Metamorfik Zararlı Yazılımların Derin Öğrenme İle Sınıflandırılması

Bu tez Bilgi Güvenliği Mühendisliği'nde
Tezli Yüksek Lisans Programının bir koşulu olarak

Ahmet Faruk Yazı
tarafından

Fen Bilimleri Enstitüsü'ne
sunulmuştur.

İSTANBUL
ŞEHİR
ÜNİVERSİTESİ

Bu tezi okuduk, kapsam ve nitelik açısından Bilgi Güvenliği Mühendisliği alanında Yüksek Lisans derecesi için tümüyle uygun olduğu görüşüne vardık.

ONAYLAYANLAR:

Prof. Dr. Ensar Gül
(Tez Danışmanı) .....

Doç. Dr. Ferhat Özgür Çatak
(Tez Eş-danışmanı) .....

Doç. Dr. Ali Fuat Alkaya .....

Dr. Öğretim Üyesi Mehmet Serkan Apaydın .....

Bu tez İstanbul Şehir Üniversitesi, Fen Bilimleri Enstitüsü tarafından belirlenen tüm koşullara uygundur.

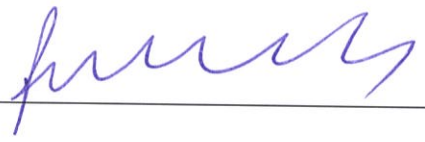
ONAY TARİHİ: 26.08.2019

MÜHÜR/İMZA:

Yazarlık Beyanı

Ben, Ahmet Faruk YAZI, başlığı, 'Metamorfik Zararlı Yazılımların Derin Öğrenme İle Sınıflandırılması' olan tezin ve içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Esaslı yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

İmza: 

Tarih: 26.08.2019

"Yapay zeka alanında tekelleşen dünyayı yönetir."

Vladimir Putin

Metamorfik Zararlı Yazılımların Derin Öğrenme İle Sınıflandırılması

Ahmet Faruk Yazı

Öz

Her geçen gün artan ve yaygınlaşan internet hizmetleri ve bu hizmetleri kullanan kişiler, kötü amaçlar içeren birçok saldırıya maruz kalmaktadır. Bu saldırılar çok farklı amaçlar içermekle birlikte, genel olarak zararlı yazılımlar kullanılarak yapılmaktadır. Bu nedenlede farklı mimari ve özelliklerde çok fazla zararlı yazılımlar üretilmektedir. Yapılan incelemlerde, bu zararlı yazılımların yetkinlik ve etkinliklerinin çok ciddi bir seviyeye ulaştıkları görülmektedir. Bu doğrultuda, zararlı yazılım ailesinin en gelişmiş üyesi olarak metamorfik zararlı yazılımlar karşımıza çıkmaktadır.

Metamorfik zararlı yazılımlar, geleneksel imza tabanlı tespit yöntemleri kullanan antivirus uygulamaları tarafından tespit edilememektedirler. Bu durumun bir sonucu olarak, bu yazılımların türlerine göre sınıflandırılması da pek mümkün olmamaktadır. Bu doğrultuda, son zamanlarda yapılan tespit ve sınıflandırmaya yönelik çalışmaların hemen hemen hepsi zararlı yazılımların davranışlarını ele almaktadır. Bu çalışma kapsamında da zararlı yazılımların davranışları göz önünde bulundurularak zararlı yazılım türlerine göre bir sınıflandırma yöntemi geliştirme amaçlanmıştır.

Öncelikle, çalışmamızda zararlı yazılımların davranışlarını temsil eden Windows işletim sistemi üzerinde yapmış oldukları API çağrılarını içeren bir veri kümesi oluşturulmuştur. Veri kümesi, *Adware*, *Backdoor*, *Downloader*, *Dropper*, *Spyware*, *Trojan*, *Virus* ve *Worm* türlerndeki gerçek zararlı yazılımların davranışlarını içermektedir.

Sınıflandırma yöntemi olarak, zamana göre sıralı gelen verilerin işlenmesinde başarılı olan ve yaygın bir şekilde kullanılan, derin öğrenme yöntemlerinden uzun-kısa süreli bellek (long-short term memory - LSTM) yöntemi kullanılmıştır. Bu sayede 7 farklı türdeki zararlı yazılımların, türlerine ait davranışları modellenerek, bir sınıflandırma yöntemi geliştirilmiştir.

Anahtar Sözcükler: Metamorfik Zararlı Yazılımlar, Windows API, Derin Öğrenme, LSTM.

Classification of Metamorphic Malware With Deep Learning

Ahmet Faruk YAZI

Abstract

Increasing and widespread internet services and the users of these services are exposed to many malicious attacks. Although these attacks have many different purposes, they are generally done using malicious software. For this reason, too many malware are produced with different architecture and features. In the examinations conducted, it is seen that the competencies and activities of these malware have reached a very serious level. In this respect, metamorphic malware is the most advanced member of malware family. Metamorphic malware cannot be detected by anti-virus applications using traditional signature-based detection methods. As a result of this situation, it is not possible to classify these software according to their types. In this respect, almost all of the recent detection and classification studies address the behavior of malware. In this study, it is aimed to develop a classification method according to malware types by considering malware behavior. First of all, in our study, a dataset was created containing API calls made on the Windows operating system, which represents the behavior of malicious software. The data set contains the behavior of real malware such as Adware, Backdoor, Downloader, Dropper, Spyware, Trojan, Virus and Worm. Long-term term memory (LSTM), which is a widely used and deep learning method, is used as the classification method. In this way, a classification method has been developed by modeling the behaviors of 7 different types of malware.

Keywords: Metamorfic Malware, Windows API, Deep Learning, LSTM

Teşekkür

Bu çalışmanın bütün aşamalarında her türlü desteğini esirgemeyen, bilgi ve deneyimi ile beni yönlendiren değerli hocam, eş-danışmanım Doç. Dr. Ferhat Özgür ÇATAK' a teşekkür ederim.

Ayrıca önemli yönlendirmeleri ve fikirleri ile bu çalışmaya destek veren değerli danışmanm Prof. Dr. Ensar GÜL hocama teşekkür ederim.

Gerek ders aşamasında, gerekse tez aşamasında bana sabır gösteren ve beni destekleyen eşim, kızım ve oğluma teşekkür ederim.

İçindekiler

Şekil Listesi

Tablo Listesi

Kısaltmalar

API	**A**pplication **P**rogramming **I**nterface
DT	**D**ecision **T**ree
GPU	**G**raphics **P**rocessing **U**nit
JSON	**J**ava**S**cript **O**bject **N**otation
kNN	**k**-**N**earest **N**eighborhood
LSTM	**L**ong **S**hort **T**erm **M**emory
PE	**P**ortable **E**xecutable
RF	**R**andom **F**orest
RNN	**R**ecurrent **N**eural **N**etwork
SVM	**S**upport **V**ector **M**achine
URL	**U**niform **R**esource **L**ocator

Bölüm 1

Giriş

Bir yazılımın, istenmeyen bir şekilde(yasadışı bir şekilde) fayda elde etmek için tasarlanmış olması, bu yazılımın zararlı bir yazılım olduğu anlamına gelmektedir. Bu yazılımlar belirli bir amaç doğrultusunda hareket etmektedirler. Günlük hayatımızda, bir sistemin çalışmasını engellemek, bir sistemde izinsiz olarak erişim hakkı kazanmak, kişisel verileri elde etmek gibi çok farklı amaçlar için kullanıldıklarını bilmekteyiz. Bu gibi amaçlar doğrultusunda, sunucular, kişisel bilgisayarlar, mobil telefonlar, kameralar gibi çok farklı platformlar hedef alınmaktadır. Her geçen gün hedef haline gelen platform sayısı artmaktadır. Bu platformlar için geliştirilen zararlı yazılımlar da çeşitlenmektedir.

Son yıllarda, zararlı yazılımların çok fazla artmasında ve yaygınlaşmasında ki en büyük faktör, büyük bir kitle tarafından çalışılan bir konu olmasıdır. Bu doğrultuda gerek bireysel çalışmalar, gerekse yetkin organizasyonel çalışmalar zararlı yazılımların sürekli gelişmesini sağlamaktadır. Özellikle 2018 yılı ilk 4 ayında, 40,000,000 adet [1] yeni zararlı yazılımın gün yüzüne çıktığı düşünülürse, bu doğrultuda çok büyük emekler sarfedildiği anlaşılmaktadır. Bu nedenle, zararlı yazılımların saldırılarından korunmak için de büyük çabalar gösterilmesi gerekmektedir.

Zararlı yazılımlarından etkili bir şekilde korunabilmek için, öncelikle zararlı yazılımların iyi tanınması ve davranışlarının iyi analiz edilmesi gerekmektedir. Bu doğrultuda sadece zararlı yazılımların tespit edilmesi yeterli olmamaktadır. Bu yazılımların başarı bir şekilde sınıflandırılması da gerekmektedir. Zararlı yazılımların sınıflandırılması, önemli bir konudur. Bir zararlı yazılımın ait olduğu sınıf, zararlı bir davranışı temsil etmektedir. Bu

davranışlara karşı alınacak önlemler zararlı yazılımların sınıflarına göre farklılık gösterebilmektedir.

Günümüzde kullanılan zararlı yazılımların türlerine göre sınıflandırma yöntemleri çok başarılı sonuçlar verememektedir. Genel olarak bu yöntemler, zararlı yazılım aileleri göz önünde bulundurularak yapılmaktadır [2]. Kullanılan sınıflandırma yöntemleri ile bir zararlı yazılımı, bir anti-virus uygulaması Trojan olarak tespit ederken, başka bir uygulama Worm olarak tespit edebilmektedir [3]. Bu durum, gelişmiş zararlı yazılımların ortaya çıkması ile daha da karmaşık bir hal almaya başlamıştır.

Zararlı yazılımların gelişimleri incelendiğinde, zararlı yazılımların yetkinlik ve etkinliklerinin çok ciddi bir seviyeye ulaştıkları görülmektedir. Bu doğrultuda, zararlı yazılım ailesinin en gelişmiş üyesi olarak metamorfik zararlı yazılımlar karşımıza çıkmaktadır. Bu yazılımlar, güvenlik bileşenlerinden kendilerini, farklı yöntemler kullanarak, farklı zamanlarda kendi kodlarında yaptıkları değişiklikler ile saklayabilmektedirler. Bu sayede kod imzaları sürekli değişmektedir [4]. Bu nedenle, geleneksel imza tabanlı tespit yöntemleri kullanan anti-virus uygulamaları tarafından tespit edilememektedirler. Bu durumun bir sonucu olarak, metamorfik zararlı yazılımların türlerine göre sınıflandırılması da pek mümkün olmamaktadır.

Bir metamorfik zararlı yazılım, farklı ortamlarda farklı kod dizilimleri ile kendini gösterebilmesine rağmen, bütün ortamlarda aynı davranışı göstermek zorundadır. Çünkü belirli bir zararlı eylemi gerçekleştirmek için oluşturulmuştur. Bu bilgi doğrultusunda, metamorfik zararlı yazılımların tespiti ve sınıflandırılması için kullanılan yöntemlerin neredeyse tamamı, zararlı yazılımın yapısal özelliklerini değil, davranışsal özelliklerini ele almaktadırlar. Bu yöntemler ile, genellikle zararlı yazılımların davranışlarını temsil eden, sistem üzerinde yapmış oldukları Windows API çağrıları gibi veriler önemli derecede kullanılmaktadır.

Herhangi bir yazılım tarafından, bir eylemi gerçekleştirmek için yapılan sistem API çağrılarının tamamı, bu yazılımın genel davranışını göstermektedir. Bu gibi davranışların doğru analiz edilmesi ile, davranışa sahip olan yazılımın zararlı bir yazılım olduğu, hangi sınıfa ait bir zararlı eyleme sahip olduğu bilgileri elde edilebilmektedir.

Zararlı yazılımların yapmış oldukları her bir API çağrısı bir veri niteliğinde olduğu gibi bu çağrıların yapılış sırası da önemli olmaktadır. Belirli API çağrılarının belirli sırada

yapılması bir davranışı temsil etmektedir. Bu gibi zamana göre sıralı gelen verilerin işlenmesinde derin öğrenme yöntemlerinden uzun-kısa süreli bellek (long-short term memory - LSTM) oldukça yaygın kullanılmaya başlanmıştır.

Biz de çalışmamızı, API çağrılarının analizi üzerine kurguladık. Amacımız, farklı türlerdeki zararlı yazılımların sistem üzerinde yaptıkları API çağrılarını analiz ederek, zararlı yazılım türüne göre API çağrı kümesi oluşturmak ve bu verileri kullanarak, zararlı yazılım türlerine ait davranışları tespit edebilecek bir yöntem geliştirmektir. Bu yöntemi de, LSTM algoritması kullanarak sınıflandırma modelleri oluşturarak yapmaktır. Bu sayede metamorfik zararlı yazılım gibi yapısal değişikliğe uğrayan fakat sistem üzerinde bir zararlı yazılım türüne ait davranış sergileyen yazılımları sınıflandırmaktır.

Zararlı yazılımların sınıflandırılmasından kastedilen, sadece zararlı yazılımların türlerine göre sınıflandırılmasıdır. Herhangi bir zararlı yazılım tespiti işlemi amaçlanmamaktadır. Bu nedenle oluşturmak istenen veri kümesinde sadece zararlı olduğu bilinen yazılımların API çağrıları bulunacaktır.

Yukarıda bahsedilen amaç doğrultusunda yapmış olduğumuz çalışmanın en önemli katkıları şu şekildedir:

- Zararlı yazılımlar ile ilgili yeni bir veri kümesi oluşturulmuştur. Bu alanda bu kapsamda bir veri kümesi bulunmamaktadır.
- Metamorfik yazılımlar yalıtılmış kumhavuzu ortamlarında çalıştırılarak analiz edilmiş ve API sıralamaları kayıt altına alınmıştır.
- Metin sınıflandırma için kullanılan LSTM algoritmasıyla bir metin sınıflandırma problemi olarak modellenmiş ve zararlı yazılım türü tespit modelleri oluşturulmuştur.
- Çok rastlanan zararlı yazılım türleri olan *Adware*, *Backdoor*, *Downloader*, *Dropper*, *Spyware*, *Trojan*, *Virus* ve *Worm* türleri kullanılmıştır.

Çalışmamızın bir çıktısı olarak hazırlanmış olan bu dokümanda; 2. bölümde, konu ile ilgili daha önceden yapılmış çalışmalara yer verilmektedir. 3. bölümde, yapmış olduğumuz çalışma kapsamında kullanılan kavramlar ile ilgili genel bilgiler verilmiştir. 4. bölümde, veri kümesi oluşturulması ve çalışmamızda uyguladığımız model oluşturma yöntemi ile

ilgili bilgi verilmiştir. 5. bölümde ise, yapmış olduğumuz analizler sonucu elde ettiğimiz deneyimler paylaşılmıştır. Son olarak 6. bölümde ise, çalışmanın sonucuna yer verilmiştir.

Bölüm 2

İlgili Çalışmalar

Zararlı yazılımların tespiti ve sınıflandırılması ile ilgili çalışmalar:

- **Classification and Detection of Metamorphic Malware using Value Set Analysis:**
  Bu çalışmadan metamorfik zararlı yazılımların yapısal değişiklikleri göz önünde bulundurularak geçirmiş oldukları değişimler incelenmiştir. VSA(Value Set Analysis) yöntemi kullanılarak, zararlı yazılımlardaki değişmeyen kod yapısı çıkarılarak tespit işlemleri yapılmıştır. Bu çalışma ile metamorfik zararlı yazılımların örneklemelerinde davranış değişikliği olmaması durumunda, %100 doğruluk ile tespit edilebildiği sonucu çıkarılmıştır [2].

- **MEDUSA: MEtamorphic malware Dynamic analysis Using Signature from API:**
  Bu çalışmada, metamorfik zararlı yazılımların Windows API çağrı dizilerini kullanılarak, zararlı yazılım imzaları oluşturulmuştur. Bu imzalar kullanılarak zararlı yazılım ailelerinin tespitini ve sınıflandırılmasını sağlayan bir yöntem önerilmiştir. Bu çalışmada VX Heavens uygulaması ile, NGVCK, MPCGEN, G2 ve IL_SMG ailelerinin her birinden 80 adet zararlı yazılım oluşturulmuştur. Bu yazılımların emulator yardımıyla Windows API çağrı dizileri elde edilmiştir. Bu veriler kullanılarak veri kümesi oluşturulmuştur. Önerilen yöntem ile elde edilen doğruluk oranları, her bir aile için, %75, %80, %80 ve %75' dir [5].

- **Analyzing Malware by Abstracting the Frequent Itemsets in API call Sequences:**

  Bu çalışmada zararlı yazılımların davranışları göz önünde bulundurularak bir tespit yöntemi geliştirilmiştir. Zararlı yazılımların Windows işletim sistemi üzerinde yapmış oldukları API çağrı dizileri Cuckoo Sandbox uygulaması yardımıyla elde edilmiştir. Bu çağrı dizilerindeki sıkça kullanılan öğeleri çıkarılarak bir analiz yöntemi geliştirilmiştir. Analiz verisi için 3131 tane zararlı yazılım kullanılmıştır. Yapılan deneyler sonucunda %94.7 doğruluk oranı elde edilmiştir [6].

- **An Information Retrieval Approach For Malware Classification Based on Windows API Calls:**

  Bu çalışma zararlı yazılımların davranışlarını çıkarma üzerine, bilgi alma teorisi (Information Retrieval Theory) kullanarak, geliştirilmiş bir sınıflandırma yöntemi içermektedir. Bal küplerinden(honeypots) elde edilen zararlı yazılımların, Cuckoo Sandbox uygulaması yardımıyla Windows API çağrı dizileri elde edilerek, zararlı davranışları temsil eden belgeler hazırlanmıştır. Bu belgeler TF-IDF ağırlık ve benzerlik ölçümleme yöntemi kullanılarak analiz edilmiş ve zararlı yazılımların benzerlik özellikleri çıkarılmıştır. Bu özellikler sayesinde sınıflandırma yapılmıştır. 0.977 doğruluk oranı elde edilmiştir [7].

- **DaCoMM: Detection and Classification of Metamorphic Malware:**

  Bu çalışmada sınıflandırma için toplam 600 zararlı yazılım kullanılmıştır. Zararlı yazılımların 268 tanesi VX Heaven kullanılarak oluşturulmuştur. Bu zararlı yazılımların kontrol akış ve API istek grafikleri çıkarılmıştır. Gerekli çağrı özelliklerini çıkarmak için Gourmand Feature Selection algoritması kullanılmıştır. Veriler Weka aracı kullanılarak, zararlı yazılım sınıflandırması yapılmıştır. Sınıflandırma zararlı yazılım ailelerine göre histogram ve Chi-square fark ölçüm formülleri kullanılarak yapılmıştır. Sınıflandırma farklı aileler için %89.00 ve %99.10 doğruluk değerleri arasındadır [8].

- **Analysis of Malware Behavior: Type Classification using Machine Learning:**

  Bu çalışmada gözetimli öğrenme (Supervised Machine Learning) teknolojisi kullanılarak, bir sınıflandırma yöntemi geliştirilmiştir. Sınıflandırma için Random Forests algoritması kullanılmıştır. Toplam 42000 zararlı yazılım davranışları Cuckoo

Sandbox uygulaması yardımıyla elde edilmiştir. Windows API çağrıları temel sınıflandırma verileri olacak şekilde, DNS istekleri, Accessed Files, Mutexes, Registry Keys verileri kullanılarak analiz raporları elde edilmiştir. Ayrıca zararlı yazılımların sınıf etiketleri için, VirusTotal servisinin sağladığı sonuçlar üzerinden Avast uygulamasının tespit ettiği etiketler kullanılmıştır. Sınıflandırma için Trojan, Potentially Unwanted Program, Adware ve Rootkit sınıfları göz önünde bulundurularak veri kümesi oluşturulmuştur. Önerilen yöntem ile 0.98'lik bir ağırlıklı ortalama AUC değeri elde edilmiştir [9].

- **A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining:**
  Bu çalışmada 91 tanesi zararlı yazılım olmak üzere toplam 188 PE dosyası kullanılmıştır. Bu dosyaların çalıştırılması sonucunda Process monitor uygulaması yardımıyla bir rapor elde edilmiştir. Bu rapordan gerekli özellikler çıkarılmış ve SVM yöntemi kullanılarak sınıflandırıcılar belirlenerek sınıflandırma işlemi yapılmıştır. Bu çalışmada %97.8 doğruluk oranı yakalanmıştır [10].

- **A Compression-Based Technique to Classify Metamorphic Malware:**
  Bu çalışmada metamorfik zararlı yazılım ailelerini belirlemek için sıkıştırma temelli bir sınıflandırma tekniği(Compression-Based Text Classification) kullanılmıştır. 13 farklı, gerçek zararlı yazılım ailesinden oluşan, yazılımlarının binary dosyaları kullanılarak bir veri kümesi oluşturulmuştur. Sınıflandırma için, bir zararlı yazılım dosyası ile farklı ailelere ait veri kümesindeki benzerlikler kullanılmıştır [11].

- **Malware Classification with LSTM and GRU Language Models And Character-Level CNN:**
  Bu çalışma üç farklı zararlı yazılım sınıflandırma yöntemi sunmaktadır. LSTM ve GRU dil modelleri kullanılarak oluşturulan yöntemler ve Character-Level CNN üzerine kurulmuş tek seviyeli bir sınıflandırma yöntemidir. Yöntemler oluşturulurken, zararlı ve zararsız olmak üzere, 75,000 adet PE dosya kullanılmıştır. Bu dosyalar çalıştırılarak, Microsoft Anti-Malware Engine uygulaması yardımıyla, her biri için API çağrı dizileri elde edilmiştir. Bu çağrı dizileri sınıflandırma veri kümesini oluşturmaktadır. Çalışmada, her bir yöntem için FPR değerinin %2 olması durumunda TPR değerinin %80 civarında olduğu yapılan deneyim sonuçlarında gösterilmiştir [12].

- **Deep Learning LSTM based Ransomware Detection:**
  Bu çalışmadan derin öğrenme yöntemlerinden LSTM modeli kullanılmıştır. Ransomware zararlı yazılım sınıfına özel bir tespit modeli oluşturulmuştur. Çalışma için 157 adet ransomware ve zararsız PE dosyaları kullanılmıştır. Sandbox uygulaması kullanılarak bu dosyalar çalıştırılarak API çağrı dizinleri elde edilmiştir. Bu diziler filtrelenerek model oluşturmak için kullanılan veri kümesi oluşturulmuştur. Elde edilen sonuçlarda doğruluk oranı %96.67' dır [13].

- **Android malware detection based on system call sequences and LSTM:**
  Bu çalışma ile mobil cihazlar üzerinde çalışan zararlı yazılımların tespitini sağlayan bir yöntem sunulmaktadır. Bu çalışmada LSTM dil modelleme yöntemi kullanılmıştır. Zararlı yazılımlar ve güvenilir yazılımlar için iki ayrı model oluşturulmuştur. 3567 adet zararlı yazılım ve 3567 adet güvenli yazılım kullanılmıştır. Bu yazılımların mobil cihaz üzerinde yaptığı sistem çağrı dizinleri elde edilerek veri kümesi oluşturulmuştur. Yapılan çalışmada %93.7 doğruluk oranı elde edilmiştir [14].

- **Framework for Detecting Metamorphic Malware Based on Opcode Feature Extraction:**
  Bu çalışmada, metamorfik zararlı yazılımların yapısal olarak değişikliğe uğrasalarda, temel algoritma yapılarının korunduğu gerçeğine dayandırmışlardır. Bu doğrultuda, yazılım kodları incelenerek, davranışları temsil eden zararlı yazılımların opcode özellikleri çıkarılmıştır. Bu veriler, makina öğrenmesi metodları kullanılarak bir yöntem geliştirilmiştir [15].

- **Research on Malicious JavaScript Detection Technology Based on LSTM:**
  Bu çalışma ile zararlı Javascript kodlarının tespit edilebilmesi için LSTM tabanlı bir yöntem geliştirmişlerdir. Bu yöntemde zararlı Javascript kodlarının, bytecode ve kelime vektörü(word vector) özellikleri çıkarılarak bir veri kümesi oluşturulmuştur. Ayıca bu çalışma içerisinde Random Forest and SVM algoritmalar kullanılarak da farklı farklı yöntemler geliştirilmiştir. LSTM tabanlı tespit yönteminin diğer yöntemlerinden daha başarılı olduğu gösterilmiştir [16].

- **MEDUSA: Malware Detection Using Statistical Analysis of System's Behavior:**
  Bu çalışma ise bu alanda farklı bir yaklaşım sunmaktadır. Yaklaşımın temel

mantığı; zararlı yazılımların tespit edilmesinde kullanılacak olan asıl veri, zararlı yazılımların nitelikleri değil, sistem üzerinde yaptıkları etkileridir. Bu nedenle önerdikleri tespit yönteminde, zararlı yazılımların yapısal ve davranışsal özellikleri ele alınmamaktadır. Olağan bir durumda ki sistem üzerinde, anormal olan davranışlar tespit edilmek istenmektedir. Bu sayede, polimorfik, metamorfik gibi gelişmiş zararlı yazılımların, sıfır gün(zero-day) zararlı yazılımların tespitinin yapılabileceği iddia edilmiştir [17].

Bölüm 3

Ön Bilgiler

Bu çalışma kapsamında genel olarak, veri kümesi oluşturmak için Windows API Çağrıları, Cuckoo Sandbox uygulaması ve VirusTotal servisi, sınıflandırma modeli oluşturmak için ise LSTM derin öğrenme yöntemi kullanılmıştır.

3.1 Windows API Çağrıları

Windows API, Windows işletim sistemi üzerinde uygulama geliştirmek için uygulama geliştiricilere sunulan bir arayüzdür. Uygulama geliştiricileri Windows API'leri kullanarak işletim sistemi ile iletişime geçebilirler. Bu nedenle işletim sistemi bir çok hizmetini API olarak sunmaktadır [18].

Windows işletim sisteminde çalışmak üzere geliştirilen bir uygulama, işletim sisteminin sunduğu bir işlevi kullanabilmesi için API olarak sunulan fonksiyonları kullanması gerekmektedir. Bu fonksiyonların kullanılmasına API çağrısı denilmektedir. Bir uygulama çalıştırıldığı süre içerisinde, gerçekleştirmek istediği eylem ile ilgili birçok kez API çağrısı yapmaktadır. Örneğin bir uygulama bir dosya oluşturmak istendiğinde CreateFileA Windows API çağrısı yapması gerekmektedir [19]. Bir uygulamanın sistem üzerinde yapmış olduğu bütün API çağrıları, bu uygulamanın gemel davranışını gösterebilmektedir. Bu nedenle dinamik analizlerde çoğu zaman API çağrıları sıklıkla kullanılmaktadır.

Bu çalışmada kullanılan veri kümesinin temel girdileri, zararlı yazılmların işletim sistemi üzerinde yapmış oldukları Windows API çağrılarıdır.

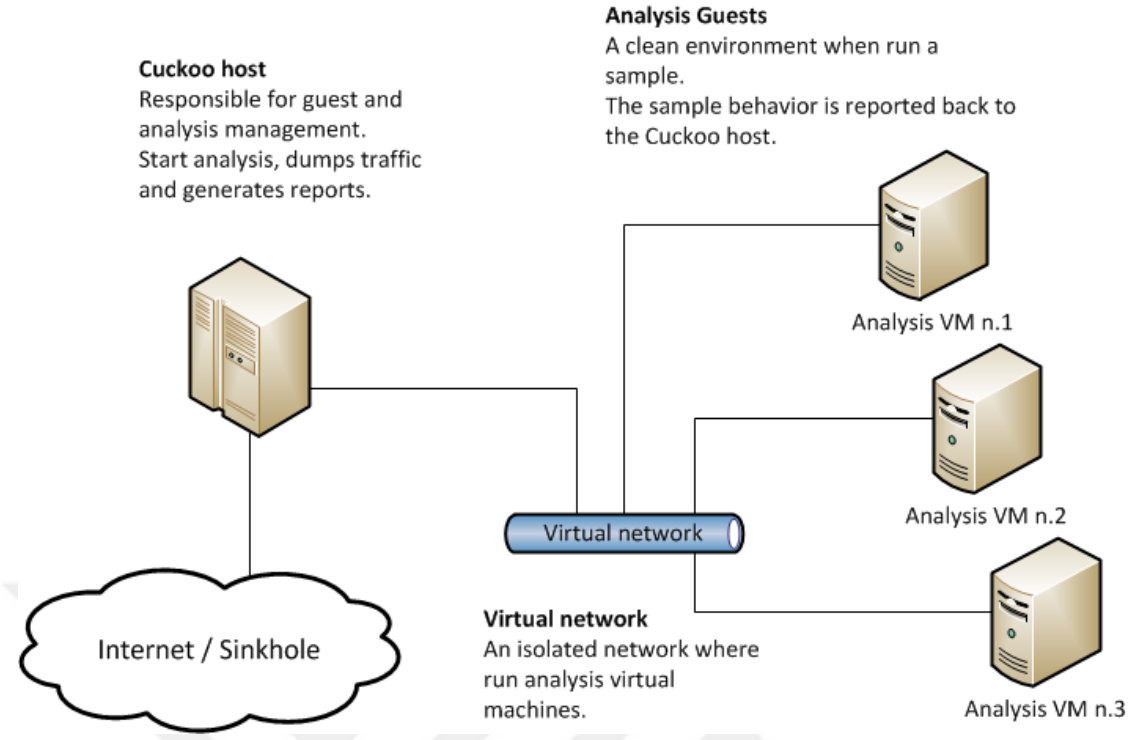
3.2 Cuckoo Sandbox (Kum Havuzu) Uygulamsı

Cuckoo Sandbox uygulaması bir kumhavuzu uygulamasıdır. Ücretsiz olarak dağıtılan ve farklı işletim sistemlerini destekleyen bir uygulamadır.

Bu uygulama ile şüpheli olarak görülen dosyalar ile ilgili detaylı analiz raporu elde edilebilmektedir. Genellikle zararlı yazılımları analiz etmek için kullanılmaktadır. Cuckoo Sandbox uygulaması ile zararlı yazılımların, gerçek çalışma ortamlarına benzer bir ortam hazırlanıp, bu ortamda çalıştırılması sağlamaktadır. Bu uygulama ile çalıştırılan zararlı yazılımların davranış ve yapısal özellikleri ile ilgili detaylı bilgiler elde edilmektedir. Zararlı yazılımın API çağrıları, oluşturduğu ağ trafiği, üzerinde çalıştığı sistemin bellek dökümü gibi birçok bilgi elde edilebilmektedir. Elde edilen veriler JSON yapısında MongoDB veritabanında tutulmaktadır.

Cuckoo Sandbox uygulamasının sisteminde iki temel bileşeni bulunmaktadır. Bu bileşenlerin ilki, zararlı yazılımların analizlerinin başlatıldığı, sonuçlarının yazıldığı veri tabanının bulunduğu, kullanıcılar için web hizmetinin sunulduğu, yönetim makinasıdır. Diğer bileşen ise, zararlı yazılımların üzerinde koşturulacağı, asıl analizin yapıldığı, zararlı yazılımların gerçek çalışma ortamı benzeri, analiz makinalarıdır. Analiz makinaları sanal veya fiziksel makinalar olabilmektedir. Şekil 3.1 'de Cuckoo sisteminin genel mimarisi gözükmektedir [20].

Zararlı yazılımlar analiz makinalarında çalıştırılarak, elde edilen analiz bilgileri yönetici makinasındaki veri tabanına kayıt edilmektedir. Yönetici uygulaması bir web servis ile ilgili analiz raporlarını anlaşılır bir şekilde kullanıcıya sunmaktadır.



Şekil 3.1: Cuckoo Sandbox Uygulama Mimarisi

Çalışmamızda, zararlı yazılımların davranışlarını temsil eden, Windows API çağrı dizileri, bu uygulama kullanılarak elde edilmiştir.

3.3 Virus Total Servisi

Virus Total servisi, dosyaları veya URL'leri analiz edebilen, internet üzerinden çevrimiçi olarak hizmet veren ücretsiz bir servistir. Analiz işlemleri için birçok antivirüs uygulama motoru ve web sitesi tarayıcıları kullanmaktadır. Zararlı olduğu düşünülen dosyalar antivirus uygulama motorlarında tek tek analiz edilmektedir. Herbir antivirüs uygulama motoru bir analiz raporu oluşturmaktadır. Yaklaşık olarak 67 adet antivirüs uygulama motoru kullanılmaktadır. Virus Total servisi bu uygulama motorlarından almış olduğu herbir analiz raporlarını yorumlamadan, bir ayüz ile kullanıcılara sunmaktadır. Aynı durum analiz edilmek istenen URL' ler içinde geçerlidir. Ayrıca, bu servis çok geniş bir analiz arşivini de içinde barındırmaktadır. Bu sayede yeni bir analiz yapılabileceği gibi, daha önceden yapılmış analiz raporları da elde edilebilmektedir [21].

Virus Total servisi web tarayıcı üzerinden hizmet verdiği gibi, web tarayıcısı kullanmadan da hizmet alınabilmesi için bir arayüz sunmaktadır(VirusTotal Public API v2.0). Bu

arayüz sayesinde, ugulama geliştiricileri bu servisi ile otomatik olarak dosyalar ve URL adresleri analiz edilebilmektedir.

Virus Total Public API kullanılarak elde edilen analiz sonucu JSON nesnesi olarak gelmektedir. Herbir antivirus uygulama motoru veya web tarayıcı analiz sonuçları, bu nesne üzerinden, ayrı ayrı olarak elde edilmektedir. Şekil 3.2 'de örnek bir analiz sonucu olarak JSON nesnesi gözükmektedir [22].


```
{
 'response_code': 1,
 'verbose_msg': 'Scan finished, scan information embedded in this object',
 'resource': '99017f6eebbac24f351415dd410d522d',
 'scan_id': '52d3df0ed60c46f336c131bf2ca454f73bafdc4b04dfa2aea80746f5ba9e6d1c-1273894724',
 'md5': '99017f6eebbac24f351415dd410d522d',
 'sha1': '4d1740485713a2ab3a4f5822a01f645fe8387f92',
 'sha256': '52d3df0ed60c46f336c131bf2ca454f73bafdc4b04dfa2aea80746f5ba9e6d1c',
 'scan_date': '2010-05-15 03:38:44',
 'positives': 40,
 'total': 40,
 'scans': {
    'nProtect': {'detected': true, 'version': '2010-05-14.01', 'result': 'Trojan.Generic.3611249', 'update': '20100514'},
    'CAT-QuickHeal': {'detected': true, 'version': '10.00', 'result': 'Trojan.VB.acgy', 'update': '20100514'},
    'McAfee': {'detected': true, 'version': '5.400.0.1158', 'result': 'Generic.dx!rkx', 'update': '20100515'},
    'TheHacker': {'detected': true, 'version': '6.5.2.0.280', 'result': 'Trojan/VB.gen', 'update': '20100514'},
    .
    .
    .
    'VirusBuster': {'detected': true, 'version': '5.0.27.0', 'result': 'Trojan.VB.JFDE', 'update': '20100514'},
    'NOD32': {'detected': true, 'version': '5115', 'result': 'a variant of Win32/Qhost.NTY', 'update': '20100514'},
    'F-Prot': {'detected': false, 'version': '4.5.1.85', 'result': null, 'update': '20100514'},
    'Symantec': {'detected': true, 'version': '20101.1.0.89', 'result': 'Trojan.KillAV', 'update': '20100515'},
    'Norman': {'detected': true, 'version': '6.04.12', 'result': 'W32/Smalltroj.YFHZ', 'update': '20100514'},
    'TrendMicro-HouseCall': {'detected': true, 'version': '9.120.0.1004', 'result': 'TROJ_VB.JVJ', 'update': '20100515'},
    'Avast': {'detected': true, 'version': '4.8.1351.0', 'result': 'Win32:Malware-gen', 'update': '20100514'},
    'eSafe': {'detected': true, 'version': '7.0.17.0', 'result': 'Win32.TRVB.Acgy', 'update': '20100513'}
  },
 'permalink': 'https://www.virustotal.com/file/52d3df0ed60c46f336c131bf2ca454f73bafdc4b04dfa2aea80746f5ba9e6d1c/analysis/1273894724/'
}
```

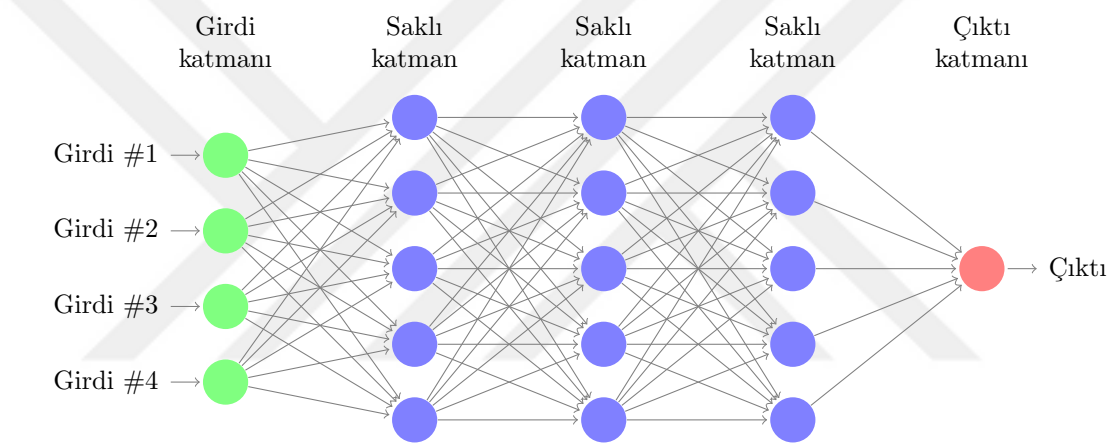

Şekil 3.2: Virus Total Public API Örnek Analzi Sonucu

Bu çalışmamızda, Virus Total Public API yardımı ile her bir zararlı yazılım dosyası analiz edilmiştir. Elde edilen sonuçlar işlenerek zararlı yazılımların türleri tespit edilmiş ve veri kümesine dahil edilmiştir.

3.4 Derin Öğrenme - Deep Learning

Derin öğrenme, yapay zeka alanında oldukça yaygın olarak kullanılmaktadır. Bilgisayarların performanslarının artması(GPU' ların kullanılabilmesi) ve internetin yaygınlaşması ile büyük veri kümelerine erişimin kolaylaşması, derin öğrenme yöntemlerine ilgiyi oldukça artırmıştır. Bu doğrultuda son zamanda yapılan sürücüsüz araç sistemleri, yüz tanıma, ses tanıma sistemleri gibi geniş bir alanda başarılı bir şekilde kullanılmaktadır.

Derin öğrenme, genel olarak insan beynini taklit eden bir öğrenme şekli ortaya koymaktadır. Makine öğrenmesi yöntemlerinden biridir. Çok katmanlı yapay sinir ağları üzeride çalışan algoritmalar ve modellerden oluşmaktadır.



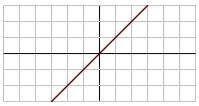
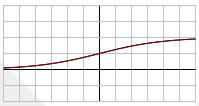
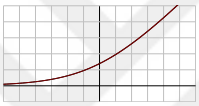
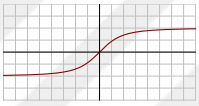
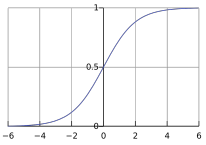
Şekil 3.3: Derin Öğrenme Modeli

Şekil 3.3'de derin öğrenmenin çok katmanlı model yapısı gösterilmektedir. Şekilde 3 temel katman (girdi katmanı, saklı katman ve çıktı katmanı) gözükmesine rağmen, derin öğrenmenin çok katmanlı olma özelli bir çok saklı katman içermesinden dolayıdır.

3.4.1 Aktivasyon Foksiyonu

Aktivasyon fonksiyonları, yapay sinir ağlarının eğitimi sırasında elde edilen değerlerin düzenlenmesinde, modellerin uygun bir şekilde eğitilmesinde kullanılmaktadır. **Model eğitimleri sırasında kullanılan**; *tanh*, *relu*, *sigmoid*, *softplus*, *softsign*, *softmax* ve *linear* aktivasyon fonksiyonlarının matematiksel gösterimi Şekil 3.4'de gösterilmektedir.

Bu çalışmamızda, sınıflandırma modelleri eğitilirken Şekil 3.4'deki aktivasyon fonksiyonları kullanılmıştır.

Fonksiyon	Şekil	Denklem	Türev
linear		$f(x) = x$	$f'(x) = 1$
tanh		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
relu		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
sigmoid		$f(x) = x \cdot \sigma(x)$	$f'(x) = f(x) + \sigma(x)(1 - f(x))$
softplus		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$
softsign		$f(x) = \frac{x}{1+\lvert x \rvert}$	$f'(x) = \frac{1}{(1+\lvert x \rvert)^2}$
softmax		$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$ *for* $i = 1, \ldots, J$	$\frac{\partial f_i(\vec{x})}{\partial x_j} = f_i(\vec{x})(\delta_{ij} - f_j(\vec{x}))$

Şekil 3.4: Aktivasyon Fonksiyonları

3.4.2 LSTM

LSTM, RNN tabanlı bir derin öğrenme yöntemidir. LSTM, RNN sinir ağlarının uzun süreli öğrenmelerde yeterince başarılı olamaması üzerine geliştirilmiştir. Vanishing Gradient problemine çözüm olarak tasarlanmıştır.

LSTM uzun süreli bağımlılıkları rastgele aralıklarla hatırlayabilen, öğrenebilen bir mimariye sahiptir. Özellikle zamana göre sıralı bir şekilde gelen verileri veya belirli bir ilişkiye sahip olayları analiz etmekte oldukça başarılı bir yöntemdir [23]. $\mathbf{x} = \{x_1, \cdots, x_T\}$ şeklinde ilerleyen sıralı bir veri olmak üzere, RNN, $\mathbf{h} = \{h_1, \cdots, h_T\}$ şeklinde ilerleyen gizli vektör sıralamasını ve $\mathbf{y} = \{y_1, \cdots, y_T\}$ şeklinde ilerleyen çıktı vektörü sıralamasını

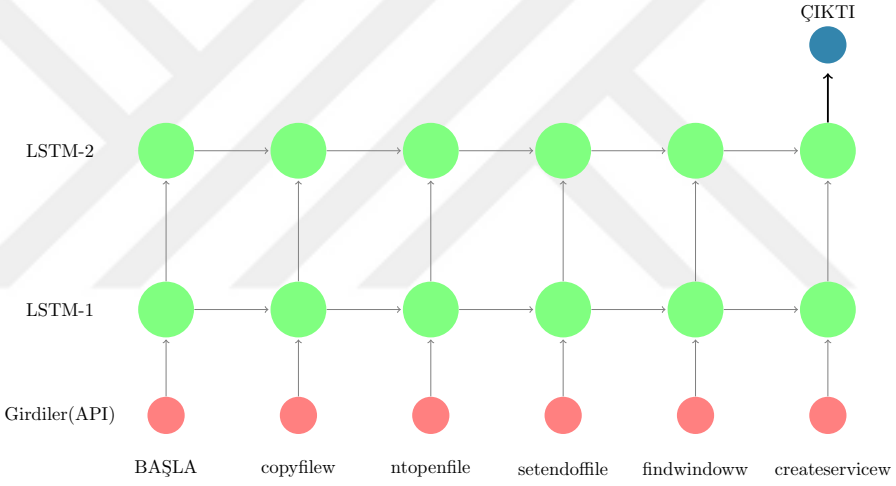
bulmaktadır. Bu hesaplama T adet yineleme ile şu şekilde bulunmaktadır.

$$\begin{aligned} h_t &= \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= \mathcal{F}(W_{hy}h_t + b_y) \end{aligned} \tag{3.1}$$

W_{xh}, W_{hh}, ve W_{hy} matrisleri, eğitim zamanında hesaplanan bağlantı ağırlıklarıdır. $\mathcal{F}$ ise sigmoid aktivaston fonksiyonudur. Sigmoid şu şekilde tanımlanmıştır.

$$\mathcal{F} = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{3.2}$$

Çalışmamızda iki katmanlı LSTM modeli kullanılmıştır. Şekil 3.5'de iki katmalı derin öğrenme model yapısı gösterilmektedir.



Şekil 3.5: İki Katmanlı LSTM Modeli

Bölüm 4

Yöntem

Bu çalışmada, öncelikle bir veri kümesi oluşturulmuştur. Sonrasında ise, bu veri kümesi kullanıarak zararlı yazılımların türlerine göre sınıflandırma yapabilen modeller oluşturulmuştur.

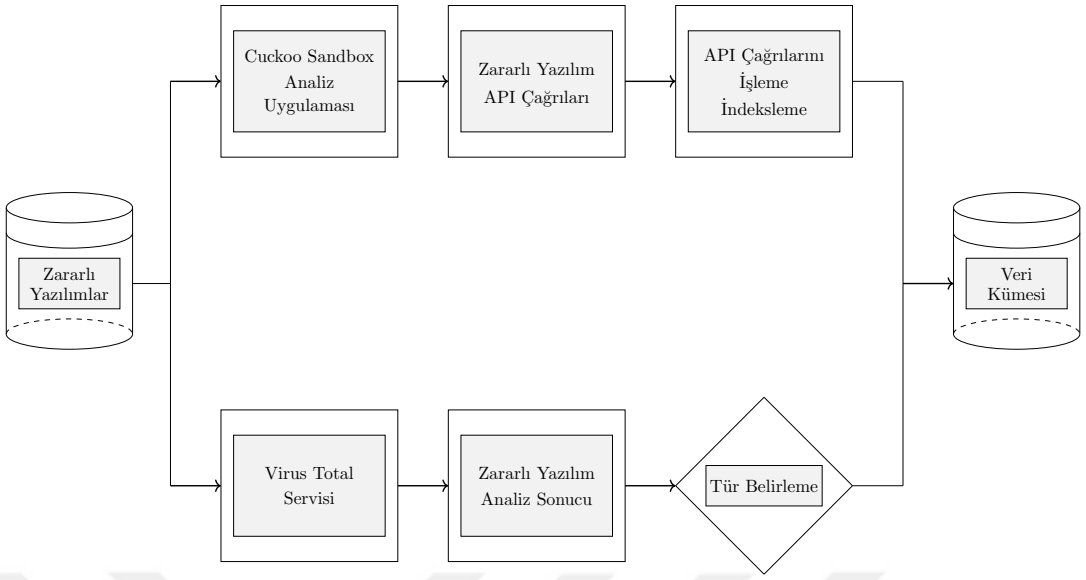
4.1 Veri Kümesi Oluşturma

Bu çalışma kapsamında bir veri kümesi oluşturulmuştur. Bu veri kümesinde, farklı farklı sınıflardan, toplam 7107 adet zararlı yazılımın analiz bilgileri bulunmaktadır. Bu veri kümesi oluşturulurken yukarıda bahsedilen Cuckoo Sandbox uygulaması, zararlı yazılımların Windows API çağrı dizilerini elde etmek için; Virüs Total Servisi zararlı yazılımların sınıflarını tespit etmek için kullanılmıştır. Ayrıca veri kümesi oluşturulurken ihtiyaç duyulan yazılım gereksinimleri için Python dili kullanılmıştır.

Şekil 4.1'de genel adımlar gösterilmektedir.

Veri kümesi oluşturulurken aşağıdaki adımlar takip edilmiştir.

Cuckoo Sandbox ortamı hazırlanması: Bu uygulamanın koşacağı makineye Ubuntu işletim sistemi kurulmuştur. Daha sonrasında Cuckoo Sandbox uygulaması yüklemiştir. Zararlı yazılımların çalıştırılıp analiz edileceği, analiz makinası sanal sunucu olarak ayağa kaldırılmıştır. Bu sunucuya Windows işletim sistemi kurulmuştur. Zararlı yazılımların çalışması sırasında herhangi bir engel oluşmaması için güvenlik duvarı kapatılmış, işletim sistemi güncellemeleri yapılmamıştır [24].



Şekil 4.1: Veri Kümesi Oluşturma

- İşletim sistemi sürümü : Ubuntu 17.10
- Cuckoo Sandbox sürümü : 2.0.4
- Analiz makinası işletim sistemi sürümü: Windows 7
- Sanallaştırma uygulaması: VirtualBox 5.1.30
- Python Sürümü: 3.6.5

Zararlı yazılımların analiz edilmesi: 20000' den fazla zararlı yazılım teker teker Cuckoo Sandbox uygulamasında çalıştırılmıştır. Uygulama tarafından, herbir zararlı yazılımın analiz bilgileri MongoDB veritabanına yazılmıştır. Bu analiz bilgilerinden, zararlı yazılımın analiz makinasında göstermiş olduğu davranış verileri elde edilmiştir. Bu veriler, zararlı yazılımın Windows 7 işletim sistemi üzerinde yapmış olduğu bütün Windows API çağrı istekleridir.

Windows API çağrılarının indekslenmesi: Windows API çağrıları incelendiğinde, 342 çeşit API çağrısının yapıldığı tespit edilmiştir. Bu API çağrıları 0-341 arası rakamlar ile indekslenerek bir veri kümesi oluşturulmuştur. Bu veri kümesinde, her bir satır bir zararlı yazılımın yapmış olduğu API çağrı dizisini temsil etmektedir.

Veri kümesinin filtrenmesi: Zararlı yazılımların analizi sırasında, zararlı yazılımların özelleşmiş bir uygulama için geliştirilmiş olması, çalışması sırasında hata alması gibi durumlar göz önünde bulundurularak bir filtreleme yapılmıştır. Her bir API çağrı dizisi

incelenerek en az 10 farklı API çağrısı barındırmayan API çağrı dizileri veri kümesinden temizlenmiştir.

Virus Total Public API yardımıyla zararlı yazılımların analiz edilmesi: Cuckoo Sandbox uygulaması ile Windows API çağrı dizileri elde edilen her bir zararlı yazılım, tek tek Virus Total servisine sorularak analiz ettirilmiştir. Ve elde edilen analiz sonuçları bir veri tabanına kaydedilmiştir. Bu sayede herbir zararlı yazılım birçok farklı antivirus motoru tarafından analiz edilmiş ve analiz sonucları kayıt altına alınmıştır.

Analiz sonuçlarının işlenmesi: Virus Total servisi, dosya analizi için yaklaşık 67 farklı antivirüs uygulama motoru kullanmaktadır. Bu servis kullanılarak elde ettiğimiz her bir analiz sonucundan, ilgili zararlı yazılımın ait olduğu sınıf belirlenmiştir. Bu işlem sırasında, aynı zararlı yazılım için farklı antivirus uygulamalarının farklı sonuçlar verdiği tespit edilmiştir. Ayrıca her zararlı yazılımı her antivirus uygulamasının tespit edemediği gözlemlenmiştir. Örneğin; 06e76cf96c7c7a3a138324516af9fce8 hash değerine sahip zararlı yazılım dosyası Virus Total servisinde analiz edildiğinde, birçok uygulama bu dosyanının bir worm olduğunu belirtirken, "DrWeb" gibi bazı uygulamalar trojan olduğunu, "Babable" uygulaması ise temiz bir dosya olduğunu belirtmektedir [25]. Bu nedenle zararlı yazılımların sınıflarını tespit ederken bir zararlı yazılım için elde edilen analizler içerisinde hangi sınıf sayıca daha fazla belirtilmiş ise, zararlı yazılımın sınıfı o sınıf olarak kabul edilmiştir. Yani antivirüs uygulamalarının çoğunluğunun mutabık kaldığı sınıf ele alınmıştır. Bu kabullenme ile her bir zararlı yazılımın dahil olduğu sınıflar belirlenmiştir.

Veri kümesinin oluşturulması: Son olarak, elde edilen Windows API çağrı dizileri ve zararlı yazılım sınıfları eşleştirilerek veri kümesi oluşturulmuştur. Veri kümesi oluşturulurken, 8 farklı sınıf göz önünde bulundurulmuştur. Ve bu sınıflar için veri kümesi filtrelenmiştir. Bu sınıflar ve bu sınıflara dahil olan zararlı yazılım sayıları şu şekildedir.

1. **Adware, 379**
2. **Backdoor, 1001**
3. **Downloader, 1001**
4. **Dropper, 891**
5. **Spyware, 832**
6. **Trojan, 1001**
7. **Virus, 1001**

8. **Worms, 1001**

Bu sayede 7107 adet zararlı yazılımın Windows API çağrı dizilerinin ve dahil oldukları sınıf bilgilerinin bulunduğu bir veri kümesi oluşturulmuştur.

4.2 Model Oluşturma

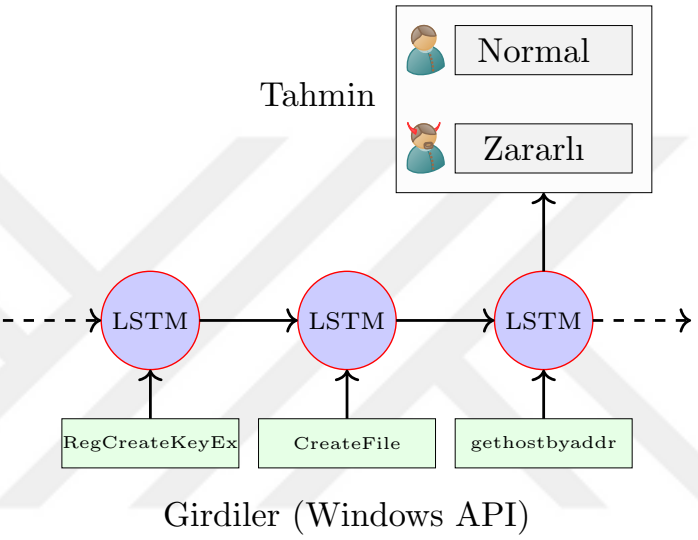
Bu çalışma kapsamında Python programlama dili kullanılarak bir uygulama geliştirilmiştir. Geliştirilen bu uygulama, zararlı yazılım türlerine göre sınıflandırma yapmamızı sağlamaktadır. Bu uygulama ile sınıflandırmak istediğimiz 8 farklı tür için ayrı ayrı sınıflandırma modelleri oluşturulabilmektedir. Herbir sınıflandırma modelinin eğitilmesi ve test edilmesi, çalışma kapsamında oluşturulan veri kümesi kullanılarak yapılmaktadır.

Uygulama şu şekilde çalışmaktadır.

1. Sınıflandırma modeli oluşturulmak istenilen zararlı yazılım türü seçilmektedir.
2. Veri kümesinde bulunan veriler seçilen zararlı yazılım türü için işlenmektedir. Modeli oluşturulmak istenilen zararlı yazılım türü bilgisine "1" etiketini, diğer tür bilgilerine ise "0" etiketini atamaktadır. Bu şekilde model çıktısı 0 (türe ait bir davranış değil) veya 1 (türe ait bir davranış) olarak karşımıza çıkacaktır.
3. İki katmanlı LSTM tabanlı bir sınıflandırma modeli tanımlanarak model oluşturulur.
4. Tanımlanan sınıflnadırma modeli, veri kümesindeki verilerin %80'ı kullanılarak eğitilir. Eğitim için ayrılan verilerin %30'u ile eğitim sırasında doğrulama işlemi yapılmaktadır.
5. Eğitilen sınıflandırma modeli veri kümesindeki verilerin %20'u kullanılarak test edilir. Test işlemi sırasında, yeni bir yazılımın API çağrıları sınıflandırma modeline verilmekte, model sonuçlarının oylanması sonucuyla sınıf etiketi tespit edilmektedir.
6. Eğitim ve test sonuçları kayıt edilmektedir.
7. Farklı aktivasyon fonksiyonları kullanılarak 3., 4., 5. ve 6. adımlar tekrar edilerek farklı sınıflandırma modelleri oluşturulmaktadır.

Yukarıda bahsedilen 1.-7. adımlar herbir zararlı yazılım türü için ayrı ayrı olarak işletilerek, 8 farklı tür için sınıflandırma modelleri oluşturulmaktadır.

Bu uygulama Python programlama dili ve makine öğrenmesi kütüphaneleri olan Keras, Tensorow ve Scikit-learn kütüphaneleri kullanılmıştır. LSTM ağlarının kurulması için Keras kütüphanesi kullanılmıştır. İki katmanlı bir LSTM yapısı oluşturulmuştur. Sınıflandırma modelleri, *tanh*, *relu*, *sigmoid*, *softplus*, *softsign*, *softmax* ve *linear* aktivasyon fonksiyonları kullanılarak ayrı ayrı olarak eğitilmiştir.



Şekil 4.2: Windows API çağırımları ile LSTM modeli oluşturulması

Model eğitimi süreci Şekil 4.2'de gösterilmiştir. LSTM ağ modeli her bir zararlı yazılımın Windows işletim sisteminde yaptığı API çağrılarını sırası ile almakta ve son adımda tahmin sınıf etiketini, $\hat{y}$ hesaplamaktadır. Her bir sınıf etiketi için model oluşturulmasından dolayı modellerimiz ikili sınıflandırıcılardır. Kayıp fonksiyonu olarak *log loss* kullanılmıştır. Log loss kayıp fonksiyonmu eşitlik 4.1'de gösterilmiştir.

$$l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{L} \sum_{l=1}^{l=|L|} -(y_l \log(\hat{y}_l) + (1 - y_l) log(1 - \hat{y}_l)) \tag{4.1}$$

Ayrıca çalışmamızda çoklu sınıflandırıcı modelleride oluşturulmuştur. Bu modeller 0-8 arasında değerler vererek çoklu sınıflandırma yapabilmektedir. Bu modeller tek ve iki katmanlı LSTM modeller olacak şekilde oluşturulmuştur.

Bölüm 5

Deneyim

5.1 Sınıflandırma Modellerinin İlklendirilmesi

Bu çalışma kapsamında uygulama geliştirmek için Python programlama dili seçilmiş ve Python 3.6 versiyonu kullanılmıştır. Ayrıca analiz modellerinin oluşturulması ve elde edilen sonuçların anlamlı hale getirilmesi için birçok kütüphane kullanılmıştır.
Sınıflandırma modellerinin ilklendirilmesi sırasında kullanılan kütüphaneler;

Tensorflow ; Derin öğrenme çalışmaları için Tensorflow kütüphanesinin, en son versiyon olan 1.12 sürümü kullanılmıştır. Bu kütüphane Google tarafından geliştirilmiş, açık kaynak kodlu olarak sunulmuş ücretsiz bir kütüphanedir. Bu kütüphane makine öğrenmesi ve derin öğrenme için kullanılmaktadır. Python programlama dili ile hızlı ve kolay bir şekilde uygulama geliştirilebilmesine olanak sağlamaktadır. Ayrıca esnek mimarisi sayesinde, yüksek işlem gücü gerektiren uygulamalar için çeşitli platformlarda(CPU, GPU, TPU) hesaplama yükünün kolay bir şekilde dağıtılmasını sağlamak amacıyla programlama yapılabilmesine olanak tanımaktadır [26].

Keras: En son versiyon olan 2.2.4 sürümü kullanılmıştır. Bu kütüphane, Theano yada Tensorflow kütüphaneleri üzerine kurulmuş bir Python kütüphanesidir. Üst seviye bir kütüphane olduğu için Tensorflow' a göre kullanılması oldukça kolaydır. Bu kütüphane ile model oluşturma, eğitme gibi işlemler çok kolay bir şekilde, kısa bir sürede programlanabilir [27].

Scikit-learn: En son versiyon olan 0.20 sürümü kullanılmıştır. Bu kütüphane de makina öğrenmesinde kullanılan, regresyon, karar ağaçları oluşturma gibi birçok yöntemi barındırmaktadır. Ayrıca test verisi oluşturma, eğitim sonuçlarının karşılaştırılması ve sınıflandırılması ile ilgili birçok yeteneğe de sahiptir. Çalışmamız da eğitilen modellerin test sonuçlarını anlamlandırmak için kullanılmıştır [28].

Ayrıca, test verilerin okunmasında ve model eğitim geçmişinin görselleştirilmesinde Pandas - 0.23 [29] ve Matplotlib - 2.2.2 [30] kütüphaneleri kullanılmıştır.

Bu çalışmada, yukarıda bahsedilen kütüphaneler kullanılarak, 8 farklı zararlı yazılım sınıfı için, sınıflandırma modelleri oluşturulmuştur. Çalışma sırasında farklı farklı parametreler göz önünde bulundurulduğu için, bir zararlı yazılım sınıfı için birden fazla sınıflandırma modeli oluşturulmuştur.

Sınıflandırma yöntemi olarak ikili sınıflandırma modelleri ve çoklu sınıflandırma modelleri oluşturulmuştur. İkili sınıflandırma modelleri sadece bir tür için sınıflandırma yapabilirken, çoklu sınıflandırma modelleri 8 farklı tür için sınıflandırma yapabilmektedir.

Sınflandırma modellerinin test sonuçları hata matrisleri ile gösterilmiştir. Hata matrisleri oluştururken sklearn.metrics kütüphanesinin confusion_matrix metodunu kullanılmıştır [31]. Oluşturulan hata matrislerinde sütunlar tahmin(predicted), satırlar gerçek(actual) değerlerini göstermektedir. Satır ve sütun değerleri sıfırdan başlayarak devam etmektedir. İkili hata matrislerinde 0(Negatif), 1(Positif); çoklu hata matrsilerinde ise 0..7 şeklindedir. Örnek hata matrisi Tablo 5.1' de gösterilmiştir

Tablo 5.1: Örnek Hata Matrisi

		Tahmin	
		Negatif	Pozitif
Gerçek	Negatif	1345	6
	Pozitif	16	55

5.2 İkili Sınıflandırma Analiz Sonuçları

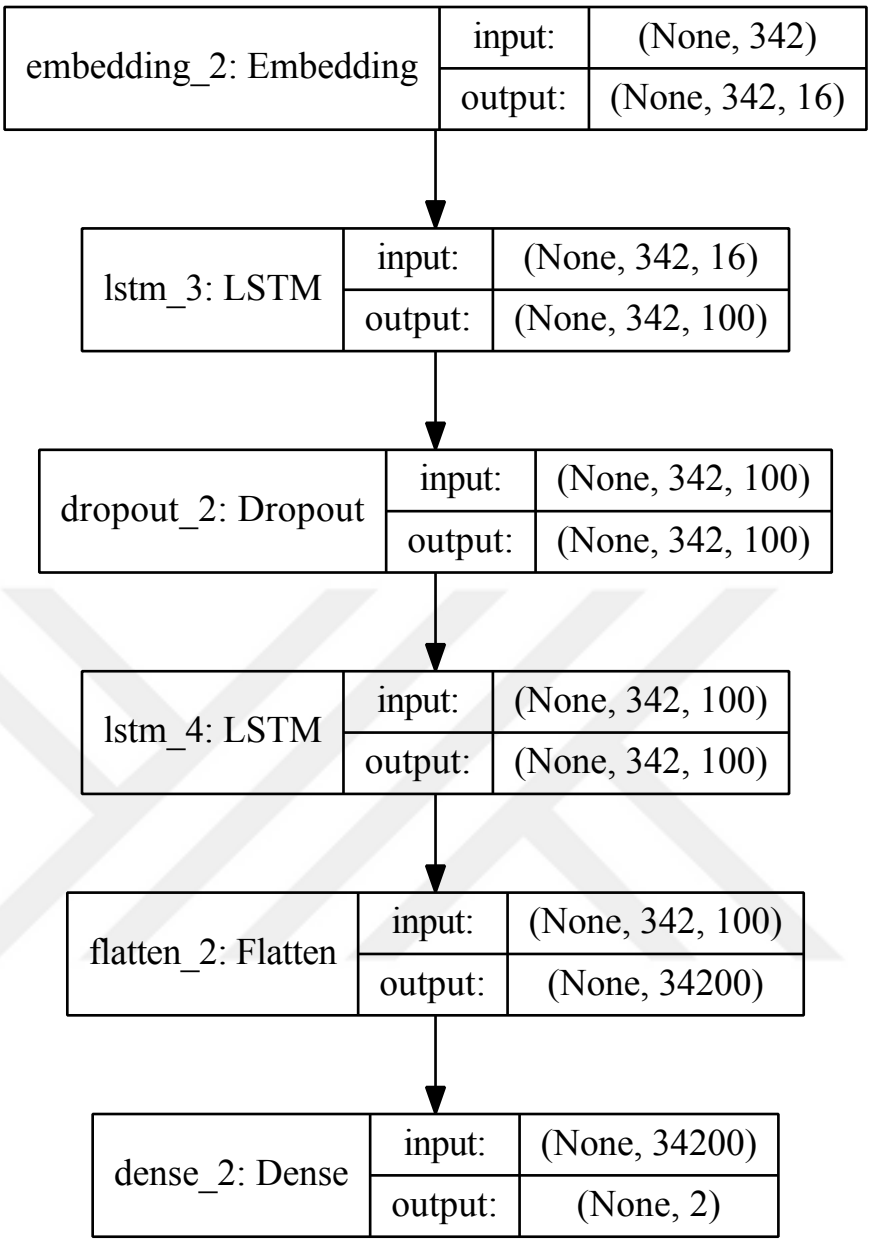
Her bir sınıf için ayrı ayrı model oluşturulmak istendiği için, veri kümesindeki sınıf bilgileri düzenlenmiştir. Analiz edilmek istenen zararlı yazılım sınıfı için '1' değeri, diğerleri için '0' değeri ataması yapılmıştır. Herbir veri sınıf modeli oluşturulurken bu işlem

yapıldığından dolayı oluşturulan modeller ikili tip sınıflandırma yapmaktadır. Örneğin Adware sınıfı için oluşturulan bir sınıflandırma modelinin sonucunu, "Adware" veya "diğer" olmaktadır.

5.2.1 İkili LSTM Analiz Sonuçları

Her bir zararlı yazılım sınıfı için *tanh*, *relu*, *sigmoid*, *softplus*, *softsign*, *softmax* ve *linear* aktivasyon fonksiyonları kullanılarak 8 farklı model oluşturulmuş.

Zararlı yazılım modelleri oluşturulurken, analiz verileri farklı farklı sayılarda olmasına rağmen, analiz aynı akış katmanları kullanılarak yapılmıştır. Modellerin akış katmanlarına örnek olarak Şekil 5.1'de Adware sınıfı model çıktısı gözükmektedir.



Şekil 5.1: Sınıflandırma Model Yapısı

İki katmanlı LSTM yapısında %20 dropout değeri kullanılmıştır. Eğitim sırasında girdi boyutu, veri kümesi oluşturulurken kullanılan farklı Windows API çağrı sayısı olarak belirlenmiştir. 342 çeşit API çağrısı kullanıldığı tespit edilmiştir.

Bu çalışma kapsamında, 8 farklı tür için farklı aktivasyon fonksiyonları kullanılarak elde edilen sınıflandırma analiz sonuçları farklı başlıklar altında, detaylı bir şekilde aşağıda verilmiştir.

5.2.1.1 Adware Zararlı Yazılım Sınıfı için Sonuçlar

Adware zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Adware sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Adware zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.2' de gösterilmektedir.

Tablo 5.2: Adware Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.985	0.95	0.981	0.95	0.981	0.972	0.95
Hassasiyet	0.90	0	0.85	0	0.83	0.78	0
Anımsama	0.77	0	0.75	0	0.77	0.61	0
F1	0.83	0	0.80	0	0.80	0.68	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.3' de verilmiştir.

Tablo 5.3: Adware Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1345	6
P	16	55

(b) sigmoid

	N	P
N	1342	9
P	18	53

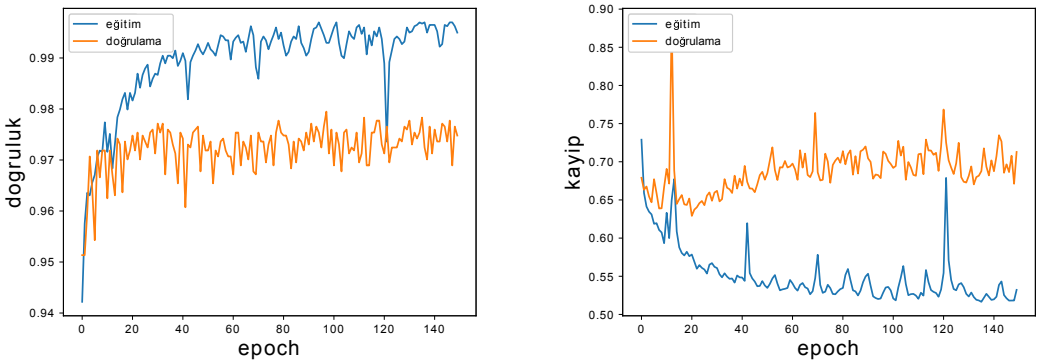
(c) softsign

	N	P
N	1340	11
P	16	55

(d) softmax

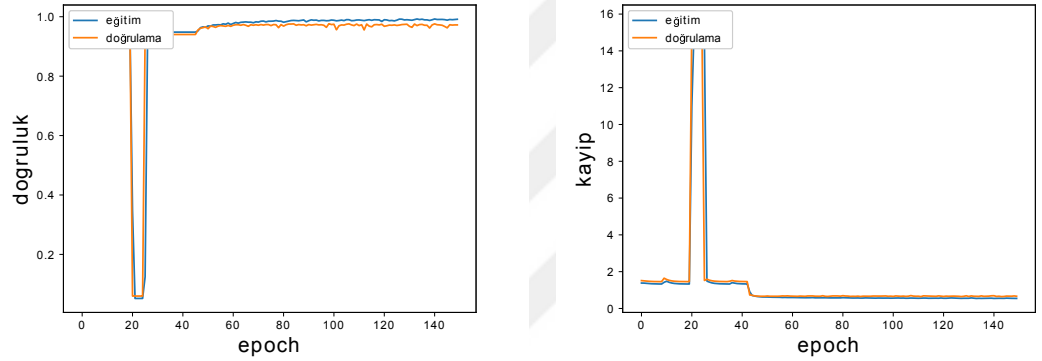
	N	P
N	1339	12
P	28	43

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.2 - 5.5' de verilmiştir.



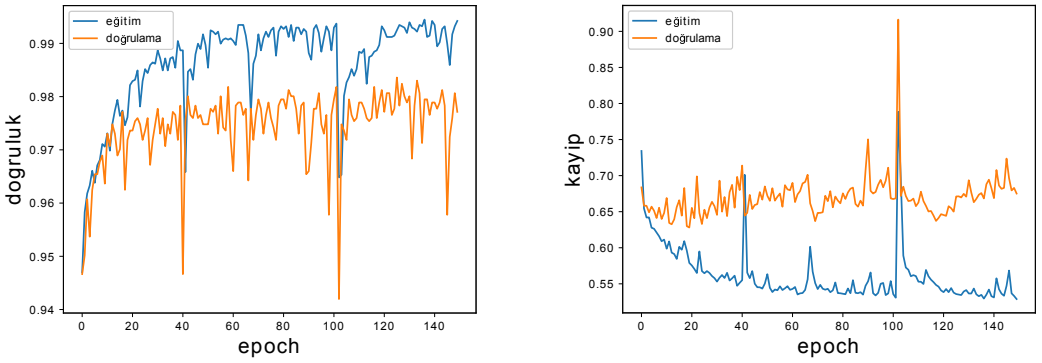
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.2: Adware-tanh modeli doğruluk-kayıp grafikleri



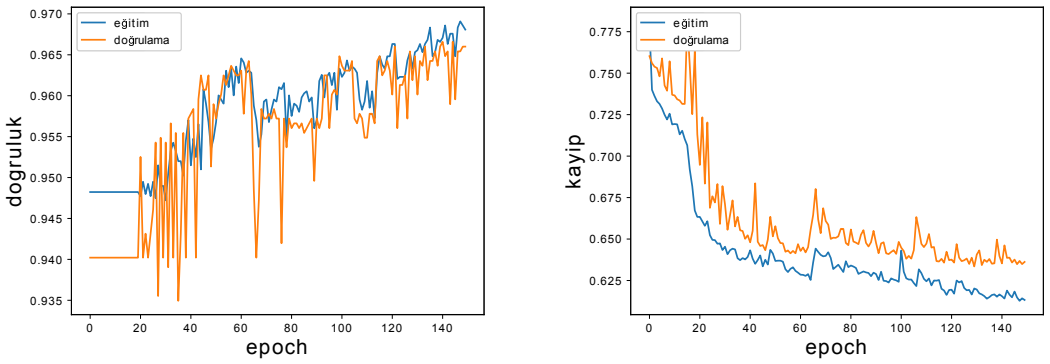
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.3: Adware-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.4: Adware-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.5: Adware-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *tanh* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %98.5 değerinde bir doğruluk oranı elde edilmiştir. Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: tanh
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.2 Backdoor Zararlı Yazılım Sınıfı için Sonuçlar

Backdoor zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Backdoor sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Backdoor zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.4' de gösterilmektedir.

Tablo 5.4: Backdoor Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.877	0.857	0.854	0.857	0.876	0.864	0.857
Hassasiyet	0.60	0	0.49	0	0.56	0.66	0
Anımsama	0.40	0	0.38	0	0.58	0.09	0
F1	0.48	0	0.43	0	0.57	0.16	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.5‘ de verilmiştir.

Tablo 5.5: Backdoor Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1165	54
P	121	82

(b) sigmoid

	N	P
N	1138	81
P	126	77

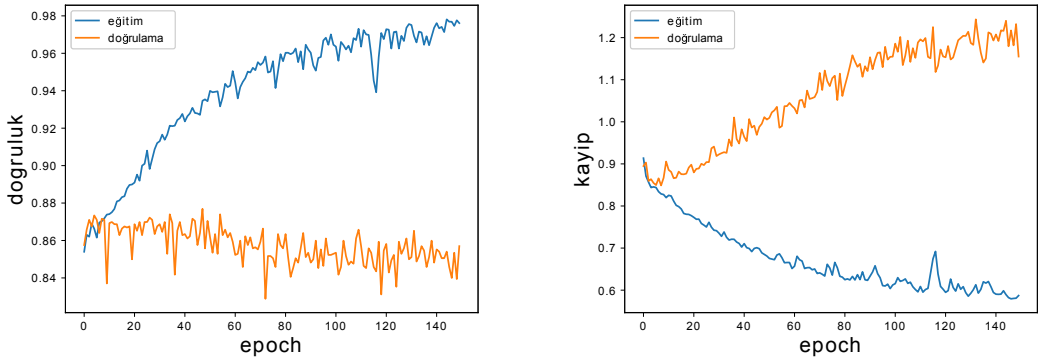
(c) softsign

	N	P
N	1128	91
P	85	118

(d) softmax

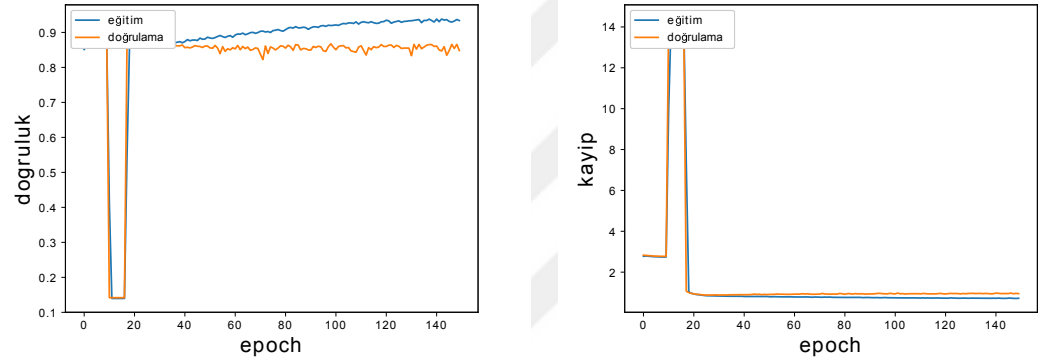
	N	P
N	1209	10
P	184	19

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.6 - 5.9‘ de verilmiştir.



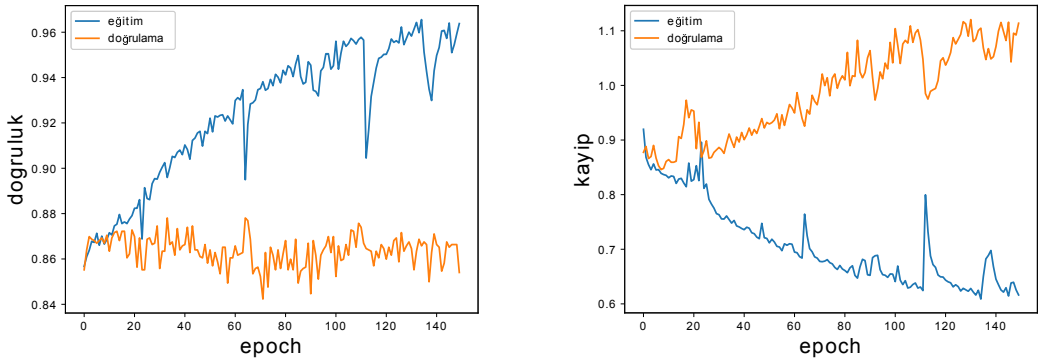
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.6: Backdoor-tanh modeli doğruluk-kayıp grafikleri



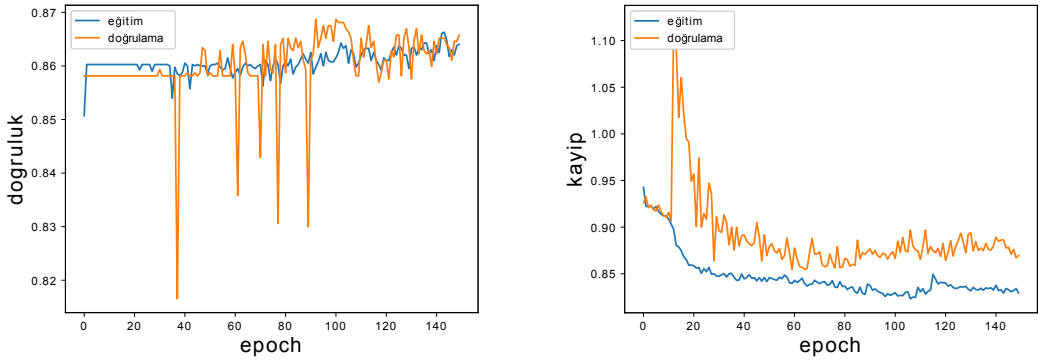
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.7: Backdoor-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.8: Backdoor-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

ŞEKIL 5.9: Backdoor-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *tanh* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %87,7 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: tanh
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.3 Downloader Zararlı Yazılım Sınıfı için Sonuçlar

Downloader zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Downloader sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Downloader zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.6' de gösterilmektedir.

Tablo 5.6: Downloader Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.895	0.86	0.905	0.860	0.909	0.823	0.860
Hassasiyet	0.65	0	0.75	0	0.72	0.42	0
Anımsama	0.56	0	0.49	0	0.57	0.72	0
F1	0.60	0	0.59	0	0.64	0.53	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.7' de verilmiştir.

Tablo 5.7: Downloader Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1163	60
P	88	111

(b) sigmoid

	N	P
N	1190	33
P	102	97

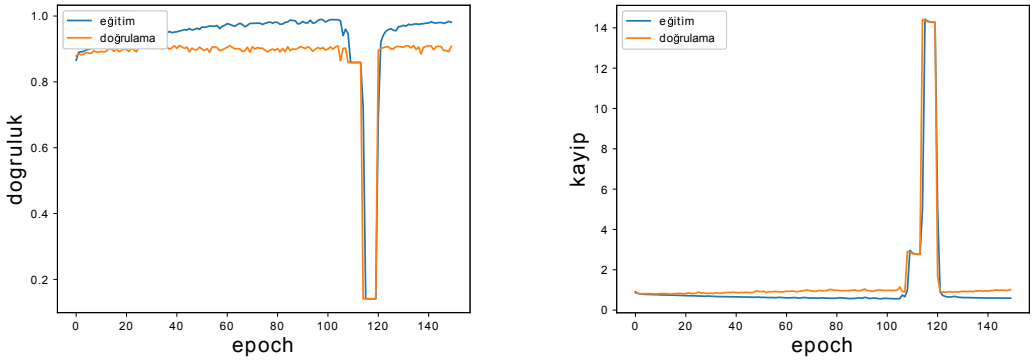
(c) softsign

	N	P
N	1179	44
P	85	114

(d) softmax

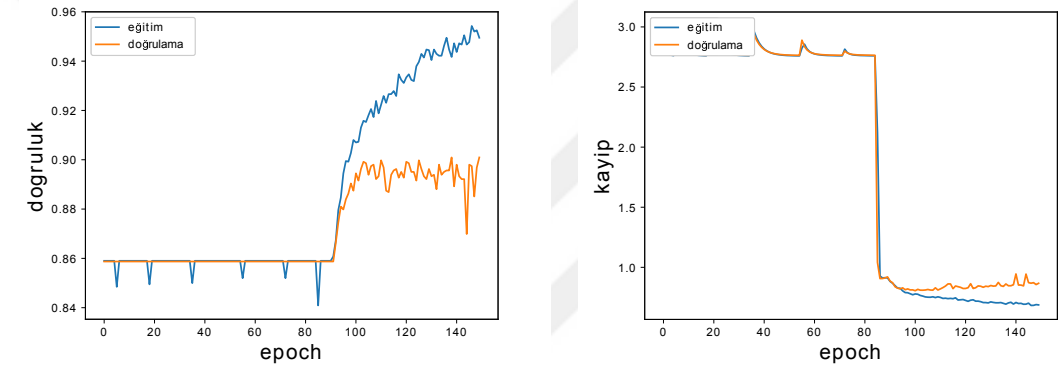
	N	P
N	1027	196
P	55	144

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.10 - 5.13' de verilmiştir.



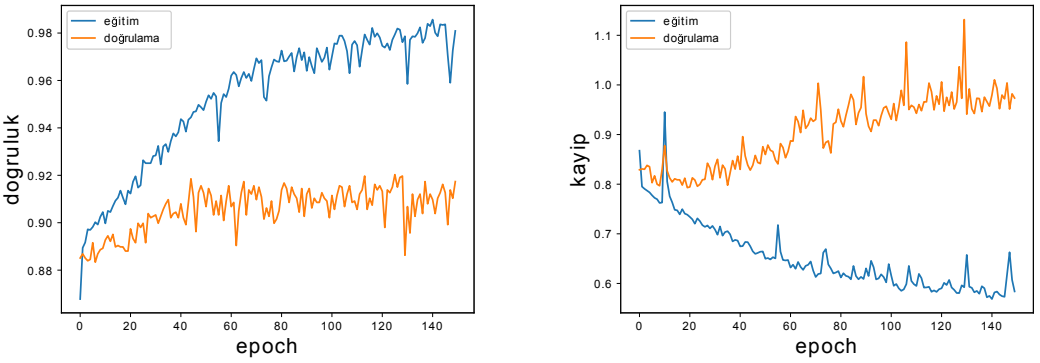
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.10: Downloader-tanh modeli doğruluk-kayıp grafikleri



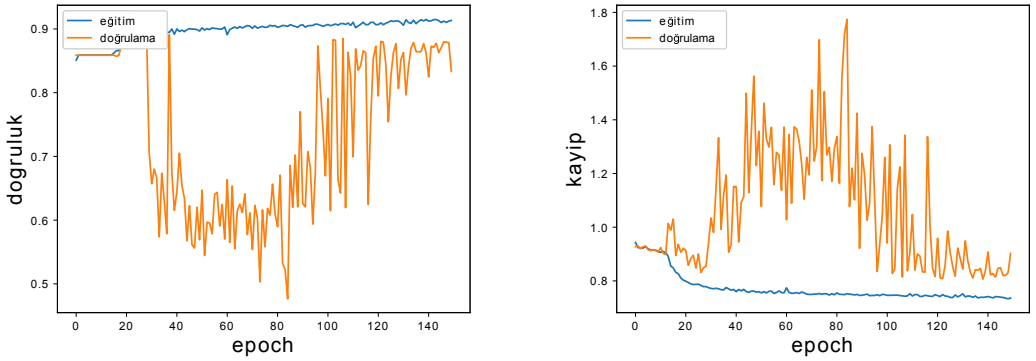
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.11: Downloader-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.12: Downloader-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.13: Downloader-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *softsign* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %90,9 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: softsign
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.4 Dropper Zararlı Yazılım Sınıfı için Sonuçlar

Dropper zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Dropper sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Dropper zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.8' de gösterilmektedir.

Tablo 5.8: Dropper Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.862	0.872	0.871	0.872	0.877	0.870	0.872
Hassasiyet	0.46	0	0.48	0	0.52	0.29	0
Anımsama	0.46	0	0.12	0	0.44	0.01	0
F1	0.46	0	0.19	0	0.48	0.02	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.9' de verilmiştir.

Tablo 5.9: Dropper Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1142	99
P	97	84

(b) sigmoid

	N	P
N	1217	24
P	159	22

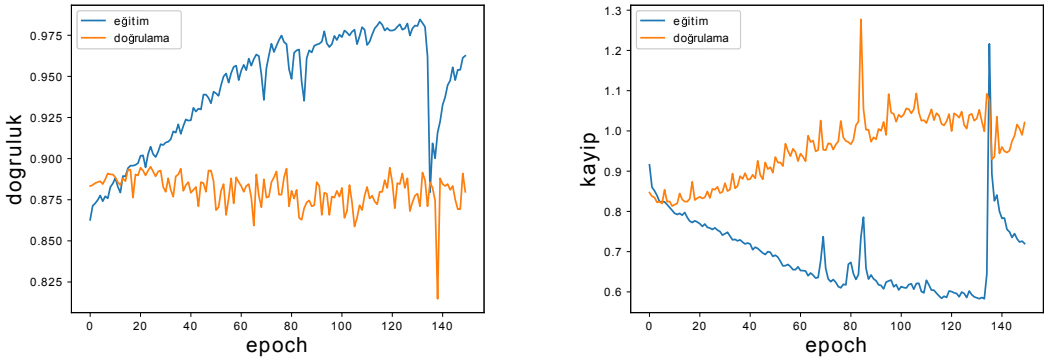
(c) softsign

	N	P
N	1169	72
P	102	79

(d) softmax

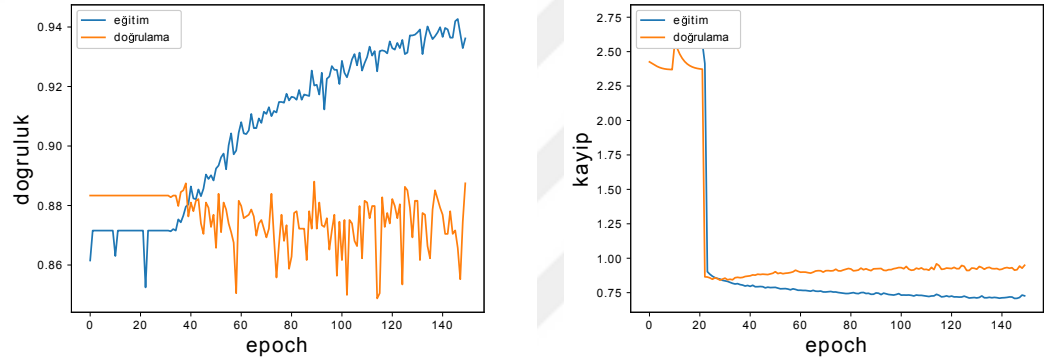
	N	P
N	1236	5
P	179	2

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.14 - 5.17' de verilmiştir.



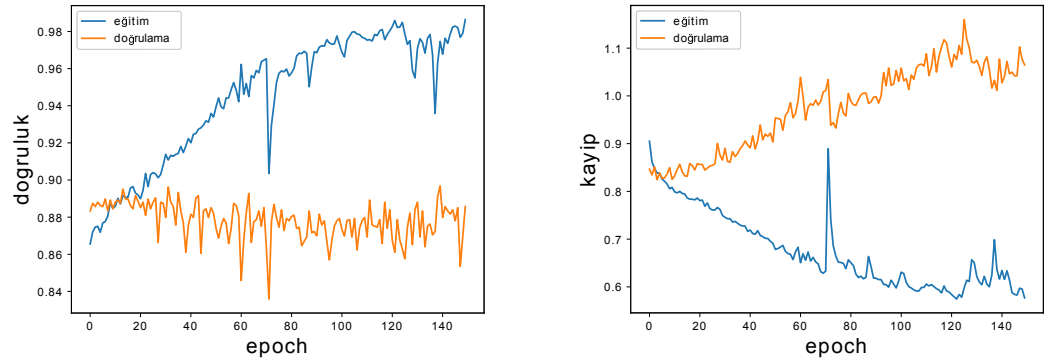
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.14: Dropper-tanh modeli doğruluk-kayıp grafikleri



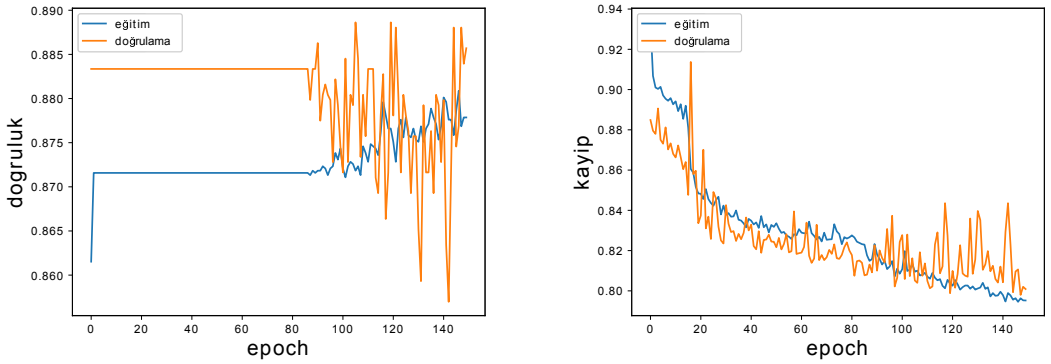
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.15: Dropper-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.16: Dropper-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

ŞEKIL 5.17: Dropper-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *softsign* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %87.7 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: softsign
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.5 Spyware Zararlı Yazılım Sınıfı için Sonuçlar

Spyware zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Spyware sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Spyware zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.10' de gösterilmektedir.

Tablo 5.10: Spyware Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.867	0.886	0.872	0.556	0.876	0.877	0.886
Hassasiyet	0.40	0	0.42	0.13	0.46	0.41	0
Anımsama	0.31	0	0.33	0.49	0.42	0.19	0
F1	0.35	0	0.37	0.20	0.44	0.26	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softplus*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.11' de verilmiştir.

Tablo 5.11: Spyware Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1182	78
P	111	51

(b) sigmoid

	N	P
N	1188	72
P	109	53

(c) softplus

	N	P
N	712	548
P	83	79

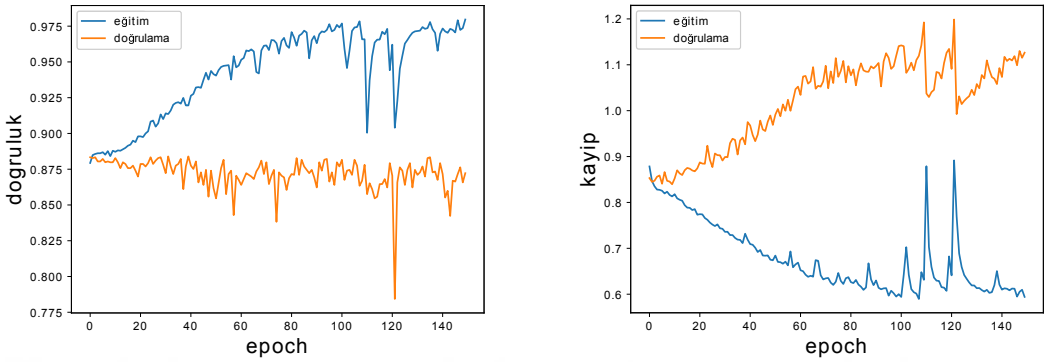
(d) softsign

	N	P
N	1179	81
P	94	68

(e) softmax

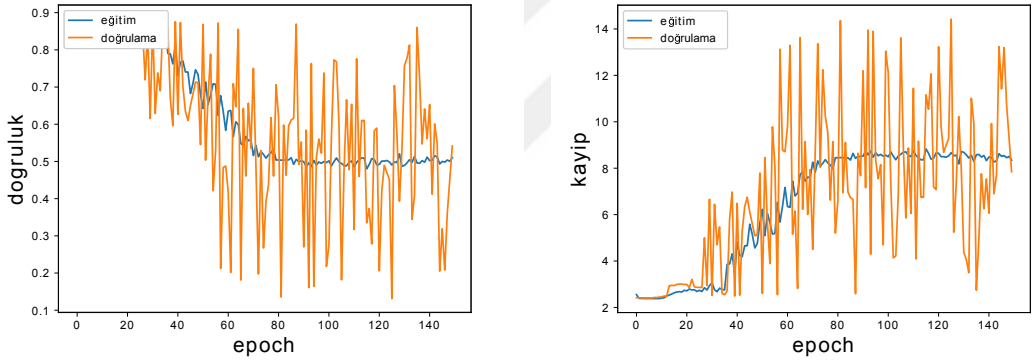
	N	P
N	1217	43
P	132	30

Tanh, *sigmoid*, *softplus* *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.18 - 5.22' de verilmiştir.



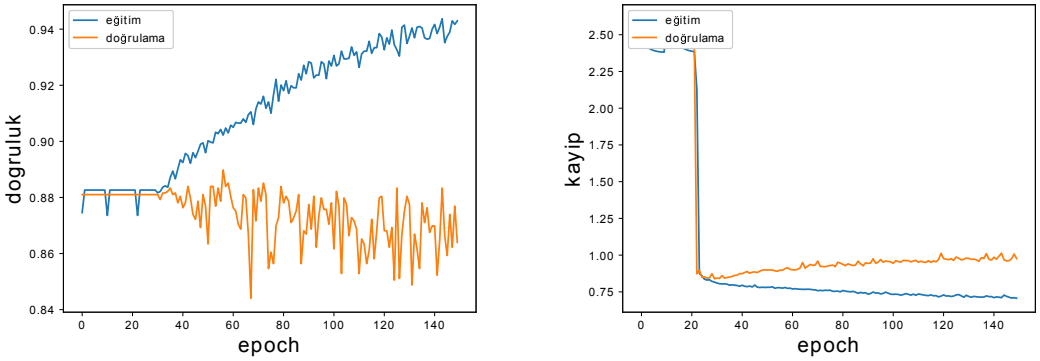
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.18: Spyware-tanh modeli doğruluk-kayıp grafikleri



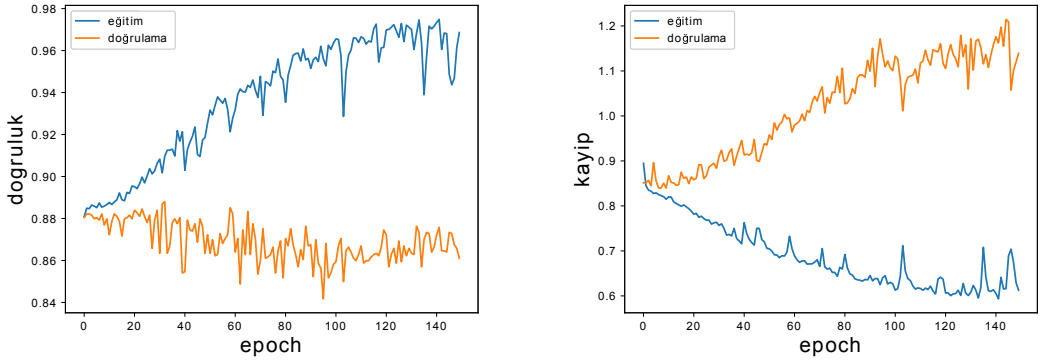
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.19: Spyware-softplus modeli doğruluk-kayıp grafikleri



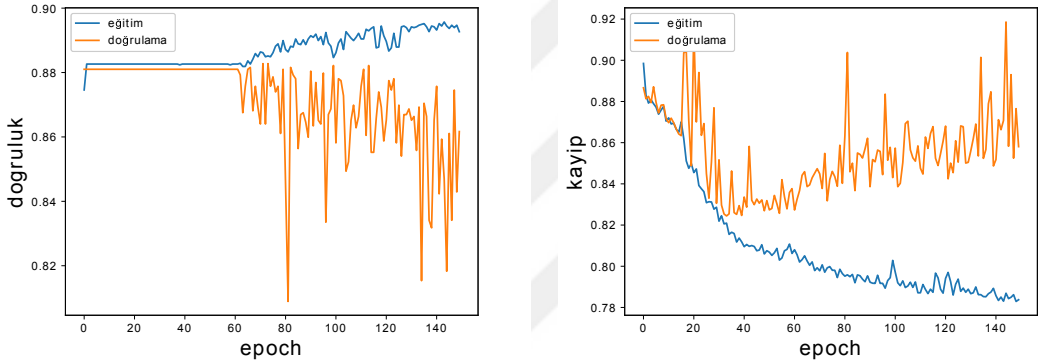
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.20: Spyware-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

ŞEKIL 5.21: Spyware-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

ŞEKIL 5.22: Spyware-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *softsign* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %87.6 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: softsign
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.6 Trojan Zararlı Yazılım Sınıfı için Sonuçlar

Trojan zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Trojan sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Trojan zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.12' de gösterilmektedir.

Tablo 5.12: Trojan Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.832	0.859	0.833	0.859	0.835	0.860	0.859
Hassasiyet	0.35	0	0.34	0	0.36	0.60	0
Anımsama	0.23	0	0.19	0	0.22	0.03	0
F1	0.27	0	0.24	0	0.27	0.06	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.13' de verilmiştir.

Tablo 5.13: Trojan Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1139	83
P	155	45

(b) sigmoid

	N	P
N	1147	75
P	162	38

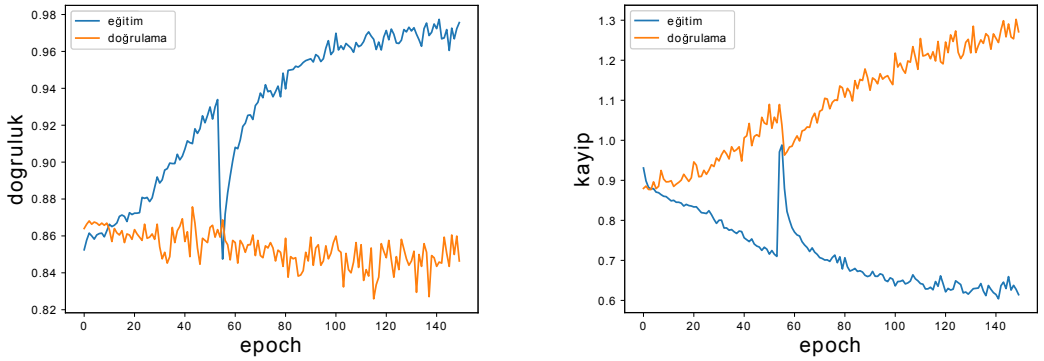
(c) softsign

	N	P
N	1144	78
P	156	44

(d) softmax

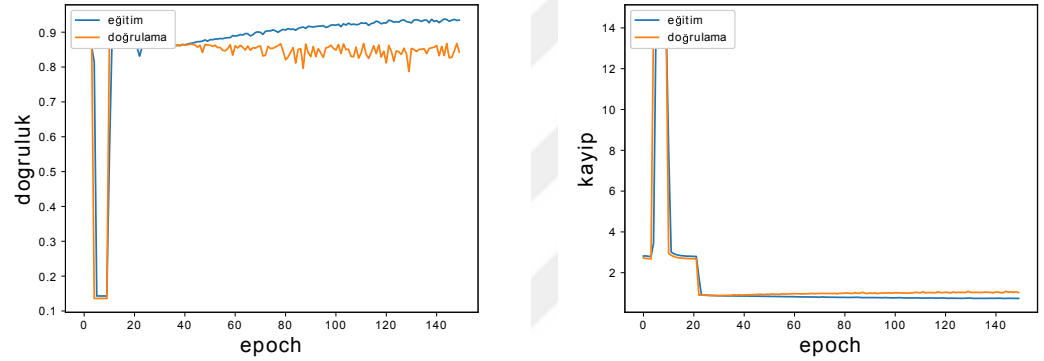
	N	P
N	1218	4
P	194	6

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.23 - 5.26' de verilmiştir.



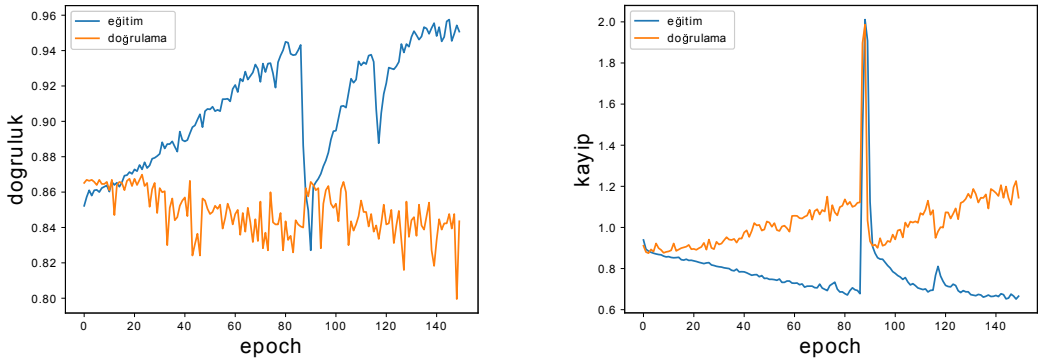
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.23: Trojan-tanh modeli doğruluk-kayıp grafikleri



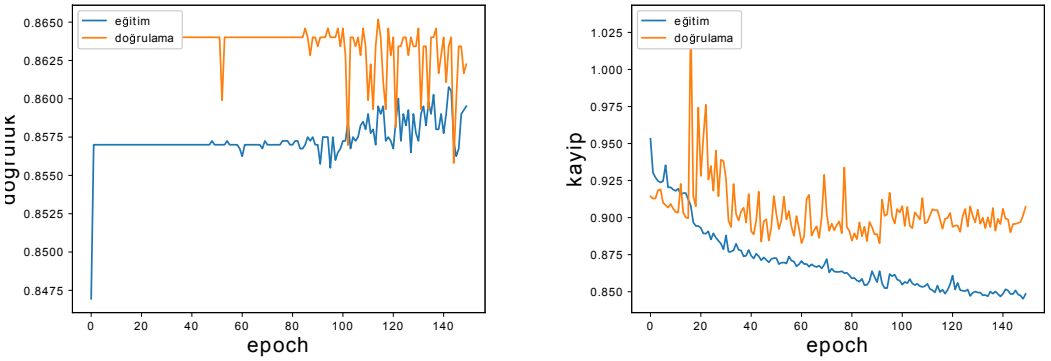
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.24: Trojan-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.25: Trojan-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.26: Trojan-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *softsign* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %83.5 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: softsign
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.7 Virus Zararlı Yazılım Sınıfı için Sonuçlar

Virus zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Virus sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Virus zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.14' de gösterilmektedir.

Tablo 5.14: Virus Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.927	0.862	0.895	0.862	0.925	0.905	0.862
Hassasiyet	0.76	0	0.60	0	0.75	0.68	0
Anımsama	0.70	0	0.74	0	0.69	0.57	0
F1	0.73	0	0.66	0	0.72	0.62	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.15' de verilmiştir.

Tablo 5.15: Virus Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1183	44
P	59	136

(b) sigmoid

	N	P
N	1129	98
P	51	3144

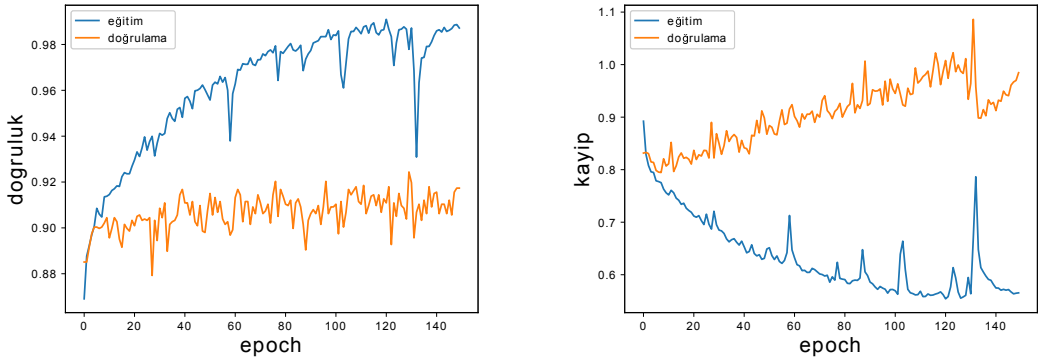
(c) softsign

	N	P
N	1182	45
P	61	134

(d) softmax

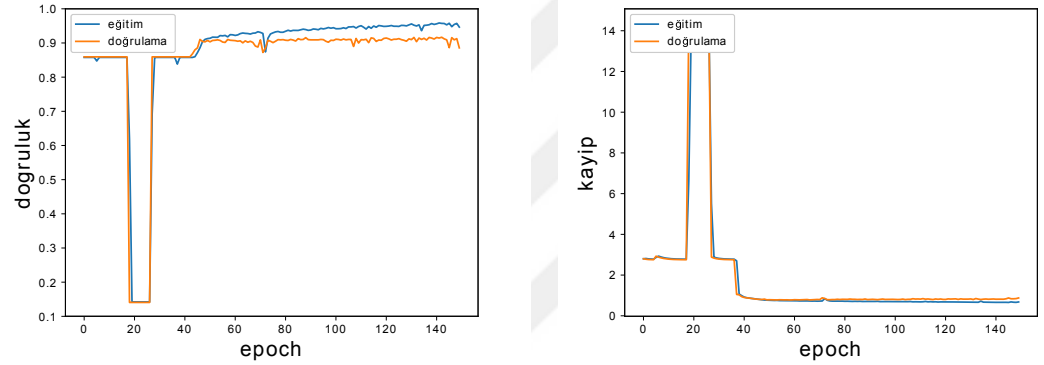
	N	P
N	1175	52
P	83	112

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.27 - 5.30' de verilmiştir.



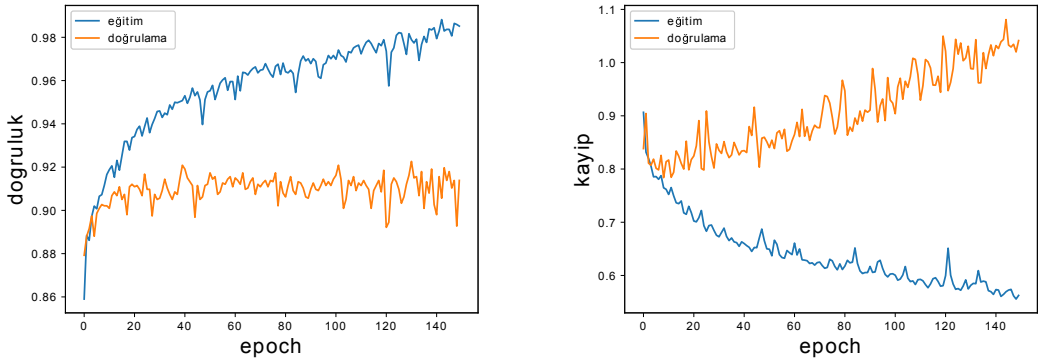
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.27: Virus-tanh modeli doğruluk-kayıp grafikleri



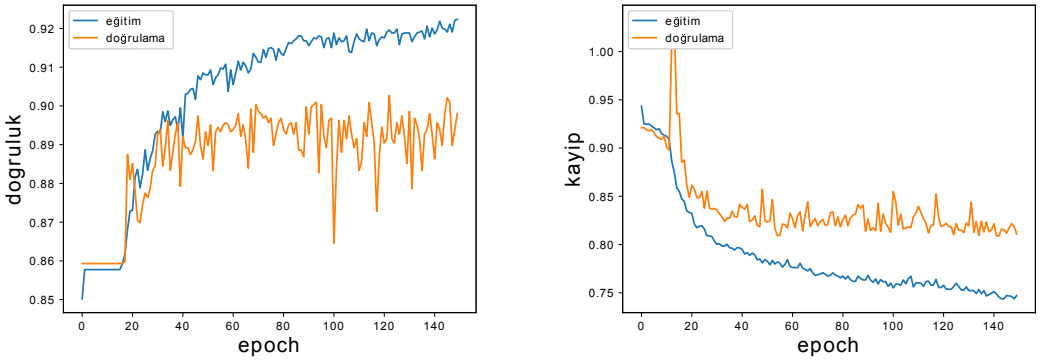
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.28: Virus-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.29: Virus-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.30: Virus-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *tanh* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %92.7 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: tanh
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.8 Worm Zararlı Yazılım Sınıfı için Sonuçlar

Worm zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Worm sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Worm zararlı yazılımları için 7 farklı aktivasyon fonksiyonu kullanılarak oluşturulan sınıflandırma modellerinin analiz sonuçları detayılı bir şekilde Tablo 5.16' de gösterilmektedir.

Tablo 5.16: Worm Sınıfı için Analiz Sonuçları

	tanh	relu	sigmoid	softplus	softsign	softmax	linear
Doğruluk	0.841	0.851	0.862	0.851	0.865	0.849	0.851
Hassasiyet	0.46	0	0.55	0	0.57	0.47	0
Anımsama	0.44	0	0.36	0	0.37	0.12	0
F1	0.45	0	0.44	0	0.45	0.20	0
Devir(epoch)	150	150	150	150	150	150	150

Herbir analiz modeli sonuçları incelendiğinde, en başarılı modellerin, *tanh*, *sigmoid*, *softsign* ve *softmax* aktivasyon fonksiyonları kullanılarak oluşturulan analiz modelleri oldukları gözükmektedir.

Bu fonksiyonlar kullanılarak elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.17' de verilmiştir.

Tablo 5.17: Worm Sınıfı Analiz Hata Matrisleri

(a) tanh

	N	P
N	1105	106
P	119	92

(b) sigmoid

	N	P
N	1150	61
P	135	76

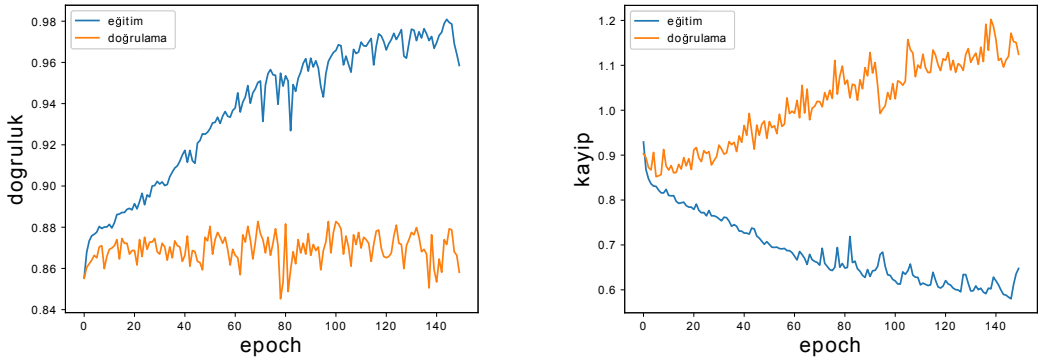
(c) softsign

	N	P
N	1153	58
P	133	78

(d) softmax

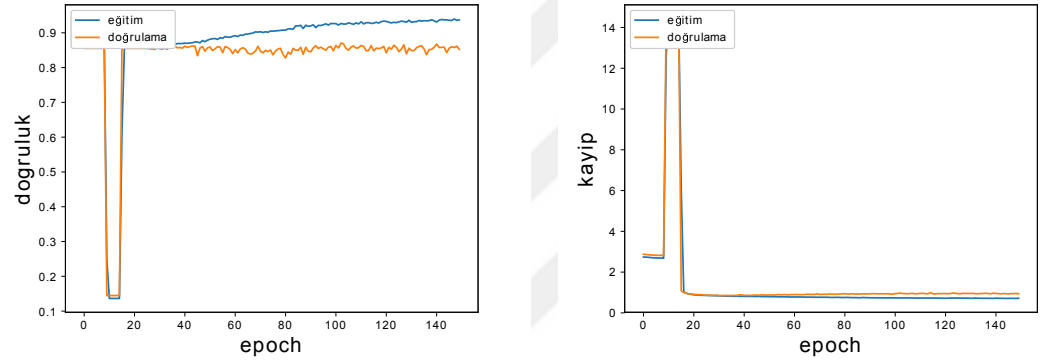
	N	P
N	1182	29
P	185	26

Tanh, *sigmoid*, *softsign* ve *softmax* aktivasyon aktivasyon fonksiyonları kullanarak oluşturulan modellerin doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.31 - 5.34' de verilmiştir.



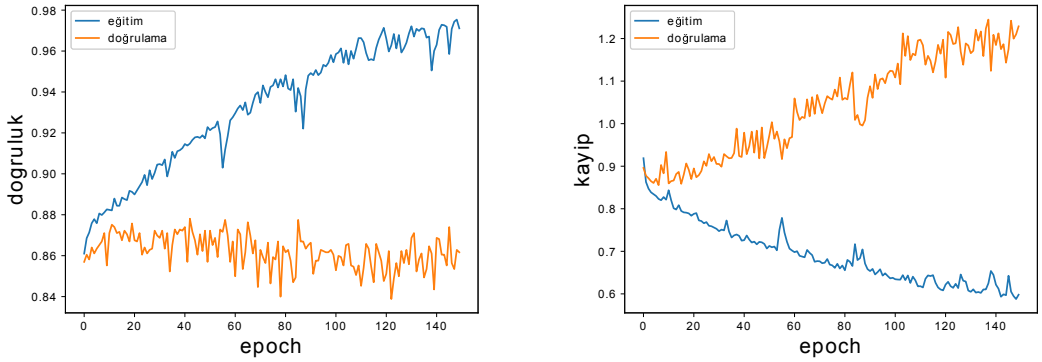
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.31: Worm-tanh modeli doğruluk-kayıp grafikleri



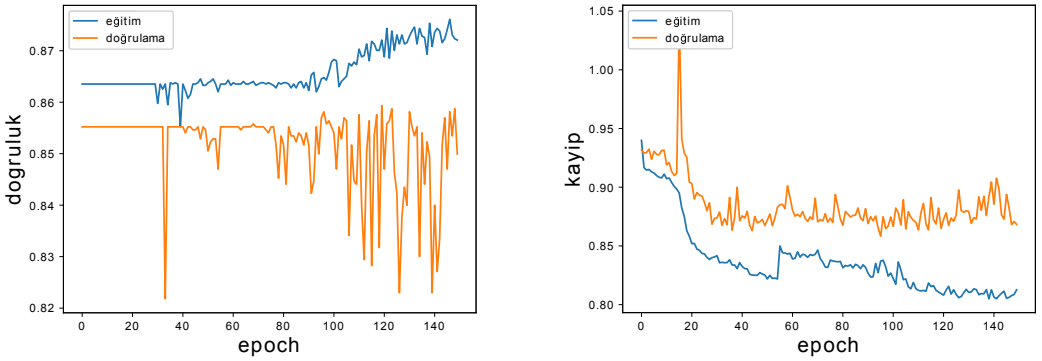
(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.32: Worm-sigmoid modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.33: Worm-softsign modeli doğruluk-kayıp grafikleri



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.34: Worm-softmax modeli doğruluk-kayıp grafikleri

Başarılı sonuç verdiği kabul edilen analiz çalışmaları içerisinde de en verimli sonuçların *softsign* fonksiyonu kullanılarak oluşturulan analiz modelinden elde edildiği gözükmektedir. Bu analiz modeli ile %86.5 değerinde bir doğruluk oranı elde edilmiştir Bu model oluşturulurken kullanılan parametreler;

- optimizer: adam
- layers: 342(Input) - 342,16 - 342,100 - 34200 - 2(Output)
- dropout: 0.2
- kernel_initializer: glorot_uniform
- activation: softsign
- epochs: 150
- Output Activation: softmax
- Loss: binary_crossentropy

5.2.1.9 Özet Sonuçlar

Tablo 5.18 incelendiğinde Adware, Backdoor ve Virus zararlı yazılım türlerinin eğitiminde *tanh* fonksiyonu, Downloader, Dropper, Spyware, Trojan ve Worm türlerinin eğitiminde ise *softsign* fonksiyonu kullanılarak en iyi modellerin elde edildiği gözükmektedir. Herbir tür için elde edilen doğruluk değerlerinin 0.835 ile 0.985 arasında olduğu gözlemlenmektedir.

Tablo 5.18: Siniflandirma Analiz Sonuçlari

	Adware	Backdoor	Downloader	Dropper	Spyware	Trojan	Virus	Worm
Doğruluk	0.985	0.877	0.909	0.877	0.876	0.835	0.927	0.865
Hassasiyet	0.90	0.60	0.72	0.52	0.46	0.36	0.76	0.57
Anımsama	0.77	0.40	0.57	0.44	0.42	0.22	0.70	0.37
F1	0.83	0.48	0.64	0.48	0.44	0.27	0.73	0.45
Fonksiyon	tanh	tanh	softsign	softsign	softsign	softsign	tanh	softsign
Devir	150	150	150	150	150	150	150	150

5.2.2 İkili Geleneksel Yöntemler Analiz Sonuçları

Bu çalışmamızda geleneksel yöntemlerden kNN(k-Nearest Neighborhood), DT(Decision Tree) ve SVM(Support Vector Machine) algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur.

5.2.2.1 Adware Zararlı Yazılım Sınıfı için Sonuçlar

Adware zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Adware sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Adware zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılıp sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.19' de gösterilmektedir.

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.20' de verilmiştir.

Tablo 5.19: Adware Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.976	0.954	0.979	0.950
Hassasiyet	0.89	0.53	1.00	0
Anımsama	0.59	0.80	0.59	0
F1	0.71	0.64	0.74	0

Tablo 5.20: Adware Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1346	5
P	29	42

(b) DT

	N	P
N	1301	50
P	14	57

(c) SVM(rbf)

	N	P
N	1351	0
P	29	42

(d) SVM(sigmoid)

	N	P
N	1351	0
P	71	0

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %97.6, DT algoritması kullanılarak %95.4 ve SVM algoritması kullanılarak %97.9 başarı elde edilmiştir.

5.2.2.2 Backdore Zararlı Yazılım Sınıfı için Sonuçlar

Backdore zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Backdore sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Backdore zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılarak sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.21' de gösterilmektedir.

Tablo 5.21: Backdore Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.834	0.824	0.875	0.864
Hassasiyet	0.38	0.39	0.91	0.87
Anımsama	0.26	0.41	0.14	0.06
F1	0.31	0.40	0.25	0.12

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.22‘ de verilmiştir.

Tablo 5.22: Backdore Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1135	84
P	151	52

(b) DT

	N	P
N	1089	130
P	119	84

(c) SVM(rbf)

	N	P
N	1216	3
P	174	29

(d) SVM(sigmoid)

	N	P
N	1217	2
P	190	13

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %83.4, DT algoritması kullanılarak %82.4 ve SVM algoritması kullanılarak %87.5 başarı elde edilmiştir.

5.2.2.3 Downloader Zararlı Yazılım Sınıfı için Sonuçlar

Downloader zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Downloader sınıfı için ’1’, diğer sınıftaki zararlı yazılımlar için ’0’ etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Downloader zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* ’kernal’ modunda çalıştırılarak sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.23’ de gösterilmektedir.

Tablo 5.23: Downloader Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.870	0.839	0.876	0.860
Hassasiyet	0.62	0.43	1.00	0
Anımsama	0.20	0.48	0.12	0
F1	0.30	0.45	0.22	0

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.24‘ de verilmiştir.

Tablo 5.24: Downloader Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1198	25
P	159	40

(b) DT

	N	P
N	1099	129
P	104	95

(c) SVM(rbf)

	N	P
N	1223	0
P	175	24

(d) SVM(sigmoid)

	N	P
N	1223	0
P	199	0

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %87, DT algoritması kullanılarak %83.9 ve SVM algoritması kullanılarak %87.6 başarı elde edilmiştir.

5.2.2.4 Dropper Zararlı Yazılım Sınıfı için Sonuçlar

Dropper zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Dropper sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Dropper zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılarak sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.25' de gösterilmektedir.

Tablo 5.25: Dropper Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.819	0.815	0.881	0.872
Hassasiyet	0.19	0.32	0.84	0
Anımsama	0.13	0.38	0.09	0
F1	0.16	0.35	0.16	0

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.26' de verilmiştir.

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %81.9, DT algoritması kullanılarak %81.5 ve SVM algoritması kullanılarak %88.1 başarı elde edilmiştir.

Tablo 5.26: Dropper Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1141	100
P	157	24

(b) DT

	N	P
N	1091	150
P	112	69

(c) SVM(rbf)

	N	P
N	1238	3
P	165	16

(d) SVM(sigmoid)

	N	P
N	1241	0
P	181	0

5.2.2.5 Spyware Zararlı Yazılım Sınıfı için Sonuçlar

Spyware zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Spyware sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Spyware zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur.SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılıp sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.27' de gösterilmektedir.

Tablo 5.27: Spyware Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.879	0.824	0.899	0.886
Hassasiyet	0.45	0.30	0.79	0
Anımsama	0.26	0.41	0.16	0
F1	0.33	0.35	0.27	0

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.28' de verilmiştir.

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %87.9, DT algoritması kullanılarak %82.4 ve SVM algoritması kullanılarak %89.9 başarı elde edilmiştir.

Tablo 5.28: Spyware Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1208	52
P	120	42

(b) DT

	N	P
N	1107	153
P	96	66

(c) SVM(rbf)

	N	P
N	1253	7
P	136	26

(d) SVM(sigmoid)

	N	P
N	1260	0
P	162	0

5.2.2.6 Trojan Zararlı Yazılım Sınıfı için Sonuçlar

Trojan zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Trojan sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Trojan zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılıp sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.29' de gösterilmektedir.

Tablo 5.29: Trojan Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.841	0.786	0.867	0.859
Hassasiyet	0.33	0.26	0.87	0
Anımsama	0.12	0.28	0.07	0
F1	0.18	0.27	0.12	0

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.30' de verilmiştir.

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %84.1, DT algoritması kullanılarak %78.6 ve SVM algoritması kullanılarak %86.7 başarı elde edilmiştir.

Tablo 5.30: Trojan Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1173	49
P	176	24

(b) DT

	N	P
N	1061	161
P	143	57

(c) SVM(rbf)

	N	P
N	1220	2
P	187	13

(d) SVM(sigmoid)

	N	P
N	1222	0
P	200	0

5.2.2.7 Virus Zararlı Yazılım Sınıfı için Sonuçlar

Virus zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Virus sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Virus zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılıp sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.31' de gösterilmektedir.

Tablo 5.31: Virus Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.859	0.856	0.888	0.862
Hassasiyet	0.47	0.48	1.00	0
Anımsama	0.21	0.52	0.18	0
F1	0.29	0.50	0.31	0

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.32' de verilmiştir.

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %85.9, DT algoritması kullanılarak %85.6 ve SVM algoritması kullanılarak %88.8 başarı elde edilmiştir.

Tablo 5.32: Virus Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1181	46
P	154	41

(b) DT

	N	P
N	1116	111
P	93	102

(c) SVM(rbf)

	N	P
N	1227	0
P	159	36

(d) SVM(sigmoid)

	N	P
N	1227	0
P	195	0

5.2.2.8 Worm Zararlı Yazılım Sınıfı için Sonuçlar

Worm zararlı yazılım sınıfı için model eğitimi öncesinde, veri kümesindeki her bir veriye Worm sınıfı için '1', diğer sınıftaki zararlı yazılımlar için '0' etiketi atanmıştır. Bu etiketler model oluşturulurken kullanılması sayesinde, oluşturulan modeller sadece ilgili zararlı yazılım sınıfı için sınıflandırma yapabilmektedir.

Worm zararlı yazılımları için kNN, DT ve SVM algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuştur. SVM algoritması *rbf* ve *sigmoid* 'kernal' modunda çalıştırılıp sonuç elde edilmeye çalışılmıştır. Elde edilen analiz sonuçları detayılı bir şekilde Tablo 5.33' de gösterilmektedir.

Tablo 5.33: Worm Sınıfı için Analiz Sonuçları

	kNN	DT	SVM(rbf)	SVM(sigmoid)
Doğruluk	0.823	0.825	0.875	0.851
Hassasiyet	0.36	0.41	0.93	0
Anımsama	0.23	0.40	0.18	0
F1	0.28	0.40	0.29	0

Başarılı sonuçlanan eğitimler sonucu elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.34' de verilmiştir.

Analiz sonuçları incelendiğinde; kNN algoritması kullanılarak %82.3, DT algoritması kullanılarak %82.5 ve SVM algoritması kullanılarak %87.5 başarı elde edilmiştir.

Tablo 5.34: Worm Sınıfı Analiz Hata Matrisleri

(a) kNN

	N	P
N	1122	89
P	162	49

(b) DT

	N	P
N	1090	121
P	127	84

(c) SVM(rbf)

	N	P
N	1208	3
P	174	37

(d) SVM(sigmoid)

	N	P
N	1211	0
P	211	0

5.2.3 Derin Öğrenme ve Geleneksel Yöntemler Analiz Sonuçları

Geleneksel yöntemler ile aldığımız sonuçların doğruluk oranları ile derin öğrenme yöntemi ile aldığımız doğruluk oranları birbirine yakın gözüksede, sınıf dağılımlarının yakın olmaması nedeniyle F1 değerleri daha anlamlı bilgiler vermektedir. F1 değeri, 'hassasiyet' ve 'anımsama' değerleri arasındaki dengeyi göstermektedir. Bu nedenle derin öğrenme yöntemi ile elde ettiğimiz F1 değerleri, geleneksel yöntemler ile elde edilen F1 değerlerinden daha iyi gözükmektedir. Bu durum, derin öğrenme yönteminin zararlı yazılım davranışlarını analiz etmede daha iyi olduğu sonucunu göstermektedir.

Tablo 5.35: Derin Öğrenme ve Geleneksel Yöntemler Analiz Sonuçları

	Adware	Backdoor	Downloader	Dropper	Spyware	Trojan	Virus	Worm
LSTM								
Doğruluk	0.985	0.877	0.909	0.877	0.876	0.835	0.927	0.865
Hassasiyet	0.90	0.60	0.72	0.52	0.46	0.36	0.76	0.57
Anımsama	0.77	0.40	0.57	0.44	0.42	0.22	0.70	0.37
F1	0.83	0.48	0.64	0.48	0.44	0.27	0.73	0.45
Fonksiyon	tanh	tanh	softsign	softsign	softsign	softsign	tanh	softsign
kNN								
Doğruluk	0.976	0.834	0.870	0.819	0.879	0.841	0.859	0.823
Hassasiyet	0.89	0.38	0.62	0.19	0.45	0.33	0.47	0.36
Anımsama	0.59	0.26	0.20	0.13	0.26	0.12	0.21	0.23
F1	0.71	0.31	0.30	0.16	0.33	0.18	0.29	0.28
DT								
Doğruluk	0.954	0.824	0.839	0.815	0.824	0.786	0.856	0.825
Hassasiyet	0.53	0.39	0.43	0.32	0.30	0.26	0.48	0.41
Anımsama	0.80	0.41	0.48	0.38	0.41	0.28	0.52	0.40
F1	0.64	0.40	0.45	0.35	0.35	0.27	0.50	0.40
SVM(rbf)								
Doğruluk	0.979	0.875	0.876	0.881	0.899	0.867	0.888	0.875
Hassasiyet	1.00	0.91	1.00	0.84	0.79	0.87	1.00	0.93
Anımsama	0.59	0.14	0.12	0.09	0.16	0.07	0.18	0.18
F1	0.74	0.25	0.22	0.16	0.27	0.12	0.31	0.29
SVM(sigmoid)								
Doğruluk	0.950	0.864	0.860	0.872	0.886	0.859	0.862	0.851
Hassasiyet	0	0.87	0	0	0	0	0	0
Anımsama	0	0.06	0	0	0	0	0	0
F1	0	0.12	0	0	0	0	0	0

5.3 Çoklu Sınıflandırma Analiz Sonuçları

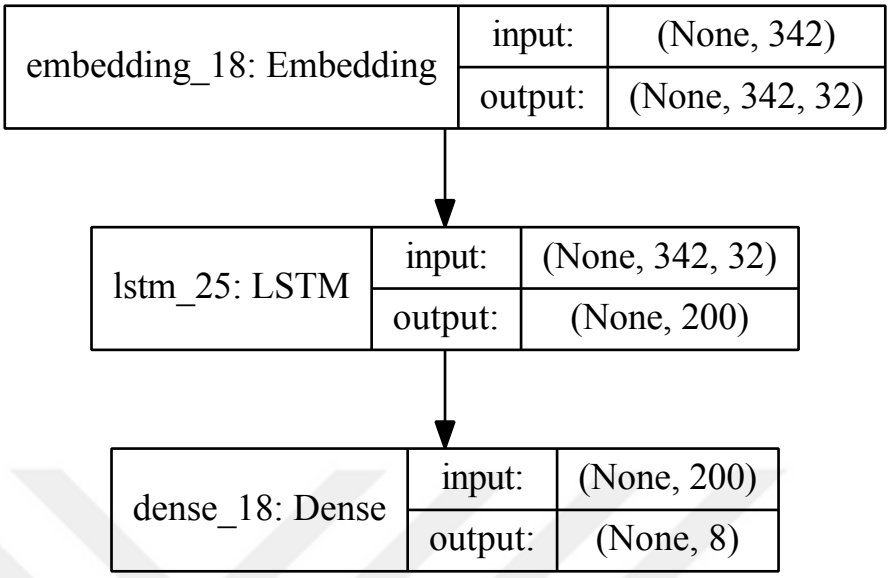
Veri kümemizde 8 farklı zararlı yazılım türü için ikili sonuç veren ayrı ayrı modellerin yanı sıra, 8 farklı sonuç verebilen sınıflandırma modelleri de oluşturulmuştur. Özellikle farklı parametreler kullanılarak en başarılı modeller elde edilmeye çalışılmıştır.

Modellerin karşılaştırılması sırasında analiz sonuçlarından F1 değeri kullanılacaktır. F1, hassasiyet, anımsama ve bu değerlerin ortalama değerlerini hesaplamak için sklearn.metrics kütüphanesindeki classification_report metodu kullanılmıştır. Bu metodun ürettiği ortalama F1, hassasiyet ve anımsama değerlerini, hesaplarken veri kümesindeki sınıf ağırlıklarınıda hesaba katmaktadır.

5.3.1 Tek Katmanlı LSTM Analiz Sonuçları

8 farklı zararlı yazılım türünü sınıflandırabilecek tek katmanlı LSTM modelleri oluşturulmuştur. Bu modeller 0-7 arasında bir çıktı üretmektedir. Bu değerler zararlı yazılım türlerini temsil etmektedir.

Tek katmanlı LSTM modelleri farklı parametreler ile oluşturulmasına rağmen genelde Şekil 5.35‘ deki katman yapısında inşa edilmiştir.



Şekil 5.35: Tek Katmanlı LSTM Sınıflandırma Model Yapısı

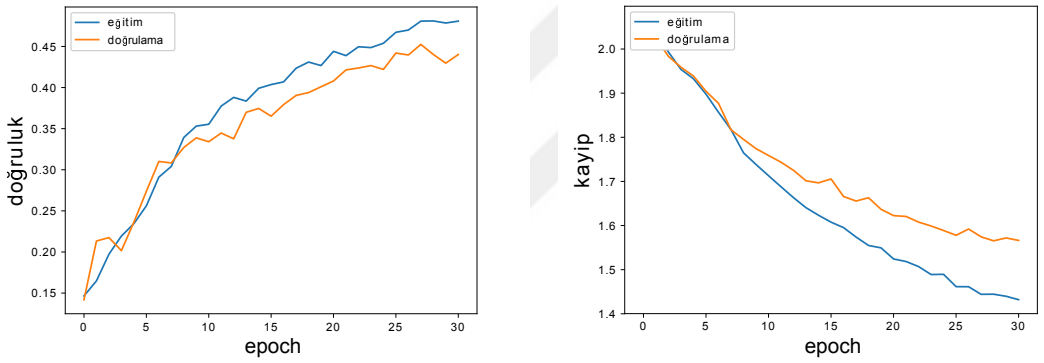
Modeller oluşturulurken bir çok farklı parametre kullanılarak en iyi sınıflandırma modelinin elde edilmesi hedeflenmiştir. Genellikle model inşa ve eğitimi sırasında: Embedding katmanı için units(5 - 300); LSTM katmanı için unit(1 - 300), activation(*tanh*, *relu*, *sigmoid*, *softplus*, *softsign*, *softmax*, *linear*), recurrent_activation(*tanh*, *relu*, *hard_sigmoid*, *softplus*, *softsign*, *softmax*, *linear*), kernel_initializer(*random_uniform*, *glorot_uniform*, *lecun_uniform*, *uniform*), dropout(0.1 - 0.9), recurrent_dropout(0.1 - 0.9); model derleme için loss(*categorical_crossentropy*, *sparse_categorical_crossentropy*, *kullback_leibler_divergence*), optimizer(*adam*, *adadelta*, *adamax*, *nadam*); model eğitimi için epochs(10 - 200), batch_size(25 - 250) gibi parametrelerin değişik kombinasyonları kullanılmıştır.

En iyi sonuç elde edilen modeli oluşturulurken kullanılan parametre değerleri aşağıdadır.

- Embedding units: 32
- units: 200
- activation: sigmoid
- recurrent_activation: hard_sigmoid
- kernel_initializer: glorot_uniform
- dropout: 0.2
- recurrent_dropout: 0.2

- Dense units: 8
- Dense activation: softmax
- loss_func: categorical_crossentropy
- optimizer: adam
- epochs: 150
- batch_size: 50

İlgili parametreler kullanılarak oluşturulan modelin eğitim geçmişi, doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.36‘ de verilmiştir. Grafikler incelendiğinde model eğitim sürecinin 30. devirde(epoch) durduğu gözükmektedir. Bu durumun sebebi, modelin aşırı uyum(overfitting) sınırına gelmesindendir. Model eğitimleri sırasında EarlyStopping parametresi kullanarak modelin aşırı uyum durumuna ulaşmadan eğitimin sonlandırılması sağlanmıştır.



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.36: Tek Katmanlı LSTM Model doğruluk-kayıp grafikleri

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.36‘ de verilmiştir.

Tablo 5.36: Tek Katmanlı LSTM Model Analiz Hata Matrisi

49	0	8	4	1	5	4	0
1	92	6	15	22	44	6	17
6	10	124	22	6	22	4	5
3	12	6	80	21	53	6	0
2	22	7	16	41	48	17	9
3	23	13	25	25	90	15	6
7	6	1	19	8	20	131	3
2	30	6	17	19	62	20	55

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen analiz sonuçları Tablo 5.37' de gösterilmektedir.

Tablo 5.37: Tek Katmanlı LSTM Model Sınıflandırma Analiz Sonuçları

	Hassasiyet	Anımsama	F1
Adware	0.67	0.69	0.68
Backdoor	0.47	0.45	0.46
Downloader	0.73	0.62	0.67
Dropper	0.40	0.44	0.42
Spyware	0.29	0.25	0.27
Trojan	0.26	0.45	0.33
Virus	0.65	0.67	0.66
Worm	0.58	0.26	0.36

Farklı modellerin karşılaştırılmasında kullanabileceğimiz, Tablo 5.37' da gösterilen verilen ortalama bilgileri Tablo 5.38' da gösterilmektedir.

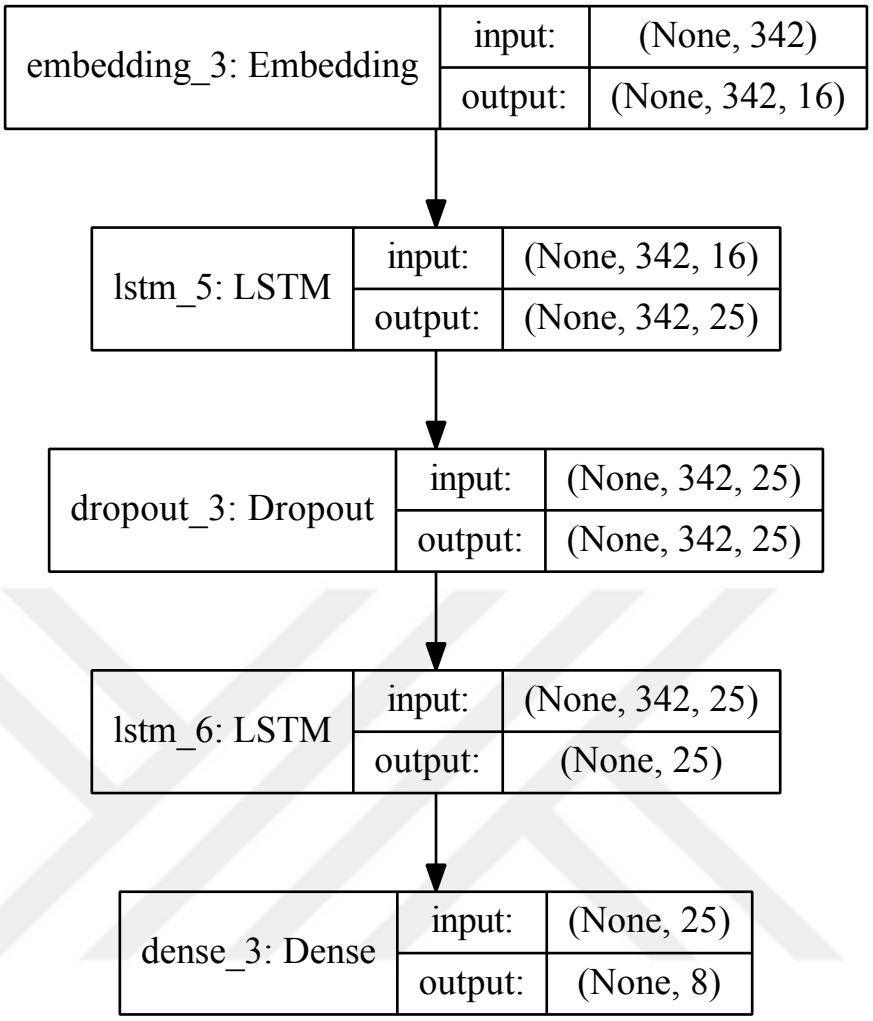
Tablo 5.38: Tek Katmanlı LSTM Model Analiz Sonuç Ortalaması

	Hassasiyet	Anımsama	F1
Ortalama	0.50	0.47	0.47

5.3.2 İki Katmanlı LSTM Analiz Sonuçları

8 farklı zararlı yazılım türünü sınıflandırabilecek iki katmanlı LSTM modelleri oluşturulmuştur. Bu modeller 0-7 arasında bir çıktı üretmektedir. Bu değerler zararlı yazılım türlerini temsil etmektedir.

İki katmanlı LSTM modelleri farklı parametreler ile oluşturulmasına rağmen genelde Şekil 5.37‘ deki katman yapısında inşa edilmiştir.



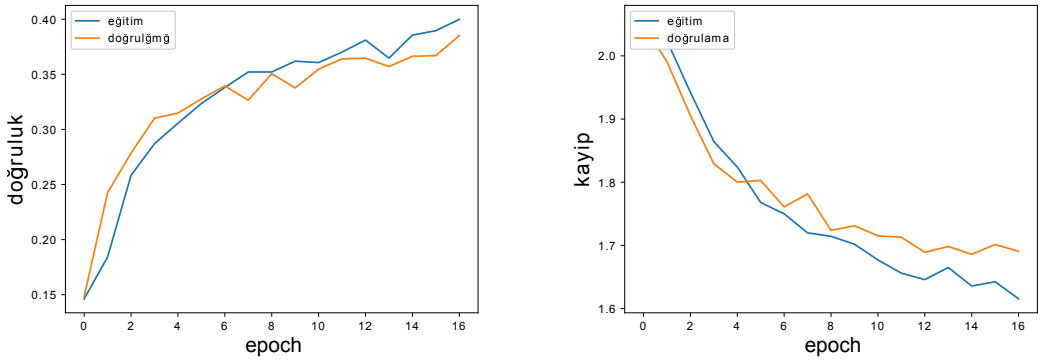
Şekil 5.37: Sınıflandırma Model Yapısı

Modeller oluşturulurken bir çok farklı parametre kullanılarak en iyi sınıflandırma modelinin elde edilmesi hedeflenmiştir. Genellikle model inşa ve eğitimi sırasında: Embedding katmanı için units(5 - 300); LSTM katmanları için unit(1 - 300), activation(*tanh*, *relu*, *sigmoid*, *softplus*, *softsign*, *softmax*, *linear*), recurrent_activation(*tanh*, *relu*, *hard_sigmoid*, *softplus*, *softsign*, *softmax*, *linear*), kernel_initializer(*random_uniform*, *glorot_uniform*, *lecun_uniform*, *uniform*), dropout(0.1 - 0.9), recurrent_dropout(0.1 - 0.9); Dropout katmanı için rate(0.1 - 0.9); Dense katmanı için kernel_regularizer(*l1*, *l2*, *l1_l2*), activity_regularizer(*l1*, *l2*, *l1_l2*); model derleme için loss(*categorical_crossentropy*, *sparse_categorical_crossentropy*, *kullback_leibler_divergence*), optimizer(*adam*, *adadelta*, *adamax*, *nadam*); model eğitimi için epochs(10 - 200), batch_size(25 - 250) gibi parametrelerin değişik kombinasyonları kullanılmıştır.

En iyi sonuç elde edilen modeli oluşturulurken kullanılan parametre değerleri aşağıdadır.

- Embedding units: 16
- units: 25
- LSTM-1 activation: softsign
- LSTM-1 recurrent_activation: hard_sigmoid
- LSTM-1 kernel_initializer: glorot_uniform
- rate: 0.2
- LSTM-2 activation: softsign
- LSTM-2 recurrent_activation: hard_sigmoid
- LSTM-2 kernel_initializer: glorot_uniform
- LSTM-2 recurrent_dropout: 0.2
- Dense units: 8
- Dense activation: softmax
- Dense kernel_regularizer: regularizers.l2(0.01)
- Dense activity_regularizer: regularizers.l1(0.01)
- loss_func: categorical_crossentropy
- optimizer: adam
- epochs: 150
- batch_size: 50

İlgili parametreler kullanılarak oluşturulan modelin eğitim geçmişi, doğruluk (accuracy) ve kayıp (loss) grafikleri Şekil 5.38‘ de verilmiştir. Grafikler incelendiğinde model eğitim sürecinin 16. devirde(epoch) durduğu gözükmektedir. Bu durumun sebebi, modelin aşırı uyum(overfitting) sınırına gelmesindendir. Model eğitimleri sırasında EarlyStopping parametresi kullanarak modelin aşırı uyum durumuna ulaşmadan eğitimin sonlandırılması sağlanmıştır.



(A) Doğruluk(accurancy) grafiği (B) Kayıp(loss) grafiği

Şekil 5.38: İki Katmanlı LSTM Model doğruluk-kayıp grafikleri

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.39‘ de verilmiştir.

Tablo 5.39: İki Katmanlı LSTM Model Analiz Hata Matrisi

53	1	2	8	4	0	2	1
2	94	7	33	21	24	7	15
4	15	122	29	10	7	8	4
12	19	14	75	19	21	8	13
4	38	11	38	31	14	13	13
4	35	18	46	29	29	25	14
6	10	7	18	8	7	133	6
2	56	11	31	13	13	43	42

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen analiz sonuçları Tablo 5.40’ de gösterilmektedir.

Tablo 5.40: İki Katmanlı LSTM Model Sınıflandırma Analiz Sonuçları

	Hassasiyet	Anımsama	F1
Adware	0.61	0.75	0.67
Backdoor	0.35	0.46	0.40
Downloader	0.64	0.61	0.62
Dropper	0.27	0.41	0.33
Spyware	0.23	0.19	0.21
Trojan	0.25	0.14	0.18
Virus	0.56	0.68	0.61
Worm	0.39	0.20	0.26

Farklı modellerin karşılaştırılmasında kullanabileceğimiz, Tablo 5.40’ da gösterilen verilen ortalama bilgileri Tablo 5.41’ da gösterilmektedir.

Tablo 5.41: İki Katmanlı LSTM Model Analiz Sonuç Ortalaması

	Hassasiyet	Anımsama	F1
Ortalama	0.40	0.41	0.39

5.3.3 DT Analiz Sonuçları

DT algoritması kullanılarak bir sınıflandırma modeli oluşturulmuştur. Bu model 0-7 arasında bir çıktı üretmektedir. Bu değerler zararlı yazılım türlerini temsil etmektedir.

Modeller oluşturulurken random_state (1 - 10), max_depth(1 - 30), class_weight(None veya ağırlıklar) parametreleri kullanılarak en iyi sınıflandırma modelinin elde edilmesi hedeflenmiştir.

En iyi sonuç elde edilen modeli oluşturulurken kullanılan parametreler aşağıdadır.

- random_state: 3
- max_depth: None
- class_weight: None

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.42‘ de verilmiştir.

Tablo 5.42: DT Model Analiz Hata Matrisi

56	0	4	1	3	3	4	0
3	96	9	14	21	25	14	21
7	12	87	26	14	16	21	16
3	20	18	58	28	20	15	19
5	15	9	15	69	21	14	14
9	33	21	17	19	43	24	34
3	20	14	14	14	15	94	21
2	12	23	17	22	34	23	78

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen analiz sonuçları Tablo 5.43’ de gösterilmektedir.

Farklı modellerin karşılaştırılmasında kullanabileceğimiz, Tablo 5.43’ da gösterilen verilen ortalama bilgileri Tablo 5.44’ da gösterilmektedir.

Tablo 5.43: DT Model Sınıflandırma Analiz Sonuçları

	Hassasiyet	Anımsama	F1
Adware	0.64	0.79	0.70
Backdoor	0.46	0.47	0.47
Downloader	0.47	0.44	0.45
Dropper	0.36	0.32	0.34
Spyware	0.36	0.43	0.39
Trojan	0.24	0.21	0.23
Virus	0.45	0.48	0.47
Worm	0.38	0.37	0.38

Tablo 5.44: DT Model Analiz Sonuç Ortalaması

	Hassasiyet	Anımsama	F1
Ortalama	0.40	0.41	0.40

5.3.4 kNN Analiz Sonuçları

kNN algoritması kullanılarak bir sınıflandırma modeli oluşturulmuştur. Bu model 0-7 arasında bir çıktı üretmektedir. Bu değerler zararlı yazılım türlerini temsil etmektedir.

Modeller oluşturulurken n_neighbors(1 - 30), weights(*uniform*, *distance*), algorithm(*auto*, *ball_tree*, *kd_tree*, *brute*), leaf_size(1 - 100), p(1 - 50) parametreleri kullanılarak en iyi sınıflandırma modelinin elde edilmesi hedeflenmiştir.

En iyi sonuç elde edilen modeli oluşturulurken kullanılan parametreler aşağıdadır.

- n_neighbors: 3
- weights: uniform
- algorithm: brute
- leaf_size: 30
- p: 2

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.45‘ de verilmiştir.

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen analiz sonuçları Tablo 5.46’ de gösterilmektedir.

Farklı modellerin karşılaştırılmasında kullanabileceğimiz, Tablo 5.46’ da gösterilen verilen ortalama bilgileri Tablo 5.47’ da gösterilmektedir.

Tablo 5.45: kNN Model Analiz Hata Matrisi

52	2	3	3	0	1	3	7
5	101	17	28	9	14	9	20
7	36	72	29	7	12	13	23
8	34	23	48	20	16	12	20
6	27	10	19	64	13	7	16
6	48	19	26	26	33	9	33
3	42	28	21	11	15	64	11
3	48	23	24	23	20	11	59

Tablo 5.46: kNN Model Sınıflandırma Analiz Sonuçları

	Hassasiyet	Anımsama	F1
Adware	0.58	0.73	0.65
Backdoor	0.30	0.50	0.37
Downloader	0.37	0.36	0.37
Dropper	0.24	0.27	0.25
Spyware	0.40	0.40	0.40
Trojan	0.27	0.17	0.20
Virus	0.50	0.33	0.40
Worm	0.31	0.28	0.29

Tablo 5.47: kNN Model Analiz Sonuç Ortalaması

	Hassasiyet	Anımsama	F1
Ortalama	0.35	0.35	0.34

5.3.5 RF Analiz Sonuçları

RF algoritması kullanılarak bir sınıflandırma modeli oluşturulmuştur. Bu model 0-7 arasında bir çıktı üretmektedir. Bu değerler zararlı yazılım türlerini temsil etmektedir.

Modeller oluşturulurken n_estimators(1 - 500), max_depth(1 - 30), class_weight(None veya ağırlıklar), min_samples_split(1 - 50) parametreleri kullanılarak en iyi sınıflandırma modelinin elde edilmesi hedeflenmiştir.

En iyi sonuç elde edilen modeli oluşturulurken kullanılan parametreler aşağıdadır.

- n_estimators: 200
- max_depth: None
- min_samples_split: 15
- class_weight: 0: 2.3072240259740258, 1: 0.8905075187969925, 2: 0.88606608478803, 3: 1.0008802816901408, 4: 1.060634328358209, 5: 0.8871722846441947, 6: 0.8816687344913151, 7: 0.8995253164556962

RF algoritması ve diğer algoritmalar kullanılarak model oluşturma sırasında class_weight parametresi kullanılmıştır. Bu parametre değeri sklearn.utils.class_weight kütüphanesindeki compute_class_weight metodunu kullanılarak elde edilmiştir.

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.48‘ de verilmiştir.

Tablo 5.48: RF Model Analiz Hata Matrisi

54	1	7	3	2	0	1	3
8	96	14	15	25	13	10	22
3	11	124	21	10	9	12	9
8	8	25	66	25	25	20	4
6	14	12	15	65	19	13	18
8	28	24	14	26	62	18	20
5	8	16	25	10	14	111	6
4	18	18	16	18	24	27	86

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen analiz sonuçları Tablo 5.49’ de gösterilmektedir.

Tablo 5.49: RF Model Sınıflandırma Analiz Sonuçları

	Hassasiyet	Anımsama	F1
Adware	0.56	0.76	0.65
Backdoor	0.52	0.47	0.50
Downloader	0.52	0.62	0.56
Dropper	0.38	0.36	0.37
Spyware	0.36	0.40	0.38
Trojan	0.37	0.31	0.34
Virus	0.52	0.57	0.55
Worm	0.51	0.41	0.45

Farklı modellerin karşılaştırılmasında kullanabileceğimiz, Tablo 5.49’ da gösterilen verilen ortalama bilgileri Tablo 5.50’ da gösterilmektedir.

Tablo 5.50: RF Model Analiz Sonuç Ortalaması

	Hassasiyet	Anımsama	F1
Ortalama	0.46	0.47	0.46

5.3.6 SVM Analiz Sonuçları

SVM algoritması kullanılarak bir sınıflandırma modeli oluşturulmuştur. Bu model 0-7 arasında bir çıktı üretmektedir. Bu değerler zararlı yazılım türlerini temsil etmektedir.

Modeller oluşturulurken kernel(*linear*, *poly*, *rbf*, *sigmoid*, *precomputed*), c(0.1 - 100.0), gamma(1.0 - 20.0, *auto*), class_weight(None veya ağırlıklar) parametreleri kullanılarak en iyi sınıflandırma modelinin elde edilmesi hedeflenmiştir.

En iyi sonuç elde edilen modeli oluşturulurken kullanılan parametreler aşağıdadır.

- kernel: rbf
- c: 1.0
- gamma: auto
- class_weight: None

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen Hata Matris (Confusion Matrix) bilgileri Tablo 5.51‘ de verilmiştir.

Tablo 5.51: SVM Model Analiz Hata Matrisi

42	0	0	0	0	29	0	0
0	37	0	0	1	164	0	1
0	0	24	1	0	174	0	0
0	0	0	18	5	157	0	1
0	1	0	0	29	129	0	3
0	3	0	3	3	188	0	3
0	1	0	0	0	158	36	0
0	1	0	0	2	169	0	39

Eğitilen sınıflandırma modelinin test edilmesi sonucunda elde edilen analiz sonuçları Tablo 5.52’ de gösterilmektedir.

Tablo 5.52: SVM Model Sınıflandırma Analiz Sonuçları

	Hassasiyet	Anımsama	F1
Adware	1.00	0.59	0.74
Backdoor	0.86	0.18	0.30
Downloader	1.00	0.12	0.22
Dropper	0.82	0.10	0.18
Spyware	0.72	0.18	0.29
Trojan	0.16	0.94	0.27
Virus	1.00	0.18	0.31
Worm	0.83	0.18	0.30

Farklı modellerin karşılaştırılmasında kullanabileceğimiz, Tablo 5.52’ da gösterilen verilen ortalama bilgileri Tablo 5.53’ da gösterilmektedir.

Tablo 5.53: SVM Model Analiz Sonuç Ortalaması

	Hassasiyet	Anımsama	F1
Ortalama	0.78	0.29	0.29

5.3.7 Derin Öğrenme ve Geleneksel Yöntemler Çoklu Analiz Sonuçları

Farklı algoritmalar kullanılarak elde edilen çoklu sınıflansırma analiz sonuçları Tablo 5.54' da verilmiştir. Tablodaki değerler analiz sonuçları ortalamalarını göstermektedir.

Tablo 5.54: Çoklu Modellerin Analiz Sonuçları Ortalamaları

	Hassasiyet	Anımsama	F1
LSTM	0.50	0.47	0.47
2LSTM	0.40	0.41	0.39
DT	0.40	0.41	0.40
kNN	0.35	0.35	0.34
RF	0.46	0.47	0.46
SVM	0.78	0.29	0.29

Sonuçlar incelendiğinde, özellikle F1 değeri göz önünde bulundurulduğunda, tek katmanlı LSTM modelinin iki katmanlı LSTM, DT, kNN ve SVM modellerinden daha başarılı olduğu söylenebilir. RF modeli ile elde edilen sonuçların tek katmanlı LSTM modeli sonuçlarına çok yakın olduğu gözlemlenmiştir. Bu durumdan; LSTM çoklu sınıflandırma modelinin, zararlı yazılımları türlerine göre sınıflandırmada tercih edilebileceği, sonucunu çıkarabiliriz.

Modellerin eğitim süreleri, Windows 7(64 bit) işletim sistemi, Intel(R) Core(TM) i7-2600 CPU@ 3.40Ghz 3.40Ghz işlemcili 6GB RAM olan makinada aşağıdaki gibidir;

- Tek Katman LSTM : 64.78 dakika
- İki Katman LSTM : 42.82 dakika
- DT : 1.36 dakika
- kNN : 5.26 dakika
- RF : 6.8 dakika
- SVM :13.62 dakika

Bu süreler sadece model eğitim süreleridir. Veri kümesinin okunması gibi süre kayıplarından arındırılmıştır.

Daha önceki bölümde 8 farklı ikili sınıflandırma modelleri oluşturulmuştur. İkili sınıflandırma modelleri bir uygulama yardımı ile, veriler ikili ikili olarak herbir modele

sorularak, alınan cevapların istatisliği alınarak, bir çoklu sınıflandırma modeli oluşturulabilir.

Bölüm 6

Sonuç

Tez çalışması kapsamında, 7107 adet zararlı yazılımın Windows 7 işletim sistemi üzerinde yaptığı sistem çağrıları, çalışma anında, elde edilerek bir veri kümesi oluşturulmuştur. Bu yazılımların dahil oldukları sınıflar, VirusTotal servisi sayesinde, belirlenmiş ve veri kümesine dahil edilmiştir. Bu sayede zararlı yazılımların çalışma anında sergiledikleri davranış verilerini ve bu yazılımların dahil oldukları sınıf verilerini içeren bir analiz veri kümesi meydana gelmiştir.

Bu veri kümesi, derin öğrenme yöntemlerinden LSTM kullanılarak analiz edilmiştir. Belirlenen zararlı yazılım sınıfları için ayrı ayrı sınıflandırma modelleri oluşturulmuştur. Elde edilen sonuçların %83.5 ile %98.5 arasında bir başarı oranları gösterdikleri gözlemlenmiştir. VirusTotal servisi kullanılarak elde edilen zararlı yazılım sınıf verilerinin yüzde yüz doğru olamayacağı göz önünde bulundurulduğunda, Derin Öğrenme - LSTM yöntemi kullanılarak yapılan sınıflandırmanın başarılı olduğu söylenebilir.

Ayrıca çoklu sınıflandırma yöntemleri kullanılarak derin öğrenme ve geleneksel yöntemler ile elde edilen sınıflandırma modelleri oluşturulmuştur. Bu modellerin sonuçları incelendiğinde LSTM kullanılarak oluşturulan çoklu sınıflandırma yöntemlerinin diğer modellere göre başarılı olduğu söylenebilir.

Metamorfik zararlı yazılımlar da, diğer aile üyelerinin yaptığı gibi, bir işletim sistemi üzerinde benzer davranışlar sergilemektedirler. Bu nedenle, yapmış olduğumuz çalışmanın metamorfik ve diğer zararlı yazılım türleri için de geçerli olduğu söylenebilir.

Sonuç olarak, Derin Öğrenme - LSTM yöntemi, metamorfik zararlı yazılımlarının sınıflandırılmasında kullanılabilir. Yapmış olduğumuz çalışma, bu sonucun kavramsal bir ispatı niteliğindedir.

Kaynaklar

[1] McAfee Labs Threats Report, Haziran 2018. URL https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf.

[2] F. Leder, B. Steinbock, P. Martini. "Classification and Detection of Metamorphic Malware using Value Set Analysis". *International Conference on Malicious and Unwanted Software (MALWARE)*, Ocak 2009.

[3] Virus Total Sınıflandırma Örneği. URL https://www.virustotal.com/#/file/c73e85e0b2de80fd187879b8704f46e976aa1e61507b8a469bdf826d2b36c09d/detection.

[4] B. B. Rad, M. Masrom, S. Ibrahim. "Camouflage in Malware: from Encryption to Metamorphism". *IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.8*, Ağustos 2012.

[5] P. Vinod, H. Jain, Y. K. Golecha, M. S. Gaur, V. Laxmi. "MEDUSA: MEtamorphic malware dynamic analysis using signature from API". *Proceedings of the 3rd International Conference on Security of Information and Networks*, Ocak 2010.

[6] Y. Qiao, Y. Yang, L. Ji, J. He. "Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences". *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Temmuz 2013.

[7] J. Y. Cheng, T. Tsai, C. Yang. "An information retrieval approach for malware classification based on Windows API calls". *International Conference on Machine Learning and Cybernetics*, Temmuz 2013.

[8] V. Mehra, V. Jain, D. Uppal. "DaCoMM: Detection and Classification of Metamorphic Malware". *Fifth International Conference on Communication Systems and Network Technologies*, Nisan 2015.

[9] R. S. Pirscoveanu, S. S. Hansen, T. M. T. Larsen, M. Stevanovic, J. M. Pedersen, A. Czech. "Analysis of Malware behavior: Type classification using machine learning". *International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, Haziran 2015.

[10] S. P. Choudhary, M. D. Vidyarthi. "A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining". *Procedia Computer Science 54 (2015) 265 – 270* , Ağustos 2015.

[11] D. Ekhtoom, M. Al-Ayyoub, M. Al-Saleh, M. Alsmirat, I. Hmeidip. "A Compression-Based Technique to Classify Metamorphic Malware". *IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, 2016.

[12] B. Athiwaratkun, J. W. Stokes. "Malware classification with LSTM and GRU language models and a character-level CNN". *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mart 2017.

[13] S. Maniath, A. Ashok, P. Poornachandran, V. G. Sujadevi, A. U. P. Sankar, S. Jan. "Deep Learning LSTM based Ransomware Detection". *Recent Developments in Control, Automation and Power Engineering (RDCAPE)*, Mayıs 2017.

[14] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, A. K. Sangaiah. "Android malware detection based on system call sequences and LSTM". *Multimedia Tools and Applications*, Eylül 2017.

[15] S. B. Prapulla, S. J. Bhat, G. Shobha. "Framework for Detecting Metamorphic Malware Based on Opcode Feature Extraction". *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, 2017.

[16] Y. Fang, C. Huang, L. Liu, M. Xue. "Research on Malicious JavaScript Detection Technology Based on LSTM". *IEEE Access PP(99):1-1*, 2018.

[17] M. E. Ahmed, S. Nepal, H. Kim. "MEDUSA: Malware Detection Using Statistical Analysis of System's Behavior". *IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, 2018.

[18] Windows API Index, Mayıs 2018. URL `https://docs.microsoft.com/en-us/windows/desktop/apiindex/windows-api-list`.

[19] CreateFileA function, Mayıs 2018. URL https://docs.microsoft.com/en-us/windows/desktop/api/FileAPI/nf-fileapi-createfilea.

[20] What is Cuckoo? URL https://cuckoo.sh/docs/introduction/what.html.

[21] Virus Total, . URL https://www.virustotal.com/gui/home/upload.

[22] VirusTotal Public API v2.0, . URL https://www.virustotal.com/tr/documentation/public-api/.

[23] S. Hochreiter, J. Schmidhuber. "Long Short-term Memory". *Neural Computation 9(8):1735-80*, 1997.

[24] Cuckoo Sandbox Installation. URL https://cuckoo.sh/docs/installation/index.html.

[25] Virus Total Ananliz Sonucu, . URL https://www.virustotal.com/#/file/2f1d11e03f1e31c0319c978cfadc25cfa3cdf03ed972c08087d056b96cd09069/detection.

[26] Tensorflow. URL https://www.tensorflow.org/.

[27] Keras. URL https://keras.io/.

[28] Scikit-Learn. URL https://scikit-learn.org/stable/.

[29] Pandas. URL https://pandas.pydata.org/.

[30] Matplotlib. URL https://matplotlib.org/api/pyplot_api.html.

[31] Confusion matrix. URL https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py.