgilerin aktarımında ve işlenmesinde giderek daha önemli hale gelmiştir. Bu sebeple, mobil cihazlarda depolanan kişisel ve ticari bilgilerin güvenliği ön plana çıkmaktadır [7]. Giderek daha fazla sayıda kullanıcı ve işletme, mobil cihazlarını hem iletişim kurmak hem de kullanıcılarının işlerini ve özel hayatını planlamak ve organize etmek için kullanmaktadır. Şirketler için de bu teknolojiler bilgi sistemlerinin organizasyonunda köklü değişikliklere neden olmakta ve bu nedenle potansiyel risklerin kaynağı haline gelmektedir. Mobil cihazlar, kullanıcının mahremiyetini ve şirketlerin fikri mülkiyetini korumak için erişimlerinin kontrol edilmesi gereken, artan miktardaki hassas bilgileri toplamakta ve işlemektedir [8].

International Data Corporation'ın 2019 raporuna göre, Android işletim sistemi akıllı telefon pazarında % 87 pazar payına sahiptir [9]. Statista'nın raporuna göre 2009'dan Eylül 2019'a kadar yapılan araştırmada Google Play Store'da 2,8 milyon Android uygulaması bulunmaktadır [10]. Statista'nın 2021 raporuna göre 2021 yılı Nisan ayı itibariyle Android'in %72,19 pazar payı ile lider mobil işletim sistemi olduğu ortaya konulmuştur. Google tarafından desteği sürdürülen Android işletim sistemi ve Apple'ın iOS işletim sistemi mobil cihaz pazar pazar payının yüzde 99'undan fazlasına sahiptir. Android işletim sistemi bahsedilen popülerliği ve açık kaynaklı yapısından dolayı saldırganların hedefi konumuna gelmiştir.

Android uygulama güvenliği ile ilgili araştırmacılar ve uygulama marketleri tarafından çalışmalar yapılmaktadır. Google tarafından 2012 yılında tanıtılan Bouncer uygulamasının, kötü amaçlı yazılımlar tarafından atlatılabildiği görülmüştür. Benzer şekilde Google Play Protect, Google tarafından 2017 yılında tanıtılmıştır. Google Play Protect, uygulama güvenliğini sağlamak için sürekli olarak çalışan ve cihazları otomatik olarak tarayan bir servistir. Google Play Protect, uygulamaların üçüncü taraf uygulama mağazalarından veya resmi Google Play Store'dan indirilip indirilmediğine bakmaksızın her gün 50 milyardan fazla uygulamayı taramaktadır [12]. Ancak McAfee tarafından 2018 yılında yayınlanan raporda, Google Play Protect servisinin başarısız olduğu belirtilmiştir. Uygulama pazarları ve servis sağlayıcıları tarafından sunulan güvenlik mekanizmalarına ek olarak, araştırmacılar Android kötücül yazılım tespiti üzerinde çalışmaktadır.

Android kötücül yazılım tespitinde temel olarak statik analiz, dinamik analiz ve hibrit yöntemler kullanılmaktadır. Statik analiz, uygulamaların izin talepleri veya uygulama programlama arayüzü (Application Programming Interface-API) çağrıları analiz edilerek gerçekleştirilir [12]. Dinamik analiz, uygulamaları emülatör adı verilen sanal ve kontrollü ortamlarda veya gerçek aygıtlar üzerinde çalıştırarak uygulamaların davranışını gözlemlemeye dayanır [13]. Hibrit analizler, statik ve dinamik analiz yöntemlerinin bir arada kullanılmasıyla gerçekleştirilir.

Arora vd. ağ trafiğini analiz ederek kural tabanlı bir sınıflandırıcı geliştirmiştir [14]. Bu çalışmada, kullanıcıların özel bilgilerini göndermek için düzenli aralıklarla uzak bir sunucuya bağlanan, sunucular tarafından uzaktan kontrol edilen kötücül yazılımların tespit edilmesi amaçlanmıştır. Geliştirilen modelin ilk aşamasında, Android kötücül yazılımlarının ağ trafiği analiz edilmekte ve kötücül trafiği normal mobil trafikten ayıran bir özellik listesi bulunmaktadır. İkinci aşamada, belirlenen ayırt edici özelliklere göre kural tabanlı bir sınıflandırıcı oluşturulmaktadır. Deneysel sonuçlar, geliştirilen modelin kötücül trafik modellerinin %90'ından fazlasını tespit ettiğini göstermektedir. Malik ve Kaushal tarafından yapılan çalışmada, CREDROID adı verilen ağ trafiği analizine dayalı bir model sunulmuştur [15]. Önerilen model, DNS sorguları ve ağ trafik günlüklerinin çevrimdışı analizi ve uzak sunuculara aktarılan verilere göre kötücül yazılım tespiti gerçekleştirmektedir. CREDROID, uzak sunuculara gönderilen hassas bilgileri tespit eden bir sistemidir. CREDROID, uygulamaların bağlandığı ve verilerin gönderildiği uzak sunucuların güvenli olup olmadığını belirlemek için iletişim protokolleri gibi çeşitli faktörleri değerlendirir. Deneysel sonuçlar, veri kümesindeki uygulamaların %63'ünün kötücül ağ trafiği oluşturduğunu

göstermektedir. Wang vd. tarafından yapılan çalışmada, ağ trafiği analizine dayalı olarak geliştirilen TrafficAV adı verilen bir kötücül yazılım tespit sistemi sunulmuştur [16]. TrafficAV uygulamasında, mobil uygulamalar tarafından üretilen ağ trafiği, veri analizi için kablosuz erişim noktasından sunucuya iletilmektedir. C4.5 karar ağacı algoritması kullanılarak geliştirilen sistem, 8312 iyicil ve 5560 kötücül uygulama kullanılarak test edilmiştir. Deneysel sonuçlar, TCP akış algılama modelinin %98,16'ya ve HTTP algılama modelinin %99,65 doğruluğa ulaştığını göstermiştir.

Milosevic vd. tarafından yapılan çalışmada, Android kötücül yazılım tespitine yönelik makine öğrenmesi tabanlı statik analiz yaklaşımları sunulmuştur [17]. Sunulan ilk yaklaşım izin analizine, ikincisi ise kelime torbası temsil modelini kullanan kaynak kodu analizine dayanmaktadır. İzin tabanlı analiz modelinde SVM, NB, C4.5 Karar ağaçları, JRIP ve AdaBoost algoritmaları kullanılmıştır. Kaynak kod tabanlı sınıflandırma modelinde C4.5 karar ağaçları, NB, SVM, RF, JRIP, Lojistik Regresyon ve AdaBoostM1 algoritmaları kullanılmıştır. Deneysel sonuçlar, kaynak koda dayalı analiz modelinin %95,1 F-1 puanına ve izin tabanlı analiz modelinin ise %89 F-1 puanına sahip olduğunu göstermiştir.

Karbab vd. tarafından yapılan çalışmada, MalDozer adı verilen örüntü madenciliğine dayalı bir model önerilmiştir [18]. Önerilen model, derin öğrenme ile API çağrılarından kötücül yazılım tespiti yapmayı ve kötücül yazılım ailelerini tanımayı amaçlamaktadır. Geliştirilen modelde, Android API çağrıları sabit boyutlu (L) vektör dizisine dönüştürülmüştür. MalDozer evrişim, MaxPooling ve tam bağlı katmanlardan oluşur. MalDozer, Malgenome (1000 örnek), Drebin (5500 örnek), MalDozer veriseti (20000 örnek) ve birleştirilmiş verisetleri gibi kötücül verisetleri ve Google Play Store'dan indirilen 38000 iyicil veriseti üzerinde test edilmiştir. MalDozer, kötücül yazılım tespitinde %99 F-1 puanı elde etmiştir. Mehtab vd. tarafından yapılan çalışmada, AdDroid adı verilen kural tabanlı bir kötücül yazılım tespit sistemi önerilmiştir [19]. Önerilen sistemde, İnternete bağlanmak, dosyaları uzak bir sunucuya yüklemek veya cihaza bir paket yüklemek gibi işlemler kural oluşturmaya dayanmaktadır. AdDroid, Adaboost'un geleneksel sınıflandırıcılarla birleştirildiği topluluk tabanlı makine öğrenmesi gerçekleştirir. Kullanılan veriseti, 910 kötücül ve 510 iyicil olmak üzere 1420 Android uygulamasından oluşmaktadır. Önerilen sistem %98,61 gerçek pozitif (TP), %99,33 gerçek negatif (TN) ve %99,11 doğruluk değerine sahip olmuştur.

Alzaylaee vd. tarafından yapılan çalışmada, Android uygulamalarının durum bilgisi girdilerini kullanarak DL-Droid adı verilen dinamik analiz tabanlı bir kötücül yazılım tespit sistemi önerilmiştir [20]. Geliştirilen sistem kullanılarak, gerçek cihazlar üzerinde 30.000'den fazla iyicil ve kötücül uygulama ile deneysel çalışmalar yapılmıştır. Ayrıca, derin öğrenme kullanılarak durum bilgisinin kullanılmadığı yaygın yaklaşımlarla durum bilgisinin kullanıldığı geliştirilen yönteminin kötücül yazılım tespit performansını ve kod kapsamını karşılaştırmak için deneyler de yapılmıştır. Deneysel sonuçlar, DL-Droid'in geleneksel makine öğrenmesi tekniklerinden daha iyi performans gösterdiğini ortaya koymuştur. Yalnızca dinamik özellikler kullanılarak %97,8'e başarı oranı elde edilmiştir. Dinamik ve statik özelliklerin birlikte kullanıldığı deneylerde ise %99,6 başarı oranı elde edilmiştir.

Lu vd. tarafından yapılan çalışmada, Deep Belief Network (DBN) ve Gated Recurrent Unit (GRU) modelleri ile oluşturulan hibrit bir derin öğrenme modeli kullanılarak bir kötücül yazılım tespit sistemi geliştirilmiştir [21]. Öncelikle Android kötücül yazılımlar analiz edilerek statik ve dinamik uygulama özellikleri çıkarılmıştır. Statik özellikler nispeten bağımsız olduğundan, statik özellikleri işlemek için DBN kullanılmıştır. Dinamik özellikler zamansal korelasyona sahip olduğundan, dinamik özellik dizisini işlemek için GRU kullanılmıştır. DBN ve GRU'nun eğitim sonuçları Backpropagation Neural Network (BP) sinir ağına girilerek nihai sınıflandırma sonuçları çıkarılmaktadır. Deneysel sonuçlar, geleneksel makine öğrenmesi algoritmaları ile karşılaştırıldığında, geliştirilen hibrit derin öğrenme modelinin Android kötücül yazılım tespitinde %96.89 ile daha yüksek bir tespit doğruluğuna sahip olduğunu ve ayrıca gizlenmiş kötücül yazılımlar üzerinde daha iyi bir tespit oranına sahip olduğunu göstermektedir.

Pektaş ve Acarman tarafından yapılan çalışmada, kötücül yazılımların çalışma zamanı sırasında izleyebileceği tüm olası yürütme yollarının bir grafik temsili olarak API çağrısı grafiği şeklinde kullanmıştır [22]. Düşük boyutlu sayısal vektör özellik setine dönüştürülen API çağrı grafikleri oluşturulan derin öğrenme modeline girdi olmuştur. Ardından, her bir ikili fonksiyon için benzerlik tespiti eğitilmiş ve test edilmiştir. Deneysel sonuçlar, sunulan kötücül yazılım tespit sisteminin %98,86 doğruluk, %98,65 F-ölçütü, duyarlılık (recall) %98,47 ve kesinlik (precison) %98,84 değeri elde edildiğini göstermektedir. Karbab ve Debbabi tarafından yapılan çalışmada, Android kötücül yazılımların doğru bir şekilde tespit edilebilmesi ve kötücül yazılım ailelerine göre kümelenmesi için statik analiz tabanlı bir çerçeve olan PetaDroid sunulmuştur [23]. PetaDroid, kötücül yazılımlarda zaman içinde yaşanabilecek özellik değişimlerine ve iyicil uygulamalardaki kod değişimlerine karşı otomatik olarak uyum sağlamaktadır. Geliştirilen sistem, kötücül yazılımların tespit edilebilmesi ve yazılım ailelerine göre kümelenmesi için doğal dil işleme ve makine öğrenmesi tekniklerini kullanmaktadır. PetaDroid farklı referans verisetleri üzerinde kapsamlı bir şekilde değerlendirilmiştir. PetaDroid, yazılımların ailelerine göre kümelenmesinde %96 başarı oranı, kötücül yazılım tespitinde ise %98,99 F1-skor değeri elde etmiştir. Bakour ve Ünver tarafından yapılan çalışmada, kötücül yazılım örneklerini tespit etmek için görüntü tabanlı özellikleri derin öğrenme teknikleri ile hibritleştirmeye dayalı DeepVisDroid adlı yeni bir hibrit derin öğrenme modeli önerilmiştir [24]. Bu amaçla, Android uygulamalarının kaynağındaki bazı dosyalar gri tonlamalı görüntülere dönüştürülerek dört adet gri tonlamalı görüntü veriseti oluşturulmuştur. Daha sonra, oluşturulan görüntü verisetlerinden yerel öznitelikler ve küresel öznitelikler olmak üzere iki tür görüntü tabanlı öznitelik çıkarılmış ve önerilen modelin eğitimi için

kullanılmıştır. Görsel kelime temsili kümesi, her bir görüntüden çıkarılan birden fazla yerel özellik tanımlayıcısından bir özellik vektörü oluşturmak için kullanılmıştır. Daha sonra, çıkarılan yerel ve küresel görüntü tabanlı öznitelikler kullanılarak 1D evrişimli katman tabanlı sinir ağı modeli oluşturulmuş ve eğitilmiştir. Deneysel sonuçlar, önerilen modelin %98 sınıflandırma doğruluğuna sahip olduğunu göstermiştir.

Tablo 1'de incelenen literatürdeki çalışmalar özetlenmiştir. Tablo 1'de incelenen literatürdeki çalışmaların referans, yayın tarihi, analiz metodu, analiz türü ve başarı oranı gibi özelliklerine göre özeti sunulmuştur. Literatürdeki incelenen çalışmaların statik ve dinamik analiz yöntemleriyle birlikte hibrit yöntemleri de kullandığı görülmektedir. Analiz metodu olarak uygulamaların ağ trafiği kullanım durumlarına göre kural tabanlı sınıflandırma, izin tabanlı analiz, uygulama durum bilgisi analizi, API çağrısı analizi, özellik analizi ve ağ trafiği analizi gibi yöntemler kullanılmıştır. Kötücül yazılım tespitinde incelenen çalışmalarda makine öğrenmesi ve derin öğrenme yöntemlerinin kullanıldığı görülmüştür. Bu sayede çalışmaların çoğunda %95 ve üzerinde başarı oranı elde edildiği görülmüştür.

Bu çalışmada, Android uygulamaların ağ trafiği verilerinden kötücül yazılımların tespit edilebilmesi için Naive Bayes (NB), Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), GRU ve Long-Short Term Memory (LSTM) modelleri uygulamalı olarak karşılaştırılmıştır. Her bir model için elde edilen doğruluk (accuracy), kesinlik, duyarlılık ve F-1 puanı (F-1 score) değerlerine göre karşılaştırmalı olarak incelenmiştir. Veriseti olarak Wireshark uygulaması kullanılarak uygulamalara ait pcap uzantılı dosyalardan elde edilen ağ trafiği verileri kullanılmıştır. Bu çalışma, Android uygulamalarının ağ trafik kullanım durumlarına yönelik derin öğrenme tabanlı yapılmış ilk çalışmadır. Ayrıca bu veriseti kullanılarak yapılan ilk dinamik analiz tabanlı çalışmadır. Bu çalışma vasıtasıyla Türkçe literatüre de katkıda bulunabilmek amaçlanmıştır.

2. ANDROİD KÖTÜCÜL YAZILIM TESPİT YÖNTEMLERİ
(ANDROID MALWARE DETECTION TECHNIQUES)

Günümüzde, İnternetin hızla gelişmesi ve kullanıcı sayısının artmasıyla birlikte kötücül yazılımlar önemli siber tehditlerden biri haline gelmiştir [25]. İzinsiz olarak sistemlere sızmak ve kötücül faaliyetlerde bulunmak için tasarlanmış programlar veya kod parçacıkları, kötücül yazılım olarak adlandırılır. Kötücül yazılımlara Truva atları, casus yazılımlar, reklam yazılımları, solucanlar ve virüsler örnek olarak verilebilir [26].

Kötücül yazılım, veri gizliliği ihlallerine ve veri kaybına neden olabilir. Bu sebeple kurumların ve bireysel kullanıcıların verilerini kötücül yazılımlara karşı korumak önemlidir. Etkili koruma, yalnızca kötücül yazılımların doğru ve zamanında tespit edilmesiyle mümkündür [27]. Kötücül yazılım tespiti için genel olarak statik analiz veya dinamik analiz teknikleri kullanılır. Statik analizler, kodu çalıştırmadan kod yapısını ve veri özelliklerini inceler [28]. Dinamik analiz teknikleri, ağdaki veya cihazlardaki kodun davranışını gözlemlemek için yazılımı çalıştırmaya dayanır [29]. Hibrit analizler ise statik analiz ve dinamik analiz

Tablo 1. Literatürdeki çalışmaların özeti (Summary of studies in the literature)

Referans	Yayın Tarihi	Analiz Metodu	Analiz Türü	Kullanılan Yöntem	Başarı Oranı
14	2014	Kural tabanlı sınıflandırma	Statik analiz	Kural tabanlı sınıflandırıcı	%90
15	2016	Ağ trafiği analizi	Statik analiz	Virustotal üzerinden uygulamaların puanlanması	%63
16	2016	Ağ trafiği analizi	Statik analiz	C4.5 algoritması	%99
17	2017	İzin analizi ve kaynak kod analizi	Statik analiz	SVM, NB, C4.5, JRIP, AdaBoost RF ve Lojistik Regresyon algoritmaları	%89
18	2018	Örüntü madenciliği	Statik analiz	CNN tabanlı bir derin öğrenme modeli	%95, %99
19	2020	Kural tabanlı sınıflandırma	Statik analiz	Kural tabanlı sınıflandırıcı	%99
20	2020	Uygulama durum bilgisi analizi	Dinamik analiz	NB, SVM, J48, PART, RF ve Derin sinir ağı (Deep Neural Network- DNN) modelleri	%99
21	2020	Statik ve dinamik uygulama özelliklerinin analizi	Hibrit analiz	Derin inanç ağları (Deep Belief Network-DBN) ve GRU modelleri	%96
22	2020	API çağrısı analizi	Statik analiz	DNN modeli	%98
23	2021	Uygulamalardaki özellik ve kod değişimlerinin analizi	Statik analiz	Doğal dil işleme ve derin öğrenme modelleri	%98
24	2021	Görüntü tabanlı özellik analizi	Hibrit analiz	CNN modeli	%98

tekniklerini birlikte kullanmaktadır. Kötücül yazılım tespitinde makine öğrenmesi yöntemlerinin kullanımı son zamanlarda yeni bir çalışma alanı haline gelmiştir. Makine öğrenmesi kullanılarak kötücül yazılım tespiti, yazılımın dinamik davranışları veya statik kod özellikleri kullanılarak yapılabilir. Bu durumda bu algoritma, problemi çözerken daha önce elde edilen sonuçlardan yararlanıyorsa, denetimli bir öğrenme algoritmasıdır. Denetimli öğrenmede algoritma, veri özelliklerini ve veri etiketlerinden oluşan nesneleri içeren bir eğitim verisi ile eğitilir. Kötücül yazılım tespitinde bu nesneler dosyalardır. Özellikler ise farklı dosya istatistikleri, kullanılan API işlevlerinin listesi olabilir. Öte yandan veri etiketleri, bilinen uygulamaları kötücül veya iyicil olarak etiketlemeye ifade etmektedir. Makine öğrenmesi algoritması daha sonra bu eğitim verisini kullanarak tahmine dayalı bir model oluşturur. Bu sayede daha önce görmediği uygulamaları kötücül veya iyicil olarak sınıflandırabilir.

2.1. Statik Analiz (Static Analysis)

Statik analizler, uygulamaların özellikleri ve kaynak kodları incelenerek yapılan analizlerdir. Statik analizler, uygulamayı çalıştırmadan ve uygulamanın kötücül davranışından etkilenmeden koddaki kötücül alanları bulmanın basit bir yöntemidir. Statik analiz için ayrıştırma, kod çözme, örüntü eşleştirme ve statik sistem çağrısı analizi gibi birçok teknik kullanılabilir. Ancak, yazılıma gömülü şifreleme ve gizleme teknikleri statik analizi zorlaştırır. Ek olarak, statik analiz geleneksel antivirüsler tarafından kullanılan kötüye kullanım algılama ve anormallik algılama olarak kategorize edilebilir.

Android uygulamaları APK (Android Package-APK) dosyası olarak oluşturulur ve derlenir. APK dosyası, apk uzantısına sahip olacak şekilde yeniden adlandırılan sıkıştırılmış bir arşiv dosyasıdır. Android uygulama dosyaları, java kodlarının yürütülebilir formatı olan Dalvik dosyalarından ve AndroidManifest.xml dosyasından oluşmaktadır. AndroidManifest.xml dosyası, uygulama için gerekli olan kaynaklardan, izinlerden ve bildirim dosyasından oluşur ve uygulamanın geliştiricisi tarafından imzalı bir sertifika kullanılarak imzalanır [30].

APK dosyası temel olarak iki klasör (META-INF ve res) ve üç dosya (Classes.dex, AndroidManifest.xml, Resources.arsc) içermektedir. Class.dex ve AndroidManifest.xml, genellikle kötücül uygulama geliştiricilerin hedefleri konumunda olan, APK dosyalarının en hassas ve önemli bileşenleridir.

Statik analizler için uygulamaların arşiv dosyası halinden ayrıştırılması gereklidir. Bu ayrıştırma işlemi için dönüştürücü araçları veya winrar gibi bir arşiv uygulaması kullanılır. Apk dosyası ayrıştırıldıktan sonra, classes.dex dosyası jar dosyasına dönüştürülür. Oluşturulan bu jar dosyası vasıtasıyla, java kodlarının görüntülenmesi ve kötücül kodların belirlenmesi sağlanabilir. Apk ayrıştırma işlemi önlemek için kod karıştırma yöntemleri kullanılmaktadır [31].

2.2. Dinamik Analiz (Dynamic Analysis)

Dinamik analiz, uygulama cihaza yüklendikten sonra kötücül yazılımları tespit eden bir analiz yöntemidir. Dinamik analiz, mobil cihazlara yüklendikten sonra uygulamaların işletim sistemindeki veya ağdaki davranışını inceler.

Dinamik analiz, uygulama davranışını gözlemlemek için uygulamanın izole bir ortamda çalıştırılmasını gerektirir. Dinamik analiz, statik analizden farklı olarak, kod yürütme süreçleri nedeniyle kötücül yazılımın doğal davranışını ortaya çıkarır. Ağ durumunun gözlenmesi, uzak sunucularla kurulan bağlantılar, gönderilen ve alınan paket bilgilerinin izlenmesi ile sistem çağrılarının algılanması dinamik analiz kullanılarak gerçekleştirilir [32]. Android uygulamaları, mobil cihazların yazılım ve donanım olarak simule edildiği sanal bir modeli olan emülatör adı verilen yazılımlar kullanılarak test edilmektedir. Test emülatörleri, Android sanal cihaz yapılandırmalarını destekler.

2.3. Hibrit Yöntemler (Hybrid Methods)

Hibrit analiz yöntemleri, statik ve dinamik analiz yöntemlerinin etkin özelliklerini birleştiren yöntemlerdir. Hibrit yöntemler, öncelikle Android uygulamalarının statik ve dinamik özelliklerini çıkarmaya çalışır. Çıkarılan bu özellikler bir algılama modeli oluşturmak için birleştirilir. Daha sonra çıkarılan statik ve dinamik özelliklere göre, geliştirilecek bir algılama modeli ile uygulamaların sınıflandırılması gerçekleştirilir.

Statik ve dinamik analiz yaklaşımlarını ortak bir zeminde birleştiren hibrit analiz, kötücül yazılım tespitinde daha fazla karmaşıklığa yol açar. Algılama sürecinin daha fazla zaman ve çaba gerektirmesi olasıdır. Hibrit yöntemler, Android kötücül yazılım tespitinde statik veya dinamik analiz yöntemlerinden daha etkili olsa da uygulanabilirliği bir sistem gerçekleştirmek zordur.

Ancak hibrit yöntemler, statik ve dinamik analiz yöntemlerinin birleşimi olduğu için bu yöntem bireysel zayıflıkların üstesinden gelebileceği gibi diğer yöntemlerin avantajlarını da öne çıkarabilir. Bu sayede hibrit yöntemler, zaman ve karmaşıklık maliyeti ile algılama sürecini güçlendirir. Hibrit yöntemler ayrıca sağlamlığı artırabilir, düzenlenen uygulamaları izleyebilir, kod kapsamını artırabilir ve güvenlik açıklarını bulabilir [33].

3. AĞ TRAFİĞİ ANALİZİ İLE DERİN ÖĞRENME TABANLI ANDROİD KÖTÜCÜL YAZILIM TESPİTİ (USING NETWORK TRAFFIC ANALYSIS DEEP LEARNING BASED ANDROID MALWARE DETECTION)

Günümüz mobil çağında, insanlar mobil cihazlara giderek daha bağımlı bir hale gelmiştir. Akıllı telefonlar, en popüler akıllı bilgi işlem cihazı olarak ortaya çıkmaktadır. Bununla birlikte, akıllı telefonları etkileyen çok sayıda güvenlik sorunu da ortaya çıkmıştır.

Son yıllarda, kötücül yazılımların davranışlarını belirlemek için mobil ağ trafiğine dayalı yaklaşımlar önerilmiştir. Bununla birlikte bu yaklaşımlar, özellikle de makine öğrenmesi yöntemlerini kullananlar, mobil trafik verisetlerini toplamanın zorluğu nedeniyle büyük ölçüde sınırlandırılmıştır.

Bu çalışmada, uygulamalar çalıştırılırken elde edilen ağ trafiği verileri kullanılarak zararlı yazılımların tespit edilmesi amaçlanmıştır. Uygulamaların gönderdiği ve aldığı paket bilgileri, iletişim kurmaya çalıştıkları IP adresi bilgileri, dış kaynaklara gönderilen ve alınan paket sayısı, gönderilen ve alınan veri hacmi gibi bilgiler kullanılarak kötücül yazılım tespiti yapılmıştır.

3.1. Veriseti (Dataset)

Bu çalışmada, DroidCollector projesinden elde edilen 4704 iyicil ve 3141 kötücül uygulamanın pcap dosyaları veriseti olarak kullanılmıştır. Uygulamaya özgü özellikler pcap dosyalarından çıkarılarak elde edilmiştir. İletişim kurmak veya bilgi aktarmak için ağ bağlantısı kurmaya çalışan uygulamaların davranışlarını analiz ederek, kötücül yazılımları tespit etmek amaçlanmıştır. Kullanılan verisetindeki uygulamalar, kötücül ve iyicil olarak, veriseti yayımcıları tarafından etiketlenmiştir. Verisetine, https://www.kaggle.com/xwolf12/network-traffic-android-malware bağlantısı aracılığıyla Kaggle üzerinden erişilebilmektedir [34].

Veriseti, 7845 uygulamaya ait 10 özellikten oluşmaktadır. Verisetinde bulunan özellikler aşağıda açıklanmıştır.

- tcp_packets, iletişim sırasında TCP tarafından gönderilen ve alınan paketlerin sayısını ifade eder.
- dist_port_tcp, TCP haricindeki toplam paket sayısını ifade eder.
- external_ips, uygulamanın iletişim kurmaya çalıştığı harici adreslerin sayısını belirtir.
- volume_bayt, uygulamadan harici sitelere gönderilen bayt sayısını ifade eder.
- udp_packets, bir iletişimde gönderilen toplam UDP paketi sayısını ifade eder.
- source_app_packets uygulamadan uzak bir sunucuya gönderilen paket sayısını ifade eder.
- remote_app_packets, harici kaynaklardan alınan paketlerin sayısını ifade eder.
- source_app_bytes, uygulama ile sunucu arasındaki bayt cinsinden iletişim hacmini ifade eder.
- remote_app_bytes, sunucudan uygulama ile emülatöre giden verilerin bayt cinsinden hacmini ifade eder.
- dns_query_times, DNS sorgularının sayısını ifade eder.

Bu çalışmada kullanılan verisetinde yukarıda sunulduğu gibi 10 farklı özellik bulunmaktadır. Her bir özelliğin oluşturulacak sınıflandırma modellerini görebilmek için her bir özellik için feature importance değerleri hesaplanmış ve Tablo 2'de sunulmuştur.

Tablo 2. Verisetindeki özelliklerin önem değerleri
(Feature importance values in the dataset)

Özellik	Önem değeri
tcp_packets	0,10321458
dist_port_tcp	0,05483772
external_ips	0,08113071
volume_bayt	0,13419372
udp_packets	0,00250063
source_app_packets	0,10741905
remote_app_packets	0,13559925
source_app_bytes	0,16523215
remote_app_bytes	0,14647571
dns_query_times	0,06939648

Tablo 2'de görüldüğü gibi source_app_bytes, remote_app_bytes, remote_app_packets, volume_bayt ve tcp_packets özellikleri en yüksek önem değerine sahip özelliklerdir.

3.2. Geliştirilen Model (Developed model)

Bu çalışmada NB, RF, SVM, MLP, CNN, RNN, GRU ve LSTM modelleri uygulamalı olarak karşılaştırılmıştır. Kullanılan verisetine, modeller uygulanmadan önce veri ön işleme işlemleri uygulanmıştır. Verisetindeki eksik ve hatalı olan alanlar kontrol edilmiştir. Veri ön işleme adımından sonra eğitim, doğrulama ve test veri kümeleri seçilmiştir. Verisetinin %67'si eğitim ve %33'ü test olmak üzere ayrılmıştır. Eğitim verilerinin %10'u doğrulama için ayrılmıştır. Model parametrelerinin optimizasyonu için doğrulama verileri kullanılmıştır. Eğitim ve test ayrımından sonra 2589 öğeden oluşan bir test veriseti elde edilmiştir. Eğitim, doğrulama ve test veri kümelerinin elde edilmesinden sonra veriler normalize edilmiştir. Normalleştirmeden sonra, k=10 değeri seçilerek çapraz doğrulama yapılmıştır. Bu sayede modelin eğitim verileri üzerinde ezberleme yapması engellenmiştir. Doğrulama verileri kullanılarak en uygun model parametrelerinin seçilmesi amaçlanmıştır. Elde edilen parametreler kullanılarak model oluşturulmuş ve sınıflandırma tahminleri yapılmıştır. Elde edilen tahmin sonuçlarına göre karışıklık matrisi oluşturularak kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri elde edilmiştir. Geliştirilen sistemin akış diyagramı Şekil 1'de sunulmuştur.

Şekil 1'de akış diyagramı sunulan geliştirilen sistemin algoritması aşağıda sunulmuştur:

Girdi: pcap dosyalarından çıkarılan uygulamalara ait özelliklerdir.
Çıktı: İyicil ya da kötücül olmak üzere yapılan tahmin sonucudur.
(1) Başla.
(2) Verideki eksik ve hatalı alanların kontrolünün yapılması (veri ön işleme).
(3) Eğitim, doğrulama ve test veri kümelerinin seçilmesi ve verilerin normalize edilmesi.
(4) Doğrulama verileri kullanılarak model parametrelerinin optimize edilmesi.

(5) k=10 değeri kullanılarak çapraz doğrulamanın yapılması.
(6) En yüksek doğruluk değerine sahip parametreler seçildi mi? Evet ise 7. adıma git, hayır ise 4. adıma git.
(7) Modelin oluşturulması.
(8) Oluşturulan model kullanılarak sınıflandırma tahminlerinin yapılması.
(9) Elde edilen tahmin sonuçlarına göre kesinlik, duyarlılık, doğruluk ve F-1 puanı değerlerinin hesaplanması.
(10) Bitir.

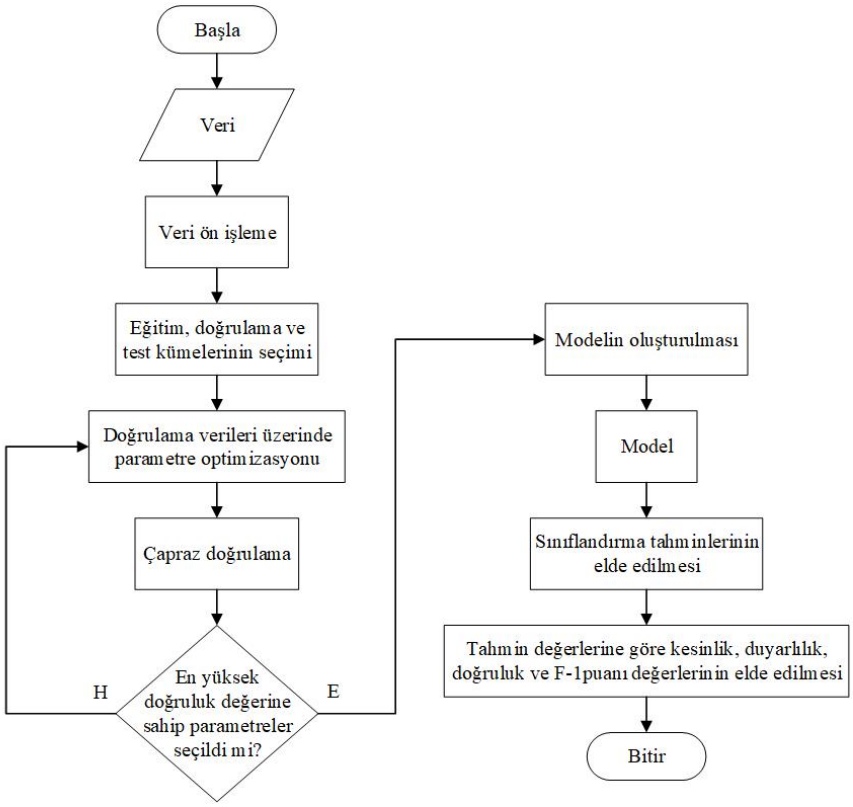
LSTM'in standart mimarisinde bir giriş katmanı, tekrarlı LSTM katmanları ve bir çıkış katmanı bulunmaktadır. Giriş katmanı, LSTM katmanlarına bağlanmaktadır. LSTM katmanındaki tekrarlı bağlantılar, doğrudan hücre çıkış birimlerinden hücre giriş birimlerine, giriş kapılarına, çıkış kapılarına ve unutma kapılarına bağlıdır. Hücre çıkış birimleri, ağın çıkış katmanına bağlıdır. Her bellek bloğunda bir hücre bulunan standart bir LSTM ağındaki toplam parametre sayısı *W*, Eş. 1'de görüldüğü gibi hesaplanabilir:

$$W = n_c \, x \, n_c \, x \, 4 + n_i \, x \, n_c \, x \, 4 + n_c \, x \, n_0 + n_c \, x \, 3 \qquad (1)$$

Burada n_c, bellek hücrelerinin sayısını, n_i giriş birimlerinin sayısını ve n_0 ise çıkış birimlerinin sayısını ifade etmektedir. Olasılıksal Dereceli Azalma (Stochastic Gradient Descent (SGD)) optimizasyon tekniği ile ağırlık ve zaman adımı başına LSTM modellerinin öğrenme hesaplama karmaşıklığı O(1)'dir. Bu nedenle, zaman adımı başına öğrenme hesaplama karmaşıklığı O(W)'dir. Nispeten az sayıda girdiye sahip bir ağ için öğrenme süresine $n_c \times (n_c + n_0)$ faktörü hâkimdir.

Tekrarlı sinir ağları, zaman içinde girdiler arasında paylaşılan nöron katmanlarını içeren bir yapay sinir ağı türüdür. Bu yapı video dizileri, hava durumu modelleri veya hisse senedi fiyatları gibi verilerin modellenmesini sağlar. Bir zaman adımından diğerine bilgi akışını kontrol eden tekrarlı bir hücre tasarlamanın birçok yolu vardır. Tekrarlı bir hücre, sinir ağı için işleyen bir bellek sağlamak üzere tasarlanabilir. En popüler tekrarlı hücre tasarımlarına RNN, GRU ve LSTM örnek verilebilir. LSTM, tekrarlı sinir ağı modellerinden olan RNN'in gelişmiş bir versiyonudur. RNN ve LSTM arasındaki temel fark, bilginin bellekte tutulduğu süredir. LSTM, RNN'e kıyasla daha avantajlıdır çünkü LSTM bellekteki bilgileri RNN'e kıyasla daha uzun süre işleyebilir. LSTM hücreleri, okunan ve kendisine yazılan hücre durumlarını korur. Giriş ve hücre durumu değerlerine bağlı olarak, hücre durumuna okuma, yazma ve çıktı verme değerlerini düzenleyen 4 kapı vardır. İlk kapı, gizli durumun neyi unuttuğunu belirler. İkinci kapı, hücre durumunun hangi kısmına yazıldığını belirlemekten sorumludur. Üçüncü kapı, yazılan içeriğe karar verir. Son kapı ise çıktı üretmek için hücre durumundan okur.

Uzun vadeli bağımlılıkların hatırlanması aşamasındaki başarısı sebebiyle LSTM diğer derin öğrenme modellerine göre ön plana çıkmaktadır. Bu sebeple, yapılan parametre analizi çalışmaları neticesinde çok katmanlı bir LSTM



Şekil 1. Geliştirilen sistemin akış diyagramı (Flow chart of the developed system)

modeli geliştirilmiş ve karşılaştırılan diğer modellere göre göre LSTM tabanlı modelin daha başarılı sonuçlar ürettiği görülmüştür.

Uygulanan modeller için doğrulama verileri kullanılarak en uygun parametrelerin belirlenmesi amacıyla doğruluk metriği kullanılarak parametre analizi çalışması yapılmıştır. RF için 10 ile 200 arasında ağaç sayısı değer aralığı ve iç düğümü bölmek için gereken örnek sayısı olan çocuk sayısı değeri ise 2 olarak seçilmiştir. Ağaç sayısı 500, iç düğümü bölmek için gereken örnek sayısı 2 ve bölme kriteri gini olduğunda en yüksek doğruluk değerine erişildiği için bu değerler deneysel çalışmalarda kullanılmak üzere seçilmiştir. Tablo 3'te RF için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

Tablo 3. Doğruluk metriğine göre RF için parametre seçimi (Parameter selection for RF)

Ağaç sayısı	İç düğümü bölmek için gereken örnek sayısı	Doğruluk
100	2	0,905
300	2	0,912
500	2	*0,914
1000	2	0,911
2000	2	0,907

*Seçilen parametreler

SVM için *c* ile ifade edilen düzenlileştirme parametresi için 0 ile 3 aralığında, kernel ise linear, poly, rbf, sigmoid ve precomputed aralığından seçilmiştir. *c* değeri 0,5 ve kernel rbf olarak seçildiğinde en yüksek doğruluk değerine erişildiği için bu değerler deneysel çalışmalarda kullanılmak üzere seçilmiştir. Tablo 4'te SVM için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

Tablo 4. Doğruluk metriğine göre SVM için parametre seçimi (Parameter selection for SVM)

c	Kernel	Doğruluk
1,5	rbf	0,608
1,0	rbf	0,611
0,5	rbf	0,615
0,5	rbf	0,617
0,5	rbf	*0,620
0,5	rbf	0,618

*Seçilen parametreler

MLP için modelin aynı anda kaç veriyi işleyeceğini ifade eden yığın boyutu parametresi 1 ile 1024 aralığında, oluşturulan derin öğrenme modelin derinliğini sağlayan ve geriye yayılım sağlayarak geçmiş bilgilerin hatırlanmasını sağlayan katman sayısı 1 ile 1024 aralığında ve eğitim tur sayısı ise 1 ile 500 aralığında seçilmiştir. Aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Yığın boyutu 128, katman sayısı 8 ve eğitim tur sayısı ise 100 olarak seçildiğinde en yüksek doğruluk değerine erişildiği için bu değerler deneysel çalışmalarda kullanılmak üzere seçilmiştir. Tablo 5'te MLP için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

Tablo 5. Doğruluk metriğine göre MLP için parametre seçimi (Parameter selection for MLP)

Yığın boyutu	Katman sayısı	Eğitim sayısı	Doğruluk
2	8	100	0,715
2	16	200	0,719
2	32	100	0,721
4	8	100	0,718
16	8	100	0,722
64	8	100	0,728
128	8	100	*0,738
256	8	100	0,731

*Seçilen parametreler

CNN için oluşturulan derin öğrenme modelin derinliğini sağlayan ve geriye yayılım sağlayarak geçmiş bilgilerin hatırlanmasını sağlayan katman sayısı 1 ile 1024 aralığında, eğitim tur sayısı 1 ile 500 aralığında ve özellik çıkarımı için kullanılacak olan kernel filtre matrisinin boyutu ise 1 ile 3 aralığında seçilmiştir. Aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Katman sayısı 64, eğitim tur sayısı 200 ve kernel sayısı 3 olarak seçildiğinde en yüksek doğruluk değerine erişildiği için bu değerler deneysel çalışmalarda kullanılmak üzere seçilmiştir. Tablo 6'da CNN için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

Tablo 6. Doğruluk metriğine göre CNN için parametre seçimi (Parameter selection for CNN)

Katman sayısı	Eğitim sayısı	Kernel sayısı	Doğruluk
32	100	1	0,886
32	100	2	0,892
32	100	3	0,897
64	100	1	0,901
64	100	2	0,903
64	100	3	0,907
128	100	1	0,881
128	100	2	0,879
128	100	3	0,875
64	200	3	*0,911
64	300	3	0,906

*Seçilen parametreler

RNN için modelin aynı anda kaç veriyi işleyeceğini ifade eden yığın boyutu parametresi 1 ile 1024 aralığında seçilmiştir. Derin öğrenme modellerinde katmanlar içinde bulunan nöronlar, ağın işlem birimleridir. Her nöron farklı girdileri işleme sokar ve bunları bir aktivasyon fonksiyonundan geçirir. Aktivasyon fonksiyonunun rolü, verileri bir sonraki katmana gönderilmeden önce ara belleğe almaktır. Oluşturulacak modelde en önemli parametrelerden biri olan gizli katmandaki nöron sayısının değeri 1 ile 1024 aralığında ve eğitim tur sayısı ise 1 ile 500 aralığında seçilmiştir. Aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Yığın boyutu 64, gizli katmandaki nöron sayısı 32 ve eğitim tur sayısı ise 100 olarak seçildiğinde en yüksek doğruluk değerine erişildiği için bu değerler deneysel çalışmalarda kullanılmak üzere seçilmiştir. Tablo 7'de RNN için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

Tablo 7. RNN parametre seçimi (Parameter selection for RNN)

Yığın boyutu	Eğitim sayısı	Nöron sayısı	Doğruluk
1	100	4	0,895
8	100	4	0,897
32	100	4	0,894
64	100	4	0,898
128	100	4	0,903
256	100	4	0,905
64	50	4	0,903
64	200	4	0,908
64	100	8	0,910
64	100	16	0,912
64	100	32	*0,918
64	100	64	0,914
64	100	128	0,911

*Seçilen parametreler

GRU için modelin aynı anda kaç veriyi işleyeceğini ifade eden yığın boyutu parametresi 1 ile 1024 aralığında, gizli katmandaki nöron sayısının değeri 1 ile 1024 aralığında ve eğitim tur sayısı ise 1 ile 500 aralığında seçilmiştir. Aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Yığın boyutu 32, gizli katmandaki nöron sayısı 32 ve eğitim tur sayısı ise 100 olarak seçildiğinde en yüksek doğruluk değerine erişildiği için bu değerler deneysel çalışmalarda kullanılmak üzere seçilmiştir. Tablo 8'de GRU için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

Tablo 8. GRU parametre seçimi (Parameter selection for GRU)

Yığın boyutu	Eğitim sayısı	Nöron sayısı	Doğruluk
1	100	4	0,920
8	100	4	0,923
32	100	4	0,927
64	100	4	0,926
128	100	4	0,925
256	100	4	0,926
32	200	4	0,932
32	300	4	0,938
32	100	8	0,939
32	100	16	0,941
32	100	32	*0,943
32	100	64	0,940

*Seçilen parametreler

Geliştirilen LSTM tabanlı modelin gizli katmanındaki yığın boyutu, eğitim sayısı ve gizli katmandaki nöron sayısı gibi parametreleri belirlemek için doğruluk metriği kullanılarak bir analiz çalışması yapılmıştır. Modelin aynı anda kaç veriyi işleyeceğini ifade eden yığın boyutu parametresi 1 ile 1024 aralığında, gizli katmandaki nöron sayısının değeri 1 ile 1024 aralığında ve eğitim tur sayısı ise 1 ile 500 aralığında seçilmiştir. Aktivasyon fonksiyonu olarak ReLU kullanılmıştır. Analiz sonuçları, LSTM'in yığın boyutu 128, eğitim sayısı 200 ve gizli katmandaki nöron sayısı 30 olduğunda en yüksek doğruluk değerine sahip olduğunu gösterdiğinden, deneysel çalışmalar için bu değerler seçilmiştir. Tablo 9'da LSTM için kullanılan parametreler ve elde edilen doğruluk değerlerinin bir bölümü sunulmuştur.

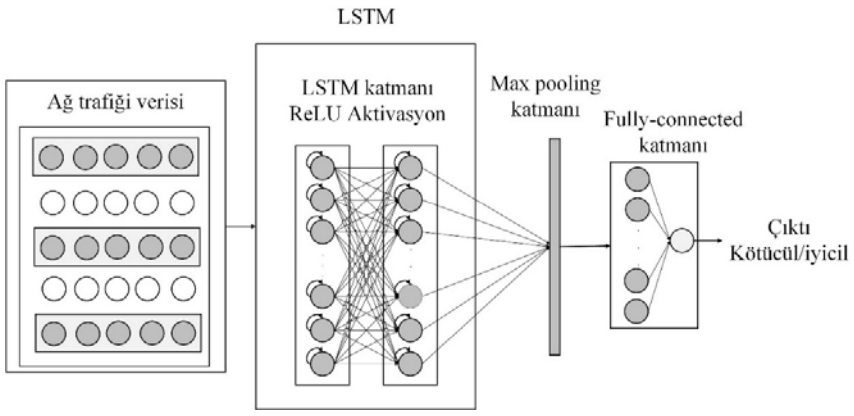
Tablo 9. LSTM parametre seçimi (Parameter selection for LSTM)

Yığın boyutu	Eğitim sayısı	Nöron sayısı	Doğruluk
1	100	4	0,923
8	100	4	0,926
32	100	4	0,931
64	100	4	0,925
128	100	4	0,937
256	100	4	0,935
512	100	4	0,933
128	200	4	0,942
128	300	4	0,938
128	400	4	0,936
128	200	8	0,941
128	200	16	0,944
128	200	32	*0,950
128	200	64	0,945
128	200	128	0,940

*Seçilen parametreler

Geliştirilen LSTM tabanlı model, eğitim verisetindeki Android uygulamalarının ağ trafiği kullanım verilerini girdi olarak almakta ve test verisetindeki uygulamalar için iyicil veya kötücül olmak üzere bir çıktı üretmektedir. Geliştirilen LSTM tabanlı modelin mimarisi Şekil 2'de görülmektedir.

Modeller verisetine uygulandıktan sonra her bir model için karışıklık matrisi hesaplanmış ve doğru pozitif (TP), yanlış pozitif (FP), doğru negatif (TN) ve yanlış negatif (FN)



Şekil 2. Geliştirilen LSTM tabanlı model (Developed LSTM-based model)

Tablo 10. Doğruluk metriğine göre her bir model ve çalıştırma adımı için elde edilen deneysel sonuçlar
(Experimental results obtained for each model and run step according to the accuracy metric)

Veriseti	NB	RF	SVM	MLP	CNN	RNN	GRU	LSTM
1	0,443	0,912	0,562	0,738	0,911	0,924	0,947	0,950
2	0,445	0,910	0,543	0,753	0,908	0,921	0,940	0,954
3	0,447	0,915	0,565	0,742	0,911	0,923	0,942	0,951
4	0,444	0,912	0,562	0,735	0,910	0,914	0,941	0,953
5	0,446	0,912	0,566	0,737	0,913	0,919	0,944	0,951
6	0,445	0,913	0,563	0,741	0,912	0,913	0,941	0,953
7	0,447	0,913	0,565	0,736	0,913	0,915	0,939	0,944
8	0,446	0,911	0,561	0,733	0,914	0,917	0,945	0,948
9	0,448	0,914	0,564	0,739	0,915	0,912	0,946	0,953
10	0,449	0,911	0,571	0,733	0,911	0,922	0,945	0,951
Ortalama	0,446	0,914	0,620	0,738	0,911	0,918	0,943	0,950

değerleri elde edilmiştir. Karışıklık matrisi kullanılarak doğruluk, kesinlik, duyarlılık, ve F-1 puanı değerleri hesaplanmıştır.

3.3. Deneysel Sonuçlar (Experimental results)

Bu çalışmada, ağ trafiği verilerinden kötücül yazılımları tespit etmek için NB, RF, SVM, MLP, CNN ve LSTM kullanılmıştır. Kullanılan her bir model ve algoritma için elde edilen doğruluk, kesinlik, duyarlılık ve F-1 puanı karşılaştırmalı olarak analiz edilmiştir.

Çalışmada, uygulanan makine öğrenmesi algoritmaları ve derin öğrenme modellerinin parametrelerinin belirlenmesi için bir parametre analizi çalışması yapılmıştır.

Uygulanan modellerde, aşırı uyum sorunu ortadan kaldırmak ve oluşturulan modellerin kalitesinin arttırmak için çapraz doğrulama kullanılmıştır. Çapraz doğrulama, henüz görülmemiş test verisetinde yüksek hata oranlarıyla karşılaşmadan önce modelin performansının test edilmesini sağlar. Çapraz doğrulama için k=10 değeri seçilmiştir. Uygulanan tüm modeller cross validation kullanılarak rastgele olarak oluşturulan 10 farklı veri kümesi üzerinde çalıştırılarak elde edilen sonuçların ortalaması alınmıştır. Tablo 10'da doğruluk metriğine göre her bir model ve çalıştırma adımı için elde edilen sonuçlar ve ortalamaları görülmektedir.

İlk olarak, NB modeli verisetine uygulanmıştır. NB için karışıklık matrisi ve deneysel sonuçlar Tablo 11 ve Tablo 12'de görülmektedir.

Tablo 11'de görüldüğü gibi, doğru sınıflandırılan kötücül uygulamaların sayısı 184, doğru sınıflandırılan iyicil uygulamaların sayısı ise 973'tür. NB algoritması, 1157 uygulamayı doğru bir şekilde sınıflandırmıştır. NB, 1432 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 12'de görüldüğü gibi NB için kesinlik değeri 0,814, duyarlılık değeri 0,116, doğruluk değeri 0,446 ve F-1 puanı ise 0,203'tür. Deneysel sonuçlar, NB'in başarısız bir sınıflandırma performansına sahip olduğunu göstermiştir.

Tablo 11. NB için karışıklık matrisi (Confusion matrix for NB)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	184	1390
	İyicil (0)	42	973

Tablo 12. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri (Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,814	0,116	0,446	0,203

RF için karışıklık matrisi ve deneysel sonuçlar Tablo 13 ve Tablo 14'te görülmektedir.

Tablo 13. RF için karışıklık matrisi (Confusion matrix for RF)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1480	94
	İyicil (0)	132	883

Tablo 3'te görüldüğü gibi, doğru sınıflandırılan kötücül uygulama sayısı 1480, doğru sınıflandırılan iyicil uygulamaların sayısı ise 883'tür. RF, 2363 uygulamayı doğru bir şekilde sınıflandırmıştır. RF, 226 uygulamayı yanlış sınıflandırmıştır.

Tablo 14. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri (Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,918	0, 940	0,912	0,929

Tablo 14'te görüldüğü gibi RF için kesinlik değeri 0,918, duyarlılık değeri 0,940, doğruluk değeri 0,912 ve F-1 puanı 0,929'dur. Deneysel sonuçlar, RF'ın kötücül yazılım tespitinde NB'den daha başarılı olduğunu göstermiştir. RF'ın NB'den daha başarılı olması, NB'in model boyutunun küçüklüğü sebebiyle modellerin karmaşık davranışlarını temsil edememesi olarak yorumlanabilir. Öte yandan, RF'ın boyutu çok büyüktür; verilerin dinamik yapısına ve değişimine uyum sağlayabilir.

SVM için karışıklık matrisi ve deneysel sonuçlar Tablo 15 ve Tablo 16'da görülmektedir.

Tablo 15. SVM için karışıklık matrisi
(Confusion matrix for SVM)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1407	167
	İyicil (0)	966	49

Tablo 15'te görüldüğü gibi, doğru sınıflandırılan kötücül uygulamalarının sayısı 1407, doğru sınıflandırılan iyicil uygulamaların sayısı ise 49'dur. SVM, 1456 uygulamayı doğru bir şekilde sınıflandırmıştır. SVM, 1133 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 16. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri (Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,592	0,893	0,562	0,712

Tablo 16'da görüldüğü gibi, SVM için kesinlik değeri 0,592, duyarlılık değeri 0,893, doğruluk değeri 0,562 ve F-1 puanı 0,712'dir. Deneysel sonuçlar, SVM'in kötücül yazılım tespitinde NB'den daha başarılı olduğunu ancak SVM'nin RF'a göre kötü bir sınıflandırma performansına sahip olduğunu göstermiştir. RF, sayısal ve kategorik özelliklerin karışımıyla çalışmaktadır. Özellikler çeşitli ölçeklerde olduğunda, RF avantajlıdır. Bu durum, RF'ın verilerin oldukları gibi kullanılabilmesini sağlar. SVM ise farklı noktalar arasındaki marjı maksimize eder ve noktalar arasındaki mesafeyi hesaplar. Sınıflandırma probleminde RF sınıfa ait olma olasılığını verirken SVM sınıflar arasındaki sınıra en yakın noktaları verir. Verideki özelliklerin kategorik ve nümerik olarak bir arada bulunmaları sebebiyle RF, SVM'e göre daha başarılı bir performans göstermiştir.

MLP için karışıklık matrisi ve deneysel sonuçlar Tablo 17 ve Tablo 18'de görülmektedir.

Tablo 17. MLP için karışıklık matrisi
(Confusion matrix for MLP)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1205	369
	İyicil (0)	307	708

Tablo 17'de görüldüğü gibi, doğru sınıflandırılan kötücül uygulama sayısı 1205, doğru sınıflandırılan iyicil uygulama sayısı ise 708'dir. MLP, 1913 uygulamayı doğru bir şekilde sınıflandırmıştır. MLP, 676 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 18. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri (Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,796	0,765	0,738	0,780

Tablo 18'de görüldüğü gibi MLP için kesinlik değeri 0,796, duyarlılık değeri 0,765, doğruluk değeri 0,738 ve F-1 puanı ise 0,780'dir. Deneysel sonuçlar, MLP'nin kötücül yazılım tespitinde NB ve SVM'den daha başarılı olduğunu ancak MLP'nin RF'dan daha kötü bir sınıflandırma performansına sahip olduğunu göstermiştir. Sinir ağı modellerinden biri olan MLP'nin burada RF'dan daha kötü bir performansa sahip olmasına neden olarak RF'ın ses, görüntü ve metin verisi gibi tablo verileri üzerinde daha başarılı bir şekilde çalışması olarak değerlendirilebilir. Sinir ağı modelleri özelliklerin ölçeklendirilmesine ihtiyaç duymaktadır. Ancak daha büyük öneme sahip özellikler ise eğitimde daha önemli olarak ele alınacaktır ve bu durum da nöronların doygun hale gelerek eğitim aşamasının verimli geçmesini engelleyecektir. Bu çalışmada kullanılan verisetinde kategorik özelliklerin de olması RF'ın MLP'den daha başarılı bir performans göstermesine neden olmuştur.

CNN için karışıklık matrisi ve deneysel sonuçlar Tablo 19 ve Tablo 20'da görülmektedir.

Tablo 19. CNN için karışıklık matrisi
(Confusion matrix for CNN)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1428	145
	İyicil (0)	84	932

Tablo 19'da görüldüğü gibi, doğru sınıflandırılan kötücül uygulama sayısı 1428, doğru sınıflandırılan iyicil uygulamaların sayısı ise 932'dir. CNN, 2360 uygulamayı doğru bir şekilde sınıflandırmıştır. CNN, 229 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 20. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri
(Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,944	0,907	0,911	0,925

Tablo 20'de görüldüğü gibi CNN için kesinlik değeri 0,944, duyarlılık değeri 0,907, doğruluk değeri 0,911 ve F-1 puanı ise 0,925'tir. Deneysel sonuçlar, CNN'nin kötücül yazılım tespitinde NB, SVM ve MLP'den daha başarılı olduğunu ancak CNN'nin RF'a göre daha kötü bir sınıflandırma performansına sahip olduğunu göstermiştir. CNN'in MLP'den daha başarılı bir sınıflandırma performansı göstermesi MLP'nin girdi olarak vektörleri alması ve CNN'in ise girdi olarak tensörü alması olarak yorumlanabilir. CNN bu sayede, özellikler arasındaki ilişkiyi daha iyi çıkarabilecektir. RF ve CNN ise birbirlerine yakın sonuçlar vermiştir. CNN'in özellik çıkarma aşamasındaki başarısı ve RF'ın tablo verileri üzerindeki başarısı yakın sonuçların alınmasını sağlamıştır.

RNN için karışıklık matrisi ve deneysel sonuçlar Tablo 21 ve Tablo 22'de görülmektedir.

Tablo 21. RNN için karışıklık matrisi
(Confusion matrix for RNN)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1482	127
	İyicil (0)	85	895

Tablo 21'de görüldüğü gibi, doğru sınıflandırılan kötücül uygulama sayısı 1482, doğru sınıflandırılan iyicil uygulamaların sayısı ise 895'tir. RNN, 2377 uygulamayı doğru bir şekilde sınıflandırmıştır. RNN, 212 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 22. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri (Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,945	0,921	0,918	0,932

Tablo 22'de görüldüğü gibi RNN için kesinlik değeri 0,945, duyarlılık değeri 0,921, doğruluk değeri 0,918 ve F-1 puanı ise 0,932'dir. Deneysel sonuçlar, RNN'in kötücül yazılım tespitinde NB, SVM, MLP ve CNN'den daha başarılı bir sınıflandırma performansına sahip olduğunu göstermiştir. RNN'in NB, RF, SVM, MLP ve CNN'den daha başarılı bir sınıflandırma performansına sahip olması CNN ve RNN'in birbirinden farklı bir mimariye sahip olması olarak yorumlanabilir. CNN, filtreler ve havuzlama katmanları kullanan ileri beslemeli sinir ağlarıdır. RNN ise sonuçları ağa geri beslerler. CNN'de girdinin boyutu ve elde edilen çıktı sabittir. Yani, bir CNN sabit boyuttaki girdileri alır ve bunları tahmininin güven düzeyi ile birlikte uygun düzeye çıkarır. RNN'de girdinin boyutu ve ortaya çıkan çıktı değişebilir. RNN'in geri beslemeli yapısı geçmişteki özelliklerin hatırlanarak ağa tekrardan girdi olarak sunulmasını ve bu sayede daha başarılı bir sonuç alınmasını sağlamıştır.

GRU için karışıklık matrisi ve deneysel sonuçlar Tablo 23 ve Tablo 24'te görülmektedir.

Tablo 23. GRU için karışıklık matrisi
(Confusion matrix for GRU)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1522	64
	İyicil (0)	83	920

Tablo 23'te görüldüğü gibi, doğru sınıflandırılan kötücül uygulama sayısı 1522, doğru sınıflandırılan iyicil uygulamaların sayısı ise 920'dir. GRU, 2442 uygulamayı doğru bir şekilde sınıflandırmıştır. GRU, 147 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 24. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri (Precision, recall, accuracy and f-1 Score values)

Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,948	0,959	0,943	0,953

Tablo 24'te görüldüğü gibi GRU için kesinlik değeri 0,948, duyarlılık değeri 0,959, doğruluk değeri 0,943 ve F-1 puanı ise 0,953'tür. Deneysel sonuçlar, GRU'nun kötücül yazılım tespitinde NB, SVM, MLP, CNN ve RNN'den daha başarılı bir sınıflandırma performansına sahip olduğunu göstermiştir. GRU'nun karşılaştırılan diğer modellere göre daha başarılı bir sınıflandırma performansı göstermesi GRU'nun RNN'den farklı olarak sahip olduğu mimarideki uzun vadeli bağımlılıkların hafızada tutulmasını sağlayan yapısıdır. Uzun vadeli bağımlılıkların öğrenilmesinin gerektiği problemlerde RNN'i eğitmek zordur. Bunun nedeni, kayıp fonksiyonunun gradyanının zamanla üstel olarak azalmasıdır (kaybolan gradyan problemi).

LSTM için karışıklık matrisi ve deneysel sonuçlar Tablo 25 ve Tablo 26'da görülmektedir.

Tablo 25. LSTM için karışıklık matrisi
(Confusion matrix for LSTM)

		Gerçek değerler	
		Kötücül (1)	İyicil (0)
Tahmin edilen değerler	Kötücül (1)	1533	40
	İyicil (0)	89	927

Tablo 25'te görüldüğü gibi, doğru sınıflandırılan kötücül yazılım sayısı 1533, doğru sınıflandırılan iyicil uygulamaların sayısı ise 927'dir. LSTM, 2460 uygulamayı doğru bir şekilde sınıflandırmıştır. LSTM, 129 uygulamayı ise yanlış sınıflandırmıştır.

Tablo 26. Kesinlik, duyarlılık, doğruluk ve F-1 puanı değerleri
(Precision, recall, accuracy and f-1 Score values)

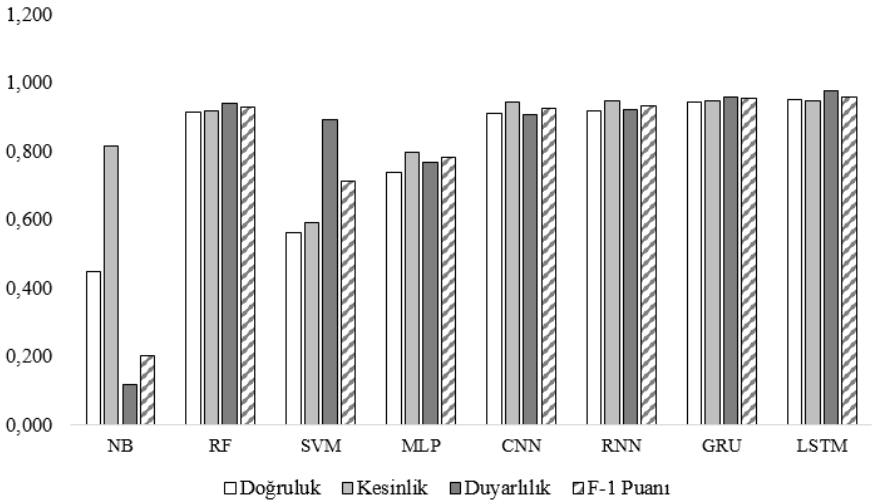
Kesinlik	Duyarlılık	Doğruluk	F-1 puanı
0,945	0,974	0,950	0,959

Tablo 26'da görüldüğü gibi LSTM için kesinlik değeri 0,945, duyarlılık değeri 0,974, doğruluk değeri 0,950 ve F-1 puanı ise 0,959'dur. Deneysel sonuçlar, LSTM'in kötücül yazılım tespitinde NB, RF, SVM, MLP, CNN, RNN ve GRU'dan daha başarılı olduğunu göstermiştir. LSTM'in karşılaştırılan diğer modellere göre başarılı bir sınıflandırma performansı göstermesi LSTM'in mimarisinin, RNN ve GRU'da da bulunan standart birimlere ek olarak özel birimler barındırmasıdır. LSTM birimleri, bilgileri uzun süreler boyunca bellekte tutabilen bir bellek hücresi içerir. Bilginin belleğe ne zaman girdiğini, ne zaman çıktığını ve ne zaman unutulduğunu kontrol etmek için bir dizi kapı kullanılır. Bu mimari, daha uzun vadeli bağımlılıkları öğrenmelerini sağlar. GRU'lar LSTM'lere benzer, ancak basitleştirilmiş bir yapı kullanır. Ayrıca bilgi akışını kontrol etmek için bir dizi kapı kullanırlar, ancak ayrı bellek hücreleri kullanmazlar ve daha az kapı kullanırlar.

Doğruluk, kesinlik, duyarlılık ve F-1 puanı değerlerine göre NB, RF, SVM, MLP, CNN, RNN, GRU ve LSTM için karşılaştırmalı deneysel sonuçlar Tablo 27 ve Şekil 3'te görülmektedir.

Tablo 27. Karşılaştırmalı deneysel sonuçlar (Comparative experimental results)

Model	NB	RF	SVM	MLP	CNN	RNN	GRU	LSTM
Doğruluk	0,446	0,912	0,562	0,738	0,911	0,918	0,943	0,950
Kesinlik	0,814	0,918	0,592	0,796	0,944	0,945	0,948	0,945
Duyarlılık	0,116	0,940	0,893	0,765	0,907	0,921	0,959	0,974
F-1 puanı	0,203	0,929	0,712	0,780	0,925	0,932	0,953	0,959



Şekil 3. Karşılaştırmalı deneysel sonuçlar (Comparative experimental results)

Tablo 27'de görüldüğü gibi LSTM, karşılaştırılan diğer modellere göre daha başarılı sonuçlara sahiptir. LSTM için doğruluk değeri 0,950, kesinlik değeri 0,945, duyarlılık değeri 0,974 ve F-1 puanı ise 0,959'dur. GRU, CNN, RNN ve RF, LSTM'den sonra en başarılı sonuçlara sahip modellerdir. GRU için doğruluk değeri 0,943, kesinlik değeri 0,948, duyarlılık değeri 0,959 ve F-1 puanı ise 0,9253'tür. RNN için doğruluk değeri 0,918, kesinlik değeri 0,945, duyarlılık değeri 0,921 ve F-1 puanı ise 0,932'dir. CNN için doğruluk değeri 0,911, kesinlik değeri 0,944, duyarlılık değeri 0,907 ve F-1 puanı ise 0,925'tir. RF için doğruluk değeri 0,912, kesinlik değeri 0,918, duyarlılık değeri 0,940 ve F-1 puanı ise 0,929'dur.

Tablo 27 ve Şekil 3'te görüldüğü gibi LSTM, diğer modellere kıyasla kötücül yazılım tespitinde daha başarılı bir sınıflandırma performansı göstermiştir. GRU, RNN, CNN ve RF, LSTM'den sonra en başarılı sonuçlara sahip modellerdir. NB ise kötücül yazılım tespitinde başarısız bir sınıflandırma performansı göstermiştir.

4. SONUÇLAR (CONCLUSIONS)

Çoklu bağlantı ve sensör gibi özelliklerle donatılmış mobil cihazlar günlük yaşamın giderek her alanında daha yaygın bir şekilde kullanılmaktadır. Mobil cihazların yaygınlaşması ve bu cihazlarda yapılan hassas kişisel işlemler ile birlikte mobil kötücül yazılımların sayısı da artmıştır. Mobil cihazlarda, bilgisayar ortamından farklı olarak, kötücül uygulamaların mobil cihazlara bulaşmasını ve yayılmasını önlemek için kullanılacak çözümlerde birçok faktör göz önünde bulundurulur. Sınırlı kaynaklar, güç ve işlem birimleri gibi çok sayıda mobil cihaz özelliği saldırganlar tarafından kötüye kullanılabilir.

Statik analizler uygulamaların durağan halinde yapılan ve uygulamanın yürütülmesini gerektirmeyen, uygulamaların çalışma esnasındaki davranışlarını incelemeyen bir analiz türüdür. Bu çalışmada ise uygulamanın yürütülmesi esnasındaki davranışlarının incelendiği dinamik bir analiz gerçekleştirilmiştir. Wireshark programı vasıtasıyla elde edilen bir veriseti üzerinde uygulamaların çalışma zamanındaki ağ trafikleri incelenmiştir. Kötücül ve iyicil oldukları bilinen uygulamalar çalıştırılırken elde edilen ağ trafiği verileri kullanılarak kötücül yazılımların tespit edilmesi amaçlanmıştır. Uygulamaların gönderdiği ve aldığı paket bilgileri, iletişim kurmaya çalıştıkları IP adresi bilgileri, dış kaynaklara gönderilen ve alınan paket sayısı, gönderilen ve alınan veri hacmi gibi bilgiler kullanılarak kötücül yazılım tespiti yapılmıştır.

Bu amaçla, derin öğrenme tabanlı bir kötücül yazılım tespit sistemi geliştirilmiştir. Geliştirilen LSTM tabanlı model, NB, RF, SVM, MLP, CNN, RNN ve GRU ile uygulamalı olarak karşılaştırılmıştır. Uygulanan modeller doğruluk, kesinlik, duyarlılık ve F-1 puanı metrikleri kullanılarak karşılaştırmalı olarak analiz edilmiştir. Deneysel sonuçlar LSTM, GRU, RNN, CNN, ML SVM, RF ve NB'nin doğruluk değerlerinin sırasıyla 0,950, 0,943, 0,918, 0,911, 0,738, 0,562, 0,912 ve 0,446 olduğunu göstermiştir. Geliştirilen LSTM tabanlı modelin, diğer modellere kıyasla doğruluk, kesinlik, duyarlılık ve F-1 puanı metriklerinin her biri için daha başarılı sonuçlar elde ettiği görülmüştür. Bu çalışmada, bilinen uygulamalar ve bilinen bir veriseti

üzerinde çalışılmıştır. Geliştirilen sistem mobil cihazlara bir uygulama olarak entegre edildiğinde anlık olarak uygulamaların ağ trafik durumları izlenecek ve öğrenmiş olan sistem yardımıyla kötücül yazılımlar tespit edilebilecektir. Gelecek çalışmalarda geliştirilen sistemin mobil cihazlara bir uygulama olarak entegre edilmesi ve anlık olarak uygulamaların ağ trafik durumlarının izlenerek öğrenmiş olan sistem yardımıyla kötücül yazılımların tespit edilmesi hedeflenmektedir.

5. KAYNAKLAR (REFERENCES)

1. Wang S., Chen Z., Yan Q., Yang B., Peng L., Jia Z., A mobile malware detection method using behavior features in network traffic, Journal of Network and Computer Applications, 133, 15-25, 2019.
2. Gezici B., Tarhan A., Chouseinoglou O., Complexity, size and internal quality in the evolution of mobile applications: An exploratory study, Journal of the Faculty of Engineering and Architecture of Gazi University, 34 (3), 1483-1500, 2018.
3. Andrade E.D.O., Viterbo J., Vasconcelos C.N., Guérin J., Bernardini F.C., A model based on lstm neural networks to identify five different types of malware, Procedia Computer Science, 159, 182-191, 2019.
4. Grønli T.M., Hansen J., Ghinea G., Younas M., Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS, In 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, Victoria, Kanada, 635-641, 13-16 Mayıs 2014.
5. Feng P., Ma J., Sun C., Xu X., Ma Y., A novel dynamic Android malware detection system with ensemble learning. IEEE Access, 6, 30996-31011, 2018.
6. Sarwar M., Soomro T.R., Impact of smartphone's on society, European journal of scientific research, 98 (2), 216-226, 2013.
7. Siau K., Lim E. P., Shen Z., Mobile commerce: Promises, challenges and research agenda, Journal of Database Management (JDM), 12 (3), 4-13, 2001.
8. Yesilyurt M., Yalman Y., Security threats on mobile devices and their effects: estimations for the future, International Journal of Security and Its Applications, 10 (2), 13-26, 2016.
9. Sheikh A.A., Ganai P.T., Malik N.A., Dar K.A., Smartphone: Android Vs iOS. The SIJ Transactions on Computer Science Engineering & its Applications (CSEA), 1 (4), 141-148, 2013.
10. Statista. Android işletim sisteminin 2018-2021 yılları arasında dünya çapındaki pazar payları. https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/. Yayın tarihi Haziran 2021. Erişim tarihi Ağustos 31, 2021.
11. Kabakus A.T., Dogru I.A., An in-depth analysis of Android malware using hybrid techniques, Digital Investigation, 24, 25-33, 2018.
12. Wu D.J., Mao C.H., Wei T.E., Lee H.M., Wu K.P., Droidmat: Android malware detection through manifest and api calls tracing, In 2012 Seventh Asia Joint Conference on Information Security, Tokyo, Japonya, 62-69, 9-10 Ağustos, 2012.
13. Kapratwar A., Di Troia F., Stamp M., Static and dynamic analysis of android malware, In ICISSP, Porto, Portekiz, 653-662, 19-21 Şubat, 2017.
14. Arora A., Garg S., Peddoju S.K., Malware detection using network traffic analysis in android based mobile devices, In 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, Oxford, İngiltere, 66-71, 10-12 Eylül, 2014.
15. Malik J., Kaushal R., CREDROID: Android malware detection by network traffic analysis, In Proceedings of the 1st acm workshop on privacy-aware mobile computing, New York, ABD, 28-36, Temmuz, 2016.
16. Wang S., Chen Z., Zhang L., Yan Q., Yang B., Peng L., Jia Z., Trafficav: An effective and explainable detection of mobile malware behavior using network traffic, In 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), Beijing, Çin, 1-6, 20-21 Haziran, 2016.
17. Milosevic N., Dehghantanha A., Choo K.K.R., Machine learning aided Android malware classification, Computers & Electrical Engineering, 61, 266-274, 2017.
18. Karbab E. B., Debbabi M., Derhab A., Mouheb D., MalDozer: Automatic framework for android malware detection using deep learning, Digital Investigation, 24, 48-59, 2018.
19. Mehtab A., Shahid W.B., Yaqoob T., Amjad M.F., Abbas H., Afzal H., Saqib M.N., AdDroid: rule-based machine learning framework for android malware analysis, Mobile Networks and Applications, 25 (1), 180-192, 2020.
20. Alzaylaee M.K., Yerima S.Y., Sezer S., DL-Droid: Deep learning based android malware detection using real devices, Computers & Security, 89, 101663, 2020.
21. Lu T., Du Y., Ouyang L., Chen Q., Wang X., Android malware detection based on a hybrid deep learning model, Security and Communication Networks, 2020.
22. Pektaş A., Acarman T., Deep learning for effective Android malware detection using API call graph embeddings, Soft Computing, 24 (2), 1027-1043, 2020.
23. Karbab E.B., Debbabi M., PetaDroid: Adaptive Android Malware Detection Using Deep Learning, In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 319-340, 2021.
24. Bakour K., Ünver H. M., DeepVisDroid: android malware detection by hybridizing image-based features with deep learning techniques, Neural Computing and Applications, 1-18, 2021.
25. Jang-Jaccard J., Nepal S., A survey of emerging threats in cybersecurity, Journal of Computer and System Sciences, 80 (5), 973-993, 2014.
26. Or-Meir O., Nissim N., Elovici Y., Rokach L., Dynamic malware analysis in the modern era—A state of the art survey, ACM Computing Surveys (CSUR), 52 (5), 1-48, 2019.

27. Rabbani M., Wang Y. L., Khoshkangini R., Jelodar H., Zhao R., Hu P., A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing, Journal of Network and Computer Applications, 151, 102507, 2020.

28. Seo S.H., Gupta A., Sallam A.M., Bertino E., Yim K., Detecting mobile malware threats to homeland security through static analysis, Journal of Network and Computer Applications, 38, 43-53, 2014.

29. Utku A, Doğru İ.A., Permission based detection system for android malware, Journal of the Faculty of Engineering and Architecture of Gazi University, 32 (4), 1015-1024, 2017.

30. Li Q., Li X., Android malware detection based on static analysis of characteristic tree, In 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Xi'an, Çin, 84-91, 17-19 Eylül, 2015.

31. Dhaya R., Poongodi M., Detecting software vulnerabilities in android using static analysis, In 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, Hindistan, 915-918, 8-10 Mayıs 2014.

32. Grégio A.R., Fernandes Filho D.S., Afonso V.M., Santos R.D., Jino M., de Geus P.L., Behavioral analysis of malicious code through network traffic and system call monitoring, In Evolutionary and Bio-Inspired Computation: Theory and Applications, 2011.

33. Galib, A. H., Hossain, B. M., A Systematic Review on Hybrid Analysis using Machine Learning for Android Malware Detection, In 2019 2nd International Conference on Innovation in Engineering and Technology (ICIET), Dhaka, Bangladesh, 1-6, 23-24 Aralık 2019.

34. Kaggle. Android ağ trafiği veriseti. https://www.kaggle.com/xwolf12/network-traffic-android-malware. Yayın tarihi 2019. Erişim tarihi Şubat 1, 2021.