TDK-dolgozat

Szász Attila

Szegedi Tudományegyetem

Informatikai Intézet

Neurális hálózatok paraméterrobusztussága

TDK Dolgozat

Készítette:
Szász Attila
informatika szakos
hallgató

Témavezető:
Dr. Bánhelyi Balázs
egyetemi docens

Szeged
2024

Tartalomjegyzék

1. fejezet

Bevezetés

A neurális hálózatok kiemelkedő figyelmet kaptak az elmúlt években, mind a felhasználói, mind a kutatási oldalról. Napjainkban számos területen alkalmazzák őket, ilyen például a számítógépes látás és a beszédfelismerés. A kutatások döntő többsége az egyre pontosabb és megbízhatóbb hálózatok előállításának irányába mozdult el. A megbízható, más néven robusztus hálózatok tanításának érdekében számos tanítási technikát javasoltak a kutatók. A módszerek döntő többsége két csoportba sorolható. Az adverzális tanítás [7] alapú algoritmusok ellenséges példák felett optimalizálják a hálózatok paramétereit. A minősített [6] (*certified*) tanítás alapú módszerek pedig befoglalást számítanak a hálózat kimeneteire, majd a korlátok alapján feltételezhető legrosszabb esetet minimalizálják. A bemenet fókuszú támadások mellett, a hálózatok paraméter támadása is előtérbe került. Az ilyen típusú támadások a hálózatok paramétereit módosítják és ezáltal váltják ki az ellenséges viselkedést. Ezek alapján kidolgoztak olyan tanítási módszereket, melyek a hálózatok paraméterstabilitásának elérését is beépítik a tanítási folyamatba, melyek közül a szakirodalomban legelterjedtebb az *Adverzális Súly Perturbáció (AWP)* [3] módszere lett. Az algoritmus fő hátránya, hogy a legrosszabb esetet rendkívül alulbecsüli, melynek eredményeként nem biztosít megfelelő védelmet a paramétertámadásokkal szemben. Kutatásunk során megmutattuk az AWP algoritmus legfőbb gyenge pontját, illetve javaslatot tettünk egy certified alapú tanítóalgoritmusra, amely számos esetben megnövelte a hálózatok ellenállóképességét a paramétertámadással szemben.

2. fejezet

Bemeneti robusztusság

Kutatásunk során a hálózatok robusztusságát két szempontból vizsgáltuk: bemeneti, illetve paraméter robusztusság szempontjából. A bemeneti robusztusság a hálózat ellenálló képességére utal a bemeneten történő változtatásokkal szemben. A szakirodalomban megjelent kutatások döntő többsége erre a robusztusságra fókuszált. Ezzel szemben a paraméter robusztusság a hálózat súly-, illetve bias paramétereinek perturbálásával szembeni ellenállására utal. Ezek alapján egy hálózatot bemenet robusztusnak nevezünk, ha a bemenet egy lefixált környezetében a hálózat minden bemenetre ugyanazt a kimenetet eredményezi, illetve paraméter robusztusnak, ha a hálózat a paramétereinek egy lefixált környezetében lévő összes *alternatív* hálózat, egy fix bemenetre ugyanazt a kimenetet eredményezi.

Bemeneti robusztus tanítás során a cél a modell olyan θ paraméterkonfigurációjának a megtalálása, amely minimalizálja a bemenet ϵ környezetében várható maximális hibát:

$$\theta = \arg\min_{\theta} \mathbb{E}_{x,y}[\max_{\check{x} \in B_p(x,\epsilon)} L_\theta(\check{x}, y)] \tag{2.1}$$

L a tanítás során használt hibafüggvényt, $B_p(x, \epsilon)$ az x bemenet ϵ sugarú környezetét jelöli, mely alapján $B_p(x, \epsilon) = \{\check{x} \mid ||\check{x} - x||_p \leq \epsilon\}$, továbbá $L_\theta(\check{x}, y)$, a θ paraméterkonfigurációjú hálózat kimeneti hibája, a $\check{x}$ bement és y elvárt kimenet esetén.

A szakirodalomban javasolt tanítóalgoritmusok két csoportba sorolhatók attól függően, hogy a belső maximalizálási feladatot hogyan optimalizálják.

Adverzális tanítás [7] során olyan bemenetek mellett tanítják a hálózatot, amely az adott környezetben maximalizálja a hibafüggvényt. Az egyik legelterjedtebb módszer a

Projektív Gradiens Ereszkedés (PGD). Az algoritmus a bemeneti gradiens alapján maximalizálja a hibát, majd a maximális hibát eredményező bemenet felett optimalizálja a modell paramétereit. A módszer könnyen elakadhat lokális optimumban, mely eredményeként a legrosszabb esetet alulbecsülheti. Az adverzális tanítású hálózatok továbbra is sebezhetőek a propagáláson [6], illetve lineáris programozáson [13] alapuló erősebb támadási módszerekkel szemben.

A **minősített** (*certified*) tanítási módszerek megbízható befoglalást számítanak a hálózat kimeneteire, majd a feltételezhető legrosszabb kimenet mellett optimalizálják a paramétereket. A kimenet befoglalására számos módszer ismert, melyek általában rendkívül túlbecsülik a kimeneti értékkészletet. Az erős túlbecslésnek köszönhetően a módszerek nagyon erős regulairizáló hatást vezetnek a tanítási folyamatba, melynek eredményeként a hálózatok kiértékelése egyszerűbb, illetve bizonyítható robusztusságuk nő, azonban a teszthalmaz feletti normál pontosságuk csökken. A pontosabb befoglalás elérésére több technika is ismert, [14], melyeket a hálózatok verifikálása során is gyakran alkalmaznak. A szakirodalomban megmutatták, hogy ezek az algoritmusok, bár csökkentenék az erős túlbecslést a tanítás során, azonban sokkal nehezebb optimalizálási feladatot eredményeznének, amely végül gyengébb teljesítményű hálózatokhoz vezetne [9]. A probléma kezelésére megjelentek olyan minősített tanítási módszerek, melyek precíz, de nem feltétlen megbízható befoglalást számítanak a kimeneti értékkészletre [5]. A szakirodalomban ezeket a módszereket *Unsound-Certified* típusú tanításoknak nevezik, illetve megmutatták, hogy számos esetben robusztusabb hálózatokat eredményeznek, mint a megbízható befoglalás mellett tanító, *Sound-Certified* algoritmusok.

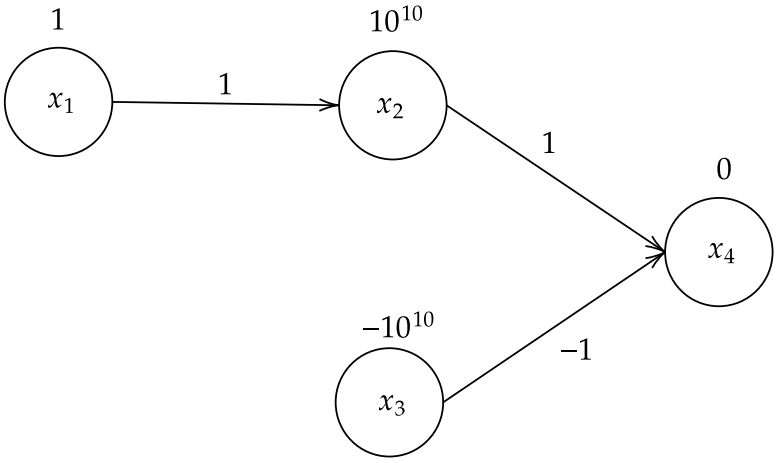
3. fejezet

Paraméter robusztusság

3.1. Motiváció

A hálózatok tanítási-, kiértékelési-, illetve használati környezete sokszor eltérő lehet, melynek eredményeként a hálózatok paramétereit konvertálni kell az aktuális környezet típusára. A konvertálás eredménye azonban egy teljesen új hálózat lesz, melynek az eredeti hálózattal való azonos működését nem lehet garantálni a 2-es fejezetben részletezett tanítási módszerek mellett. Vizsgáljuk például a 3.1-es ábrán látható hálózat kimenetét 32, illetve 64 bites architektúrán kiértékelve. Az éleken a súlyok, a neuronok felett a bias értékek láthatóak. 64 biten $10^{10} + 1$ ábrázolható ($eps(10^{10}) \approx 1.9e - 6$), majd ebből kivonva 10^{10}-t, az x_4 kimenete 1 lesz. 32 biten azonban $eps(10^{10}) = 1024$, mely eredményeként $10^{10} + 1$ továbbra is csak 10^{10} lesz, így az x_4 kimenetben számított különbség eredménye 0 lesz. A hálózatot hozzáillesztve más hálózatokhoz, a kiértékelési architektúrától függően befolyásolhatjuk a hálózat kimenetét. Könnyen elérhető, hogy a hálózat az általunk választott kiértékelési pontosság mellett mindig hibásan működjön, viszont a másik pontosság mellett a tanulás eredményének megfelelő válaszokkal szolgáltasson.

A bemeneti robusztusságra koncentráló algoritmusok explicit módon a bemeneti-hiba teret optimalizálják adverzális bemenetek (*adversarial training* [7]) vagy propagált korlátok (*certified training* [6]) alapján előálló legrosszabb eset alapján. A módszerek egy közös jellemzője, hogy a legrosszabb esetet mindig lokálisan, az aktuális bemenet alapján számítják. Adverzális tanítás során bemenetenként számított ellenséges perturbáció alap-



3.1. ábra. 32-64 bit kapcsoló háló

ján maximalizáljuk az adverzális hibát, certifed tanításnál pedig bemenetenként számított korlátok alapján optimalizálunk. Ezzel szemben a paraméterrobusztusság során ellenséges viselkedést kiváltó paramétereket (adverzális hálózatokat) keresünk, melyek akár a teljes tanítóhalmaz feletti hibát maximalizálhatják, ezáltal egy modell-szintű, globális legrosszabb esetet eredményeznek.

Egy hálózat paraméter robusztussága összefüggésben van a robusztus általánosítási réssel (*robust generalization gap*), amely a normál, illetve az adverzális pontosság különbségére utal. Tanítás során a bemeneti robusztusságra koncentráló módszerek esetén gyakran megfigyelhető, hogy még az eredeti teszthalmaz feletti pontosság javul, az adverzális pontosság egy idő után csökkenni kezd. A szakirodalomban megmutatták [3], hogy az általánosítási rés azoknál a hálózatoknál volt minimális, amelyeknél a paraméter-hiba tér a lehető legsimább volt a paraméterek kis környezetében, amely szintén a paraméterrobusztusságra utal. A szakirodalomban használt AWP algoritmus explicit módon, a paraméter-hiba tér simítását is beépíti a tanítási folyamatba, amellyel több esetben növelni tudták a hálózatok bemeneti robusztusságát az egyszerű adverzális tanításhoz képest.

Az előzőek alapján a hálózatok paraméterperturbációjának kérdése egy hasznos és fontos kiegészítése lehet a csak bemeneti perturbációra koncentráló módszereknek. A szakirodalomban alkalmazott tanítási módszert a 3.2.1-es fejezet szemlélteti.

3.2. Szakirodalomban használt algoritmusok

3.2.1. Adversarial Weight Perturbation (AWP) algoritmus

A paraméterrobusztusságra is koncentráló tanítás során, a 2.1-es célfüggvény a következőre módosul:

$$\theta = \arg\min_{\theta} \mathbb{E}_{x,y}\Big(\max_{\check{\theta}\in B_{p_1}(\theta,\lambda)}\big(\max_{\check{x}\in B_{p_2}(x,\epsilon)} L_{\check{\theta}}(\check{x},y)\big)\Big) \tag{3.1}$$

A 3.1-es célfüggvényben λ, a paramétereken megengedett maximális perturbációt jelöli. Mivel a hálózatban a súlyok eloszlása rétegről rétegre változhat, így abszolút távolság helyett, relatív távolságot szokás korlátozni.

$$||\theta_i||_{p_1} \leq \lambda \, ||\check{\theta}_i||_{p_1} \tag{3.2}$$

A 3.1-es célfüggvény optimalizálására használt AWP [3] algoritmus, a belső maximalizálási feladatokat egy alternáló, gradiens alapú módszerrel oldja meg.Az algorimtus első lépésben, PGD segítségével adverzális bemenetet keres a hálózathoz, majd a megtalált bemenet felett, szintén PGD módszerrel maximalizálja a paraméterperturbáció melletti hibát. Ezt követően a megtalált adverzális bemenet és hálózat alapján frissíti a modell paramétereit, majd a frissített modellből kivonva az adverzális perturbációt, a következő iteráció kezdete előtt visszatér a középponti modellre.

3.2.2. Adverzális Paraméter Támadás

A szakirodalomban számos paramétertámadási módszer ismert, azonban közülük csak néhány támadja explicit módon a hálózatok robusztusságát. Adverzális Paraméter Támadás [4] során a hálózat paramétereinek kis környezetében keresnek olyan ellenséges hálózatot, melynek az eredeti teszthalmazon számított pontossága csak minimálisan változik, azonban bemeneti robusztussága drasztikusan csökken.

A támadás két fázisra bontható. Az első fázisban a hálózat adverzális hibáját maximalizálják, a megengedett maximális paraméterperturbáció mellett. A második fázisban az eredeti hiba, illetve az adverzális hiba hányadosából képzett célfüggvényt minimalizálják.

A szakirodalomban megmutatták [4], hogy még az AWP tanításon átesett hálózatok is sebezhetőek a támadással szemben, azonban ennek okát nem vizsgálták.

3.3. Algoritmusok csoportosítása

A kutatásunk egyik célja, a paraméterrobusztusságot is figyelembe vevő algoritmusok elhelyezése a robusztus tanítási módszerek osztályozásában. Az osztályozás során az adverzális és minősített tanítási fogalmak kiterjesztése mellett definiáltuk az osztályokat, melyekben a korábbi megközelítésekkel ellentétben, a hálózatok paraméterei felett értelmezett perturbációt is figyelembe vettük. Fontos megemlíteni, hogy a kutatásunk során csak *white box* támadásokra koncentráltunk és az osztályozást is ilyen módszerek köré építettük fel.

A kialakított osztályozás a 3.1-es táblázatban látható. A $\check{x}$ és $\check{\theta}$ jelölések a 3.1-es célfüggvényben szereplő, legrosszabb esetet eredményező bemenetet és hálózatot jelölik. *Fix* hálózat, illetve bemenet esetén a tanítási módszer nem tételez fel perturbációt az adott komponensre. Ezeknél az algoritmusoknál $\check{x} = x$ és $\check{\theta} = \theta$. A $\check{x}$ és $\check{\theta}$ oszlopaiban rendre a $\check{x}$ és $\check{\theta}$ meghatározására használt algoritmuscsoportok találhatóak. A *PGD* a gradiens alapú, a legrosszabb esetet általában alulbecsülő algoritmusokat, az *IA*, pedig a befoglalás alapú, a legrosszabb esetet általában túlbecsülő algoritmusokat jelölik. A *PGD* és *IA* (Intervallum aritmetika) jelölés rendre az adverzális és minősített tanítási módszerek legelterjedtebb képviselőire utalnak. Az osztályok közötti legfontosabb különbség, a 3.1-es célfüggvény, maximalizálási részfeladatának (3.3) az optimalizálási módja.

$$WorstCase(\theta, x, \epsilon, \lambda) = \max_{\check{\theta} \in B_{p_1}(\theta, \lambda)} \big(\max_{\check{x} \in B_{p_2}(x, \epsilon)} L_{\check{\theta}}(\check{x}, y) \big) \tag{3.3}$$

A 3.1-es táblázat első három sorában szereplő módszerek $\check{\theta} = \theta$, azaz fix hálózat feltevése mellett optimalizálják a 3.3-as részfeladatot. Az osztályok megegyeznek a szakirodalomban is használt adverzális és minősített tanítási módszerek osztályaival. A bemeneti támadásokkal szemben hatékonyak, azonban a paramétertámadásokkal szemben nem biztosítanak elegendő védelmet [4].

A Model-Certified és Model-Adversarial algoritmusok védelmet nyújtanak az olyan paramétertámadásokkal szemben, melyek nem a bemeneti robusztusságot támadják, azonban a bemeneti robusztusságot is szem előtt tartó támadásokkal szemben nem biztosítanak kellő védelmet [3].

A bemeneti- és paraméterrobusztusságot is figyelembe vevő algoritmusok két fő cso-

$\check{x}$ (*Bemenet*)	$\check{\theta}$ (*Hálózat*)	Osztály	Implementáció
PGD	fix	Input-Adversarial	Adversarial training [7], ...
IA	fix	Input-Certified (sound)	IA, IBP [6] ...
PGD+IA	fix	Input-Certified (unsound)	TAPS [5], SABR [10], ...
fix	IA	Model-Certified	-
fix	PGD	Model-Adversarial	-
IA	IA	Certified-Sound	-
IA	PGD	Certified-Unsound	-
PGD	IA		**APP** (jelen dolgozat, 3.4-es fejezet)
PGD	PGD	Adversarial	AWP [3]

3.1. táblázat. Robusztus tanítási módszerek osztályai

portba sorolhatóak. Az *Adversarial* algoritmusok a 3.3-as részfeladat mindkét komponensét gradines módszerrel optimalizálják. A módszerek hátránya, hogy a legrosszabb esetet rendkívül alulbecsülik. Az így tanított hálózatok adverzális pontossága több esetben is javulást mutatott az Input-Adversarial algoritmusokhoz képest, azonban az erősebb bemeneti támadásokkal, illetve paramétertámadásokkal szemben nem biztosított kellő védelmet. A szakirodalomban használt AWP algoritmus is ebbe az osztályba tartozik.

A *Certified* módszerek befoglalást számítanak a 3.3-as részfeladatra, majd a feltételezhető legrosszabb eset alapján minimalizálják a 3.1-es célfüggvényt. Legyen $\mathbf{o}$ a hálózat kimeneteit tartalmazó vektor, $\overline{\mathbf{o}}$ és $\underline{\mathbf{o}}$ pedig $\mathbf{o}$ befoglalásához tartozó alsó és felső korlátokat tartalmazó vektorok. Ekkor y elvárt címke esetén, a legjobb, illetve legrosszabb kimenetek a következőképpen definiálhatóak:

$$\mathbf{o}_{best_i} = \begin{cases} \overline{\mathbf{o}}_i & \text{, ha } i = y \\ \underline{\mathbf{o}}_i & \text{, egyébként.} \end{cases} \tag{3.4}$$

$$\mathbf{o}_{worst_i} = \begin{cases} \underline{\mathbf{o}}_i & \text{, ha } i = y \\ \overline{\mathbf{o}}_i & \text{, egyébként.} \end{cases} \tag{3.5}$$

Az algoritmusok további alcsoportba sorolhatóak attól függően, hogy a számított befoglalás biztosan tartalmazza-e a tényleges kimeneti értékkészletet. Ezek alapján definiálható a megbízható (*Sound*), illetve nem megbízható (*Unsound*) alcsoport. A gyakorlat-

ban a *Certified-Sound* algoritmusok nagyobb hálózatok esetén szinte használhatatlanok. Ahogyan a 3.4.2-es fejezetben majd részletezzük, a paraméterek befoglalása melletti kiértékelések megnehezítik a propagálási módszerek használatát, melynek hatására a naiv intervallum aritmetika melletti tanítások meglehetősen instabilak lesznek a durva túlbecslésnek köszönhetően.

3.4. Adverzális Paraméter Propagálás (APP)

3.4.1. Az algoritmus

Ebben a fejezetben az általunk javasolt, Adverzális Paraméter Propagálás algoritmust mutatjuk be, amely a 3.1-es táblázat osztályai közül, a *Certified (Unsound)* osztályba tartozik. A módszerünk lényege, hogy a kimeneti értékkészleteket a hálózat paramétereinek intervallumos befoglalása mellett számolja, ezzel egy megbízható, de durván túlbecsült befoglalást adva a paraméterek szerinti legrosszabb esetre. A durva túlbecslés kompenzálása érdekében, a 3.3-as részfeladat $\check{x}$ komponensét PGD-vel optimalizáljuk. Mivel a PGD nem garantál globális optimumot, így az $\check{x}$ mellett számított korlátok sem fogják biztosan tartalmazni a globális legrosszabb esetet (*Unsound* módszer), azonban a kimenetek jelentősen szűkebbek lesznek, mint a tisztán intervallumos (*Certified-Sound*) megoldásnál, amelynek köszönhetően kevésbé lesz jelen a túlbecslések okozta tanítási instabilitás.

A tanítás első lépése a $\check{x}$ meghatározása, mely során olyan ellenséges bemenetet kerestünk, amely a hálózat intervallumos befoglalása mellett számított hiba alsó korlátját maximalizálja. A 3.4-es és 3.5-ös képlet alapján, a hibafüggvény értékkészletének befoglalása a 3.6-os képlettel számítható, melyben x a hálózat bemenetét, y az elvárt kimenetét, $\mathbf{o}_{best}(x)$ és $\mathbf{o}_{worst}(x)$ pedig a paramétereinek a befoglalása mellett számított legjobb, illetve legrosszabb kimenetet jelöli.

$$[L_\theta(\mathbf{o}_{best}(x), y), L_\theta(\mathbf{o}_{worst}(x), y)] \tag{3.6}$$

Mivel tanítás során a cél a 3.6-os befoglalás felső korlátjának minimalizálása, így a támadás során olyan bemeneteket kerestünk, melyek maximalizálják a 3.6-os befoglalás alsó korlátját. Az előzőek alapján, a PGD bemeneti támadás képlete, a 3.7-es képletre

módosult.

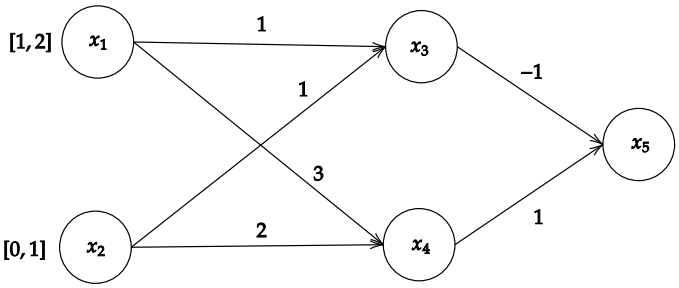
$$\check{x} = \underset{x \in B_p(x,\epsilon)}{\arg\max}(L_\theta(\mathbf{o}_{best}(x), y)) \tag{3.7}$$

Második lépésben a hálózat paramétereinek a frissítése történik, ahol a hálózat befoglalása, illetve az $\check{x}$ adverzális bemenet mellett feltételezhető legrosszabb kimenet mellett számítjuk a gradienseket.

$$\Theta \leftarrow \Theta - \alpha \nabla_\Theta L_\theta(\mathbf{o}_{worst}(\check{x}), y) \tag{3.8}$$

3.4.2. A bemeneti robusztusság vizsgálata során használt propagálási módszerek hiánya

Az Input-Certified algoritmusok gyakran propagálási módszerekkel kompenzálják a naiv intervallum aritmetika függőségi problémája miatt keletkező túlbecslést. Az algoritmusok, a hálózatok linearizálása mellett csökkentik a számítási lánc hosszát, mellyel jelentősen csökkentik a túlbecslési hibát is. Az általunk javasolt algoritmus, a hálózatok paramétereinek befoglalása mellett számítja a kimeneti értékkészletet, melynek hatására ellehetetleníti a bemeneti robusztusságvizsgálatnál is használt propagációs módszereket. A probléma demonstrálására tekintsük a 3.2-es és 3.3-es ábrákat.



3.2. ábra. Fix hálózati paraméterek és intervallum bemenet

A 3.2-es ábra, a bemeneti robusztusságra optimalizáló algoritmusok problémáját szemlélteti. x_1 és x_2 a bemeneti neuronok, x_5 a kimeneti neuron. Az aktivációs függvény ReLU

az x_3 és x_4 neuronokban. Az Input-Certified algoritmusok, a naiv intervallum aritmetika túlbecslési hibáját szimbolikus módszerek segítségével kompenzálják. Ezekben a módszerekben a hálózat bemeneteit gyakran változóknak tekintik, melyeket felhasználva minden neuronhoz felírható egy megfelelő lineáris kifejezést. A hálózat kiértékelése során nem a neuronok értékkészlete, hanem a meghatározott lineáris kifejezések felett értelmezik a rétegtranszformációkat. A szimbolikus propagálás hatására a függőségi információk jobban nyilvántarthatóak, melynek köszönhetően jelentősen csökken a kimeneti túlbecslés. A 3.2-es hálózat naiv befoglalásának eredményeként $[0, 7]$ kimeneti értékkészletet kapunk. A 7-es felső korlát azonban sosem éles, az egzakt felső korlát az 5. Szimbolikus kiértékelés során, a propagált $2x_1 + x_2$ kimeneti kifejezés kiértékelése mellett már az egzakt felső korlátot kapjuk. A propagálás általában nem az egzakt korlátokat eredményezi, hiszen az aktivációs függvények linearizálása, majd a kifejezések naiv kiértékelése mellett azért megjelenik túlbecslés, viszont a kimeneti értékkészlet sokkal pontosabb lesz, mint a teljesen naiv módszer esetében.

Az általunk javasolt APP algoritmusban fix bemenet és intervallum hálózati paraméterek mellett számítjuk a kimeneti értékkészletet. A problémát a 3.3-as ábra szemlélteti, melyen a w_{ij} jelölés az ij súly intervallumos befoglalását jelenti. Stabil és aktív x_3, illetve x_4 neuronokat feltételezve, a propagálás melletti kimeneti kifejezést a 3.9-es képlet jelöli.

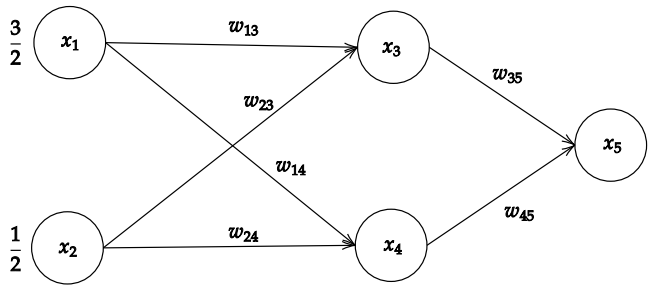
$$x_5 = \frac{3}{2}(w_{13}w_{35} + w_{14}w_{45}) + \frac{1}{2}(w_{23}w_{35} + w_{24}w_{45}) \tag{3.9}$$

A 3.9-es képletben jól látható, hogy a propagált kifejezésekben nem a bemenet, hanem a hálózat paraméterei határozzák meg a változókat, így a 3.2-es propagálási problémával ellentétben, itt nem végezhetőek el az összevonások a számítás során, melynek hatására a naiv, illetve a propagált korlátok megegyeznek.

3.4.3. Tanítást segítő regularizációk

A következő fejezetben olyan regularizációs technikákat javasoltunk, melyek csökkentik a kimeneti szélességeket, ezáltal stabilabbá teszik a tanítást.

A kimeneti szélességnövekedést leginkább a hálózatok korai rétegeiben megjelenő, 1-nél nagyobb abszolút értékű paraméterek válthatják ki. A tanítás korai szakaszaiban ez



3.3. ábra. Intervallum hálózati paraméterek és fix bemenet

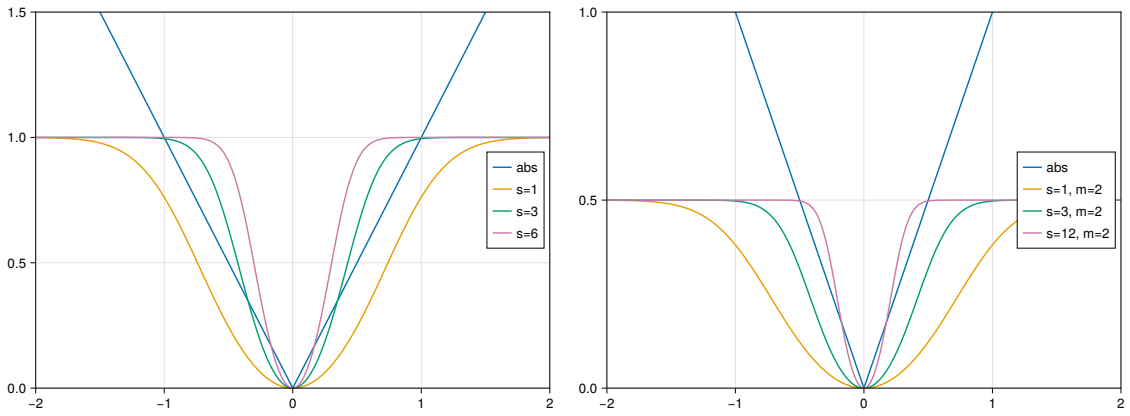
gyakran megfigyelhető, mely során olyan széles kimenetek keletkezhetnek, amelyek instabillá tehetik az aktuális tanítási iterációt. A tanítás későbbi fázisaiban azonban kis abszolút értékű súlyok a jellemzőek, ami többek között a szakirodalomban gyakran alkalmazott L_1 és L_2 regularizációnak is köszönhető. Annak érdekében, hogy a tanítás korai szakaszaiban is elkerüljük a drasztikus kimeneti szélességek miatti instabilitást, a következő két regularizációs technikával bővítettük a tanítást:

A paraméterek relatív sugarait nem az abszolút értékük alapján, hanem az általunk javasolt 3.10-es, folytonos approximáció mellett határoztuk meg.

$$ParameterRadius(x) = \frac{\tanh(sx^2)}{m}\lambda \tag{3.10}$$

A 3.10-es képletben λ a tanításnál meghatározott relatív sugarat jelöli, s, illetve m pedig az approximáció beállítandó paramétereit. A 4.4-es ábra szemlélteti a függvényt néhány paraméterkonfigurációra. A paraméterek beállítása mellett meghatározhatjuk a regularizáció erősségét. Az s paraméter növelésével szigoríthatjuk a relatív sugarak nagyságát, az m paraméter beállításával a maximálisan megengedett szélességet állíthatjuk be.

A nagy abszolút értékű súlyok elkerülése érdekében, a paraméterek értékkészletének a korlátozását is bevezettük a tanítási folyamatba. A paraméterek frissítése után minden paramétert visszavetítettünk egy előre meghatározott értékkészletbe. Az értékkészlet meghatározása során fontos figyelembe venni a 3.10-es regularizációnál használt paramétereket is. A 3.4a ábrán látható, hogy a regularizáció használata mellett, az 1-nél nagyobb súlyok esetében is, csak az 1-hez tartozó relatív környezetben követeljük meg a stabilitást, így ezek a súlyok sebezhetőséget jelentenek a paramétertámadással szemben.



(a) $m = 1$ (b) $m = 2$

3.4. ábra. A 3.10-as approximáció vizualizációja

3.4.4. A korlátok hatékony számítása

Az implementáció során kulcsfontosságú a korlátok hatékony számítása. A hálózatokat általában GPUn tanítják a gyorsaság növelése érdekében. A CPU-GPU adatmozgatás meglehetősen költséges folyamat, így a korlátok számítását is célszerű GPUn végezni.

Bemeneti robusztusságra koncentráló algoritmusok esetén a hálózat bemenetei intervallumok, a hálózat súly és bias paraméterei továbbra is lebegőpontos értékek maradnak. Ebben az esetben, a hálózat kiértékelése során csak olyan szorzatok fordulnak elő, melyeknek csak az első tagja tartalmaz intervallumokat és a második tagja mindig lebegőpontos értékekből álló mátrix lesz. Az intervallum-lebegőpontos mátrixszorzás viszont visszavezethető négy lebegőpontos mátrixszorzásra, amely már hatékonyan számítható grafikus gyorsítóval is.

Paraméterrobusztusság esetén a hálózat súly és bias paraméterei felett értelmezünk befoglalást. Lebegőpontos bemenetet feltételezve, az első réteg kimenetei már intervallumok lesznek, így a második réteg kimenetének meghatározása során már intervallum-intervallum szorzatokat kell számítanunk. Intervallumok szorzatának számítása a 3.11-es képlet alapján történik.

$$[a, b] \cdot [d, c] = [\min(ad, ac, bd, bc), \max(ad, ac, bd, bc)] \tag{3.11}$$

Általános esetben, az elemenkénti minimum és maximumszámítás miatt, az intervallummátrixok szorzása nem vezethető vissza lebegőpontos mátrixok szorzására. Közelítő algoritmusok ismertek, amelyek a naiv befoglalás túlbecslése mellett vezetik vissza a számítást lebegőpontos mátrixszorzatokra. Mivel már a naiv módszer is jelentősen túlbecsüli az egzakt eredményt, így ezek az algoritmusok használhatatlanok a neuronhálók kiértékelése során. Az általános eseten túl, néhány speciális esetben létezik megoldás az egzakt visszavezetésre [11]. Ilyen eset például, amikor az egyik mátrix nem tartalmaz negatív korláttal rendelkező intervallumokat. Ebben az esetben az $A \cdot B = O$ intervallummátrixok szorzata a 3.12 összefüggés alapján számítható. Jelölje $\overline{A}$ a felső és $\underline{A}$ az alsó korlátokból álló lebegőpontos mátrixokat (B és O esetén hasonlóan), továbbá igaz, hogy $0 \leq \overline{A}_{i,k}$ és $0 \leq \underline{A}_{i,k}$ minden i, k-ra.

$$\begin{aligned}
\underline{B}^{-} &= \min(0, \underline{B}) \\
\underline{B}^{+} &= \max(0, \underline{B}) \\
\overline{B}^{-} &= \min(0, \overline{B}) \\
\overline{B}^{+} &= \max(0, \overline{B}) \\
\underline{O} &= \underline{B}^{+}\underline{A} + \underline{B}^{-}\overline{A} \\
\overline{O} &= \overline{B}^{+}\overline{A} + \overline{B}^{-}\underline{A}
\end{aligned} \tag{3.12}$$

A kutatásunk során ReLU aktivációs függvénnyel rendelkező hálózatokat tanítottunk és vizsgáltunk. Ilyen hálózatok esetén teljesül a rétegek bemeneteit tartalmazó intervallummátrixokra a ≥ 0 feltétel, tehát a kiértékelés során használhatóak a 3.12-es összefüggések. A módszer nagy előnye, hogy a korlátok számítása csak lebegőpontos mátrixok szorzását és összeadását igényli, így a teljes folyamat hatékonyan számítható GPU-n is. A mátrixszorzáson túl, a konvolúció is hasonlóan számítható, amennyiben teljesül ≥ 0 feltétel.

4. fejezet

Kiértékelés

A rendszerünket julia [1] programozási nyelven implementáltuk a Flux [2] tanítási keretrendszer használatával. A vizsgált tanítási algoritmusokat egy telepíthető, julia csomagban összegeztük. A kiértékelés során főként a hálózatok bemeneti robusztusságának, illetve a paramétertámadással szembeni ellenállásának mértékét hasonlítottuk össze az AWP, illetve az általunk javasolt algoritmus (APP) által tanított hálózatokon.

4.1. Az osztályozási feladat és az architektúra

A tanítóalgoritmusunk vizsgálata érdekében számos hálózatot tanítottunk a CIFAR-10 [8] adathalmazon. Az adathalmaz 60000 32X32-es, 3 színcsatornás képet tartalmaz, 10 különböző kimeneti osztályból. A vizsgálatok során CNN4 [10] architektúrájú hálózatokat tanítottunk, melyek 4 konvolúciós és 1 teljesen összefüggő, aktivációval is rendelkező rétegből állnak. A konvolúciós rétegek 4X4-es filtereket tartalmaznak, illetve rendre 32,64,128,128 csatornából épülnek fel. A lépésköz minden rétegnél 2 volt. Az aktivációval rendelkező, teljesen összefüggő réteg 512 neuronból állt, melyet a kimeneti réteg követett 10 neuronnal. Az aktivációs függvény minden rétegben ReLU volt.

Fontos megemlíteni, hogy a kutatásunk során nem a jelenleg elérhető, legpontosabb hálózatok teljesítményét szerettük volna felülmúlni, ezért sem választottunk nagyobb hálózati architektúrát a kiértékelés során. A célunk az AWP és az APP algoritmusok, minél több aspektusában történő összehasonlítása volt, legfőképpen a paramétertámadással

szembeni védelemre koncentrálva.

4.2. A bemeneti robusztusság vizsgálata

Elsőként a hálózatok bemeneti robusztusságát vizsgáltuk az AWP, illetve APP algoritmus által tanított hálózatok esetében. Az összehasonlítás során különböző bemeneti, illetve paraméter perturbáció mellett tanított hálózatok mellett elemeztük a teszthalmaz feletti normál, illetve adverzális pontosságot. A 4.1-es táblázatban ϵ a bemeneti, λ pedig a paraméter perturbáció megengedett sugarát jelöli. Az általunk javasolt APP algoritmust két konfigurációban vizsgáltuk, *tanh* esetén az intervallumsugarakat a 3.10-es közelítés segítségével határoztuk meg, *abs* esetén egyszerűen az abszolút értékeket használtuk. Az adverzális pontosságot 2 módszerrel is mértük. *PGD-50* esetén 3 újraindítással, 50 interáción keresztül maximalizáltuk a bemeneti hibát, majd az így megtalált ellenséges bemenetek aránya alapján számítottuk az adverzális pontosságot. A szakirodalomban elérhető *Auto Attack* [12] algoritmussal is kiértékeltük a hálózatainkat. Az algoritmus több adverzális támadás futtatása mellett következtet a robusztusságra. Az algoritmust az alap beállítások mellett futtattuk.

A táblázatban látható, hogy minden konfigurációban az APP algoritmus, a 3.10-as regularizáció használata mellett érte el a legjobb eredményeket. Az AWP algoritmus motivációja során már megmutatták [3], hogy a bemeneti teljesítmény növelése érdekében, paraméterrobusztusság szempontjából, a paraméterek kis környezetében történő, minél erősebb paraméterrobusztusság biztosítása a legfontosabb. Az AWP algoritmus, bár explicit módon a paraméter-hiba tér simítását is beépíti a tanításba, a gradiens alapú (PGD) optimalizálás miatt gyengébb simító hatást fejt ki a hibafüggvényre, mint a befoglalás alapú APP.

Az APP algoritmusok esetén azért lett magasabb az APP (tanh) teljesítménye, mert az APP (abs) erős simító hatása melletti tanítás eredményeként, a tanítási sugár széleinél lévő hálózatok hibaértékei a középpont felé haladva sem csökkennek. A tanh befoglalás mellett a simító hatás gyengébb, ezáltal a *középponti* hálózat alacsonyabb hibát képes produkálni, mint a befoglalás széleinél lévő hálózatok.

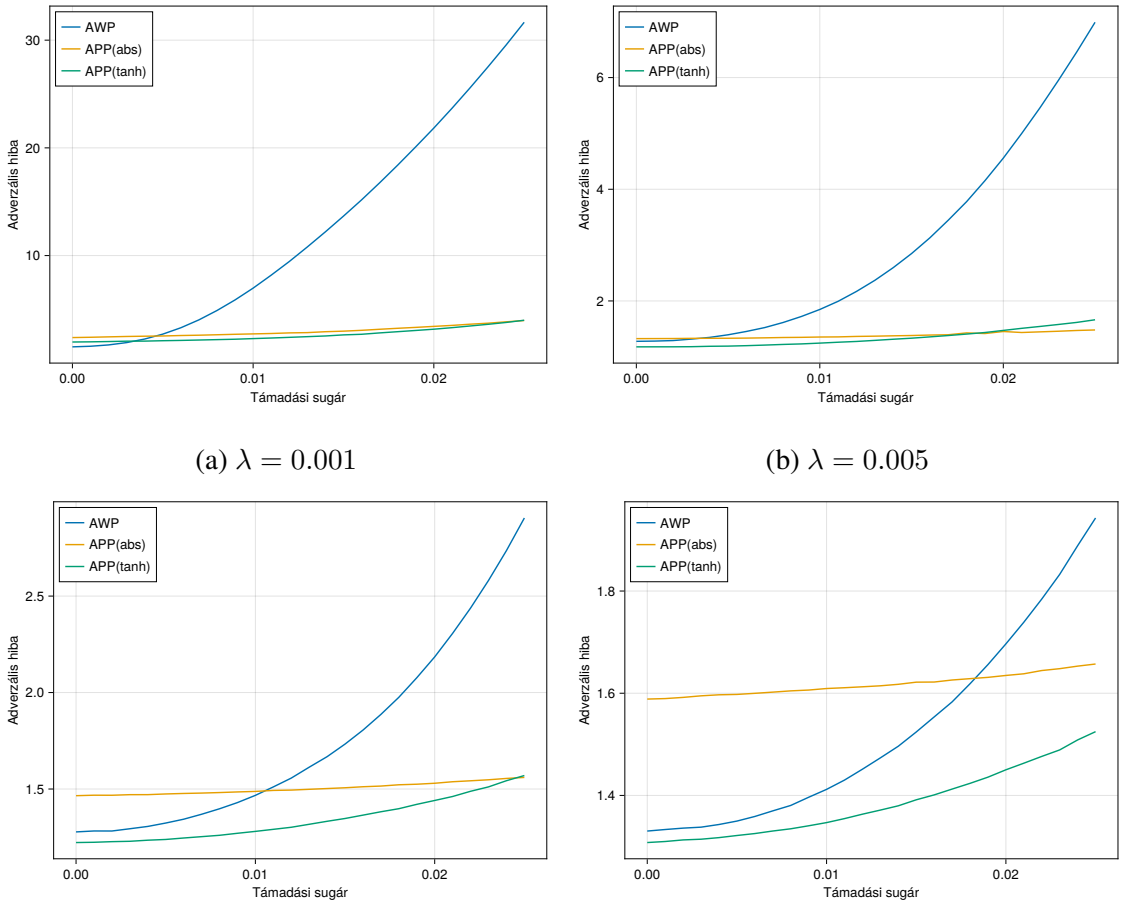
Az előző észrevételek szemléltetése érdekében vizsgáltuk azt is, hogy különböző λ pa-

raméterekkel tanított hálózatok esetében, a megengedett maximális paraméterperturbációtól függően, hogyan változik a maximális adverzális hiba. A vizsgálat során különböző λ paraméter mellett tanított hálózatok mellett maximalizáltuk az adverzális hibát (PGD-50). A megengedett paraméterperturbáció 0, illetve 0.025 között mozgott (lefedve ezzel a vizsgát tanítási sugarakat), 0.001-es lépésközzel. A 4.1-es ábrán jól látható, hogy az APP (abs) algoritmus befoglalás alapú regularizációjának köszönhetően, a tanítási környezetben történő támadás esetén szinte alig növekedett az adverzális hiba, illetve a tanítási környezeten túl ennél az algoritmusnál volt a legkisebb a növekedés mértéke.Az erős simító hatásnak egy negatív következménye, hogy a nagyobb tanítási környezetben eredményezett hibaértékek, a tanításnál kisebb környezetben is hasonlóak lesznek.

A hálózat paraméterein megengedett perturbáció sugarának megválasztása fontos kérdés a tanítás során. Megfelelő λ paraméter mellett javíthatjuk a hálózat bemeneti robusztusságát, a szakirodalomban 0.001 és 0.005 közötti sugarak mellett érték el a legjobb eredményeket [3]. Ahogyan a 4.1-es táblázatban is látható, a λ növelése egy idő után a normál, illetve az adverzális pontosság csökkenését eredményezi. A paramétertámadásokkal szembeni védelem elérése érdekében a λ nem lehet túlságosan kis érték sem. Az előzőek alapján, a megfelelő paraméter sugár megválasztásakor mérlegelnünk kell, hogy a hálózat teljesítményre, vagy pedig a paramétertámadásokkal szembeni védelem a fontosabb számunkra.

A 4.2-es, illetve 4.3-es ábrákon jól látható, hogy még az AWP esetén a λ növelése nem befolyásolta drasztikusan sem a normál, sem az adverzális pontosságot, addig az APP esetében ez sokkal jelentősebb volt, amely szintén az erős simító hatás következménye. A 4.4-es, illetve 4.3-es ábrákon megfigyelhető, hogy még a legjobb normál pontosságot mindkét esetben a legkisebb környezetben ($\lambda = 0.001$) értük, addig az adverzális pontosság a 0.005 környezetben tanított hálózatok esetén volt a legmagasabb.

APP tanítás során a λ értékét folyamatosan növeltük az első 80 epochban, amíg el nem érte a célértékét. Egyből a célértékkel történő befoglalás számítása mellett, a tanítás instabillá válik, ami a normál, illetve az adverzális pontosság romlását eredményezi. A 4.4-es, illetve 4.3-es ábrákon jól látható, hogy körülbelül a 20., illetve 80. epoch között folyamatosan csökkent a pontosság, amely a λ növelése által gerjesztett, egyre erősebb regularizáló hatásnak volt köszönhető.



(a) $\lambda = 0.001$

(b) $\lambda = 0.005$

(c) $\lambda = 0.01$

(d) $\lambda = 0.02$

4.1. ábra. Adverzális hiba alakulása

A pontosságot vizsgáltuk $\frac{8}{255}$ bemeneti környezetben tanított hálózatok esetén is. A λ paraméter növelése hasonlóan befolyásolta az eredményeket, mint a $\frac{2}{255}$ bemeneti környezetben tanított hálózatok esetében. A legmagasabb normál pontosságot az APP (tanh, $\lambda = 0.001$), a legmagasabb adverzális pontosságot pedig az APP (tanh, $\lambda = 0.01$) tanítású hálózat érte el.

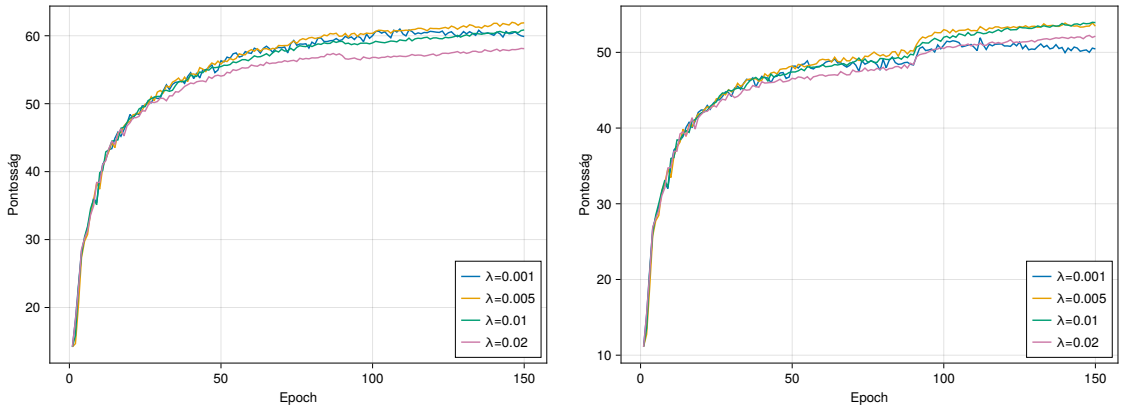
ϵ	λ	Algoritmus	Pontosság	PGD-50	AutoAttack	Rés (PGD-50)
$\frac{2}{255}$	0.001	APP (tanh)	68.15 %	57.24 %	42.75%	10.9%
		APP (abs)	59.48%	51.3%	36.08%	8.18%
		AWP	59.88%	50.45%	39.01%	9.43%
	0.005	APP (tanh)	67.17%	60.87%	49.92%	6.3%
		APP (abs)	60.15%	52.75%	44.92%	7.4%
		AWP	61.9%	53.46%	43.33%	8.44%
	0.01	APP (tanh)	63.8%	59.44%	49.63%	4.36%
		APP (abs)	52.71%	48.28%	42.85%	4.43%
		AWP	60.68%	53.93%	44.16%	6.75%
	0.02	APP (tanh)	60.77%	58.30%	48.18%	2.47%
		APP (abs)	48.81%	45.89%	41.94%	2.92%
		AWP	58.1%	52.14%	43.70%	5.96%
$\frac{8}{255}$	0.001	APP (tanh)	67.56 %	35.83 %	21.98%	31.73%
		APP (abs)	63.88%	34.44%	20.1 %	29.44%
		AWP	60.97%	32.89%	19.89%	28.08%
	0.005	APP (tanh)	60.06%	41.02%	23.31%	19.04%
		APP (abs)	57.69%	35.13%	20.96%	22.56%
		AWP	58.93%	35.21%	21.88%	23.72%
	0.01	APP (tanh)	55.42%	41.43%	23.9%	13.99%
		APP (abs)	51.9%	35.80%	21.01%	16.1%
		AWP	57.08%	35.52%	22.91%	21.56%
	0.02	APP (tanh)	50.38%	40.57%	22.12%	9.81%
		APP (abs)	38.27%	32.78%	21.10%	5.49%
		AWP	54.37%	35.26%	23.11%	19.11%

4.1. táblázat. Hálózatok pontossága

4.3. Ellenállás a paramétertámadással szemben

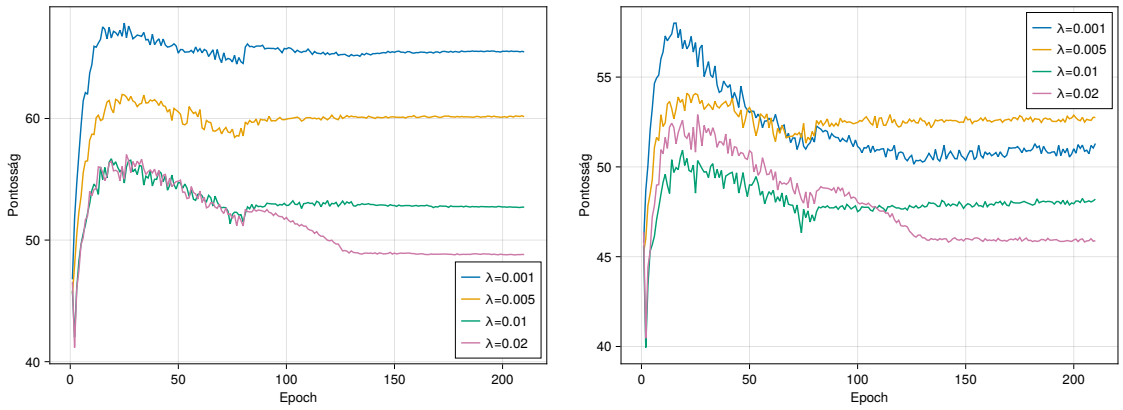
A kiértékelés során vizsgáltuk azt is, hogy az APP és AWP tanítású hálózatok milyen mértékben ellenállóak az adverzális paramétertámadással szemben. Különböző tanítású hálózatok esetén elemeztük, hogy $\check{y}$ támadási sugár mellett, hány **százalékkal sikerült csökkenteni** az eredeti hálózat adverzális pontosságát. Mivel a támadás sok esetben támaszkodik random értékekre, így minden támadást háromszor végeztünk el, majd a maximális pontosságcsökkenést eredményező értékeket összegeztük a 4.2-es táblázatban.

A támadások során azokat az eredményeket jegyeztük fel, melyek esetén nem csökkent



(a) Normál pontosság (b) Adverzális pontosság

4.2. ábra. AWP tanítású hálózatok, különböző λ paraméter mellett

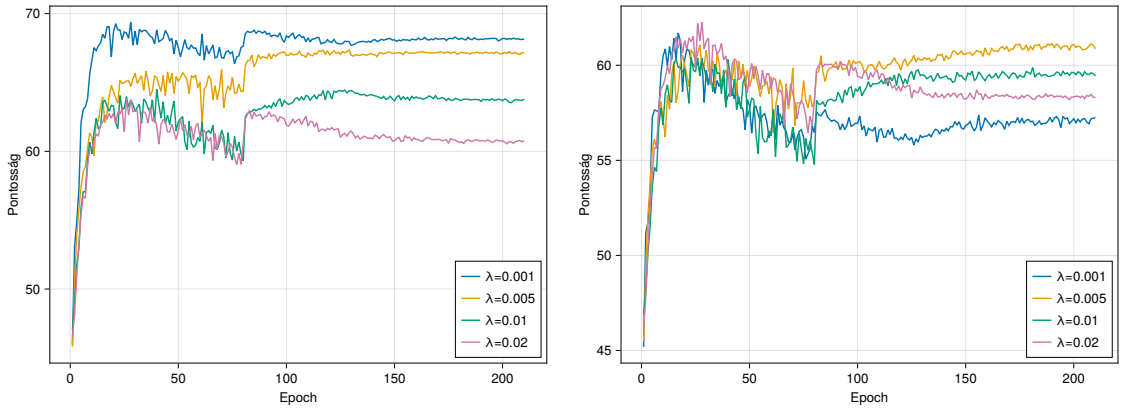


(a) Normál pontosság (b) Adverzális pontosság

4.3. ábra. APP(abs) tanítású hálózatok, különböző λ paraméter mellett

jelentősen a hálózat eredi pontossága. Jelen esetben 80%-os küszöbértéket alkalmaztunk. Az ϵ, illetve a λ, a hálózat tanítása során a bemeneten, illetve a paramétereken értelmezett perturbáció sugarát jelöli.

A bemeneti támadások vizsgálata során, a tanítási, illetve a támadási sugár általában megegyezik. Paramétertámadás során azonban ez sok esetben elvárhatatlan. A paraméterrobusztusság megkövetelése mellett fontos szempont, hogy a hálózatok továbbra is használhatóak legyenek és az eredeti tesztpéldákon mért teljesítményük ne csökkenjen draszti-



(a) Normál pontosság

(b) Adverzális pontosság

4.4. ábra. APP(tanh) tanítású hálózatok, különböző λ paraméter mellett

kusan. Ahogyan a 4.1-es táblázatban is látszik, a paramétersugár növelésének hatására, az összes algoritmus esetén csökken az eredeti példákon mért pontosság. A szakirodalomban is alkalmazott adverzális paramétertámadás során viszont 0.02 és 0.1 közötti támadási sugarakat vizsgáltak. Az előző észrevételek alapján fontos, hogy paraméterrobusztusságra is koncentráló algoritmusok, ne csak a tanítási környezetben, hanem azon kívül is biztosítani tudják a robusztusságot.

A 4.2-es táblázatban megfigyelhető, hogy a legszűkebb, 0.001-es paraméterperturbációval tanított hálózatoknál, az összes támadási sugár mellett, az APP(abs) algoritmus teljesített a legjobban. A tanítási sugár növelése hatására megfigyelhető, hogy még a kisebb támadási környezetben az AWP, nagyobb támadási környezetben az APP(abs) algoritmus ért el jobb eredményeket.

A $\check{\lambda}$ oszlopaiban félkövérrel kijelölt értékek azokat a hálózatokat jelölik, melyeknek az adott támadási sugár mellett, a legkevésbé csökkent az adverzális pontosságuk. Megfigyelhető, hogy még kis környezetben (0.02 és 0.04) az AWP algoritmus mellett tanított hálózatok teljesítettek a legjobban, addig a nagyobb környezetben (0.06-0.1) az APP tanítású hálózatok.

A paramétertámadással szembeni ellenállást vizsgáltuk $\frac{8}{255}$ bementi környezetben tanított hálózatok esetén is. Kisebb támadási környezetben ismét az AWP tanítású hálózatok teljesítettek jobban, azonban a 0.08 és 0.1 környezetben az APP(abs) algoritmus érte el a

ϵ	λ	Algoritmus	$\check{\lambda}$				
			0.02	0.04	0.06	0.08	0.1
$\frac{2}{255}$	0.001	APP (tanh)	27%	31%	36%	36%	41%
		APP (abs)	14%	18%	**19%**	**24%**	**27%**
		AWP	14%	28%	30%	37%	45%
	0.005	APP (tanh)	19%	24%	28%	30%	28%
		APP (abs)	13%	24%	24%	33%	29%
		AWP	8%	18%	34%	44%	49%
	0.01	APP (tanh)	12%	25%	22%	27%	**27%**
		APP (abs)	9%	18%	20%	**24%**	**27%**
		AWP	**6%**	17%	26%	38%	38%
	0.02	APP (tanh)	13%	19%	29%	31%	31%
		APP (abs)	9%	23%	27%	28%	28%
		AWP	21%	**13%**	23%	37%	32%
$\frac{8}{255}$	0.001	APP (tanh)	41%	31%	52%	66%	81%
		APP (abs)	12%	36%	52%	68%	75%
		AWP	35%	44%	58%	79%	71%
	0.005	APP (tanh)	24%	50%	56%	64%	64%
		APP (abs)	15%	46%	64%	69%	76%
		AWP	15%	36%	53%	78%	88%
	0.01	APP (tanh)	30%	47%	44%	62 %	63%
		APP (abs)	19%	35%	44%	54%	68%
		AWP	**10%**	26%	42%	63%	80%
	0.02	APP (tanh)	33%	42%	52%	59%	60%
		APP (abs)	35%	39%	48%	**53%**	**61%**
		AWP	23%	**25%**	**36%**	62%	76%

4.2. táblázat. Adverzális pontosság csökkenése különböző $\check{\lambda}$ támadási sugár mellett

legjobb eredményeket.

Összességében elmondható, hogy a nagyobb támadási sugarak mellett, az APP-vel tanított hálózatok ellenállóbban a paramétertámadással szemben, függetlenül attól, hogy $\frac{2}{255}$, vagy $\frac{8}{255}$ bementi környezetben lettek tanítva.

4.4. A különböző tanítások hatása a maximális hibára

A kiértékelés során vizsgáltuk azt is, hogy a különböző tanítási módszerek mellett, az AWP és APP algoritmusok legrosszabb eset becslései hogyan viszonyulnak egymáshoz. A 4.3-as táblázatban az látható, hogy az egyszerű bemeneti adverzális, illetve AWP és APP tanítású hálózatok esetén hogyan alakul a 3.3-as célfüggvény maximális értéke. A vizsgálat során négy, már tanított hálózatot hasonlítottunk össze. A bemeneti tanítási sugár $\frac{2}{255}$ volt, továbbá az AWP és APP algoritmusok esetében 0.01 paraméter sugarat használtunk. A hibabecsléseket a teszthalmaz első száz képén mért legnagyobb hibaértékek átlagaként számítottuk.

Tanítás \ Hiba mérési technika	AWP	APP (tanh)	APP (abs)
Input-Adversarial	6.93	23700	26671
AWP	1.18	4923	6044
APP (tanh)	2.07	1.32	3.26
APP (abs)	1.36	1.7	1.59

4.3. táblázat. Maximális hiba becslések

A 4.3-as táblázatban megfigyelhető, hogy a csak bemeneti adverzális tanításon átesett hálózat esetén az AWP és az APP maximális hibabecslései is jelentősen megnőttek, a paraméterperturbáció nélküli esethez képest. Az APP algoritmus kiugróan magas hibaértékei, az intervallumszámítások miatt megjelenő túlbecslésnek volt köszönhető.

Az AWP tanításon átesett hálózat esetén már látható, hogy a tanítás során is használt gradiens alapú paramétertámadási módszer által számított maximális hiba, jelentősen alacsonyabb volt, mint a paraméterperturbáció nélkül tanított hálózat esetében. Ezzel szemben az APP hibabecslései, a befoglalási pontatlanság miatt megjelenő erős túlbecslésnek köszönhetően, továbbra is jelentősen magasabbak voltak, mint az AWP esetében.

Az APP algoritmus eredményeinél megfigyelhető, hogy az intervallumos befoglalás alapján tanított hálózatok már jelentősen ellenállóbbak a befoglalás alapú módszerek maximális hiba túlbecslésével szemben. Mivel az APP algoritmus által számított hibák általában felső korlátjai a AWP által számított hibáknak, így APP tanítás során, az AWP maximális hibabecslését is minimalizáljuk.

4.5. A λ paraméter optimális értéke

Az általunk vizsgált módszerek esetén, a tanítás egyik legfontosabb paramétere a paraméterperturbáció sugara. Mint ahogyan már a korábbi fejezetekben említettük, a paramétertámadással szembeni védelem növelése érdekében célszerű nagyobb sugarat választani. A nagyobb paramétersugár azonban megnehezítheti a tanítást, melynek hatására a normál pontossága, illetve a bemeneti támadásokkal szembeni ellenállóképessége romlik a hálózatnak. A tanítás megkezdése előtt mérlegelnünk kell, hogy a hálózat pontosságára, illetve bemeneti robusztusságára, vagy pedig a paramétertámadásokkal szembeni védelemre koncentrálunk.

A 4.5-ös ábrán, különböző λ paraméter mellett tanított hálózatok esetén vizsgáltuk a normál és adverzális pontosságot, illetve a paramétertámadás eredményeként megjelenő robusztusságcsökkenést. A bemeneti perturbáció sugara $\frac{2}{255}$ volt a tanítások, illetve a támadások során is.

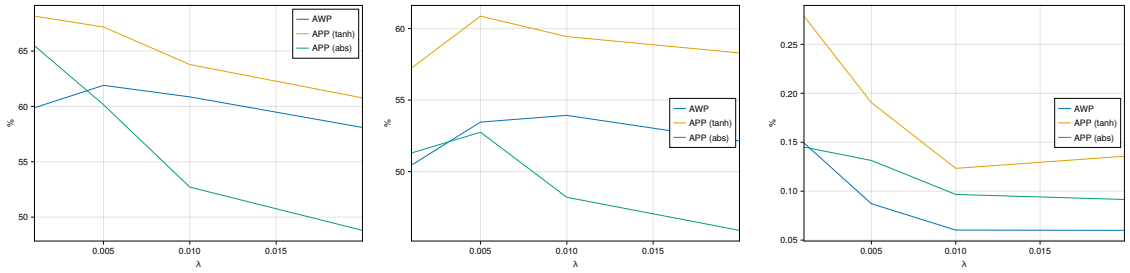
A 4.5a ábrán látható, hogy a λ növelése a normál pontosság csökkenését eredményezte. Megfigyelhető, hogy az APP (abs) esetén a teljesítménycsökkenés sokkal drasztikusabb volt, mint a másik két algoritmus mellett, mely a túlbecslések miatt megjelenő, erős regularizáló hatásnak volt köszönhető. Az AWP, illetve a APP (tanh) teljesítménycsökkenésének mértéke hasonló volt.

A 4.5b ábrán az adverzális pontosságokat (PGD-50) hasonlítottuk össze. Mindhárom algoritmus esetén megfigyelhető, hogy a paramétersugár kis mértékű növelése az adverzális pontosság növekedését eredményezte. Az APP agoritmusok esetén $\lambda = 0.005$, az AWP esetén $\lambda = 0.01$ paraméterértékek mellett értük el a legjobb teljesítményt. A λ nagyobb értékei hatására hasonló teljesítménycsökkenési tendencia figyelhető meg, mint a normál pontosságok vizsgálata során.

A 4.5c ábrán, a paramétertámadás eredményeként megjelenő robusztusságcsökkenés mértékét ábrázoltuk különböző λ paraméterek mellett. A támadási sugár 0.02 volt. A λ növelésével mindhárom algoritmus mellett megfigyelhető a paramétertámadással szembeni ellenállóképesség növekedése. A teljesítménynövekedési tendencia 0.01-es értékig volt a legjelenősebb, ezt követően lelassult.

A 4.5-es ábra, illetve 4.1 táblázat alapján elmondható, hogy az általunk vizsgált háló-

zati architektúra és osztályozási feladat esetén, a maximális bemeneti robusztusságot $\lambda = 0.005$, a paramétertámadással szembeni legellenállóbb hálózatokat pedig $\lambda = \{0.001, 0.01\}$ ($\epsilon = \frac{2}{255}$) és $\lambda = 0.02$ ($\epsilon = \frac{8}{255}$) esetén értük el az APP algoritmussal.



(a) Pontosság (b) Adverzális pontosság (c) Paraméter támadás

4.5. ábra. λ paraméter hatása

4.6. A paraméterek értékkészletének korlátozása

A tanítás során a paraméterek befoglalását relatív módon, a paraméterek abszolút értékét figyelembe véve határoztuk meg. A nagy abszolút értékű paraméterek miatt megjelenő széles intervallumoknak köszönhetően könnyen instabillá válhat a tanítás. A 3.10-es befoglalás alkalmazásán túl, a paraméterek értékkészletének a korlátozását is bevezettük a tanítási folyamatba. Ez többek között azért is volt fontos, mivel a 3.10-es befoglalás felső korlátot ad a relatív sugarakra, melynél nagyobb abszolút értékű súlyok esetén, a sugarak a felső korlát által számított értékek lesznek.

A megfelelő értékkészlet meghatározása érdekében különböző hálózatokat tanítottunk, melyek paramétereit minden tanítási iterációban visszavetítettünk a $[-R, R]$ intervallumba. A 4.4-es táblázatban különböző R vetítési sugár mellett, APP(abs) algoritmussal tanított hálózatok esetén vizsgáltuk a pontosságot, illetve az adverzális pontosságot (PGD-50). A táblázatban jól látható, hogy a legjobb eredményeket a $[-1, 1]$ értékkészletbe vetített hálózatok esetén értük el, ebben az esetben volt a legmagasabb a eredeti, illetve az adverzális pontosság is. A 4.2 és 4.3 fejezetekben részletezett, APP mellett tanított hálózatok esetén a vetítési sugár mindig 1 volt.

R	Pontosság	Adverzális pontosság
0.25	39.67%	28.48%
0.5	45.2%	29.77%
0.75	45.27%	30.27%
1	**52.71%**	**48.2%**
1.25	50.42%	47.1%
-	45.54%	42.66%

4.4. táblázat. Paraméterek értékkészletének korlátozása

4.7. A tanh paramétereinek megválasztása

A 3.10-es befoglalással kapcsolatban vizsgáltuk azt is, hogy az s paraméter különböző értékei hogyan befolyásolják a teljesítményt. Az m paraméter értékét mindig 1-re állítottuk, a 4.6-os fejezetben megállapított értékkészletkorlátnak megfelelően. Az s paraméter különböző értékeitől függően vizsgáltuk az eredeti és adverzális (PGD-50) pontosságot, valamit a paramétertámadással szembeni ellenállóképességet, a 4.5-ös táblázatban látható $\check{\lambda}$ értékek mellett. A hálózatok 0.01 paraméter és $\frac{2}{255}$ bemeneti perturbációs sugarak mellett lettek tanítva.

s		1	3	**6**	9
Pontosság		67.63%	64.13%	63.78%	55.64%
Adverzális pontosság		62.06%	61.09%	59.44%	52.03%
$\check{\lambda}$	0.02	27.6 %	22.22%	12.06%	12.3%
	0.04	37.7%	24.78%	23.08%	23.96%
	0.06	38.34%	27.2%	22.76%	23.58%

4.5. táblázat. Tanh befoglalás, s paraméterének vizsgálata

A táblázatban megfigyelhető, hogy az s növelése hatására megjelenő, egyre erősebb simító hatás következtében az eredeti és az adverzális pontosság is folyamatosan romlott. A paramétertámadással szembeni ellenállóképesség $s = 6$ értékig javult, majd ezt követően romlani kezdett. Mivel a paramétertámadással szembeni védelem $s = 6$ tanítás mellett volt maximális, illetve ekkor a hálózat bemeneti teljesítményromlása sem volt még drasztikus, így az APP (tanh) tanításaink során mindig ezt az értéket használtuk.

5. fejezet

Összegzés

A munkánk során egy új, befoglaláson alapuló tanítási módszert hoztunk létre, amely a bemeneti robusztusságon felül, a hálózatok paraméterrobusztusságának elérését is beépíti a tanítási folyamatba. A rendszerünket összehasonlítottuk a szakirodalomban is elérhető AWP módszerrel, főként a bemeneti robusztusságra, illetve az adverzális paramétertámadással szembeni védelemre koncentrálva.

A kiértékelés során megállapítottuk, hogy az algoritmusunk erősebb simító hatásának köszönhetően több esetben is javítani tudtunk a bemeneti robusztusságon, illetve az adverzális paramétertámadással szembeni ellenállóképességen is. A bemeneti robusztusság vizsgálata során, az általunk javasolt algoritmus a 3.10-es regularizáció használata mellett ért el jelentős teljesítményelőnyt az AWP-vel szemben. Paramétertámadással szembeni védelem során az APP(abs) algoritmus ért el jobb eredményeket, főként nagyobb támadási környezetben.

A kiértékelés során közepes méretű (CNN4) hálózatokat vizsgáltunk, melyeknél még kezelhető mértékben volt jelen a naiv intervallum aritmetika miatti túlbecslés. Jövőbeli kutatási céljaink közé tartozik a módszerünk továbbfejlesztése, intervallumos túlbecsléseinek további csökkentése, majd kiterjesztése nagyobb hálózatokra.

Irodalomjegyzék

[1] Julia, The Julia Programming Language. *URL: https://julialang.org*.

[2] Innes, M., Saba, E., Fischer, K., Gandhi, D., Rudilosso, M., Joy, N., Karmali, T., Pal, A. & Shah, V. Fashionable Modelling with Flux. *CoRR*. abs/1811.01457 (2018), https://arxiv.org/abs/1811.01457

[3] Wu, D., Shu-Xia & Wang, Y. Adversarial Weight Perturbation Helps Robust Generalization. (2020)

[4] Yu, L., Wang, Y. & Gao, X. Adversarial Parameter Attack on Deep Neural Networks. (2022)

[5] Mao, Y., Müller, M., Fischer, M. & Vechev, M. TAPS: Connecting Certified and Adversarial Training. (2023)

[6] Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T. & Kohli, P. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. (2019)

[7] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. (2019)

[8] Krizhevsky, A. Learning multiple layers of features from tiny images. (2009)

[9] Jovanović, N., Balunović, M., Baader, M. & Vechev, M. On the Paradox of Certified Training. (2022)

[10] Müller, M., Eckert, F., Fischer, M. & Vechev, M. Certified Training: Small Boxes are All You Need. (2023)

[11] Nguyen, H. Efficient implementation of interval matrix multiplication. *Para 2010: State Of The Art In Scientific And Parallel Computing*. (2010,6), https://inria.hal.science/inria-00469472

[12] Croce, F. & Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *ICML*. (2020)

[13] Tjeng, V., Xiao, K. & Tedrake, R. Evaluating Robustness of Neural Networks with Mixed Integer Programming. *ArXiv Preprint ArXiv:1711.07356*. (2017)

[14] Szász, A. & Bánhelyi, B. Effective inclusion methods for verification of ReLU neural networks. *Annales Mathematicae Et Informaticae*. (2024)

Köszönetnyilvánítás

Köszönetet szeretnék mondani témavezetőmnek, Dr. Bánhelyi Balázsnak a támogatásáért, valamint a Continental Autonomous Mobility Hungary Kft-nek, akik által meghirdetett ösztöndíjprogram finanszírozásával valósult meg a kutatásunk.