SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Evaluacija metoda napada i detekcije napada na duboke modele

Luka Družijanić
Voditelj: *prof. dr. sc. Siniša Šegvić*

Zagreb, svibanj 2023.

SADRŽAJ

1. Uvod **1**

2. Metode napada **3**

2.1. BadNets 3

2.2. WaNet 4

3. Metode obrane **6**

3.1. Neural Cleanse 6

3.2. Activation Clustering 7

4. Eksperimenti **10**

5. Rezultati **11**

6. Zaključak **12**

7. Literatura **13**

8. Sažetak **15**

1. Uvod

Duboki modeli su danas ključni u raznim domenama, od računalnog vida i generiranja slika, do razumijevanja govora i prevođenja teksta. Međutim, kako bi postigli zadovoljavajuće performanse, duboki modeli zahtijevaju velike količine podataka i parametara. Za učenje ovih modela je stoga potrebno puno procesorske snage. Treniranje može trajati danima ili tjednima na jakim procesorima ili grafičkim karticama.

Budući da nemaju svi dovoljno procesorske snage, razne tvrtke su počele nuditi uslugu treniranja dubokih modela. Druga često korištena opcija je preuzeti već naučeni duboki model za neki općeniti zadatak i onda ga dodatno naučiti za sličan, specifičniji zadatak (engl. *transfer learning*).

U oba slučaja postoje razne sigurnosne ranjivosti. Model koji dobivamo ima potencijalno milijune parametara koje je nemoguće interpretirati, pa ne možemo biti sigurni što smo točno dobili. Naravno, provjerit ćemo da model dobro obavlja svoj zadatak, no moguće je da model u sebi ima i namjerno ugrađena stražnja vrata - tzv. *backdoor*.

Model sa stražnjim vratima osjetljiv je na bilo kakav skriveni uzorak (*okidač*) koji će uzrokovati neočekivano ponašanje modela. U većini slučajeva, stražnja vrata vrlo je teško ili nemoguće primijetiti bez zaključivanja na podacima s okidačem. Model sa stražnjim vratima mogao bi, primjerice, u prisutnosti određenog okidača krivo klasificirati sve primjere u specifičan razred. Stražnja vrata mogla bi se i ručno ugraditi namještanjem parametara, no takva izmjena bila bi ili očita ili teška za ugraditi. Opasnija varijanta je da se stražnja vrata nauče kroz podatke, tako da se istovremeno ostvari i normalna funkcionalnost na čistim podacima.

Ako se stražnja vrata uče iz podataka, napadač će umetnuti okidač na maleni udio podataka (tzv. *zatrovani* podaci) kako bi model i dalje dobro učio rješavati originalni zadatak. Ovdje se nazire još jedan problem - čak i ako sami učimo model, vjerojatno opet preuzimamo skup podataka za učenje, a podaci mogli bi biti neprimjetno zatrovani.

Ovaj seminar nudi pregled nekoliko metoda za umetanje stražnjih vrata te načina detekcije (*obrane*) od tih napada. Postojeće metode obrane se mogu ugrubo podijeliti

u tri kategorije:

- Obrana prije učenja - obrana se provodi prije učenja modela, samo na podacima za učenje. Cilj je detektirati otrovane primjere, te ili ukloniti primjere iz skupa podataka, ili ukloniti okidač sa primjera [2].
- Obrana tijekom učenja - obrana nastoji spriječiti učenje stražnjih vrata iz zatrovanih podataka. Primjerice, ABL [7] iskorištava činjenicu da model puno brže nauči zatrovane primjere, pa ih detektira po vrijednostima funkcije gubitka u ranim epohama učenja.
- Obrana nakon učenja - cilj je detektirati sadrži li dani model stražnja vrata, te potencijalno i ukloniti ili umanjiti stražnja vrata. Većina obrana pripada u ovu kategoriju, poput NC [8] i AC [1].

2. Metode napada

2.1. BadNets

Napad BadNets [3] temelji se na umetanju okidača u mali broj podataka za učenje, s ciljem da model nauči i stražnja vrata i originalni klasifikacijski zadatak.

Pretpostavka napada je da napadač dobiva od korisnika definiciju dubokog modela $f(\boldsymbol{x}, \boldsymbol{\Theta})$ i skup podataka za učenje $D_{train} = \{(\boldsymbol{x_i}, y_i)\}$, gdje je $\boldsymbol{x_i} \in [0,1]^d$ i $y_i \in \{1, ..., C\}$. Napadač provodi postupak za učenje i vraća korisniku naučene parametre modela $\boldsymbol{\Theta'}$. Alternativno, moguće je i da korisnik želi sam naučiti model, no od napadača preuzima samo skup podataka za učenje.

Korisnik provjerava točnost naučenog modela $f(\boldsymbol{x}, \boldsymbol{\Theta'})$ na validacijskom skupu podataka D_{val}. Ukoliko točnost nije dovoljno dobra, korisnik će odbaciti model. Stoga napadač mora osigurati da model s ugrađenim stražnjim vratima i dalje postiže dobre rezultate na čistim validacijskim podacima.

Osim točnosti na čistom, validacijskom skupu podataka, napadaču je cilj ugraditi stražnja vrata. Stražnja vrata će osigurati da model, u prisustvu odabranog okidača, klasificira svaki primjer u ciljani razred. Kako bi ovo postigao, napadač uzima podskup $D' \in D_{train}$ koji sadrži do p_a (uobičajeno oko 10%) podataka, te iz njega generira zatrovani skup D'_p:

$$D'_p = \{(\boldsymbol{x'_i}, y'_i) \mid \boldsymbol{x'_i} = A(\boldsymbol{x_i}, \boldsymbol{m}, \boldsymbol{p}), y'_i = y^t, (\boldsymbol{x_i}, y_i) \in D'\} \tag{2.1}$$

U prikazanoj jednadžbi, y_t označava ciljni razred u koji se klasificiraju svi podaci sa okidačem, a $A(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{p})$ predstavlja funkciju koja na dani $\boldsymbol{x}$ postavlja okidač definiran parametrima $\boldsymbol{m}$ i $\boldsymbol{p}$ gdje je $\boldsymbol{m} \in [0,1]^d$ maska koja određuje poziciju okidača, a $\boldsymbol{p} \in [0,1]^d$ uzorak okidača:

$$A(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{p}) = (1 - \boldsymbol{m}) \cdot \boldsymbol{x} + \boldsymbol{m} \cdot \boldsymbol{p} \tag{2.2}$$

Konačno, napadač uči klasifikacijski model $f(\boldsymbol{x}, \boldsymbol{\Theta})$ na otrovanom skupu podataka

$(D \setminus D') \cup D'_p$.

Okidači će uobičajeno biti vrlo maleni uzorci, jer je cilj učiniti ih neprimjetnima. Razmatrat ćemo slučaj kada imamo jedan okidač za jedan ciljni razred, no mogući su slučajevi sa više okidača za isti razred ili sa više ciljnih razreda, svaki sa svojim okidačima.

2.2. WaNet

Umjesto "nalijepljenih" okidača koji direktno mijenjaju piksele na vrlo vidljiv način, cilj WaNet napada je dizajnirati okidače koje će čovjek vrlo teško primijetiti. WaNet to postiže sa perturbacijskim poljima. Funkcija trovanja je sada:

$$A(\boldsymbol{x}) = \mathcal{W}(\boldsymbol{x}, \mathbf{M}) \tag{2.3}$$

$\mathcal{W}$ je operacija perturbiranja (engl. *warping*) koja generira novu sliku uzorkovanjem piksela stare slike sa posmaknutih lokacija. $\mathbf{M}$ je perturbacijsko polje koje definira koji piksel stare slike se uzorkuje za generiranje piksela nove slike. Dopuštamo i uzorkovanje necjelobrojnih indeksa, te u tom slučaju bikubično interpoliramo vrijednost piksela.

Za uspješnost i nevidljivost napada ključno je generirati dobar $\mathbf{M}$. Perturbacija bi trebala biti slaba i glatka kako bi bila neprimijetna. Također, perturbacija ne bi trebala uzorkovati točke izvan granica slike.

Kako bismo osigurali glatkoću perturbacije, prvo nasumično generiramo manje perturbacijsko polje P veličine $k \times k \times 2$, koje ćemo kasnije naduzorkovati na potrebnu veličinu:

$$\mathbf{P} = \psi(rand_{[-1,1]}(k, k, 2)) \times s \tag{2.4}$$

Funkcija $rand_{[-1,1]}(k, k, 2)$ vraća tensor oblika $k \times k \times 2$ sa vrijednostima između -1 i 1, s označava jakost perturbacijskog polja, a ψ je funkcija koja normalizira snagu perturbacije:

$$\psi(\mathbf{A}) = \frac{\mathbf{A}}{\frac{1}{size(\mathbf{A})} \sum_{a_i \in \mathbf{A}} |a_i|} \tag{2.5}$$

Sada naduzorkujemo $\mathbf{P}$ na potrebnu veličinu $h \times w \times 2$, što označavamo sa $\uparrow$, te primijenjujemo funkciju podrezivanja (engl. *clipping*) ϕ koja osigurava da nijedna točka uzorkovanja ne pada izvan slike. Konačno deformacijsko polje $\mathbf{M}$ dobivamo prema sljedećem izrazu:

$$\mathbf{M} = \phi(\uparrow (\psi(rand_{[-1,1]}(k, k, 2)) \times s)) \tag{2.6}$$

Kao i u BadNet napadu, ova perturbacija se dodaje na p_a primjera iz skupa za učenje, te se njihova oznaka mijenja u ciljnu oznaku y_t. No, WaNet uvodi još jedno poboljšanje u proces napada - osim trovanja p_a primjera dodavanjem okidača i promjenom ciljnog razreda, dodatno se na p_n primjera dodaje i slučajan šum sličan okidaču bez promjene ciljnog razreda. Poanta ovog šuma je otežati detekciju napada, budući da će sada model morati naučiti samo točnu perturbaciju kao odgovarajući okidač za stražnja vrata. Točnije, za p_n primjera definira se perturbacija:

$$\mathbf{M}' = \mathbf{M} + rand_{[-1,1]}(h, w, 2) \tag{2.7}$$

Sljedeća jednadžba prikazuje konačan napad WaNet:

$$(\boldsymbol{x}, y) \mapsto \begin{cases} (\boldsymbol{x}, y) & \text{za } 1 - \rho_a - \rho_n \text{ za primjera} \\ (\mathcal{W}(\boldsymbol{x}, \mathbf{M}), c_t) & \text{za } \rho_a \text{ za primjera} \\ (\mathcal{W}(\boldsymbol{x}, \mathbf{M}'), y) & \text{za } \rho_n \text{ za primjera} \end{cases} \tag{2.8}$$

WaNet je itekako skriveniji od napada BadNets, no BadNets je praktičniji za primjene u fizičkom svijetu. Primjerice, ako je zadatak modela prepoznavanje prometnih znakova, napadač naučeni okidač može fizički zalijepiti na znakove u prometu, dok se WaNet perturbacija može samo računalno unijeti. Učenje napada WaNet je također puno zahtijevnije od BadNets - modeli bi mogli imati slabije performanse na čistim podacima, ili bi mogli zahtijevati više epoha i podataka da dostignu iste performanse.

Slika 2.1: Primjer perturbirane slike aviona. Perturbacija je neuočljiva.

3. Metode obrane

3.1. Neural Cleanse

Cilj obrane Neural Cleanse [8] je detekcija razreda na koje se stražnja vrata odnose te rekonstrukcija okidača za dani naučeni model $f(\boldsymbol{x})$.

Glavna pretpostavka obrane je da će okidači biti vrlo maleni uzorci na podacima. Drugim riječima, ukoliko imamo primjer bez okidača koji pripada razredu $y \neq y_t$, onda je potrebna vrlo malena promjena primjera kako bi ga model krivo klasificirao u razred y_t. Neural Cleanse pokušava izmjeriti veličinu te promjene za svaki od razreda, te tako detektirati razred na kojeg se odnose stražnja vrata. Ova pretpostavka nije istinita za svaki napad, primjerice WaNet okidači su perturbacije po cijeloj slici, pa na takvim napadima Neural Cleanse neće uspjeti.

Za svaki od razreda, Neural Cleanse pokušava pronaći neki okidač $(\boldsymbol{m}, \boldsymbol{p})$ koji će, kada je primijenjen na bilo koji primjer $\boldsymbol{x_i}$, klasificirati taj primjer u razred c. Takav okidač bi imao vrlo malen ukupni gubitak $\mathcal{L}$:

$$\mathcal{L} = \sum_{\boldsymbol{x_i}} \ell(c, f(A(\boldsymbol{x_i}, \boldsymbol{m}, \boldsymbol{p}))) \tag{3.1}$$

Prikazana jednadžba pretpostavlja da je ℓ neka standardna funkcija gubitka, primjerice unakrsna entropija. Primijetimo da ovakav okidač uvijek postoji - trivijalan primjer bi bio "okidač" koji je zapravo neki $\boldsymbol{x_i}$ iz skupa podataka koji model ispravno klasificira u razred c. Primjerice, ako tražimo okidače koji će natjerati model da svaki znak klasificira kao "stop", onda je sam znak "stop" validan "okidač".

Očito, takvi veliki "okidači" nas ne zanimaju jer ne podržavaju hipotezu da su podatci zatrovani. Neural Cleanse pokušava pronaći što manje okidače $(\boldsymbol{m}, \boldsymbol{p})$ koji daju što manji gubitak 3.1. Točnije, cilj Neural Cleanse-a je riješiti sljedeći optimizacijski problem:

$$\boldsymbol{m}, \boldsymbol{p} = \underset{\boldsymbol{m},\boldsymbol{p}}{\arg\min} \left[\sum_{\boldsymbol{x_i}} \ell(c, f(A(\boldsymbol{x_i}, \boldsymbol{m}, \boldsymbol{p}))) + \lambda \cdot |\boldsymbol{m}| \right] \tag{3.2}$$

Pri tome je $|\boldsymbol{m}|$ L1 norma maske, a λ je hiperparametar koji određuje koliko jako kažnjavamo velike maske.

Uz pretpostavku da nam je dostupna definicija modela f i sve njegove težine, izraz (3.2) sada možemo riješiti nekim gradijentnim postupkom kao što je Adam [5]. Optimizaciju rješavamo za svaki razred c, čime dobivamo rekonstruirane potencijalne okidače za svaki razred. Rekonstruirani okidač za razred na kojeg se stražnja vrata odnose bi trebao imati značajno manju L1 normu maske od ostalih.

Nakon što smo izračunali potencijalne okidače za svaki razred, koristimo neku od metoda za detekciju stršećih vrijednosti nad nizom L1 normi maski. Neural Cleanse koristi srednju apsolutnu devijaciju (engl. *Median Absolute Deviation, MAD*), čime konačno detektiramo razred koji je povezan sa stražnjim vratima.

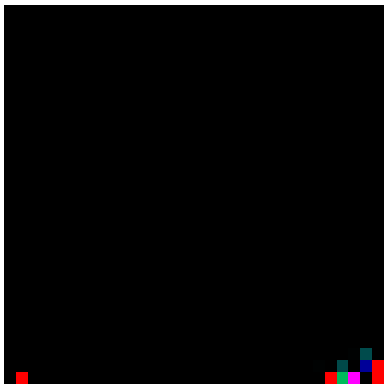
Jednom kad imamo rekonstruiran približno točan okidač, možemo pokušati i ukloniti stražnja vrata tako da na dio podataka za učenje ipak primijenimo okidač, ali ostavimo ispravnu oznaku podatku. Model bi sada na ovakvom skupu podataka trebao naučiti predviđati ispravne oznake i u prisustvu okidača te tako "zatvoriti" stražnja vrata.

3.2. Activation Clustering

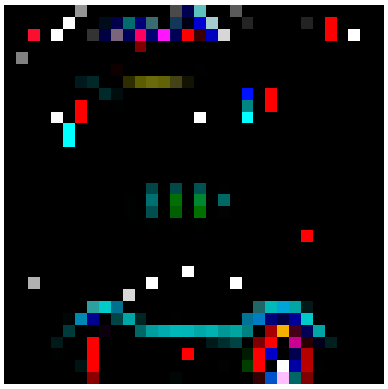
Iako primjere sa umetnutim okidačem model klasificira u isti razred kao i primjere koji stvarno pripadaju tom razredu, razlog zašto ih model pridjeljuje razredu je drukčiji. Za primjere bez okidača, model uočava značajke koje je naučio da su karakteristične za taj razred. S druge strane, za primjere s okidačem model uočava i značajke izvornog razreda i značajke okidača. Activation Clustering [1] se zasniva na ideji da bi se ta razlika u zaključivanju trebala vidjeti i u aktivacijama modela.

Pretpostavka Activation Clustering algoritma je da imamo skup podataka, potencijalno zatrovan, na kojem je model treniran. Modelom klasificiramo sve primjere i sakupimo aktivacije zadnjeg sloja modela. Ideja AC-a je da, za svaki od razreda u koji su primjeri klasificirani, grupiramo aktivacije modela. Ako neki razred nema ugrađena stražnja vrata, tada bi aktivacije trebale biti međusobno slične među svim primjerima. Ako neki razred ima ugrađena stražnja vrata, tada bi aktivacije trebale biti grupirane u dvije različite grupe - jedna grupa aktivacija za otrovane primjere, druga grupa za obične primjere.

(a) BadNets okidač za razred 0



(b) Okidač kojeg je NC rekonstruirao za razred 0



(c) Okidač kojeg je NC rekonstruirao za razred 1

Slika 3.1: BadNets okidač i okidače koje NC rekonstruira. Iako NC nije potpuno točno rekonstruirao originalni okidač za razred 0, on je i dalje puno manji od okidača kojeg NC rekonstruira za razred bez stražnjih vrata.

Algoritmi grupiranja često loše rade na visokodimenzionalnim podacima, stoga prvo koristimo neki algoritam smanjenja dimenzionalnosti poput metode nezavisnih komponenti (engl. *Independent Component Analysis, ICA*). Nadalje koristimo algoritam K-sredina s K=2. Algoritam K-sredina s K=2 će nam uvijek vratiti dvije grupe, stoga svakako nakon grupiranja moramo još odrediti je li ijedna grupa otrovana, i ako je - koja.

Prva predložena metoda je da treniramo novi model bez jedne od grupa, i onda klasificiramo tu grupu s novim modelom. Ukoliko je grupa sadržavala otrovane primjere, novi model će te primjere klasificirati drugačije od staroga. S druge strane, ako grupa nije sadržavala otrovane primjere, novi model će klasificirati primjere skoro jednako kao i stari. Međutim, ova metoda je vrlo skupa budući da moramo trenirati nove modele.

Jednostavnija opcija je gledati samo veličine grupa. Ukoliko nema otrovanih podataka za neki razred, očekujemo slične veličine grupa. Ukoliko je model učen sa skupom podataka koji sadrži $p \cdot N$ otrovanih podataka, gdje je N broj svih podataka a $p < 1$, tada će jedna grupa imati otprilike $p \cdot N$ podataka, a druga $(1 - p) \cdot N$ podataka. Uobičajeno je p malen broj, do 10%, stoga ukoliko je jedna grupa puno manja od druge, vjerojatno je riječ o stražnjim vratima.

4. Eksperimenti

Promatrat ćemo sljedeće metrike modela nad skupom za testiranje:

- *C-Acc* (engl. *clean accuracy*) - točnost predikcije čistih primjera u ispravne razrede
- *ASR* (engl. *attack success rate*) - točnost predikcije otrovanih primjera u ciljni razred
- *R-Acc* (engl. *robust accuracy*) - točnost predikcije otrovanih primjera u originalne razrede

Cilj napadača je postići visok C-Acc i ASR, a time i malen R-Acc. Cilj obrane, ukoliko pokušava ukloniti stražnja vrata, je visok C-Acc i nizak ASR. Dodatno, poželjno je da smanjenje ASR-a dovodi do povećanja R-Acc, što bi značilo da smo uklanjanjem stražnjih vrata uspjeli rekonstruirati ispravne oznake otrovanih primjera.

Svi eksperimenti su provedeni na skupu podataka CIFAR-10 [6] sa PreAct-Resnet18 [4] modelom. Model učimo kroz 30 epoha. Za napad BadNet generiramo 3×3 crveni kvadrat u donjem lijevom kutu kao okidač, i stavljamo ga na $p_a = 1\%$ ili $p_a = 10\%$ primjera iz skupa za učenje. Za napad WaNet nasumično generiramo perturbacijsko polje i stavljamo ga na $p_a = 10\%$ primjera iz skupa za učenje, te dodatno uvodimo perturbacijski šum u još $p_n = 20\%$ primjera iz skupa za učenje.

Za Neural Cleanse obranu, provodimo 20 epoha po razredu za detekciju okidača, te kroz još 30 epoha odučavamo model primjenom rekonstruiranog okidača na 20% primjera, zadržavajući originalne oznake. Za Activation Clustering obranu, nakon algoritma K-sredina uvijek odbacujemo manju grupu te provodimo učenje novog modela kroz 30 epoha.

5. Rezultati

Rezultati su prikazani u tablici 5.1. Za svaku od primijenjenih napada i obrana, prikazane su metrike modela C-Acc, ASR i R-Acc.

BadNets napad se pokazuje dobrim i za $p_a = 1\%$ i $p_a = 10\%$, iako je jači za veći p_a. U oba slučaja, obrana Neural Cleanse je točno identificirala koji razred je otrovan, no samo za veći p_a je okidač približno ispravno rekonstruiran i odučavanje modela je uspjelo - ASR je spušten sa 95% na samo 3.5%. Točnost nad čistim podacima je čak i poboljšana, te su skoro svim otrovanim primjerima rekonstruirane originalne oznake sa 89.07% R-Acc. Za Activation Clustering, situacija je obratna - za $p_a = 10\%$ nema skoro nikakvog efekta, dok za $p_a = 1\%$ uspijeva sniziti ASR na 17%.

WaNet sa $p_a = 1\%$ uopće nije ni uspio, stoga nema smisla ni pokretati obrane na modelu pa je i ostatak retka prazan. Uz to, WaNet napad ima i malo manji C-Acc od BadNetsa, što nam sugerira da WaNet zahtijeva više kapaciteta. No, jednom kada je ugrađen, vrlo je težak za detektirati - Neural Cleanse je donio zaključak da model nema ugrađena stražnja vrata, pa odučavaje nije ni pokrenuto, dok je Activation Clustering donekle i uspio spustiti ASR, no opet nedovoljno.

Tablica 5.1: C-Acc, ASR i R-Acc modela nakon primijenjenih napada i obrana

	Bez obrane			Neural Cleanse			Activation Clustering		
	C-Acc	ASR	R-Acc	C-Acc	ASR	R-Acc	C-Acc	ASR	R-Acc
$p_a = 1\%$									
BadNets	90.67	85.54	13.74	91.67	74.39	24.62	84.67	16.92	73.13
WaNet	89.26	4.69	84.30	-	-	-	-	-	-
$p_a = 10\%$									
BadNets	89.35	94.76	4.91	91.34	3.46	89.07	82.18	92.59	6.60
WaNet	86.86	96.66	3.13	-	-	-	79.33	65.84	24.81

6. Zaključak

Napadi na duboke modele su relativno nova briga, te se razvijaju sve bolji napadi, a uz njih i metode obrane. U ovom radu pogledali smo kako rade BadNets i WaNet napadi. BadNets je relativno jednostavan napad, koji na maleni udio slika u skupu podataka stavlja običan, statičan okidač, te mijenja oznaku slike u neku drugu, ciljnu. Model tijekom treniranja nauči da bi slike s okidačem trebao klasificirati u ciljni razred. WaNet je sličan, no umjesto običnih okidača preko slike, koristi jedva vidljive perturbacije.

Razmotrili smo i dva napada - Neural Cleanse i Activation Clustering. Neural Cleanse pokušava optimizacijskim postupkom rekonstruirati potencijalne okidače za svaki od razreda, te onda odučiti stražnja vrata modela stavljanjem rekonstruiranog okidača na dio slika skupa za učenje zadržavajući originalnu oznaku. Activation Clustering zasniva se na grupiranju aktivacija posljednjeg sloja modela pri klasifikaciji svake od slika, te pokušava analizom nastalih grupa zaključiti koji primjeri su otrovani.

Neural Cleanse i Activation Clustering su se pokazali dovoljno dobri za detekciju BadNets napada, iako njihova uspješnost i dalje ovisi o udjelu otrovanih slika u skupu za učenju. Međutim, nijedna metoda obrane nije uspjela ukloniti WaNet napad.

7. Literatura

[1] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, i Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[2] Bao Gia Doan, Ehsan Abbasnejad, i Damith C Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. U *Annual Computer Security Applications Conference*, stranice 897–912, 2020.

[3] Tianyu Gu, Brendan Dolan-Gavitt, i Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Identity mappings in deep residual networks. U *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, stranice 630–645. Springer, 2016.

[5] Diederik P Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Alex Krizhevsky, Vinod Nair, i Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[7] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, i Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912, 2021.

[8] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, i Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor at-

tacks in neural networks. U *2019 IEEE Symposium on Security and Privacy (SP)*, stranice 707–723, 2019. doi: 10.1109/SP.2019.00031.

8. Sažetak

Duboki modeli su danas uspješni u mnogim zadacima računalnog vida, no za postizanje dobrih rezultata zahtijevaju velike količine podataka za učenje. Pri treniranju dubokih modela, podaci se često preuzimaju s drugih izvora, što nudi napadačima priliku da neprimjetno izmijene skup za učenje. Ovaj seminar nudi pregled nekoliko metoda "backdoor" napada, u kojima napadači ciljano izmijene maleni udio skupa za učenje dodavanjem okidača. Učen na ovakvim podacima, duboki model će naučiti prepoznavati okidače i uz njihovu pojavu na novim podacima će ih krivo klasificirati. Osim napada, razmatramo i nekoliko metoda detekcije ovakvih napada.