Sustav za detekciju zlonamjernih programa temeljen na konvolucijskim neuronskim mrežama

Prpić, Juraj

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 624

SUSTAV ZA DETEKCIJU ZLONAMJERNIH PROGRAMA TEMELJEN NA KONVOLUCIJSKIM NEURONSKIM MREŽAMA

Juraj Prpić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 624

SUSTAV ZA DETEKCIJU ZLONAMJERNIH PROGRAMA TEMELJEN NA KONVOLUCIJSKIM NEURONSKIM MREŽAMA

Juraj Prpić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 624

Pristupnik:	**Juraj Prpić (0036511440)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Tomislav Hrkać

Zadatak: **Sustav za detekciju zlonamjernih programa temeljen na konvolucijskim neuronskim mrežama**

Opis zadatka:

Zlonamjerni programi predstavljaju značajan problem sigurnosti računala i podataka te se nameće potreba za njihovom automatskom detekcijom i onemogućavanjem. Potencijalno zanimljiv pristup raspoznavanju zlonamjernih programa sastoji se u uporabi dvodimencionalne konvolucijske neuronske mreže na čiji se ulaz dovodi nepoznati program predočen u obliku slike, pri čemu se svaki bajt programskog koda nepoznatog programa promatra kao piksel. U okviru diplomskog rada potrebno je proučiti najznačajnije klasifikacijske konvolucijske neuronske mreže te ispitati mogućnost njihove uporabe za raspoznavanje i detekciju zlonamjernih programa. Programski ostvariti takav sustav temeljen na prikladnom konvolucijskom modelu i uz korištenje prikladnih biblioteka. Pripremiti skup primjera za učenje i ispitivanje sustava, analizirati ponašanje ostvarenog sustava te prikazati i ocijeniti ostvarene rezultate. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne podatke i rezultate, uz potrebna objašnjenja i dokumentaciju te navesti korištenu literaturu.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Zlonamjerni programi predstavljaju problem, problem koji iz godine u godinu sve više dolazi do izražaja. Jedan od razloga tog rasta sve je veća digitalizacija svih sfera života s kojom ujedno raste i isplativost ucjene i krađe podataka. Iz istog razloga čestom metom postale su veće tvrtke koje posjeduju puno veće količine potencijalno vrijednih podataka.

Postoji veći broj različitih pristupa kojima se pokušava riješiti problem detekcije i klasifikacije zlonamjernih programa, neki od kojih koriste strojno učenje i u aktivnoj su u upotrebi. Iako postoji veći broj pristupa, i dalje se vode istraživanja o tome kako poboljšati detekciju i klasifikaciju, bilo izmjenama ili proširivanjem već postojećih metoda, bilo analiziranjem i razvijanjem novih metoda.

U posljednjih nekoliko godina počinje se razmatrati ideja upotrebe konvolucijskih neuronskih mreža za detekciju i klasifikaciju zlonamjernih programa. U ovom radu opisan je navedeni pristup. Točnije, razmatra se navedeni pristup uz korištenje prednaučenih modela ResNet i EfficientNet. Navedeno je detaljnije opisano u 2. poglavlju, *Konvolucijske neuronske mreže*.

Ukratko, ideja je slijedeća, datoteke iz skupa podataka, detaljnije opisanog u 1. poglavlju, *Zlonamjerni programi*, transformiraju se u slike kvadratnog oblika pristupom 1 bajt – 1 piksel. Dobivene slike se potom dodatno modificiraju kako bi bile prigodni ulazi u prethodno navedene prednaučene modele. Prednaučeni modeli se zatim prilagođavaju skupu podataka postupkom finog podešavanja, tj. dodatnim učenjem nad tim skupom podataka. Navedeni postupak, modificirani modeli ResNet i EfficientNet te rezultati dobiveni njihovom upotrebom opisani su u 3. poglavlju, *Konvolucijske neuronske mreže iz programskog dijela rada*.

U 4. poglavlju, *Analiza i usporedba rezultata*, provedene su analiza i usporedba navedena dva modela.

1. Zlonamjerni programi

Zlonamjerni softver (engl. malware, spoj riječi „malicious“ i „software“) je datoteka ili kod koji zaražava, istražuje, krade podatke ili izvršava proizvoljne naredbe zlonamjernog aktera – napadača [1].

Cilj zlonamjernog softvera najčešće je barem jedno od slijedećeg: zlouporaba uređaja za slanje spam poruka, krađa osjetljivih podataka, prikupljanje informacija o zaraženom uređaju i lokalnoj mreži na koju je uređaj spojen te omogućavanje udaljenog pristupa napadaču [1].

Napad zlonamjernim softverom sastoji se od dvije komponente: nosive komponente (engl. payload) i vektora napada (engl. attack vector) [2]. Nosiva komponenta je zlonamjerni kod koji napadač namjerava dostaviti žrtvi, a vektor napada je metoda kojom se nosiva komponenta dostavlja žrtvi [2].

Neki od najčešćih vektora napada su putem: društvenog inženjeringa, ranjivosti sustava, prenosivih medija, napada na lanac opskrbe (engl. supply chain attacks), osobnih uređaja u poslovnim mrežama, reklama koje preusmjeravaju na zlonamjerne stranice (tzv. malvertising) ili lažnog softvera – softvera koji nalikuje legitimnom, ali sadrži zlonamjerni softver [2].

Neke od kategorija u koje se zlonamjerni softver može svrstati su: računalni virusi, računalni crvi, botneti, cryptojackeri, bez-datotečni zlonamjerni softver (engl. fileless malware), trojanski konji, rootkitovi, scareware, spyware, adware, ucjenjivački softver (engl. ransomware), zlonamjerni softver udaljenog pristupa (engl. remote access malware) [2].

1.1. Vrste zlonamjernih programa

Vrste zlonamjernih programa koje se pojavljuju u skupu podataka „Microsoft Malware Classification Challenge“, opisanom u potpoglavlju 1.2., su računalni crv, adware,

trojanski konj, stražnji ulaz i obfuscirani programi kao šira kategorija. Navedene vrste detaljnije su opisane u nastavku, unutar potpoglavlja 1.1.1. – 1.1.5.

1.1.1. Računalni crv

Računalni crv (engl. computer worm) vrsta je zlonamjernog programa sa sposobnošću automatske propagacije ili samorepliciranja što mu omogućuje širenje na ostala računala unutar mreže bez dodatne ljudske intervencije [3].

Iskorištava ranjivosti softvera za krađu osjetljivih informacija, instalaciju stražnjih vrata, oštećivanje datoteka te ostale slične nakane [4]. Generalno se širi kroz ranjivosti u softveru, putem privitaka unutar elektronske pošte ili slanjem trenutačnih poruka (engl. Instant Messaging, IM) [4]. Kada korisnik otvori datoteku preusmjeruje ga se na zlonamjernu stranicu ili se računalni crv direktno preuzme na korisnikov uređaj [4]. Potom se može dalje širiti bez da zaraženi korisnik to primijeti [4].

Zlonamjerni akteri računalne crve mogu iskoristiti za izvođenje DDoS napada (distribuirani napad uskraćivanjem resursa, engl. Distributed Denial of Service attack), ucjenjivanje, krađu osjetljivih podataka, širenje drugih zlonamjernih programa ili brisanje datoteka [3].

Računalni crvi mogu se detaljnije podijeliti prema načinu njihovog propagiranja, neki od kojih su ranije pojašnjeni. Primjerice: email crv, IM (Instant Messenger) crv, IRC (Internet Relay Chat) crv, net crv i P2P (peer-to-peer) crv [4].

Neki od poznatih primjera računalnih crva su: Morris, SQL Slammer, Mydoom, Storm Worm, Duqu i ILOVEYOU [3].

1.1.2. Adware

Adware je termin nastao spojem engleskih riječi „advertising“ (hrv. reklamiranje) i „software“, tj. kraći je naziv za „advertising-supported software“ [5, 6].

Vrsta je zlonamjernog softvera koji se potajno instalira na žrtvino računalo te prikazuje neželjene reklame i pop-upove [7]. Generalno zaražava računalo putem preuzetog sadržaja koji omogućava pristup zlonamjernom softveru na računalo, učestalo se pojavljuje zajedno

sa sharewareom ili freewareom [6]. Developeri adwarea potencijalno mogu zarađivati na svakom prikazivanju reklame ili kliku na reklamu, što je jedan od razloga zbog kojeg ih poneke legitimne kompanije umeću sa svojim softverom [6].

Adware generalno ne stvara veću štetu poput ostalih vrsta zlonamjernih programa, najčešće nije ništa više od obične naporne smetnje [6]. Unatoč tome, treba uzeti u obzir da neke varijante adwarea mogu prikupljati podatke o korisniku i nadgledati njegov uređaj, što može uzrokovati štetu [6].

Najčešća varijanta adwarea je browser hijacker koji modificira postavke Internet preglednika bez korisnikovog znanja i dopuštenja, promjeni početnu stranicu i osnovne postavke pretraživača [7]. Posljedično, korisnik će pri pretraživanju interneta konstantno biti zatrpavan mnogim reklamama, često u obliku pop-upa [7].

Slika 1.1. Adware pop-upovi s lažnim informacijama o pronađenim virusima na računalu [8]

Primjer adwarea koji korisniku lažno prikazuje da su na računalu pronađeni „virusi“ prikazan je na Slika 1.1. Kod ove varijante adwarea najčešće je cilj navesti potencijalno neupućenog i/ili prestrašenog korisnika da izvrši novčanu uplatu za pristup programu koji će „ukloniti viruse“ s računala ili tome slično.

1.1.3. Trojanski konj

Trojanski konj (engl. trojan horse), također poznat pod nazivom trojanac, vrsta je zlonamjernog programa koji se prikriva kao legitimni kod ili softver [9].

Nakon što uspješno zarazi računalo ili mrežu, napadaču omogućuje da izvodi sve akcije koje korisnik može izvoditi, tj. za koje ima prava, primjerice izvoz datoteka, modifikaciju podataka ili brisanje datoteka [9]. Trojanci do računala mogu doći upakirani zajedno s piratskim verzijama legitimnog softvera pri preuzimanju, primjerice s video igrama, alatima, aplikacijama ili softverskim zakrpama (engl. patch) [9]. Drugi načini na koje se računalo može zaraziti su putem društvenog inženjeringa (engl. social engineering) ili putem spoofing i phishing napada [9]. Ključno je operativni sustav i sav softver održavati ažurnima zbog sprječavanja postojanja ranjivosti koje bi trojanci mogli iskoristiti [9].

Ovisno o varijanti, trojanac se može ponašati na više različitih načina: kao samostalni zlonamjerni program, kao alat za dostavljanje nosive komponente, za komunikaciju sa zlonamjernim akterom, itd. [10]. Cilj trojanaca je oštetiti datoteke, preusmjeriti internetski promet, nadgledati korisnikovu aktivnost, ukrasti osjetljive podatke ili postaviti stražnja vrata [9].

Neke od kategorija na koje se trojanci mogu podijeliti su: Exploit trojan, Downloader trojan, Ransom trojan, Backdoor trojan, Distributed Denial of Service (DDoS) attack trojan, Fake AV trojan, Rootkit trojan, SMS trojan, Banking trojan i trojan GameThief [9].

Neki od poznatih primjera trojanaca su: Zloader, QakBot i Andromeda [9].

Trojan downloader

Trojan downloader je vrsta trojanca koja potajno preuzima zlonamjerne datoteke s udaljenog poslužitelja (engl. remote server) te ih instalira i pokreće [11].

To izvodi na način da provjerava je li dostupna Internetska veza, ukoliko nije – čeka da postane dostupnom, ukoliko je – kontaktira udaljenog poslužitelja s kojeg ili direktno preuzima dodatne zlonamjerne datoteke ili čeka da zlonamjerni akter pošalje naredbe [11].

Poznati primjer Trojan Downloadera je Emotet [12]. Emotet je sofisticirani zlonamjerni program koji ima sposobnost krađe osjetljivih informacija poput korisničkih imena, lozinki i bankovnih informacija korisnika [12]. Često se širi putem phishing e-mailova i preuzima i instalira dodatan zlonamjerni softver poput ransomwarea i bankovnih trojanaca [12].

1.1.4. Stražnja vrata

Stražnja vrata (engl. backdoor) bilo su kakva metoda pomoću koje autorizirani ili neautorizirani korisnici mogu zaobići sigurnosne mjere i dobiti udaljeni pristup računalu, mreži ili aplikaciji. [13] Udaljeni pristup resursima unutar aplikacija – bazama podataka, datotečnim poslužiteljima (engl. file server) – zlonamjernim akterima omogućuje udaljeno izvršavanje naredbi (engl. Remote Code Execution, RCE) [14].

Često ih se klasificira kao trojance [13]. Trojanci su detaljnije opisani u prethodnom potpoglavlju 1.1.3.

Jednom kad zlonamjerni akteri uspješno pridobiju pristup, stražnja vrata se mogu iskoristiti za eskalaciju privilegija, širenje na druge sustave unutar mreže (tzv. lateralno kretanje), krađu osjetljivih podataka, preotimanje poslužitelja (engl. server hijacking), sudjelovanje u DDoS napadu, zaražavanje posjetitelja web stranice ili za bilo kakvo ometanje općenito [14, 15].

Developeri ponekad koriste stražnja vrata pri razvoju softvera u svrhu testiranja i razrješavanja bugova te, iako je namjera ukloniti ih prije objave finalne verzije, može se dogoditi da ih greškom ostave unutar programskog koda [13].

Ponekad su stražnja vrata namijenjena za upotrebu od strane autoriziranih korisnika te ih proizvođač instalira namjerno i direktno pri razvoju softvera [13]. Postoji veći broj razloga zašto, jedan od potencijalnih razloga je da korisnička podrška može razriješiti problem korisnika koji je sam sebi greškom u potpunosti onesposobio mogućnost pristupa svom uređaju [13]. Ipak, iako je namjena da ih koriste autorizirani korisnici, njihovo postojanje predstavlja rizik iz razloga što ih neautorizirani korisnici također mogu zloupotrijebiti ukoliko ih pronađu.

1.1.5. Obfuscirani program

Obfuskacija je tehnika koja binarne i tekstualne datoteke čini nečitljivima ili teškima za razumjeti [16].

Zlonamjerni akteri često koriste tehnike kriptiranja i kodiranja kako bi prikrili zlonamjerne programe od detekcije pomoću sigurnosnih programa [17]. Osim što otežava detekciju, kasnije inženjerima, analitičarima, dodatno otežava reverzno inženjerstvo i analizu programa [17, 18]. Kao posljedica otežane detekcije, povećava se vjerojatnost da će se zlonamjerni softver zadržati na zaraženom računalu na duži period (engl. persistence) [18].

Razlozi za korištenje tehnika obfuskacije mogu biti zlonamjerni, ali mogu biti i legitimni, primjerice u svrhu prevencije reverznog inženjerstva i piratizacije legitimnog softvera [16].

Tijekom samog procesa obfuskacije, često se obfusciraju određeni stringovi unutar koda koji bi mogli ukazivati na manipulaciju ključeva registra (engl. registry keys) te URL-ovi zlonamjernih stranica [16]. Često se umeće i beskoristan kod (engl. dead code), koji se uopće ne izvršava ili ne radi ništa korisno, već mu je cilj otežati analizu od strane inženjera, analitičara [18]. Neki zlonamjerni programi mogu biti obfuscirani u cijelosti, to se postiže upotrebom packera [16].

Obfuskacija se može postići na različite načine, neki od načina su slijedeći: korištenje XOR operacija, base64 kodiranje, ROT13 kodiranje, upotreba packera [16].

Primjer obfusciranog programa konvertiranog u PNG format prikazan je na Slika 3.8.

1.2. Skup podataka „Microsoft Malware Classification Challenge“

Skup podataka koji se koristi u sklopu ovog rada je skup podataka iz „Microsoft Malware Classification Challenge“ natjecanja [19].

Ovaj skup podataka odabran je iz razloga što sadrži velik broj primjera zlonamjernih programa, točnije sadrži ~500 GB primjera podijeljenih u skupove za učenje i testiranje u omjeru od oko 50:50. Od navedenih 500 GB više od polovice otpada na datoteke disasembliranog koda, dok ostatak otpada na binarne datoteke. Binarne datoteke iz ovog skupa podataka ne sadrže PE zaglavlje (engl. header) kako bi se osigurala sterilnost skupa

[19]. Za potrebe programskog dijela rada koriste se isključivo binarne datoteke iz ovog skupa podataka. Dodatno, dio primjera koji je u skupu podataka originalno namijenjen za učenje koristi se i za učenje i za testiranje u programskom dijelu rada. Razlog tome je što je taj dio primjera označen, dok je dio primjera koji je originalno namijenjen za testiranje neoznačen, što bi uvelo dodanu problematiku pri provjeri točnosti pri korištenju tog skupa podataka u svrhu testiranja.

U skupu se nalazi 9 različitih zlonamjernih programa, to su: Ramnit, Lollipop, Vundo, Tracur, Simda, Kelihos_ver1, Kelihos_ver3, Gatak i Obfuscator.ACY. Svaka od navedenih vrsta, osim Obfuscator.ACY, detaljnije je opisana u potpoglavljima 1.2.1. – 1.2.7. U navedenim potpoglavljima nisu nužno opisani svi detalji ponašanja svih varijanti navedenih zlonamjernih programa. Razlog tome je taj da smisao tih potpoglavlja nije da se navedeni zlonamjerni programi i njihovo ponašanje pojasne do najmanjeg koraka već da ukratko opišu vrste zlonamjernih programa koji se koriste u programskom dijelu rada. Obfuscator.ACY je prethodno opisan u potpoglavlju 1.1.5., to je vrlo općenita i široka klasifikacija koju se može pridijeliti bilo kojem obfusciranom programu.

Navedeni skup podataka je iz 2015. godine, stoga ne sadrži varijacije ranije navedenih 9 zlonamjernih programa nastalih nakon te godine.

Činjenica da modeli u ovom radu uče na temelju „samo“ 9 različitih vrsta zlonamjernih programa nije problem s obzirom da je svrha rada demonstrirati pristup detekcije i klasifikacije korištenjem konvolucijskih neuronskih mreža i usporediti kakve rezultate daju naspram već postojećih pristupa. Modeli bi se mogli učiti i na skupovima s većim brojem različitih vrsta zlonamjernih programa uz eventualne određene izmjene unutar modela. Isto vrijedi i za činjenicu da binarne datoteke skupa podataka ne sadrže PE zaglavlje.

1.2.1. Ramnit

Ramnit se prvi put pojavljuje 2010. godine [20]. Iako je originalna varijanta bila samoreplicirajući računalni crv, kasnije varijante ovog zlonamjernog programa postaju sofisticiranije te dolazi do promjena u tehnikama napada i području djelovanja [20].

Originalna varijanta Ramnita radi na način da zarazi EXE, DLL i HTML datoteke, krade podatke za prijavu (engl. credentials) putem FTP-a i kolačiće pretraživača te se dalje širi na

druge uređaje zloupotrebljavanjem zaraženog uređaja [20]. To je ujedno i varijanta Ramnita koja se nalazi unutar korištenog skupa podataka.

Kasnije varijante Ramnita imaju širi spektar sposobnosti, zaraženi uređaji mogu se iskoristiti kao udaljeno kontrolirani botovi koji komuniciraju s C&C poslužiteljem (engl. Command and Control (C&C) server), tj. kao dio botneta, u svrhu krađe bankovnih informacija, lozinki i ostalih osjetljivih podataka ili za uspostavljanje udaljenog pristupa unutar mreža tvrtki ili financijskih ustanova [20].

U trenutku kada je sustav uspješno zaražen Ramnitom, zlonamjernom akteru – napadaču, omogućava se instalacija dodatnih zlonamjernih programa [20].

Ramnit botnet koristili su kriminalci u svrhu krađe osobnih i bankovnih informacija, lozinki i za onesposobljavanje antivirusnih zaštita [21]. Širio se putem različitih vektora poput poveznica unutar spam e-mailova i posjećivanjem zaraženih web stranica od strane korisnika [21]. Ramnit botnet operaciju uspješno su obustavili Europol, Microsoft i Symantec 2015. godine [22].

Primjer Ramnit crva konvertiranog u PNG format prikazan je na Slika 3.1.

1.2.2. Lollipop

Lollipop je adware, prikazuje reklame za vrijeme korištenja web preglednika, može preusmjeriti rezultate internetskog pretraživača, nadgledati akcije korisnika na računalu, preuzimati aplikacije i zlonamjernom akteru slati prikupljene informacije o računalu [23].

Vrsta reklama koje Lollipop prikazuje su pop-up reklame unutar web preglednika, a prikazane reklame ovise o ključnim riječima koje korisnik pretražuje i o geografskoj lokaciji korisnika [23].

Jedan od načina na koje Lollipop može doći na računalo je prilikom instalacije legitimnog softvera kao što je prikazano na Slika 1.2. [23].

Lollipop se instalira na lokaciju %LOCALAPPDATA%\Lollipop, a perzistenciju na računalu uspostavlja uređivanjem stavki unutar registra ili postavljanjem Lollipop.lnk kratice u Windows startup mapu [23].

Slika 1.2. Instaliranje Lollipop adwarea putem alata za instalaciju programa GIMP [23]

Primjer Lollipop adwarea konvertiranog u PNG format prikazan je na Slika 3.2.

1.2.3. Vundo

Vundo je trojanac, iako neke varijante imaju i obilježja adwarea, poput pop-up reklama [24]. Ovisno o varijanti, može preuzimati i izvršavati proizvoljne datoteke [24].

Varijanta pop-up reklama koje Vundo prikazuje je tzv. lažni sigurnosni softver (engl. rogue security software), pop-up korisnika lažno navodi da se na njegovom računalu nalaze zlonamjerni programi [25]. Primjer takvog pop-upa prikazan je na Slika 1.1.

Određene varijante prikupljaju informacije o sustavu na kojem se nalaze poput IP adrese, verzije Windowsa, MAC adrese, verzije web preglednika, itd. [25]. Također može prikupljati i zlonamjernom akteru slati povijest pretraživanja web preglednika ili može preusmjeravati korisnika na reklamne web stranice [25].

Generalno se distribuira putem spam e-maila ili drugih kanala kao DLL datoteka te se bez korisnikovog pristanka instalira na računalo kao Browser Helper Object (BHO) [24, 25].

Pri izvršavanju Vundo sam sebe kopira u %AppData% mapu kao netprotocol.exe datoteku [24]. Također može stvoriti zapis unutar registra kako bi osigurao svoje izvršavanje pri svakom pokretanju računala [24]. Alternativno, stvara EXE datoteku u %System% direktoriju i DLL datoteku u %Temp% direktoriju [25]. Također se pokušava injektirati unutar aktivnih procesa, s posebnim naglaskom na procese vezane uz sigurnosni softver [25]. Ukoliko ga korisnik pokuša ukloniti, neke od varijanti će se pokušati oporaviti koristeći Windows File Protection (WFP) tehnike [25].

Primjer Vundo trojanca konvertiranog u PNG format prikazan je na Slika 3.4.

1.2.4. Tracur

Tracur je trojanac koji preusmjerava rezultate internetskog pretraživača u svrhu zarade putem reklamnih prevara [26]. Osim preusmjeravanja, Tracur također može preuzimati i pokretati datoteke, uključujući drugi zlonamjerni softver, i zlonamjernom akteru dati kontrolu nad zaraženim računalom [26].

Zaražava računalo na način da ga instalira neki drugi zlonamjerni softver, tako da korisnik otvori nepoznatu poveznicu ili privitak unutar e-maila, putem exploit kita (npr. Blacole), putem downloadera ili putem društvenog inženjeringa [26].

Neke od varijanti pokušavaju se spojiti s poslužiteljem koristeći slučajni TCP port te čekaju naredbe zlonamjernog aktera koji tada može, između ostalog, preuzimati i pokretati datoteke te kontrolirati kako će nosive komponente biti preusmjerene [26]. Određene druge varijante pokušavaju iskoristiti netsh.exe kako bi se nadodale na listu iznimki Windows vatrozida radeći izmjene unutar registra [26].

U svrhu osiguravanja perzistencije, Tracur modificira zapise unutar registra, stvara DLL file u jednu od podmapa unutar %%USERPROFILE%% mape te sprema kopiju sebe kao EXE i DLL u sistemsku mapu koristeći naziv neke od već postojećih DLL datoteka [26].

Primjer Tracur trojanca konvertiranog u PNG format prikazan je na Slika 3.6.

1.2.5. Simda

Simda je trojanac stražnjih vrata (engl. backdoor trojan), primarna svrha Simde je omogućiti udaljeni pristup i kontrolu nad računalom putem stražnjih vrata te prikupiti korisnikove lozinke i informacije o zaraženom računalu [27]. Računala koja Simda zarazi postaju dio botneta [28]. Prvi je put uočen 2009. godine, a 2015. godine bio je meta koordinirane internacionalne operacije s ciljem njegova uklanjanja (engl. takedown) [28].

Varijante Simde generalno se distribuiraju putem kompromitiranih stranica koje korisnike preusmjeravaju na stranice koje poslužuju exploit kitove [28].

Način na koji neke od varijanti na zaraženom računalu pokušavaju malicioznom akteru omogućiti pristup je tako da otvore portove računala [28]. Također pokušavaju uspostaviti vezu s C&C poslužiteljem kako bi mogli dobiti naredbe od malicioznog aktera [28]. C&C poslužitelju se potom mogu slati prikupljeni podatci s računala [28].

Ukoliko je zaraženi korisnik administrator, Simda može dodati zakazani zadatak (engl. scheduled task) da mu se pri svakom pokretanju računala pridijele administratorska prava [27]. U slučaju da trenutni korisnik nije administrator, neke od varijanti pokušati će se prijaviti kao administrator koristeći listu čestih lozinki [28].

Simda svoju originalnu datoteku briše u dvije situacije: nakon uspješne instalacije ili u slučaju da detektira da se nalazi unutar virtualne mašine ili sandboxa [27]. Ukoliko uspješno podigne svoje privilegije na administratorsku razinu, Simda pokušava injektirati DLL u procesni prostor od winlogon.exe [27]. Dodatno, ovisno o varijanti, Simda se uopće neće instalirati u slučaju da detektira da su pokrenuti određeni procesi vezani uz sigurnosni softver ili da postoje registarski ključevi koji su također vezani uz sigurnosni softver [27].

Primjer Simda trojanca konvertiranog u PNG format prikazan je na Slika 3.5.

1.2.6. Kelihos

Kelihos P2P (peer-to-peer) botnet prvi se put pojavljuje u prosincu 2010. godine [29]. Bio je jedan od najvećih i najduže operativnih kibernetičkih kriminalnih infrastruktura sve dok njegovu posljednju verziju nije obustavilo i preotelo (engl. takeover) Ministarstvo pravosuđa SAD-a uz pomoć CrowdStrikea u travnju 2017. godine [29]. Jedna je od prvih kriminalnih infrastruktura zlonamjernog softvera koja je koristila P2P tehnologiju [29].

Kelihos trojanac zarazi računalo kao dio drugog zlonamjernog softvera poput trojan downloadera ili putem poveznica na zlonamjerne web stranice [30]. Može se širiti i putem prijenosnih medija na kojima mape zamjeni s LNK datotekama koje izvršavaju Kelihos te potom otvore i prikažu stvarni sadržaj mape [29]. Nakon što zarazi računalo, pokušava komunicirati s udaljenim poslužiteljem i instalirati dodatne komponente poput stražnjih vrata kako bi zlonamjerni akter dobio udaljeni pristup računalu [30]. S udaljenim poslužiteljima komunicira slanjem kriptiranih poruka koristeći HTTP protokol (TCP, port 80) čime pokušava izbjeći detekciju i filtere [31].

Kelihos zaražene uređaje pokušava povezati u Kelihos botnet te ih iskoristiti za slanje spam e-mailova [30]. Spam e-mailovi koje generira koriste različite kombinacije i varijacije zaglavlja i izbjegava korištenje konzistentnih uzoraka kako bi pokušao zaobići e-mail filtere [29].

Pojedine varijante Kelihosa koriste WinPcap za nadgledanje mrežnog prometa s ciljem prikupljanja korisničkih podataka za prijavu iz FTP, POP3 i SMTP prometa [31].

Iako rijetko korišten u tu svrhu, Kelihos botnet ima sposobnost izvršavanja DDoS napada [29].

Primjer Kelihos_ver1 konvertiranog u PNG format prikazan je na Slika 3.7, a primjer Kelihos_ver3 na Slika 3.3.

1.2.7. Gatak

Gatak je trojanac stražnjih vrata (engl. backdoor trojan), prvi se put pojavljuje 2012. godine [32].

Također se naziva i Stegoloader, zbog činjenice što komunicira sa C&C poslužiteljima koristeći steganografiju, tehniku prikrivanja informacija unutar slika, u ovom slučaju prikrivanja zlonamjernog koda, naredbi ili konfiguracijskih podataka zlonamjernih programa unutar PNG ili JPG datoteka [32]. Preciznije, Gatak se povezuje sa svojim C&C poslužiteljem te zatražuje nove naredbe [32]. Umjesto da primi HTTP mrežni zahtjev, računalo zaraženo Gatkom prima podatke naizgled bezazlene slike što nalikuje uobičajenom mrežnom prometu [32]. Gatak potom čita prikrivene podatke iz slike i izvršava naredbe [32].

Njegov modularni dizajn omogućava zlonamjernom akteru da koristi samo dijelove Gatka ovisno o potrebi, što umanjuje izlaganje njegovih mogućnosti prilikom istrage i analize pomoću reverznog inženjerstva čime se otežava procjena stvarnih nakana zlonamjernog aktera [33].

Primjer Gatak trojanca konvertiranog u PNG format prikazan je na Slika 3.9.

2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (engl. Convolutional Neural Networks, CNNs) dobile su ime prema matematičkoj funkciji konvolucije [34]. Koriste se za probleme poput klasifikacije slika, računalnog vida ili obrade prirodnog jezika [34].

Konvolucija je proces u kojem se mala matrica, jezgra (engl. kernel), rednim slijedom primjenjuje na ulaznu sliku kako bi prepoznala ključne značajke slike. Jezgra se pomiče po slici te se njene vrijednosti množe s vrijednostima na položaju u slici nad kojim se trenutno nalazi te se potom pribroje [35].

Konvolucijske neuronske mreže sastoje se od većeg broja slojeva: ulaznog, konvolucijskih, aktivacijskih, slojeva sažimanja (engl. pooling layer), potpuno povezanih i izlaznog [34]. Općenitije: ulazni sloj, skriveni slojevi i izlazni sloj [36].

Cilj početnog sloja mreže je prepoznavanje osnovnih značajki slika – rubova, kutova i ostalih jednostavnih oblika [35]. Naredni slojevi postepeno prepoznaju sve kompleksnije značajke [35].

Izlaz iz posljednjeg sloja mreže je izračun kolika je vjerojatnost, od 0 do 1, da ulazni primjer pripada klasi, za svaku od klasa [35]. Što je broj bliži jedinici, to je vjerojatnost da je klasa ispravna veća [35].

2.1. Prednaučeni modeli

Prednaučeni modeli (engl. pretrained models) su modeli strojnog, ili dubokog, učenja koji su naučeni na velikom skupu podataka za rješavanje određene vrste problema [37]. Ti se modeli potom mogu koristiti direktno ili se kasnije mogu fino podesiti (engl. fine-tuning) za precizniju upotrebu ili za primjenu nad kakvim drugačijim problemom koristeći neki drugi skup podataka [37]. Koriste se u područjima poput obrade prirodnog jezika, računalnog vida i prepoznavanja glasa [37].

S obzirom da je učenje „od nule“ vremenski zahtjevno i iziskuje ogromne količine podataka, koji su potencijalno skupi za pribaviti ili ih može biti teško pribaviti, smisao

prednaučenih modela je ušteda vremena i resursa pošto se njihovim korištenjem izbjegava višestruko učenje modela od samog početka [37, 38]. Dodatno, s obzirom da su prednaučeni modeli generalno učeni nad vrlo velikim skupovima podataka, korištenje istih kao polaznom točkom može rezultirati boljim performansama [37]. Ipak, postoji i rizik od uvođenja dodatne pristranosti i lošije generalizacije u slučaju da je prednaučeni model učen koristeći vrlo specijalizirane skupove podataka [37].

Jedan od skupova podataka nad kojima su prednaučeni modeli učeni je ImageNet [39]. Skup podataka ImageNet detaljnije je opisan u slijedećem potpoglavlju 2.1.1.

Postupak zaključivanja modela (engl. model inference), tj. predviđanja ispravne klase temeljem ulazne slike je slijedeći: učitava se ulazna slika; izvode se transformacije na slici poput promjene dimenzija, podrezivanja ili normalizacije; radi se prolaz unaprijed koristeći prednaučene težine te se kao rezultat dobivaju vjerojatnosti za svaku od klasa [39].

Prednaučeni modeli ocjenjuju se koristeći sljedećih pet kriterija: Top-1 greška, Top-5 greška, vrijeme zaključivanja korištenjem procesora, vrijeme zaključivanja korištenjem grafičke kartice i veličina modela [39].

Do top-1 greške dolazi kad predviđena klasa s najvećom vjerojatnošću nije jednaka stvarnoj klasi primjera [39]. Isto vrijedi i za top-5 grešku, s tim da u tom slučaju stvarna klasa nije u predviđenih pet s najvećim vjerojatnostima [39].

Pod „veličinom modela“ misli se na prostor na disku koji zauzima PTH file prednaučenog modela [39].

Dobar model ima nisku stopu Top-1 i Top-5 grešaka, nisko vrijeme zaključivanja i kad se koristi procesor i kad se koristi grafička kartica te malu veličinu modela [39].

Neki od ključnih aspekata prednaučenih modela su slijedeći: prijenosno učenje (engl. transfer learning), izvlačenje značajki, bolje performanse, smanjena prenaučenost (engl. overfitting), širok spektar primjena, doprinos istraživačkoj zajednici [40].

U sklopu programskog dijela rada korištena su dva različita prednaučena modela: ResNet i EfficientNet. Navedeni modeli detaljnije su opisani su u naredna dva potpoglavlja – 2.1.2. i 2.1.3.

Pri odabiru varijacija ResNet i EfficientNet modela uzet je u obzir omjer preciznosti rezultata naspram računske i vremenske zahtjevnosti zbog ograničenosti računalnih resursa i vremenskog ograničenja izrade. Stoga su odabrane varijacije ResNet i EfficientNet

modela koje daju nešto manje precizne rezultate, ali koje su ujedno računski i vremenski manje zahtjevne.

2.1.1. Skup podataka ImageNet

Skup podataka ImageNet sastoji se od 14 milijuna slika [39]. Održava ga Sveučilište Stanford [39]. Korišten je za učenje ResNet i EfficientNet modela. Preciznije, korištena je varijanta ImageNet-1k, također poznata kao skup podataka ILSVRC 2012, koja je podskup skupa podataka ImageNet.

ImageNet-1k sadrži 1000 klasa, oko 1.28 milijuna slika za učenje, 50 000 slika za validaciju i 100 000 slika za testiranje [41]. Temeljem navedenog, svaka pojedina klasa sadrži 732-1300 slika za učenje, 50 slika za validaciju i 100 slika za testiranje [41].

ImageNet je izgrađen koristeći hijerarhijsku strukturu WordNeta kao okosnicu [42]. WordNet je velika leksička baza podataka engleskog jezika [43]. Riječi – imenice, glagoli, pridjevi i prilozi – su grupirane u skupove sinonima, svaki od kojih predstavlja različiti koncept [43]. Dodatno, svaki od skupova međusobno je povezan temeljem semantičkih i leksičkih odnosa [43]. S obzirom da mu je WordNet okosnica, ImageNet također održava međusobne veze različitih skupova [42].

Kod skupa podataka ImageNet jedna od ključnih stavki je da kategorije sadrže primjere s velikim brojem različitih varijacija: raznolik izgled, pozicija, točka gledišta, poza, pozadinska buka i djelomična prekrivenost subjekta [42].

2.1.2. ResNet model

Resnet je konvolucijska neuronska mreža koja se može koristiti u svrhu klasifikacije slika [44].

U originalnom radu u kojem je opisan ResNet, *Deep Residual Learning for Image Recognition* [45], (hiper)parametri pri učenju i ostale mjere definirani su kao što slijedi u nastavku.

Dimenzije ulazne slike transformiraju se tako da je njena kraća stranica duljine u rasponu od 256 do 480 piksela [45]. Potom se uzima isječak slike, ili horizontalno zrcaljene slike, u dimenzijama 224x224, s tim da se oduzima srednja vrijednost po pikselu [45].

Nakon svake pojedine konvolucije i prije aktivacija radi se normalizacija po grupi (engl. batch norm) [45].

Koristi se stohastički gradijentni spust (SGD) s veličinom mini-grupe 256 [45].

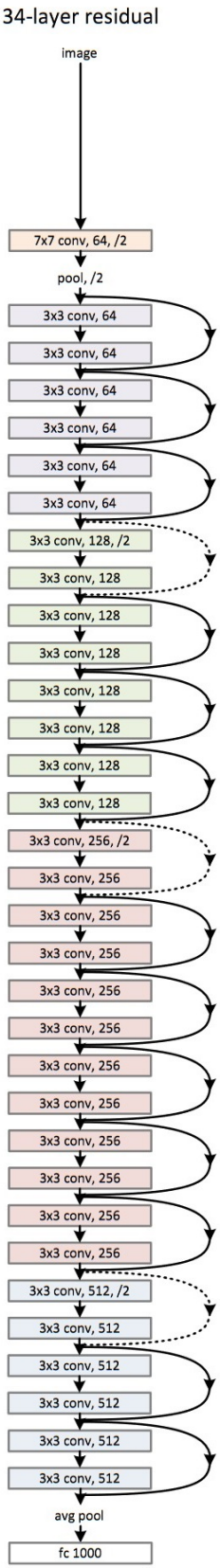
Stopa učenja postavljena je na 0.1 te se dijeli s 10 kada greška prestane padati i počne stagnirati [45]. Propadanje težina (engl. weight decay) postavljeno je na 1e-4, a zalet (engl. momentum) na 0.9 [45].

Učenje traje do najviše 60x104 iteracija [45].

Ispadanje (engl. dropout) se ne koristi [45].

ResNet-34 ima top-1 grešku od 25.03% kada se validira na ImageNet skupu podataka na kojem je i naučen [45]. Ovisno o kojoj je implementaciji ResNeta riječ, top-5 greške su 7.76%, 7.46% ili 7.40% [45].

U sklopu ovog rada koristi se ResNet-34, varijanta ResNeta s 34 sloja. Slojevi ResNet-34 modela prikazani su na Slika 2.1.



Slika 2.1. Slojevi i veze među slojevima ResNet-34 modela [45]

2.1.3. EfficientNet model

EfficientNet model opisali su istraživači Google AI-ja 2019. godine [46]. Koristi se za rješavanje mnogih izazovnih zadataka poput prepoznavanja objekata, segmentacije slike i obradu jezika [46]. Glavni razlog popularnosti ovog modela je izniman balans računske efikasnosti i performansi modela, dva kritična faktora u dubokom učenju [46].

Tradicionalni modeli često moraju tražiti kompromis između preciznosti i utroška resursa, dok EfficientNet uvodi pristup složenog skaliranja (engl. compound scaling) [46]. Dimenzije modela – širina, dubina i rezolucija – skaliraju se uniformno, koristeći skup fiksnih koeficijenata skaliranja [47].

U radu *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* [48] model ima postavljene postavke i hiperparametre kao što slijedi.

EfficientNet model učen je na ImageNet skupu podataka, koristi SiLU aktivaciju, RMSProp optimizator i AutoAugment [48].

Propadanje optimizatora postavljeno je na 0.9, a zalet na 0.9 [48]. Zalet normalizacije po grupi postavljen je na 0.99, a propadanje težine na 1e-5 [48].

Početna stopa učenja postavljena je na 0.256 te opada za 0.97 svake 2.4 epohe [48].

Stohastička dubina ima vjerojatnost preživljavanja 0.8 [48].

Ovisno o varijanti EfficientNet modela, omjer ispadanja je između 0.2 i 0.5, 0.2 za EfficientNet-B0, 0.5 za EfficientNet-B7, s linearnim porastom između [48].

Najosnovniji model, EfficientNet-B0, ima top-1 preciznost 77.1% i top-5 preciznost 93.3%, tj. top-1 grešku 22.9% i top-5 grešku 6.7% [48].

Najkompleksniji model, EfficientNet-B7, ima top-1 preciznost 84.3% i top-5 preciznost 97.0%, tj. top-1 grešku 15.7% i top-5 grešku 3.0% [48].

U sklopu ovog rada koristi se EfficientNet-B0 model. Originalno je odabran EfficientNet-B4, no zbog problema s ograničenom radnom memorijom ipak je naknadno odabran manje precizan model EfficientNet-B0. Pozitivna strana odabira modela EfficientNet-B0 naspram EfficientNet-B4 je što su ulazne dimenzije modela EfficientNet-B0 224x224. Navedene dimenzije jednake su ulaznim dimenzijama modela ResNet-34 što omogućuje realniju usporedbu rezultata modela.

3. Konvolucijske neuronske mreže iz programskog dijela rada

3.1. Priprema podataka

Za pripremu podataka koristi se Python skripta *convert_to_png.py* napisana za potrebe ovog rada. Koristi se verzija Pythona 3.12. Osim ugrađenih, koriste se i biblioteke *numpy* (verzija 1.26.3) i *Pillow (PIL)* (verzija 10.3.0).

Skripta se pokreće s 2 ulazna argumenta: direktorij koji sadrži binarne datoteke i direktorij u koji će se pohraniti slikovne datoteke u PNG formatu.

Svaka od ulaznih binarnih datoteka obrađuje se na slijedeći način. Datoteka se otvara koristeći funkciju `open()` sa „rb“ (read-binary) zastavicom te se čita pomoću funkcije read(). Svaki bajt datoteke zasebno se pohranjuje u dvodimenzionalno polje koristeći funkciju iz biblioteke Numpy: `np.frombuffer(<izlaz_funkcije_read>, dtype=np.uint8)`.

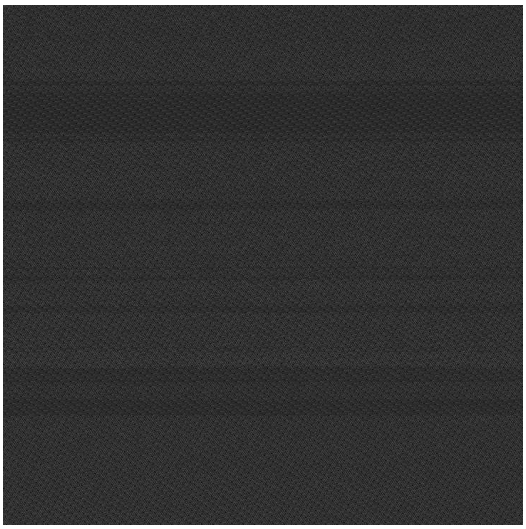
Koristeći jednostavne matematičke funkcije iz biblioteke math, računaju se dimenzije slike: `math.ceil(math.sqrt(len(<2D_polje_bajtova>)))`. S obzirom da slika mora biti kvadratnog oblika, zadnji se redak do kraja popunjava nulama u slučaju da nije u potpunosti popunjen.

Dobiveni rezultat se potom pohranjuje kao slika sivih nijansi (engl. grayscale) u PNG formatu koristeći funkciju `Image.fromarray()` iz biblioteke Pillow. Funkcija se poziva sa zastavicom „L“ koja određuje da se slika pohranjuje u sivim nijansama, gdje 0 predstavlja u potpunosti crni piksel, a 255 u potpunosti bijeli piksel.

Na slikama Slika 3.1-Slika 3.9 prikazan je po jedan primjer za svaku od 9 varijanti zlonamjernih programa iz Microsoftovog skupa podataka [19] konvertiranih u PNG format.

U drugim radovima pretvaranje datoteka u slike rađeno je na različite načine. Za potrebe ovog rada odlučeno je da se datoteke mapiraju u slike kvadratnog oblika. Pri učenju modela ih se potom skalira kako bi svi primjeri bili jednakih dimenzija. Važno je

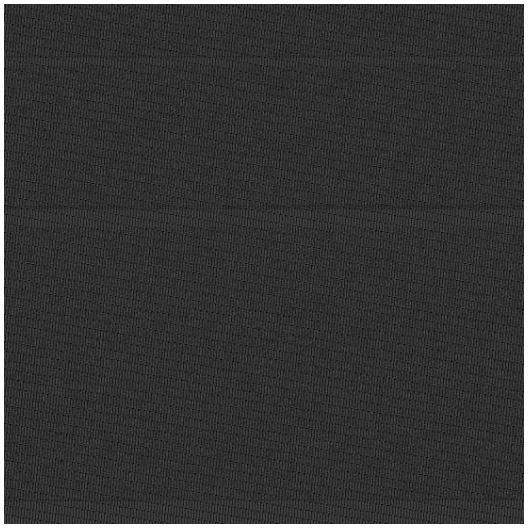
napomenuti da se odabirom ovog pristupa, naspram pristupu gdje se mapira u slike fiksne širine s različitim visinama, gubi dio informacija originalne strukture podataka. Pretpostavka je da će ovo imati negativan utjecaj nad rezultatom. Razlog zašto je ovaj pristup odabran unatoč gubljenju informacija je taj da kvadratni oblik sliku čini spremnom za upotrebu u modelima konvolucijskih neuronskih mreža bez potrebe za većom količinom manipulacija slika.



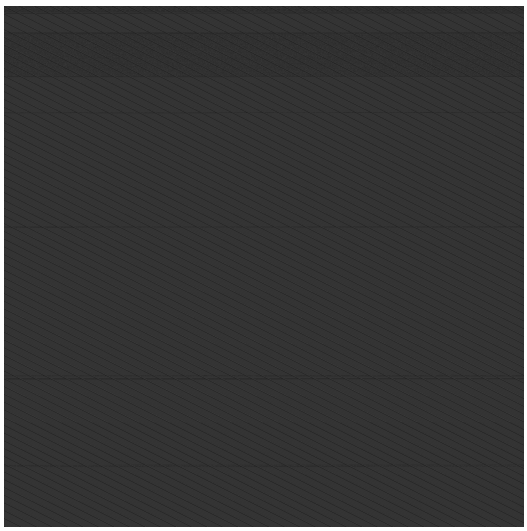
Slika 3.1 Ramnit

Slika 3.2 Lollipop

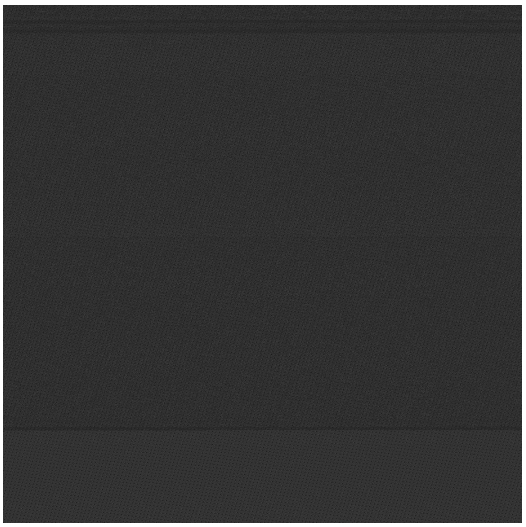
Slika 3.3 Kelihos_ver3



Slika 3.4 Vundo

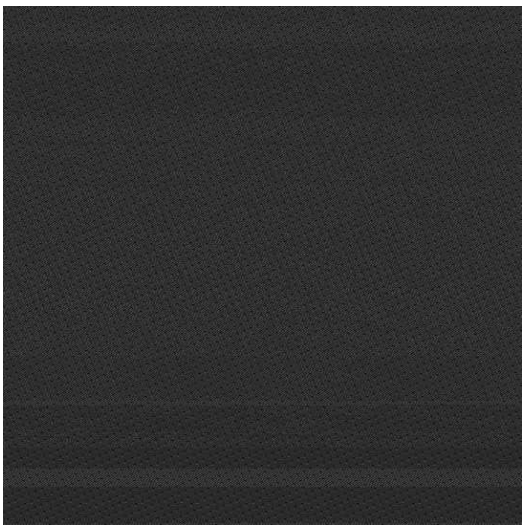


Slika 3.5 Simda

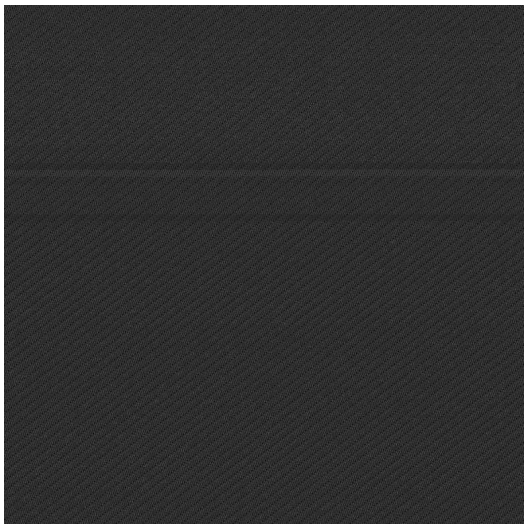


Slika 3.6 Tracur

Slika 3.7 Kelihos_ver1



Slika 3.8 Obfuscator.ACY



Slika 3.9 Gatak

3.2. Opis modela

Procesor računala na kojem se programi izvršavaju je *Intel Core i5-8300H*. Karakteristike procesora prikazane su na Slika 3.10. Grafička kartica je *Nvidia GTX 1050 Ti* te se uz pomoć CUDA-e koristi u svrhu bržeg izvođenja programa opisanog u potpoglavlju 3.3. Verzija CUDA-e koja se koristi je 12.1. Kod je pisan u programskom jeziku Python, verzija 3.12.

Osim ugrađenih, koriste se i slijedeće biblioteke: *numpy* (verzija 1.26.3), *Pillow (PIL)* (verzija 10.3.0), *torch* (verzija 2.4.1+cu121), *torchvision* (verzija 0.19.1+cu121), *sklearn* (verzija 1.5.2), *matplotlib* (verzija 3.9.2), *pandas* (verzija 2.2.2).

Modeli naučeni u sklopu ovog rada temelje se na Resnet-34 i EfficientNet-B0 prednaučenim modelima, prethodno opisanim u potpoglavljima 2.1.2. *ResNet model* i 2.1.3. *EfficientNet model*.

Base speed:	2,30 GHz
Sockets:	1
Cores:	4
Logical processors:	8
Virtualization:	Enabled
L1 cache:	256 KB
L2 cache:	1,0 MB
L3 cache:	8,0 MB

Slika 3.10. Karakteristike procesora Intel Core i5-8300H

Programski kod modela pohranjen je u datoteku *malware_image_cnn.py*.

Program prima 5 ulaznih parametara:

1. Naziv prednaučenog modela – „resnet“ ili „efficientnet“
2. Put do direktorija koji sadrži slikovne datoteke zlonamjernih programa
3. Put do CSV datoteke s oznakama primjera
4. Put do direktorija u koji će se pohraniti naučeni model
5. Put do direktorija u koji će se pohraniti grafovi i rezultati

Kod je pokretan veći broj puta koristeći različite hiperparametre kao što je prikazano u Tablica 3.1. Analiza i usporedba rezultata dobivenih pokretanjem s navedenim razlikama provedena je u poglavlju 4. *Analiza i usporedba rezultata*.

Tablica 3.1. Razlike hiperparametara korištenih tijekom različitih pokretanja koda

model	veličina grupe	stopa učenja
ResNet-34	16	1e-4
ResNet-34	16	1e-5
ResNet-34	32	1e-4
ResNet-34	32	1e-5
EfficientNet-B0	16	1e-4
EfficientNet-B0	16	1e-5
EfficientNet-B0	32	1e-4
EfficientNet-B0	32	1e-5

Ostale, fiksne, karakteristike modela opisane su u nastavku.

Broj klasa modela je 10. Maksimalan broj epoha je 30, omogućeno je rano zaustavljanje u slučaju da je u 5 epoha zaredom gubitak validacije veći ili jednak dotad najmanjem izračunatom gubitku validacije.

Funkcija gubitka je gubitak unakrsne entropije, u kodu:

```
torch.nn.CrossEntropyLoss()
```

Algoritam optimizacije je Adam, u kodu:
`torch.optim.Adam(model.parameters(), lr=<0.0001 ili 0.00001>)`

Koristi se skaler gradijenta `GradScaler()`. Razlog njegove upotrebe je što CUDA prema svojim zadanim postavkama koristi tip podataka *float16* [49]. Pri unatražnom prolazu, vrijednosti gradijenta vrlo malih magnituda potencijalno nisu reprezentativne u tipu podataka *float16* te se pritežu u nulu [50]. Skaler gradijenta pomaže izbjeći navedenu situaciju skalirajući vrijednosti s faktorom čija je zadana vrijednost definirana unutar same funkcije `GradScaler()` iz torch biblioteke [50].

Pri učenju se također koristi i `autocast()` koji omogućuje automatsko prilagođavanje tipa podataka koji se koriste unutar operacija koje se izvode na grafičkoj kartici u svrhu poboljšanja performansi modela [49].

Generalno se `autocast()` i `GradScaler()` koriste u paru [50].

Ako se koristi ResNet, učitava se odabrani model te se potom modificira njegov potpuno povezani sloj kako bi izlaz modela bio jednak broju klasa koje iziskuje skup podataka koji se koristi, tj. 10 klasa. Navedeno se obavlja sa slijedeće dvije linije koda:

```
model = models.resnet34(weights=ResNet34_Weights.DEFAULT)
model.fc = torch.nn.Linear(model.fc.in_features, num_classes)
```

Ako se koristi EfficientNet, učitava se odabrani model te se potom također modificira njegov potpuno povezani sloj iz istog razloga, no u ovom slučaju navedeno se obavlja sa slijedeće dvije linije koda:

```
model = models.efficientnet_b0(
        weights=EfficientNet_B0_Weights.DEFAULT)
model.classifier[1] = torch.nn.Linear(
        model.classifier[1].in_features, num_classes)
```

S obzirom da ulazi oba navedena modela iziskuju slike dimenzija 224x224, potrebne transformacije slika su jednake u oba slučaja. Transformacije su slijedeće:

```
transforms = v2.Compose([
    v2.Resize((224, 224)),
    v2.Compose([v2.ToImage(), v2.ToDtype(torch.float32,
                scale=True)]),
    v2.Normalize(mean=[0.485, 0.456, 0.406],
                 std=[0.229, 0.224, 0.225])
])
```

Nakon provođenja navedenih transformacija nad slikovnim reprezentacijama zlonamjernih programa kao rezultat dobivaju se tenzorske reprezentacije istih.

Skup podataka (tenzori i pripadajuće oznake) odvaja se u skupove za učenje, validaciju i testiranje u omjeru 70:15:15 dvaput koristeći funkciju `train_test_split()`. Koristi se `stratify` kako bi se osiguralo da se sve klase podjele ravnomjerno. U svrhu ponovljivosti izvođenja, `random_state` je prvo postavljen na 26 te potom na 8.

U Kôd 3.1. prikazan je dio koda koji definira proces učenja, tj. finog podešavanja, modela. Izuzete su linije koda koje se tiču računanja gubitka i točnosti. Pri finom podešavanju, na kraju svake epohe, validira se trenutno stanja modela. Kao što je već prethodno navedeno, u slučaju da je gubitak validacije veći ili jednak od prethodno najmanjeg gubitka pet epoha zaredom, proces finog podešavanja rano završava.

```
for epoch in range(num_epochs):
    model.train()
    for images, labels in dataloader_train:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()

        with autocast(device_type):
            model_outputs = model(images)
            loss = loss_function(model_outputs, labels)

        scaler.scale(loss).backward()
        scaler.step(optimizer)
        scaler.update()
```

Kôd 3.1. Učenje, fino podešavanje, modela

Nakon procesa finog podešavanja, dobiveni se model testira na odvojenom skupu podataka.

Rezultati koji se računaju i pohranjuju su slijedeći:

- Ukupno vrijeme učenja
- Broj epoha (zbog potencijalnog ranog zaustavljanja)
- Top-1 točnost za svaku klasu pojedino
- Top-1 pogreška za svaku klasu pojedino
- Top-1 točnost

- Top-1 pogreška
- Top-5 točnost
- Top-5 pogreška
- Lista gubitaka učenja po epohi
- Lista točnosti učenja po epohi
- Lista gubitaka validacije po epohi
- Lista točnosti validacije po epohi

Također se iscrtavaju i pohranjuju slijedeći grafovi:

- Promjena gubitka učenja i validacije po epohama
- Promjena točnosti učenja i validacije po epohama
- Stupčasti graf top-1 točnosti po klasama
- Stupčasti graf top-1 greške po klasama

3.3. Rezultati modela

U slijedećim potpoglavljima, 3.3.1.-3.3.4., prikazani su grafovi dobiveni kao rezultat izvođenja modela i pripadajuće tablice s točnim vrijednostima. Iznosi u tablicama zaokruženi su na četiri decimale. Usporedba navedenih rezultata provedena je u poglavlju 4. *Analiza i usporedba rezultata*.

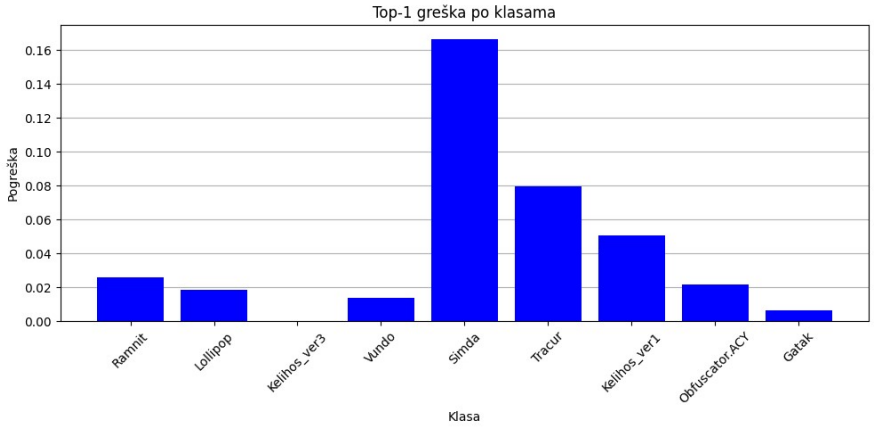
Prikazane su i top-1 greške po klasama, potpoglavlje 3.3.1., i top-1 točnost po klasama, potpoglavlje 3.3.2., iako je prikaz obojeg redundantan. Razlog zašto su unatoč redundantnosti oboje prikazani je što su iznosi grešaka manji naspram iznosa točnosti te se u tim grafovima preciznije može vidjeti i usporediti iznose. Istovremeno, razlog zašto je izdvojeno prikazana i točnost po klasama je kako bi se jasnije mogao prikazati odnos s promjenom točnosti po epohama za koje su također prikazani grafovi u potpoglavlju 3.3.3.

U potpoglavlju 3.3.4. prikazane su promjene gubitka učenja i validacije po epohama.

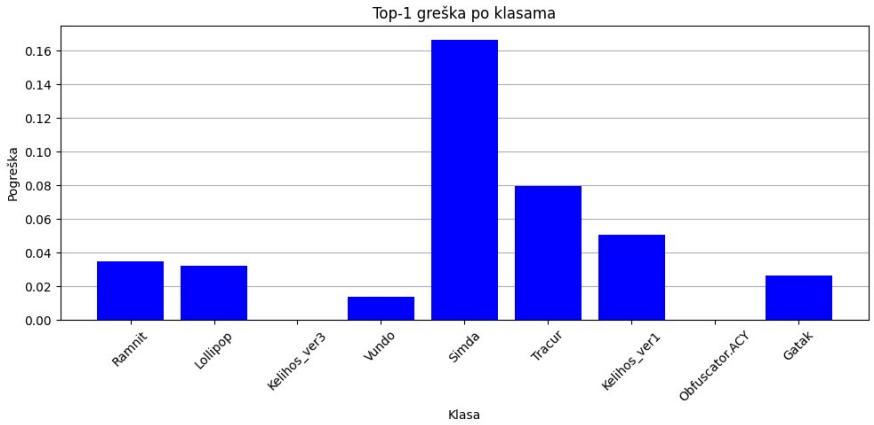
3.3.1. Greška po klasama

Tablica 3.2. Top-1 greške ovisno o modelu, stopi učenja, veličini grupe i vrsti zlonamjernog programa

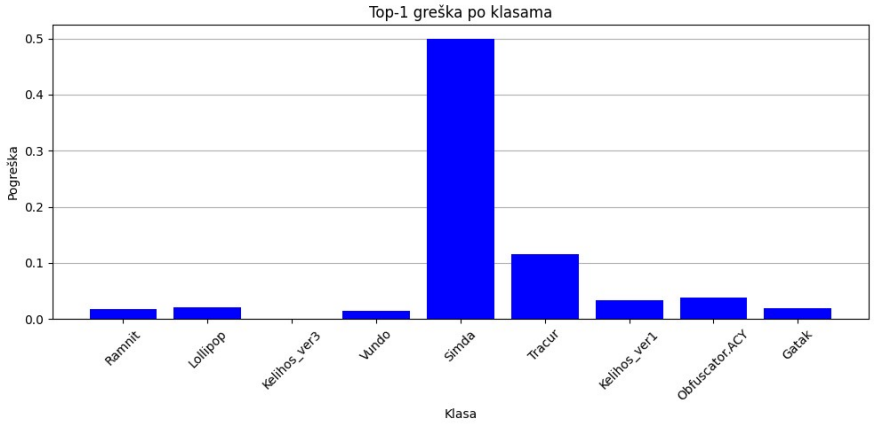
	ResNet lr=1e-5 batch=16	ResNet lr=1e-4 batch=16	ResNet lr=1e-5 batch=32	ResNet lr=1e-4 batch=32	Efficient Net lr=1e-5 batch=16	Efficient Net lr=1e-4 batch=16	Efficient Net lr=1e-5 batch=32	Efficient Net lr=1e-4 batch=32
Ramnit	0.0260	0.0346	0.0173	0.0303	0.0260	0.0260	0.0390	0.0260
Lollipop	0.0188	0.0323	0.0215	0.0161	0.0134	0.0188	0.0108	0.0054
Kelihos_ver3	0.0	0.0	0.0	0.0	0.0	0.0	0.0023	0.0023
Vundo	0.0139	0.0139	0.0139	0.0	0.0139	0.0139	0.0139	0.0139
Simda	0.1667	0.1667	0.5	0.1667	0.1667	0.1667	0.5	0.3333
Tracur	0.0796	0.0796	0.1150	0.0796	0.0885	0.0442	0.0619	0.0442
Kelihos_ver1	0.0508	0.0508	0.0339	0.0169	0.0	0.0169	0.0	0.0
Obfuscator.ACY	0.0217	0.0	0.0380	0.0054	0.0435	0.0217	0.0380	0.0272
Gatak	0.0066	0.0263	0.0197	0.0132	0.0066	0.0526	0.0197	0.0461



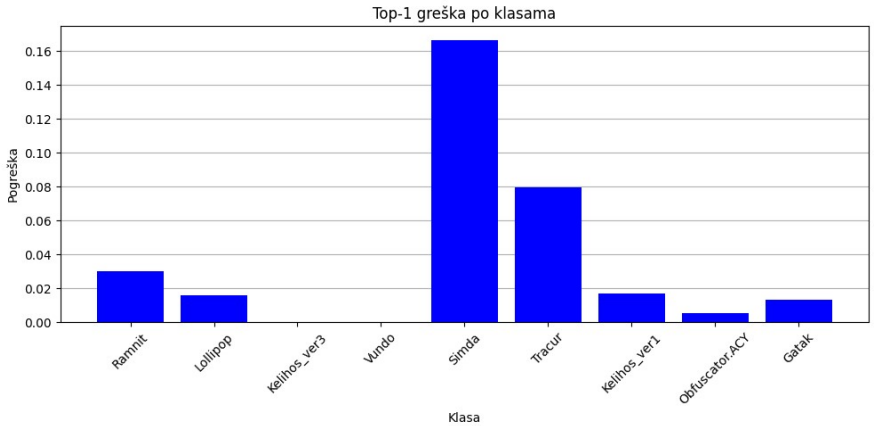
Slika 3.11. ResNet Top-1 greška po klasama sa stopom učenja 1e-5 i veličinom grupe 16



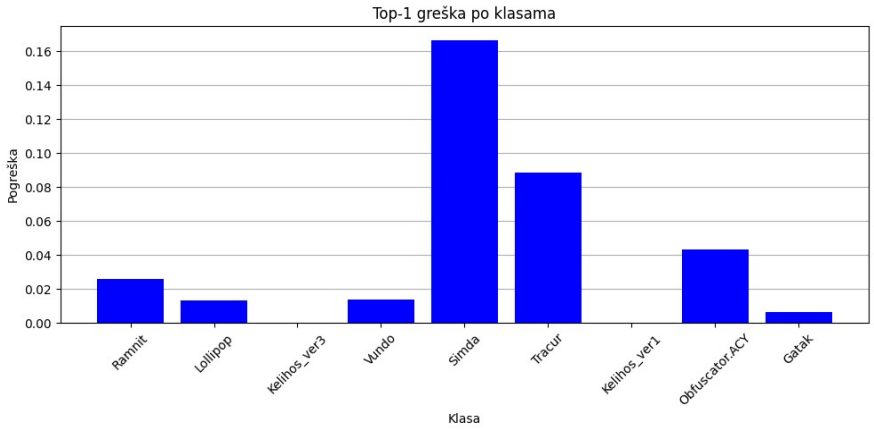
Slika 3.12. ResNet Top-1 greška po klasama sa stopom učenja 1e-4 i veličinom grupe 16



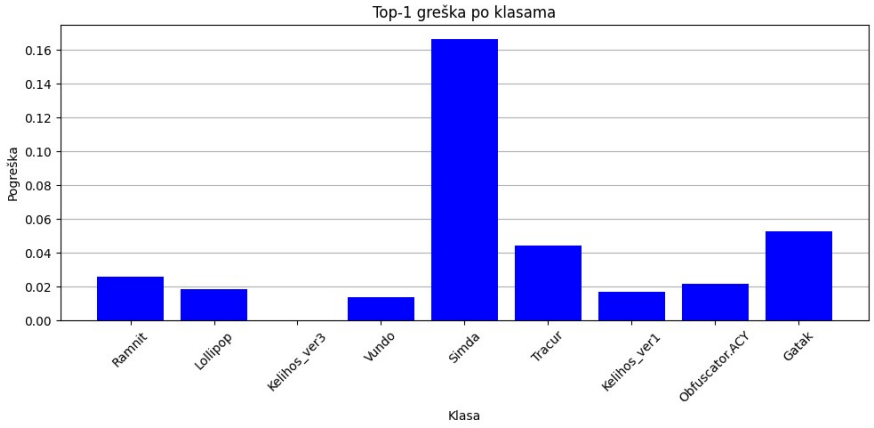
Slika 3.13. ResNet Top-1 greška po klasama sa stopom učenja 1e-5 i veličinom grupe 32



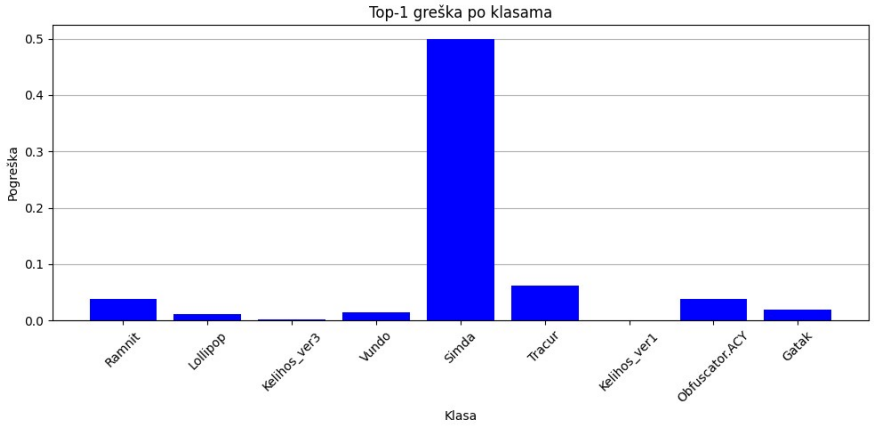
Slika 3.14. ResNet Top-1 greška po klasama sa stopom učenja 1e-4 i veličinom grupe 32



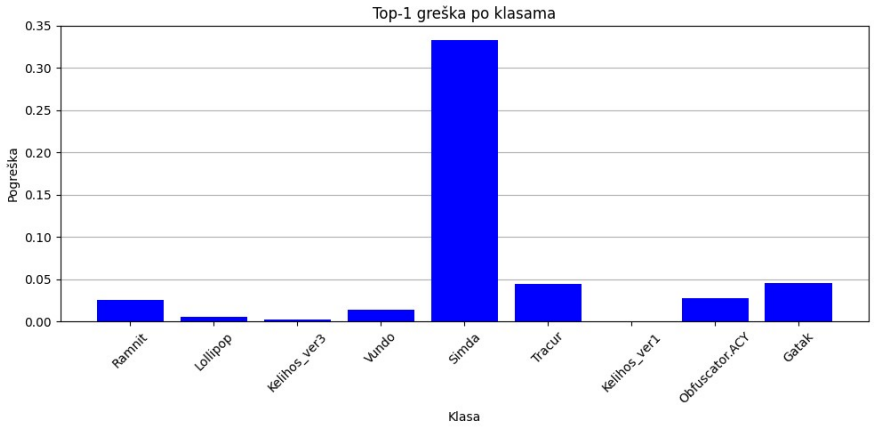
Slika 3.15. EfficientNet Top-1 greška po klasama sa stopom učenja 1e-5 i veličinom grupe 16



Slika 3.16. EfficientNet Top-1 greška po klasama sa stopom učenja 1e-4 i veličinom grupe 16



Slika 3.17. EfficientNet Top-1 greška po klasama sa stopom učenja 1e-5 i veličinom grupe 32

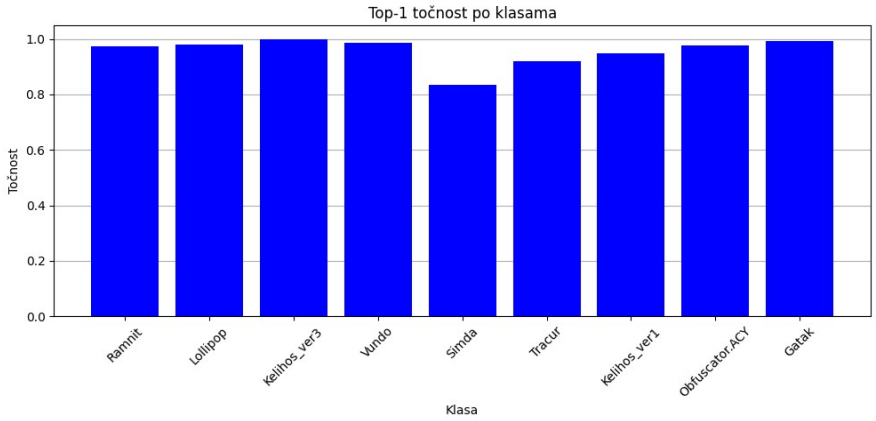


Slika 3.18. EfficientNet Top-1 greška po klasama sa stopom učenja 1e-4 i veličinom grupe 32

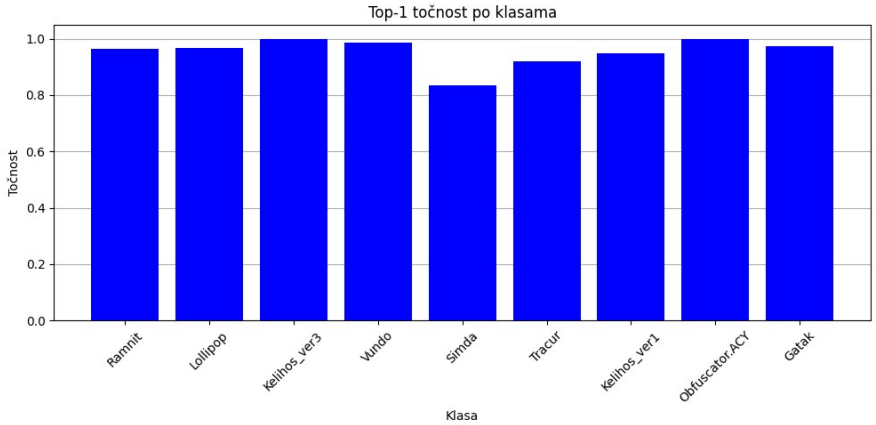
3.3.2. Točnost po klasama

Tablica 3.3. Top-1 točnosti ovisno o modelu, stopi učenja, veličini grupe i vrsti zlonamjernog programa

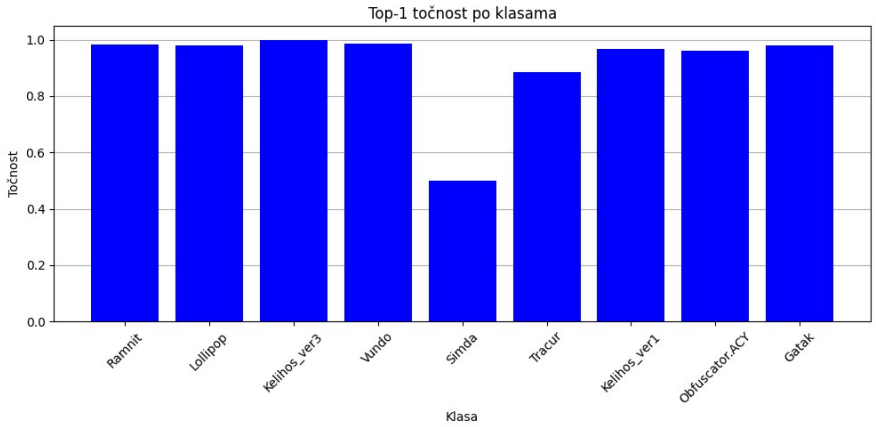
	ResNet lr=1e-5 batch=16	ResNet lr=1e-4 batch=16	ResNet lr=1e-5 batch=32	ResNet lr=1e-4 batch=32	Efficient Net lr=1e-5 batch=16	Efficient Net lr=1e-4 batch=16	Efficient Net lr=1e-5 batch=32	Efficient Net lr=1e-4 batch=32
Ramnit	0.9740	0.9654	0.9827	0.9697	0.9740	0.9740	0.9610	0.9740
Lollipop	0.9812	0.9677	0.9785	0.9839	0.9866	0.9811	0.9892	0.9946
Kelihos_ver3	1.0	1.0	1.0	1.0	1.0	1.0	0.9977	0.9977
Vundo	0.9861	0.9861	0.9861	1.0	0.9861	0.9861	0.9861	0.9861
Simda	0.8333	0.8333	0.5	0.8333	0.8333	0.8333	0.5	0.6667
Tracur	0.9204	0.9204	0.8850	0.9204	0.9115	0.9558	0.9381	0.9558
Kelihos_ver1	0.9492	0.9492	0.9661	0.9831	1.0	0.9831	1.0	1.0
Obfuscator.ACY	0.9783	1.0	0.9620	0.9946	0.9565	0.9783	0.9620	0.9728
Gatak	0.9934	0.9737	0.9803	0.9868	0.9934	0.9474	0.9803	0.9539



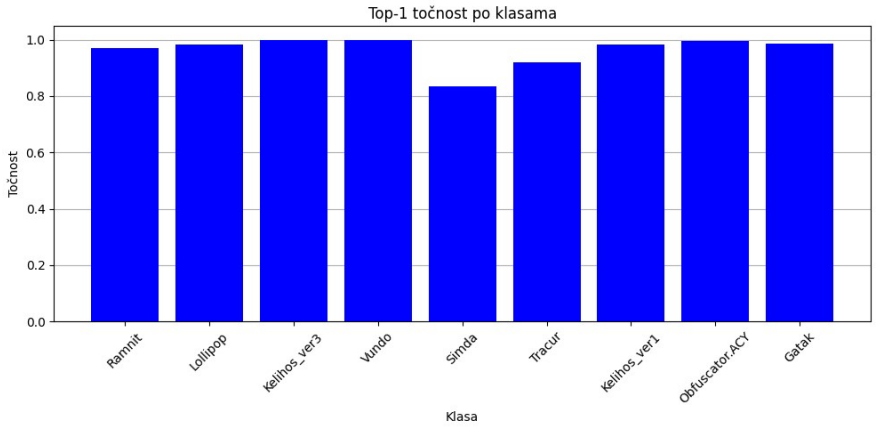
Slika 3.19. ResNet Top-1 točnost po klasama sa stopom učenja 1e-5 i veličinom grupe 16



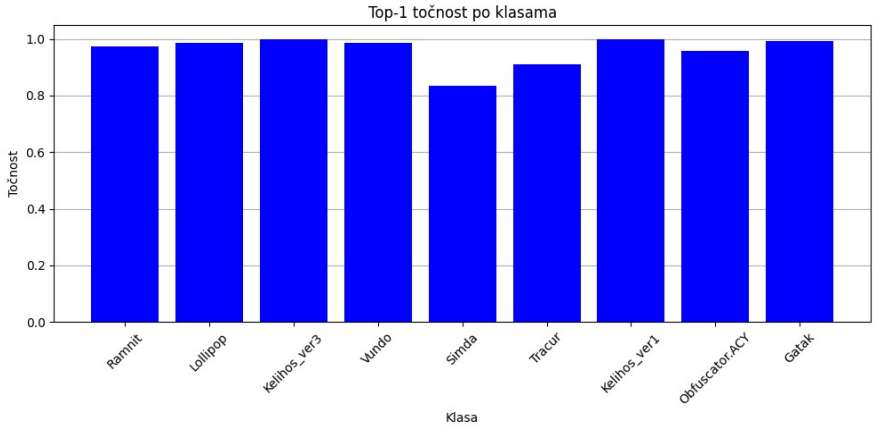
Slika 3.20. ResNet Top-1 točnost po klasama sa stopom učenja 1e-4 i veličinom grupe 16



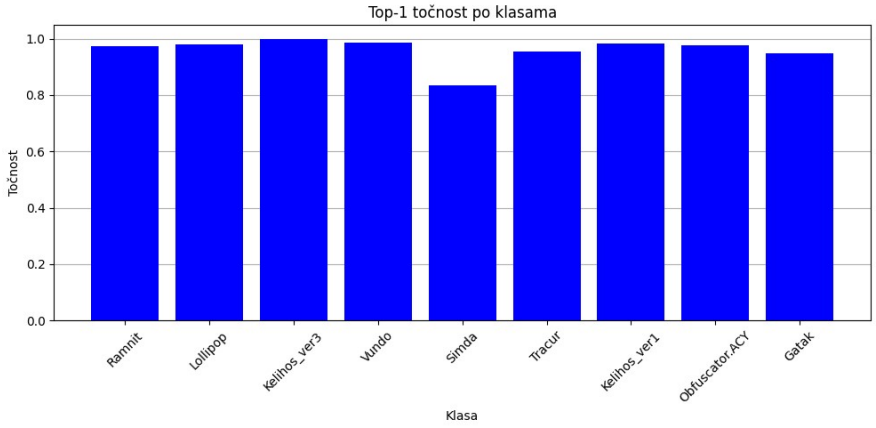
Slika 3.21. ResNet Top-1 točnost po klasama sa stopom učenja 1e-5 i veličinom grupe 32



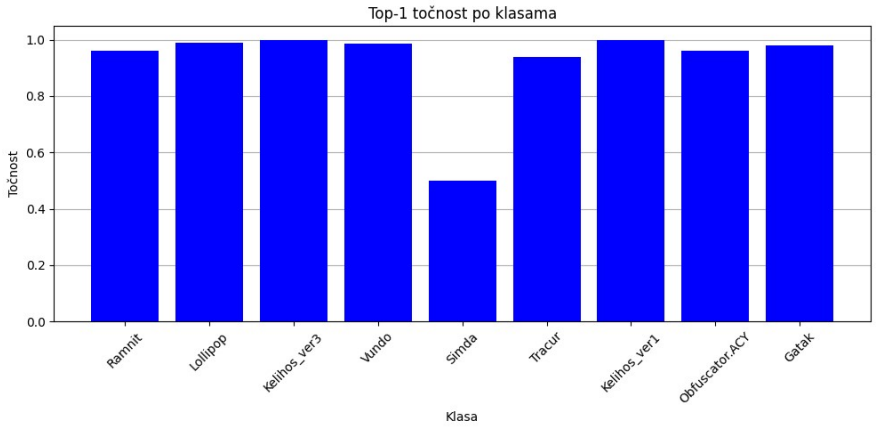
Slika 3.22. ResNet Top-1 točnost po klasama sa stopom učenja 1e-4 i veličinom grupe 32



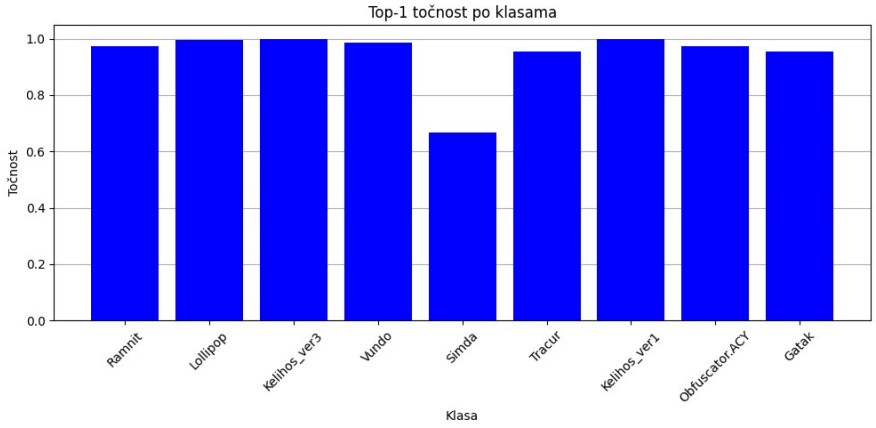
Slika 3.23. EfficientNet Top-1 točnost po klasama sa stopom učenja 1e-5 i veličinom grupe 16



Slika 3.24. EfficientNet Top-1 točnost po klasama sa stopom učenja 1e-4 i veličinom grupe 16



Slika 3.25. EfficientNet Top-1 točnost po klasama sa stopom učenja 1e-5 i veličinom grupe 32

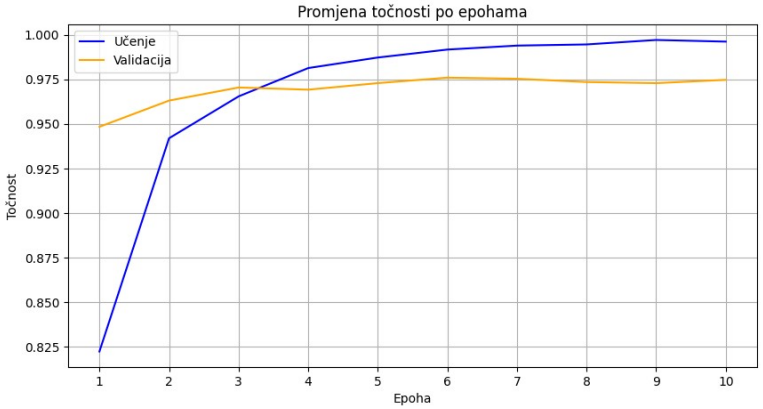


Slika 3.26. EfficientNet Top-1 točnost po klasama sa stopom učenja 1e-4 i veličinom grupe 32

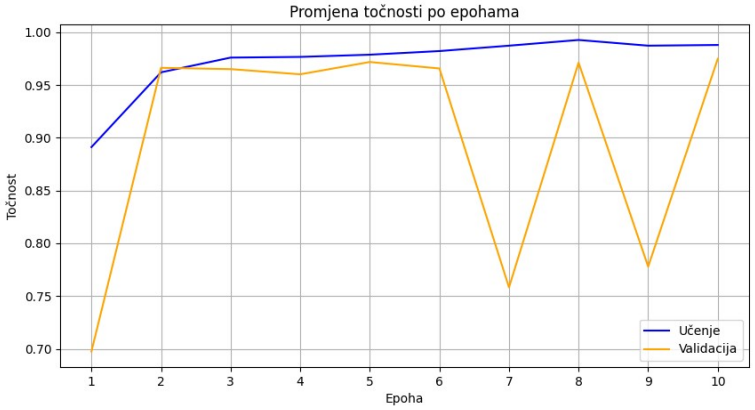
3.3.3. Točnost po epohama

Tablica 3.4. Točnost ovisno o modelu, stopi učenja, veličini grupe u posljednjoj epohi učenja i validacije te pri testiranju

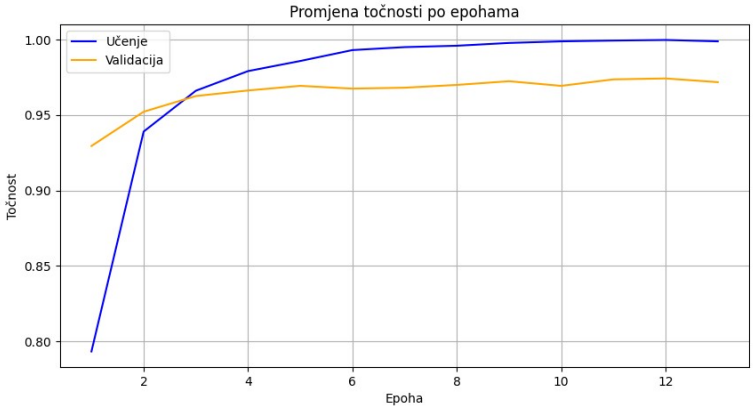
	ResNet lr=1e-5 batch=16	ResNet lr=1e-4 batch=16	ResNet lr=1e-5 batch=32	ResNet lr=1e-4 batch=32	Efficient Net lr=1e-5 batch=16	Efficient Net lr=1e-4 batch=16	Efficient Net lr=1e-5 batch=32	Efficient Net lr=1e-4 batch=32
Učenje	0.9963	0.9879	0.9988	0.9875	0.9924	0.9941	0.9891	0.9950
Validacija	0.9748	0.9748	0.9718	0.9785	0.9755	0.9822	0.9779	0.9742
Testiranje	0.9822	0.9767	0.9749	0.9834	0.9804	0.9798	0.9785	0.9822



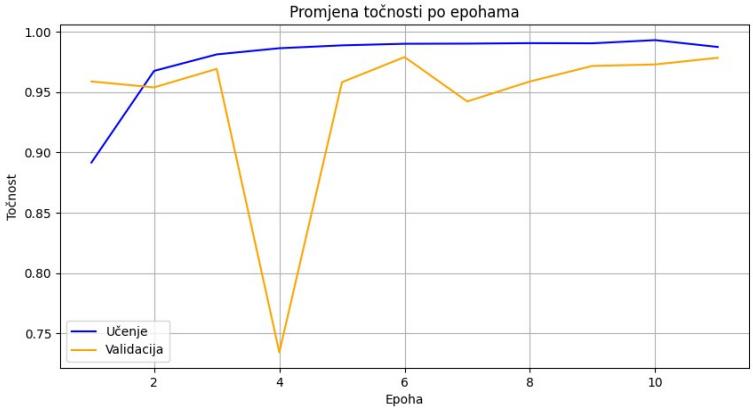
Slika 3.27. ResNet promjena točnosti po epohama sa stopom učenja 1e-5 i veličinom grupe 16



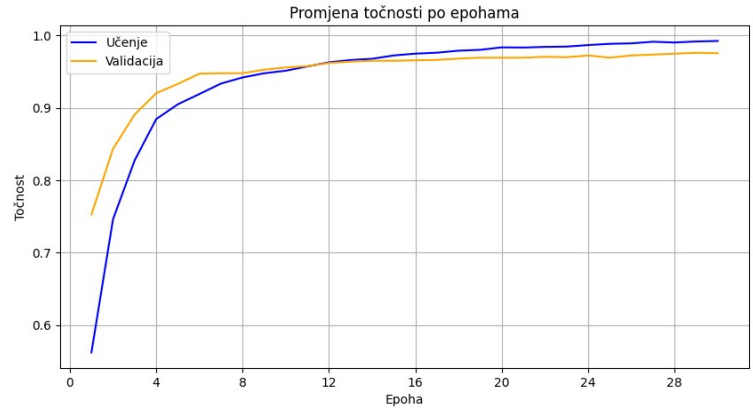
Slika 3.28. ResNet promjena točnosti po epohama sa stopom učenja 1e-4 i veličinom grupe 16



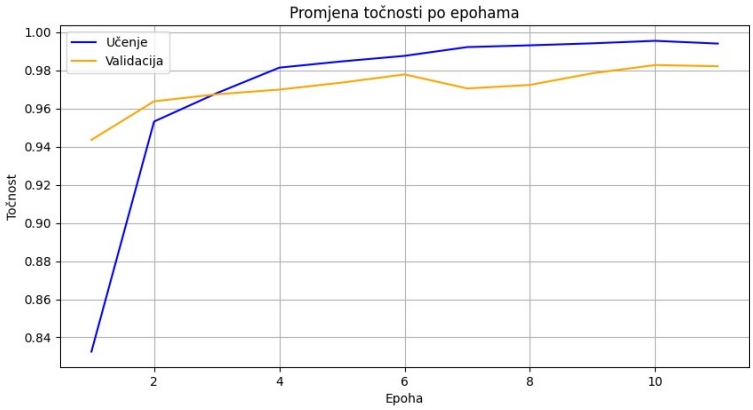
Slika 3.29. ResNet promjena točnosti po epohama sa stopom učenja 1e-5 i veličinom grupe 32



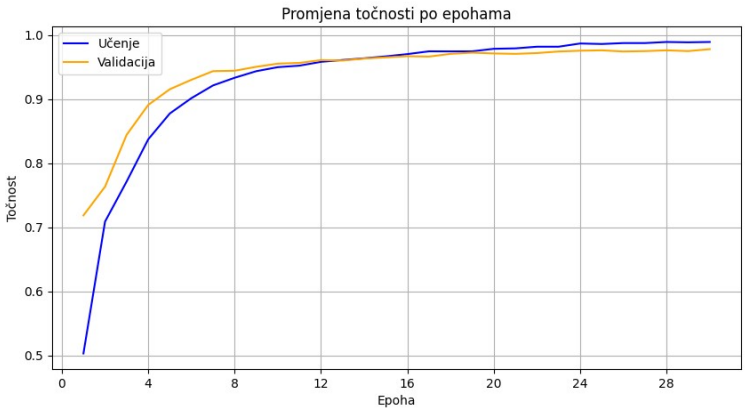
Slika 3.30. ResNet promjena točnosti po epohama sa stopom učenja 1e-4 i veličinom grupe 32



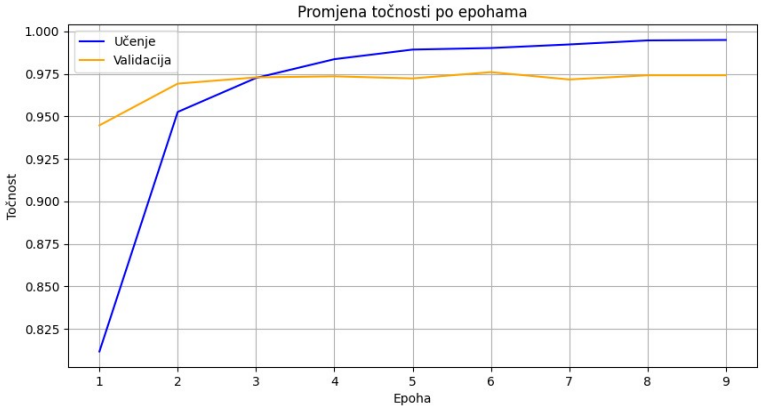
Slika 3.31. EfficientNet promjena točnosti po epohama sa stopom učenja 1e-5 i veličinom grupe 16



Slika 3.32. EfficientNet promjena točnosti po epohama sa stopom učenja 1e-4 i veličinom grupe 16



Slika 3.33. EfficientNet promjena točnosti po epohama sa stopom učenja 1e-5 i veličinom grupe 32

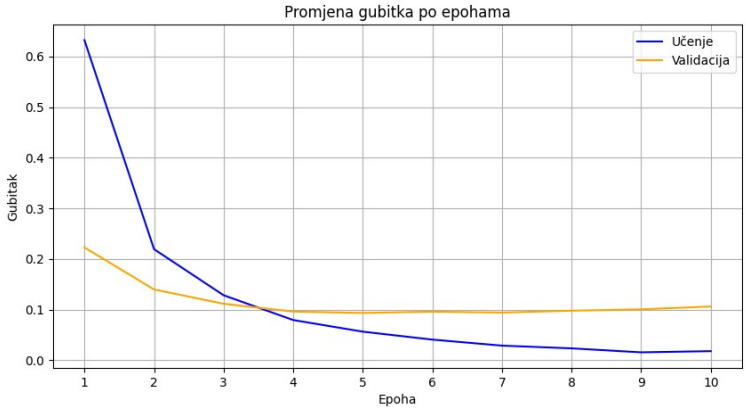


Slika 3.34. EfficientNet promjena točnosti po epohama sa stopom učenja 1e-4 i veličinom grupe 32

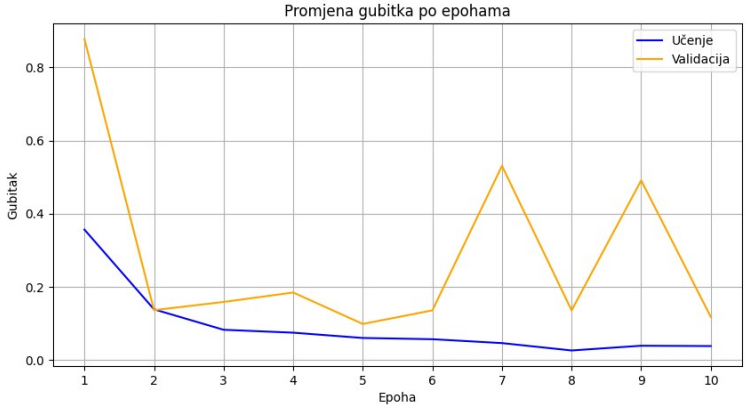
3.3.4. Gubitak po epohama

Tablica 3.5. Gubitak ovisno o modelu, stopi učenja, veličini grupe u posljednjoj epohi učenja i validacije

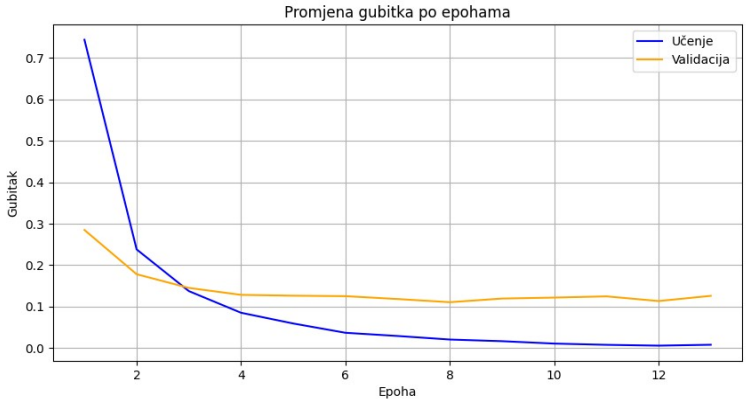
	ResNet lr=1e-5 batch=16	ResNet lr=1e-4 batch=16	ResNet lr=1e-5 batch=32	ResNet lr=1e-4 batch=32	Efficient Net lr=1e-5 batch=16	Efficient Net lr=1e-4 batch=16	Efficient Net lr=1e-5 batch=32	Efficient Net lr=1e-4 batch=32
Učenje	0.0179	0.0385	0.0082	0.0399	0.0294	0.0239	0.0380	0.0171
Validacija	0.1060	0.1182	0.1261	0.0878	0.0961	0.0871	0.0958	0.1020



Slika 3.35. ResNet promjena gubitka po epohama sa stopom učenja 1e-5 i veličinom grupe 16

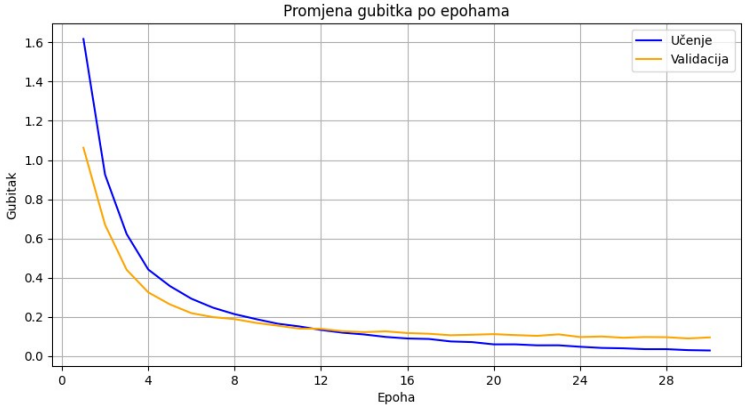


Slika 3.36. ResNet promjena gubitka po epohama sa stopom učenja 1e-4 i veličinom grupe 16



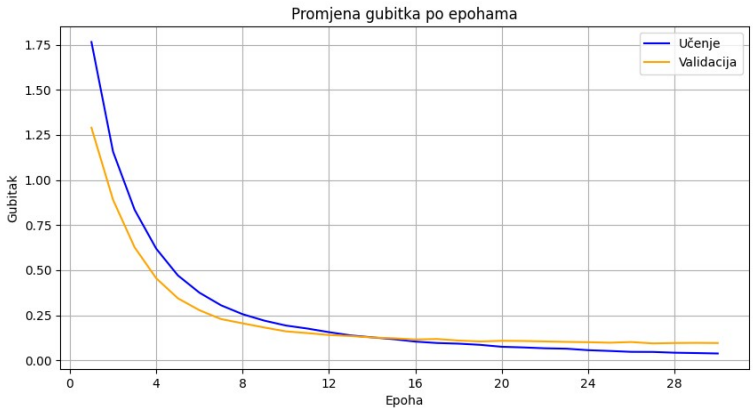
Slika 3.37. ResNet promjena gubitka po epohama sa stopom učenja 1e-5 i veličinom grupe 32

Slika 3.38. ResNet promjena gubitka po epohama sa stopom učenja 1e-4 i veličinom grupe 32

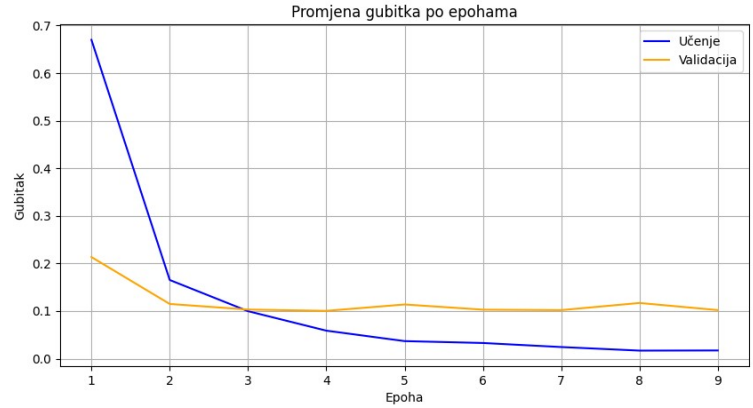


Slika 3.39. EfficientNet promjena gubitka po epohama sa stopom učenja 1e-5 i veličinom grupe 16

Slika 3.40. EfficientNet promjena gubitka po epohama sa stopom učenja 1e-4 i veličinom grupe 16



Slika 3.41. EfficientNet promjena gubitka po epohama sa stopom učenja 1e-5 i veličinom grupe 32



Slika 3.42. EfficientNet promjena gubitka po epohama sa stopom učenja 1e-4 i veličinom grupe 32

4. Analiza i usporedba rezultata

Temeljem rezultata iz potpoglavlja 3.3. *Rezultati modela*, provedena je analiza i usporedba.

Uspoređujući točnosti po epohama prema vrijednostima iz Tablica 3.1. i uzimajući u obzir isključivo točnosti iz skupa za testiranje, najvišu točnost od 98.34% ima ResNet sa stopom učenja 1e-4 i veličinom grupe 32. Ako se u obzir uzima isključivo EfficientNet, varijacija s najboljom točnošću od 98.22% je varijacija sa stopom učenja 1e-4 i veličinom grupe 32. Dakle i ResNet i EfficientNet rezultiraju najboljom točnošću s jednakim hiperparametrima.

Pad točnosti u 4. epohi kod ResNet modela sa stopom učenja 1e-4 i veličinom grupe 32, Slika 3.30., ne ukazuje nužno na problem s obzirom da se odmah u slijedećoj epohi vraća na otprilike jednaku točnost koju je model imao u prethodnoj epohi. Ipak, s obzirom da je pad točnosti velik, s više od 95% na ispod 75%, navedeno potencijalno ukazuje na to da je stopa učenja prevelika. Slično se događa kod ResNet modela sa stopom učenja 1e-4 i veličinom grupe 16, Slika 3.28., s tim da u ovom slučaju postoje dva nagla pada, u 7. i 9. epohi, a u 10. se epohi fino podešavanje rano zaustavlja. S obzirom da navedene dvije varijacije ResNet modela imaju različite veličine grupa i jednake stope učenja te da se u preostalim varijacijama modela ovakvi padovi točnosti ne događaju, uključujući i EfficientNet modele, ovo pridodaje prethodnoj tezi da je stopa učenja vrlo vjerojatno prevelika kod ResNet modela. Navedeno ukazuje da varijanta ResNet modela s najvišom točnošću skupa za testiranje nije najbolji odabir.

Varijanta ResNet modela sa stopom učenja 1e-5 i veličinom grupe 32, naspram svih varijacija modela, ima najvišu točnost na skupu za učenje, a najmanje na skupovima na validaciju i testiranje što ukazuje na to da je potencijalno prenaučena.

Kada je riječ o varijacijama ResNet modela, eliminacijom prethodno navedene tri varijacije, kao najsigurniji odabir preostaje varijacija sa stopom učenja 1e-5 i veličinom grupe 16. Navedena varijacija je pri finom podešavanju rano zaustavljena nakon 10. epohe, Slika 3.27., a na skupu za učenje ima točnost od 99.63%, na skupu za validaciju 97.48% te na skupu za testiranje 98.22%, što je druga po redu najviša točnost.

U slučaju varijacija EfficientNet modela, posebnih odstupanja, kao što je slučaj kod ResNet modela, nema. Iz grafova sa slika Slika 3.31., Slika 3.32., Slika 3.33. i Slika 3.34.

vidljivo je da u slučajevima kada je stopa učenja 1e-5 nema ranog zaustavljanja, dok se pri stopi učenja 1e-4 fino podešavanje rano zaustavlja nakon 9. i 11. epohe.

S obzirom da je za rano zaustavljene varijante vjerojatnost da su prenaučene manja, u obzir se uzimaju varijante sa stopom učenja 1e-4. Varijanta s veličinom grupe 32 ima veću točnost na skupu za učenje od 99.50% naspram 99.41% varijante s veličinom grupe 16, no isto tako ima i veću točnost na skupu za testiranje od 98.22% naspram 97.98%, što varijantu s veličinom grupe 32 čini boljim odabirom. Navedeno potvrđuje prvi odabir s početka potpoglavlja da je u slučaju varijanti ResNet modela najbolji odabir varijanta sa stopom učenja 1e-4 i veličinom grupe 32.

Sveukupno sagledano, odabrane varijante ResNet i EfficientNet modela imaju jednaku točnost na skupu za testiranje od 98.22%, dok kod skupa za učenje imaju točnosti od 99.63% i 99.50% redom. Navedeno čini varijante modela ResNet sa stopom učenja 1e-5 i veličinom grupe 16 te EfficientNet sa stopom učenja 1e-4 i veličinom grupe 32 podjednako dobrim i najboljim odabirom od navedenih u potpoglavlju 3.3.

Iz Tablica 3.5. i grafova na slikama Slika 3.35.-Slika 3.42. vidljivo je da sve prethodno navedeno jednako vrijedi i u slučaju analiziranja i usporedbe gubitaka.

Uspoređujući top-1 točnosti po klasama iz Tablica 3.3. do izražaja dolaze Kelihos_ver3 i Simda. Kelihos_ver3 iz razloga što ima točnost od 100% na skupu za testiranje u šest od osam varijacija modela, dok u preostale dvije ima točnost od 99.77%. Simda iz u potpunosti suprotnog razloga, u svim varijacijama modela ima daleko najnižu točnost – 50% u najgorem, a 83.33% u najboljem slučaju.

Temeljem top-1 točnosti po klasama može se odrediti koja je od prethodno odabrane dvije varijacije modela zapravo najbolji odabir.

Varijanta ResNet modela sa stopom učenja 1e-5 i veličinom grupe 16 ima prosječnu točnost po grupama, izuzev Simde, 97.28%, dok varijanta EfficientNet modela sa stopom učenja 1e-4 i veličinom grupe 32 ima malo višu prosječnu točnost, 97.94%. Unatoč tome, s obzirom da su točnosti Simde 83.33% i 66.67% redom, što je velika razlika, varijanta ResNet modela je bolji odabir.

Zaključno, od osam varijacija, najbolji odabir je varijacija ResNet modela sa stopom učenja 1e-5 i veličinom grupe 16.

Dodatna napomena, iako se inače koristi kao jedna od stavki za usporedbu modela, Top-5 točnost i greška izostavljeni su u potpoglavlju 3.3. *Rezultati modela* i u ovom potpoglavlju. Razlog zašto je da korišteni skup podataka sadrži 9 različitih klasa te u ovom slučaju usporedba Top-5 točnosti i grešaka nema pretjeranog smisla s obzirom da uključuje više od polovice mogućih klasa.

Zaključak

Temeljem provedene analize i usporedbe dobivenih rezultata pristupa koji se temelji na upotrebi konvolucijskih neuronskih mreža za detekciju i klasifikaciju zlonamjernih programa s prednaučenim modelima kao polaznom točkom, može se ustvrditi da je navedeni pristup potencijalno prikladan za tu svrhu.

Iako je ovom pristupu točnost manja od pristupa koji su danas u upotrebi, točnost modela je visoka, 98.22%, te se može reći da pristup nije za odbaciti, ali i da ima prostora za napredak. Zaključak proizlazi iz činjenice da je točnost dobivena minimalnim izmjenama prednaučenih modela, točnije, modifikacijom isključivo potpuno povezanog sloja. Točnost se potencijalno može dodatno povećati dodatnim modifikacijama ostalih slojeva modela. Također treba uzeti u obzir da su u ovom radu zbog ograničenih računalnih resursa korištene manje precizne varijante ResNet-34 i EfficientNet-B0. Upotreba složenijih varijanti modela poput ResNet-152 i EfficientNet-B7 trebala bi rezultirati većom točnošću. Uz navedeno, osim modela ResNet i EfficientNet postoji velik broj drugih prednaučenih modela koje bi trebalo analizirati za ovu svrhu pošto također postoji mogućnost da bi rezultirali većom točnošću.

Potrebno je naglasiti da neovisno o potencijalu koji postoji za daljnja istraživanja ovog pristupa, kad je riječ o detekciji zlonamjernih programa, točnost detekcije i klasifikacije mora biti čim bliža 100%. Ipak, pristup ne treba biti savršen. Čak i u slučaju da se ovaj pristup, ili njemu sličan, u nekom trenutku počne koristiti u sklopu antivirusnih i ostalih programa tog tipa, zasigurno se ne bi koristio samostalno, već paralelno s ostalim pristupima koji se trenutno koriste.

Dodatno, iako ima manju točnost i potencijalno ne detektira određene varijante zlonamjernih programa koje bi drugi pristupi detektirali, isto tako bi potencijalno mogao detektirati određene varijante koje drugi pristupi ne bi. Iz ovog slijedi da bi se kombinacijom pristupa potencijalno međusobno pokrili postojeći nedostatci te bi se ukupan postotak ispravnih detekcija povećao, a istovremeno bi se smanjila i količina lažnih pozitiva i lažnih negativa.

Literatura

[1] *Malware | What is Malware & How to Stay Protected from Malware Attacks*, Palo Alto Networks, (nepoznato). Poveznica: https://www.paloaltonetworks.com/cyberpedia/what-is-malware; pristupljeno: 12. rujna 2024.

[2] *What is malware?*, IBM, (nepoznato). Poveznica: https://www.ibm.com/topics/malware; pristupljeno: 12. rujna 2024.

[3] Lenaerts-Bergmans B., *What is a computer worm?*, CrowdStrike, (2023, srpanj). Poveznica: https://www.crowdstrike.com/cybersecurity-101/malware/computer-worm/; pristupljeno: 1. lipnja 2024.

[4] What Is a Worm Virus?, Fortinet, (nepoznato). Poveznica: https://www.fortinet.com/resources/cyberglossary/worm-virus; pristupljeno: 1. lipnja 2024.

[5] *virus, računalni*, Hrvatska enciklopedija, mrežno izdanje, Leksikografski zavod Miroslav Krleža (2013-2024). Poveznica: https://www.enciklopedija.hr/clanak/virus-racunalni; pristupljeno: 1. lipnja 2024.

[6] Lenaerts-Bergmans B., *What is Adware and How To Remove It*, CrowdStrike, (2023, svibanj). Poveznica: https://www.crowdstrike.com/cybersecurity-101/adware/; pristupljeno: 1. lipnja 2024.

[7] *ADWARE*, Malwarebytes, (2023, kolovoz). Poveznica: https://www.malwarebytes.com/adware; pristupljeno: 1. lipnja 2024.

[8] Olds D., *PC infected notifications in Windows 10*, IT Office Blog, The Queen's College, Oxford, (2021, srpanj). Poveznica: https://it.queens.ox.ac.uk/2021/07/22/pc-infected-notifications-in-windows-10/; pristupljeno: 1. lipnja 2024.

[9] Baker K., *What is a Trojan Horse?*, CrowdStrike, (2022, lipanj). Poveznica: https://www.crowdstrike.com/cybersecurity-101/malware/trojans/, pristupljeno: 1. lipnja 2024.

[10] *Trojan horse - Virus or malware?*, Malwarebytes, (2023, kolovoz). Poveznica: https://www.malwarebytes.com/trojan; pristupljeno: 1. lipnja 2024.

[11] *Trojan-Downloader*, F-Secure, (nepoznato). Poveznica: https://www.f-secure.com/v-descs/trojan-downloader.shtml; pristupljeno: 1. lipnja 2024.

[12] S12 - H4CK, *Trojan Downloader Malware*, Medium (2023, kolovoz). Poveznica: https://medium.com/@s12deff/trojan-downloader-malware-48711ccab225; pristupljeno: 1. lipnja 2024.

[13] *Backdoor computing attacks*, Malwarebytes, (2023, kolovoz). Poveznica: https://www.malwarebytes.com/backdoor; pristupljeno: 1. lipnja 2024.

[14] *Backdoor Attack*, imperva, (nepoznato). Poveznica: https://www.imperva.com/learn/application-security/backdoor-shell-attack/; pristupljeno: 1. lipnja 2024.

[15] Lenaerts-Bergmans B, *Backdoor Attacks*, CrowdStrike, (2023, srpanj). Poveznica: https://www.crowdstrike.com/cybersecurity-101/attack-types/backdoor-attack/; pristupljeno: 1. lipnja 2024.

[16] Cannell J., *Obfuscation: Malware's best friend*, Malwarebytes, (2013, ožujak). Poveznica: https://www.malwarebytes.com/blog/news/2013/03/obfuscation-malwares-best-friend; pristupljeno: 1. lipnja 2024.

[17] Virgillito D., *What is Malware Obfuscation?*, InfoSec Institute, (2020, veljača). Poveznica: https://www.infosecinstitute.com/resources/malware-analysis/what-is-malware-obfuscation/; pristupljeno: 1. lipnja 2024.

[18] Ait Ichou M., *Malware Obfuscation*, Medium, (2023. listopad). Poveznica: https://medium.com/@aitichoumustapha/malware-obfuscation-55fb2f933469; pristupljeno: 1. lipnja 2024.

[19] Ronen R, Radu M., Feuerstein C., Yom-Tov E., Ahmadi M., *Microsoft Malware Classification Challenge*. arXiv:1802.10135, (2018)

[20] *Guess who's back? Virus.Ramnit is here (to stay?)*, ReasonLabs, (2020, lipanj). Poveznica: https://reasonlabs.com/blog/theyre-baaack-virus-ramnit-and-its-harmful-programs; pristupljeno: 2. lipnja 2024.

[21] *Botnet taken down through international law enforcement cooperation*, Europol, (2015, veljača). Poveznica: https://www.europol.europa.eu/media-press/newsroom/news/botnet-taken-down-through-international-law-enforcement-cooperation; pristupljeno: 2. lipnja 2024.

[22] The Associated Press, *EU police operation takes down malicious computer network*, Phys.org (2015, veljača). Poveznica: https://phys.org/news/2015-02-eu-police-malicious-network.html; pristupljeno: 2. lipnja 2024.

[23] *Adware:Win32/Lollipop*, Microsoft, (2013, srpanj). Poveznica: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Adware:Win32/Lollipop&threatId=198706; pristupljeno: 2. lipnja 2024.

[24] *Trojan:Win32/Vundo.gen!AW*, Microsoft, (2011, svibanj). Poveznica: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Vundo.gen!AW; pristupljeno: 2. lipnja 2024.

[25] *Trojan.Vundo*, FireEye, (nepoznato). Poveznica: https://mil.fireeye.com/edp.php?sname=Trojan.Vundo; pristupljeno: 2. lipnja 2024.

[26] *Win32/Tracur*, Microsoft, (2011, lipanj). Poveznica: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=win32%2Ftracur pristupljeno: 2. lipnja 2024.

[27] *Win32/Simda*, Microsoft, (2013, rujan). Poveznica: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32/Simda; pristupljeno: 2. lipnja 2024.

[28] *Backdoor:W32/Simda*, F-Secure, (nepoznato). Poveznica: https://www.f-secure.com/v-descs/backdoor-w32-simda.shtml; pristupljeno: 2. lipnja 2024.

[29] Stone-Gross B., Werner T., Hartley B., *Farewell to Kelihos and ZOMBIE SPIDER*, CrowdStrike, (2018, prosinac). Poveznica:

https://www.crowdstrike.com/blog/farewell-to-kelihos-and-zombie-spider/; pristupljeno: 2. lipnja 2024.

[30] *Backdoor.Kelihos*, F-Secure, (nepoznato). Poveznica: https://www.f-secure.com/v-descs/backdoor-w32-kelihos.shtml; pristupljeno: 2. lipnja 2024.

[31] *Win32/Kelihos*, Microsoft, (2011, rujan). Poveznica: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=win32%2Fkelihos; pristupljeno: 2. lipnja 2024.

[32] Cimpanu C., *Keygen Websites Spreading Gatak Backdoor Trojan*, Bleeping Computer, (2016, studeni). Poveznica: https://www.bleepingcomputer.com/news/security/keygen-websites-spreading-gatak-backdoor-trojan/; pristupljeno: 2. lipnja 2024.

[33] Dell SecureWorks Counter Threat Unit Threat Intelligence, *Stegoloader: A Stealthy Information Stealer*, Secureworks, (2015, lipanj). Poveznica: https://www.secureworks.com/research/stegoloader-a-stealthy-information-stealer; pristupljeno: 2. lipnja 2024.

[34] Albawi S., Mohammed T.A., Al-Zawi S., *Understanding of a Convolutional Neural Network*. ICET 2017, Antalya, Turska, (2017)

[35] Potrimba P., *What is a Convolutional Neural Network?*, Roboflow, (2023, veljača). Poveznica https://blog.roboflow.com/what-is-a-convolutional-neural-network/; pristupljeno: 31. kolovoza 2024.

[36] *Convolutional Neural Network*, Nvidia, (2023, svibanj). Poveznica: https://www.nvidia.com/en-us/glossary/convolutional-neural-network/; pristupljeno: 31. kolovoza 2024.

[37] Gomede E., *The Power and Potential of Pretrained and Prebuilt Models in Machine Learning*, Medium, (2024, siječanj). Poveznica: https://medium.com/the-modern-scientist/the-power-and-potential-of-pretrained-and-prebuilt-models-in-machine-learning-4d4948a62a28; pristupljeno: 7. rujna 2024.

[38] Lee A., *What Is a Pretrained AI Model?*, Nvidia, (2022, prosinac). Poveznica: https://blogs.nvidia.com/blog/what-is-a-pretrained-ai-model/; pristupljeno: 7. rujna 2024.

[39] *Pre Trained Models for Image Classification – PyTorch for Beginners*, LearnOpenCV, Big Vision, (2019, lipanj). Poveznica: https://learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/; pristupljeno: 7. rujna 2024.

[40] Masood D., *Pre-trained CNN architectures designs, performance analysis and comparison*, Medium , (2023, listopad). Poveznica: https://medium.com/@daniyalmasoodai/pre-train-cnn-architectures-designs-performance-analysis-and-comparison-802228a5ce92; pristupljeno: 7. rujna 2024.

[41] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A. C., Fei-Fei L., *ImageNet Large Scale Visual Recognition Challenge*. IJCV, (2015)

[42] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L., *ImageNet: A Large-Scale Hierarchical Image Database*. IEEE Computer Vision and Pattern Recognition (CVPR), Miami, Florida, SAD, (2009)

[43] *What is WordNet?*, WordNet, Princeton University, (2010). Poveznica: https://wordnet.princeton.edu/; pristupljeno 11. rujna 2024.

[44] Solawetz J., *Use Resnet34 for Image Classification*, Roboflow, (2020, rujan). Poveznica: https://blog.roboflow.com/custom-resnet34-classification-model/; pristupljeno: 7. rujna 2024.

[45] He K., Zhang X., Ren S., Sun J., *Deep Residual Learning for Image Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Nevada, SAD, (2016), str. 770-778.

[46] Potrimba P., *What is EfficientNet? The Ultimate Guide.*, Roboflow, (2023, kolovoz). Poveznica: https://blog.roboflow.com/what-is-efficientnet/; pristupljeno: 7. rujna 2024.

[47] DhanushKumar, *EfficientNet — Scaling Depth,Width,Resolution*, Medium, (2024, ožujak). Poveznica: https://medium.com/@danushidk507/efficientnet-scaling-depth-width-resolution-11e2d4311357; pristupljeno: 7. rujna 2024.

[48] Tan M., Le Q.V., EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946v5, (2020)

[49] *Automatic Mixed Precision examples*, PyTorch, PyTorch documentation, (nepoznato). Poveznica: https://pytorch.org/docs/stable/notes/amp_examples.html; pristupljeno: 14. rujna 2024.

[50] *Automatic Mixed Precision package - torch.amp*, PyTorch, PyTorch documentation, (nepoznato). Poveznica: https://pytorch.org/docs/stable/amp.html; pristupljeno: 14. rujna 2024.

Sažetak

Naslov: Sustav za detekciju zlonamjernih programa temeljen na konvolucijskim neuronskim mrežama

Ključne riječi: zlonamjerni programi; konvolucijske neuronske mreže; detekcija; klasifikacija; prednaučeni modeli; ResNet; EfficientNet

Sažetak: Kao posljedica sve veće digitalizacije svih sfera života, isplativost ucjene i krađe podataka raste. Paralelno tome upotreba zlonamjernih programa i pojava različitih varijanti su u uzlaznom trendu. Postoji više različitih pristupa problemu detekcije i klasifikacije zlonamjernih programa, neki od kojih uključuju strojno učenje. U radu je opisan sustav za detekciju zlonamjernih programa temeljen na konvolucijskim neuronskim mrežama. Kao polazišne točke koriste se prednaučeni modeli ResNet i EfficientNet. Prikazani su dobiveni rezultati te je provedena njihova analiza i usporedba.

Summary

Title: Convolutional Neural Network Based Malware Detection System

Keywords: malware; convolutional neural networks; detection; classification; pretrained models; ResNet; EfficientNet

Summary: As a consequence of the increasing rate of digitalisation of all our spheres of life, it is becoming more and more profitable to try to steal data and to attempt extortion. Alongside this, the usage of malware and the appearance of new variants show a rising trend. There are multiple different approaches to malware detection and classification problems, some of which are based on machine learning. In this thesis, a malware detection system based on convolutional neural networks is described. Pretrained models ResNet and EfficientNet are used as starting points. The obtained results are presented and an analysis and comparison are made between them.