

OTDK-dolgozat

Halmosi Levente Ferenc

Szegmentáló Neurális Hálózatok Robusztussága

Halmosi Levente Ferenc

II. évf. programtervező informatikus

Témavezető: Jelasity Márk (egyetemi tanár)

SZTE TTIK Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszék

Kivonat

Napjainkban az ellenséges (adversarial) példák jelensége egy erősen kutatott terület a neuronháló alapú számítógépes látáson belül, amelynek lényege, hogy a bemenet szabad szemmel láthatatlan ellenséges perturbációival egy neuronháló kimenetét tetszőleges módon meg lehet változtatni. A képklasszifikáció területén ez a téma egységes módszertannal rendelkezik, ebből adódóan az ebben a témában megjelenő munkák összehasonlítása könnyedén lehetséges, az eredmények igazolhatóak. Ezzel szemben az ellenséges példák jelensége egy nem egységes, alulkutatott terület a neurális hálózat alapú képszegmentálás területén. A kutatás során létrehoztunk egy validáló algoritmus csomagot, amellyel sikerült igazolnunk azon hipotézisünket, miszerint a jelenleg elérhető legnagyobb robusztusságú szegmentáló modell valójában nem robusztus. Ezen túlmenően létrehoztunk egy olyan szegmentáló modellt, amely az eddig fellelhető hálónál jelentősen robusztusabb. Mindezzel együtt megalkottunk egy módszertant, amely reményeink szerint alkalmas lesz a módszertani kohézió megteremtésére a témán belül.

Tartalomjegyzék

1. BEVEZETÉS	6
1.1 ELLENSÉGES PÉLDÁK JELENTŐSÉGE	6
1.2 KUTATÁS MOTIVÁCIÓJA	6
1.3 CÉLKITŰZÉSEK.....	7
1.4 KONTRIBÚCIÓK.....	8
2. SZAKIRODALOM ÉS KAPCSOLÓDÓ MUNKÁK	8
2.1 KAPCSOLÓDÓ MUNKÁK A KÉPKLASSZIFIKÁCIÓBAN.....	9
2.1.1 Ellenséges példák a szakirodalomban	9
2.1.2 Gradiens alapú támadások.....	9
2.1.3 Gradiens mentes támadások.....	10
2.1.4 Auto Attack	10
2.1.5 Robust bench	11
2.1.6 Transzferálódás és valós ellenséges példák	11
2.1.7 Robusztus tanítás	11
2.1.8 Gradiens obfuszkáció	12
2.2 KAPCSOLÓDÓ MUNKÁK SZEGMENTÁLÁSBAN	12
2.2.1 Gradiens alapú támadások.....	12
2.2.2 Gradiens mentes támadások.....	12
2.2.3 Auto Attack és Robust Bench.....	13
2.2.4 Transzferálódás	13
2.2.5 Robusztus tanítás és gradiens obfuszkáció	13
3. HÁTTÉR ÉS PROBLÉMA	14
3.1 SZEGMENTÁCIÓS NEURÁLIS HÁLÓZATOK	14
3.2 SZEGMENTÁCIÓS NEURÁLIS HÁLÓ FORMALIZÁLÁSA	15
3.3 VESZTESÉGFÜGGVÉNY FORMALIZÁLÁSA	16
3.4 TANÍTÁS MINIMALIZÁLÁSI PROBLÉMAKÉNT	17
3.5 METRIKÁK FORMALIZÁLÁSA	17
3.6 AZ ELLENSÉGES PÉLDÁK FORMALIZÁLÁSA	18
3.7 AZ ELLENSÉGES PÉLDÁK JELENTŐSÉGE.....	19
3.8 ELLENSÉGES PÉLDÁK SZEGMETÁLÁSHOZ	20
3.9 ROBUSZTUSSÁG FORMALIZÁLÁSA	21
3.10 ROBUSZTUS TANÍTÁS	22
4. EGYSZERŰ TÁMADÁSOK FORMALIZÁLÁSA	23
4.1 FGSM	23
4.1.1 Célzatlan támadás	23
4.1.2 Célzott támadás	23
4.2 BIM.....	24
4.2.1 Célzatlan támadás	24
4.2.2 Célzott támadás	25
4.3 ADAM PGD	25
4.3.1 Célzatlan támadás	26
4.3.2 Célzott támadás	26
5. SAJÁT TÁMADÁSOK FORMALIZÁLÁSA	27

5.1 COSINE INNER REPRESENTATION ATTACK (CIRA)	27
5.1.1 Veszteségfüggvény részhálóhoz	27
5.1.2 Célzatlan támadás	28
5.1.3 Célzott Támadás	29
5.1.4 Előnyök és Hátrányok.....	30
5.1.5 Példák.....	31
5.2 COSINE INNER REPRESENTATION ATTACK + ADAM PGD(CIRA+AP).....	32
5.2.1 Célzatlan támadás	32
5.2.2 Célzott támadás	33
5.2.3 Előnyök és hátrányok	33
5.2.4 Példák.....	34
5.3 AUTO ATTACK SZEGMETÁLÁSHOZ.....	35
6. A ROBUSZTUS TANÍTÁS PÁRHUZAMOSÍTÁSA	35
6.1 IMPLEMENTÁCIÓ ÉS FELHASZNÁLT TECHNOLÓGIÁK.....	37
6.2 PÁRHUZAMOSÍTÁS ELŐNYEI ÉS HÁTRÁNYAI.....	38
7. KÍSÉRLETEK	38
7.1 HÁLÓK ROBUSZTUSÁGÁNAK VIZSGÁLATA	39
7.1.1 PSNET hálók robusztuságának vizsgálata	39
7.1.2 DeepLabV3 hálók robusztuságának vizsgálata	40
7.2 SAJÁT HÁLÓK ROBUSZTUSSÁGA	40
8. EREDMÉNYEK	41
8.1 PSNET HÁLÓK ROBUSZTUSÁGÁNAK VIZSGÁLATA.....	41
8.2 HÁLÓK ROBUSZTUSSÁGÁNAK VIZSGÁLATA.....	43
8.3 SAJÁT HÁLÓK ROBUSZTUSSÁGA	44
9. KONKLÚZIÓ	45
10. KÖSZÖNETNYILVÁNÍTÁS	46
11. HIVATKOZÁSOK.....	47

1. Bevezetés

Ebben a fejezetben be szeretném mutatni azokat gondolati alapvetéseket, melyek mentén felépítettük a kutatást. Ez a fejezet fogja tárgyalni a kutatás célkitűzéseit. Illetve itt kerülnek részletezésre a témában jelentős írásokkal szembeni kontribúciók is.

1.1 Ellenséges példák jelentősége

Az ellenséges példák (*adversarial examples*) lényegében olyan legtöbb esetben meglepően könnyen létrehozható inputok egy neurális hálózathoz, amelyek emberi szemmel nézve azonosnak tűnnek egy, a háló által korábban tökéletesen azonosított bemenettel. Mégis a modell az eredeti inputhoz képest radikálisan más kimenetet ad, a támadó céljának megfelelően [1]. Az ellenséges példákat intő jelként lehet értelmezni a neurális hálók megbízhatóságát illetően. A jelenség léte önmagában gátat szab a neurális hálózatok kritikus alkalmazásokon belüli felhasználásnak. Az ellenséges példák témaköre nem csupán elméleti probléma, nagyon is valós, ugyanis ellenséges példákat lehetséges létrehozni a valóságban is [2]. Mindezek mellett a probléma megoldását sürgeti az a tény, miszerint egyre elterjedtebbek a részben vagy egészen neurális hálózatokat alkalmazó kritikus alkalmazások. Ilyenek például az orvosi képfeldolgozó alkalmazások vagy akár az önvezető járművek is. A biztonságon és a megbízhatóságon túlmenően van még egy kérdéskör, mely plusz motivációként szolgálhat az ellenséges példák vizsgálatához. Ez pedig nem más, mint az interpretálhatóság kérdése. A neurális hálózatok működése még nem eléggé ismert, ezért a legtöbb alkalmazásban feketedoboz modellként tekintünk rájuk. Azonban ismert már pár kezdetleges, egészen hatékonynak tekinthető interpretációs módszer [3][4]. Az ellenséges példák vizsgálata által közelebb kerülhetünk a neurális hálózatok hibás működésének megértéshez, és ezáltal újabb lépéseket tehetünk az interpretálhatóság megteremtése felé.

1.2 Kutatás motivációja

Napjainkban az ellenséges példák vizsgálata a számítógépes látás területén belül egy népszerű, egyre erősebben kutatott terület. A szakirodalomban folyamatosan jelennek meg

írások a képklasszifikációs hálózatok robusztusságával kapcsolatban. Az utóbbi években az *auto attack*-nak és *robust bench*-nek köszönhetően egy egységes módszertan is létrejött a képklasszifikációs hálók ellenséges példáinak és robusztusságnak vizsgálatához [5][6]. Mindezek mellett létezik egységes, jól meghatározott szabály és vizsgálati szempontrendszer ahhoz, hogy meghatározzuk, hogy egy háló robusztus-e vagy sem. Ezzel szemben az egyszerű klasszifikációs problémáknál komplexebb feladatok esetén a robusztusság alulkutatott témának számít. Ilyen terület például a szemantikai szegmentáció, ahol jelenleg még nem létezik egységes módszertan a robusztusság vizsgálatához. Ebből adódóan nincsen egységes kritériumrendszer sem, amellyel meggyőződhetnénk arról, hogy egy háló robusztus. A témán belül csupán néhány cikk létezik, amely támadásokat próbál létrehozni szemantikai szegmentáláshoz [7][8]. Még kevesebb olyan cikk van, amelyben robusztus szegmentáló neurális hálót próbál létrehozni [9][10]. A szemantikai szegmentálás a képklasszifikáció után talán az egyik legfontosabb feladat a számítógépes látáson belül.

Kettő fő ok van, ami igazán érdekessé teszi a robusztus szegmentálást. Az első, hogy egyre elterjedtebbek az olyan kritikus alkalmazások, amelyek valamilyen formában alkalmaznak robusztus szegmentáló hálókat, mint például a tumor szegmentáló rendszerek vagy az önvezető járművek. A második, hogy a képklasszifikációs hálók csak nagyon szűk spektrumban tudják vizsgálni az ellenséges példákat. Ugyanakkor a robusztus szegmentáció az összetettségéből adódóan mélyebb betekintést adhat az ellenséges példák működésébe.

1.3 Célkitűzések

Jelen dolgozat célkitűzése egy olyan tesztelő csomag létrehozása, amely alkalmas arra, hogy ellenséges példákat hozzon létre szegmentáló neurális hálózatokhoz. Ennek a rendszernek a régebbi klasszikus algoritmusok mellett még két általunk fejlesztett támadást is tartalmaznia kell. A célunk ezzel a csomaggal egy egységes tesztelő eszköz létrehozása, mely gyors, hatékony, és bárki számára elérhető. Az algoritmus csomag hatékonyságát kísérletekkel kívánjuk igazolni. A kutatás során célunk a jelenlegi legmagasabb robusztusságú szegmentáló modellekről bebizonyítani, hogy nem robusztusak csupán gradienst obfuszkálnak. Mindemellett a saját módszertani kritérium rendszerünk alapján létre akarunk hozni egy robusztus szegmentáló hálót, amely az eddigi eredményeket meghaladó robusztusságot tud elérni.

1.4 Kontribúciók

(1) A kutatás során a jelenleg elérhető legmagasabb robusztuságú szegmentáló modelleket vettük alapul [11]. Az első eredményünk annak a ténynek a megállapítása, miszerint ezek a modellek valójában nem robusztusak. A támadó algoritmusok iterációjának növelésével jelentősen lehet csökkenteni a mért robusztusság értékét. Mindemellett kimutattuk, hogy a modellek egyáltalán nem robusztusak az egyik általunk fejlesztett támadó algoritmussal szemben.

(2) A második eredményünk, hogy igazoltuk a gradiens obfuszkáció létezését szemantikai szegmentáló modellek esetén. Emellett sikeresen létrehoztunk egy támadó algoritmus csomagot, mely az általunk vizsgált esetben képes volt leleplezni a gradiens obfuszkációt.

(3) A harmadik és egyben legjelentősebb eredményünk, hogy képesek voltunk betanítani egy PSNET típusú szemantikai szegmentáló modellt, amely a most szabadon elérhető hálók közül a legmagasabb robusztusságú.

(4) Mindezek mellett létrehoztunk egy eddigieken új, több videokártyán működni képes tanító környezetet, amely párhuzamosítani tudja a tanítást és az ellenséges példák generálását. A párhuzamosításnak köszönhetően ez egy gyors és hatékony eszköz lehet a robusztus szegmentáló modellek tanítására a jövőben.

2. Szakirodalom és kapcsolódó munkák

Ebben a fejezetben részletezem a kutatáshoz kapcsolódó szakirodalmakat és munkákat. Mivel a kutatás során célunk, hogy a robusztus képklasszifikáció módszertanához hasonló rendszert hozzunk létre a szemantikai szegmentáláshoz, ezért szükségeszerű, hogy rövid áttekintést végezzünk az ellenséges példák szakirodalmi alapjain a szegmentálás és klasszifikáció esetén is. Ezután már könnyedén láthatóak lesznek azok a kulcspontok, melyek a képklasszifikációnál már léteznek, de a robusztus szemantikai szegmentálásnál még nem részei a módszertannak.

2.1 Kapcsolódó munkák a képklasszifikációban

2.1.1 Ellenséges példák a szakirodalomban

Az ellenséges példák témakörének alapjául szolgáló cikke 2014-ben jelent meg Ian J. Goodfellow, Jonathon Shlens és Christian Szegedy munkásságaként [1]. A cikk az ellenséges példák létének felvetésén túlmenően részletezi az *fgs*m algoritmust, amely gyakorlatilag az első mintaadó gradiens alapú támadás. Az ellenséges példák kutatásának egyik meghatározó mérföldköve volt annak a ténynek az igazolása, hogy a valóságban is lehet olyan textúrákat létrehozni, mellyel félre lehet vezetni egy képfelismerő neuronhálót [2]. Ezáltal az ellenséges példák a gépi tanulás gyakorlatilag minden területét befolyásoló kérdéssé váltak.

Ellenséges példákat létre lehet hozni képklasszifikációhoz [12], objektum detektáláshoz [13] és szemantikai szegmentáláshoz is [14]. Ha kritikusabb alkalmazásokat nézünk akkor az önvezető járművekhez [15] vagy mellkasi röntgen elemző rendszerekhez is léteznek ellenséges példák [16]. Az ellenséges példák témaköre túlmutat a számítógépes látáson, a jelenséget ugyanúgy kutatják a természetes nyelvfeldolgozás keretein belül is [17] [18]. Emellett napjainkban a Reinforcement Learning alapú alkalmazásokhoz [19], a beszédfelismerő rendszerekhez [20], vagy éppen ajánló rendszerekhez is tudunk ellenséges példákat létrehozni [21]. Első hallásra könnyen úgy tűnhet, hogy az ellenséges példák létrehozása csupán neurális hálózatokhoz lehetséges. Valójában ellenséges példákat létre lehet hozni a tanuló algoritmusok többségéhez, mint például *Decision Trees*-hez [22] vagy akár az *SVM*-hez is [23]. Az ellenséges példák problémája gyakorlatilag a mesterséges intelligencia minden területét befolyásolja, jelenleg a tudományág egyik legmeghatározóbb kérdésköre.

2.1.2 Gradiens alapú támadások

Az ellenséges példák létrehozásához a legeffektívebb megoldások többnyire gradiens alapú algoritmusok. A legtöbb gradiens alapú támadás alapjául a *pgd* algoritmus szolgál [24]. A klasszifikációs problémák esetén a legelterjedtebb algoritmusok mind iteratív támadások, melyek white-box, vagyis ismert modell szcenárióra vannak felkészítve. A területen a legeffektívebb algoritmusok napjainkban a *apgd-ce* vagy a *apgd-dlr* [25]. Mindemellett képklasszifikációhoz léteznek olyan algoritmusok is, amelyek minimális perturbáció létrehozására törekednek, ezek közül az egyik legeffektívebb a *fab* algoritmus [26].

2.1.3 Gradiens mentes támadások

A gradiens alapú támadások mellett elterjedtek még a gradiensmentes megoldások is. Ezek a megoldások többnyire black-box ismeretlen modell szcenárióból indulnak ki. Ebben az esetben nem ismerjük a modell súlyait, ezért nincsen közvetlen információnk a modell gradienséről. A jelenleg legmeghatározóbb algoritmus a témakörön belül a *square attack*, amely a gradiensmenteségen túlmenően még iterációs lépések számában is az egyik legeffektívebb támadás [27]. A *square attack-on* kívül léteznek még genetikusan alapú támadások is, bár ezek hatékony alkalmazása erőforrásigényükből adódóan nehézkes [28]. Továbbá léteznek olyan támadások is, amelyek ellenséges rotációt vagy eltolást keresnek egy képhez [29]. Ezen algoritmusok jelentősége főként a robusztus tanításon belül keresendő és nem pedig a támadásoknál.

2.1.4 Auto Attack

A neurális hálózatok validálása nehéz és erőforrásigényes folyamat, még kis méretű háló esetén is. Közepes és nagy méretű neurális hálózatok esetén pedig egyáltalán nem kivitelezhető a napjainkban rendelkezésre álló erőforrás mennyiség mellett. Ezért az elméleti robusztusság közelítésére, azaz annak a vizsgálatára, hogy egy modellhez egyáltalán létre lehet-e hozni ellenséges példát vagy sem, támadó algoritmusokat használunk.

A támadó algoritmusok diverzitásából és széles körű paraméterezhetőségéből adódóan a szakirodalomban olvasható eredmények összehasonlítása nehézkes volt. Ezt a problémát a képklasszifikációnál az *auto attack* oldotta fel [5]. Az *auto attack* lényegében a korábban említett *apgd-ce*, *apgd-dlr*, *fab*, és *square attack* kombinációjából áll össze. A támadás során mind a négy algoritmust lefuttatja fix paraméterezéssel. A végeredmény az adott képhez a négy algoritmus által elért legalacsonyabb pontosság lesz. Összességében kijelenthető, hogy képklasszifikációs problémák esetén az *auto attack* a legelfogadottabb eszköz az elméleti robusztusság közelítéséhez.

2.1.5 Robust bench

A klasszifikációs hálók robusztusságának összehasonlításhoz szükségszerű volt létrehozni egy olyan felületet, ahol mindenki számára nyilvánosan hozzáférhetőek az egyes modellek eredményei. Erre a célra jött létre a *robust bench*, ahol top lista szerűen láthatóak a különböző modellek eredményei az *auto attack*-al szemben, a CIFAR-10, CIFAR-100 és az IMAGE-NET adathalmazokon [6]. Ezzel a *robust bench* létrehozott egy platformot, ami képes volt a módszertani kohézió létrehozására a robusztus klasszifikációs hálók témakörén belül.

2.1.6 Transzferálódás és valós ellenséges példák

Az ellenséges példákat fontos problémává két eléggé kézzelfogható tény teszi. Az első, hogy ellenséges példákat lehetséges a való életben is létrehozni [2]. A második a transzferálódás jelensége [30]. A transzferálódás azt jelenti, hogy ha létrehozunk egy ellenséges példát egy adott modellhez, akkor az egy másik modellen is működik. A transzferálódás egyik legszélsőségesebb esetei az úgynevezett univerzális perturbációk. Az univerzális perturbációk gyakorlatilag egy adott adathalmazon betanított összes modellel szemben működőképes ellenséges példák [31].

2.1.7 Robusztus tanítás

Az ellenséges példákkal szemben kevés hatékony védekezési módszer létezik. A jelenleg ismert leghatékonyabb módszer az úgynevezett robusztus tanítás [1]. Robusztus tanítás során már eleve ellenséges példákon tanítjuk az adott modellt és ezáltal az adott modell egyfajta természetes immunitást épít ki az ellenséges példákkal szemben [32]. Fontos megjegyeznünk, hogy a robusztus tanítás mind erőforrásigényben, mind alap pontosság tekintetében is hátrányos tulajdonságokkal rendelkezik a normál tanítással szemben. Általánosságban kijelenthető, hogy erőforrásigény tekintetében jelentősen nagyobb számítási kapacitást igényel, és matematikailag belátható, hogy képklasszifikációs problémák esetén a pontossága a normál tanítás pontosságához képest csökken [33].

2.1.8 Gradiens obfuszkáció

A gradiens obfuszkáció egy fontos és képklasszifikációs problémák esetén ismert jelenség. A képklasszifikációs hálók robusztus tanítása során könnyen előfordulhat, hogy a modellünk robusztusnak tűnik, pedig nem az. A modell valójában nem épít ki tényleges robusztuságot csupán a keresési tér struktúrájának módosításával ellehetetleníti azt, hogy a gradiens alapú algoritmusok megtalálják az ellenséges példákat. Ezt a jelenséget gradiens obfuszkációnak nevezzük [34]. Az ilyen típusú problémáknak a leleplezésére az *auto attack* hatékony megoldás [5].

2.2 Kapcsolódó munkák szegmentálásban

2.2.1 Gradiens alapú támadások

A szemantikai szegmentáláshoz gyártott támadó algoritmusok szakirodalma főleg a képklasszifikációból merít. Alapvetően ezen a területen is használják a klasszikus algoritmusokat mint az *fgsm* vagy a *pgd* [1]. Ezek mellett azonban léteznek sokkal

kifinomultabb megoldások is, ilyen algoritmusok például a *dag* vagy az *alma-prox* [14][35]. A szemantikai szegmentáláshoz jelentősen kevesebb támadó algoritmus jelenik meg, mint a képklasszifikációhoz. A képklasszifikáció támadásaihoz képest a szegmentáláshoz használt támadások számítási igénye jelentősen magasabb. Például az *alma-prox* esetén majdnem egy normál szegmentáló háló tanulási idejével egyenlő, ami nagyságrendekkel nagyobb, mint egy pgd vagy fgsm erőforrásigénye.

2.2.2 Gradiens mentes támadások

A kutatás során nem találtunk olyan támadó algoritmust, amely gradiensmentes módszerrel támadna szegmentáló neurális hálózatokat. Ez önmagában egy fontos hiányterület, hiszen képklasszifikációs hálózatoknál lehetséges úgy elrejteni egy hálóban ellenséges példákat, hogy gradiens alapú módszerrel nehezen vagy egyáltalán ne lehessen megtalálni.

2.2.3 Auto Attack és Robust Bench

Jelenleg a robusztus szegmentálás témakörén belül nincsen egységes tesztkörnyezet mellyel a szegmentáló hálózatok robusztusságát vizsgálhatnánk. Ily módon nincs megfelelője az *auto attack*-nak és ezáltal nem is létezik egységes toplista a *robust bench*-hez hasonlóan, ahol láthatnánk a modelleket egy egységes összehasonlítás keretein belül. Ez önmagában ellehetetleníti a szakirodalomban megjelenő eredmények hatékony összehasonlítását. Az előző ponthoz hasonlóan ez is egy hiányterület.

2.2.4 Transzferálódás

Szemantikai szegmentálásnál a képklasszifikációhoz hasonlóan jelen van a transzferálódás jelensége [36]. Könnyedén lehetséges olyan ellenséges példákat létrehozni, amelyek képesek akár egyszerre több különböző modellt is félrevezetni a támadó céljának megfelelően.

2.2.5 Robusztus tanítás és gradiens obfuszkáció

A szemantikai szegmentáláson belül a robusztus tanítás egy eléggé alulkutatott terület. A robusztus szegmentáláshoz általában hasonló algoritmusokat használnak, mint amiket a robusztus klasszifikációnál szokásos alkalmazni. Ilyen algoritmusok például az *fgsm* vagy a *pgd*. Ezeknél erősebb támadások alkalmazása nehézkes, mivel ez az erőforrásigény jelentős növekedésével jár. Ennek a megoldására már léteznek egészen hatékony megoldások, ilyen például *ddc-at* [11]. Azonban a gyenge tesztelő eljárásokból adódóan, amik a legtöbb esetben csupán klasszikus algoritmusok (például a *pgd*) pár alkalommal való iterációját jelentik, a gradiens obfuszkáció jelensége nem kizárható. Éppen ezért jogos a kritikus hozzáállás ezekhez az eredményekhez.

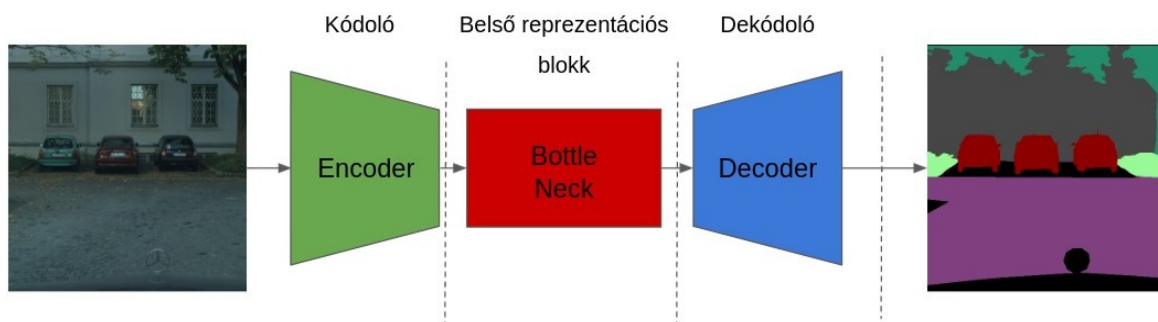
3. Háttér és probléma

Ebben a fejezetben kerülnek tárgyalásra a kutatáshoz kötődő technológiák és módszerek, ideértve a szegmentáláshoz használt neurális hálózatok struktúráját, az ellenséges példákat, a robusztusság fogalmának meghatározását, és támadó algoritmusok matematikai formalizálását.

3.1 Szegmentációs neurális hálózatok

Napjainkban a szemantikai szegmentálás problémáját neurális hálózatok alkalmazásával oldják meg. Ezek többnyire konvolúciós *autoencoderek*, de egyre népszerűbbek a transformer alapú megoldások is. Az *autoencoder* a neurális hálózat egy fajtája. Főbb jellemzője, hogy a kimeneti és a bemeneti mátrix szélessége és magassága azonos. Szemantikai szegmentálás során az autoencoder célja az eredeti kép minden pixelének egy osztályhoz való rendelése, ezáltal egy szegmentációs maszk létrehozása az eredeti képhez. Ez egy $3 \times n \times n$ -es kép esetén egy $c \times n \times n$ -es mátrix lesz, ahol c a lehetséges osztályok száma. A szegmentációs maszk vagy prediktált maszk minden eleme az adott pixelhez a háló által az osztályokhoz becsült valószínűségek eloszlása. Ha a prediktált maszkon minden pixelhez vesszük a legmagasabb valószínűségű osztálycímét, akkor egy $n \times n$ -es mátrixot kapunk, ez a háló által becsült szemantikai maszk.

Az *autoencoder* típusú neurális hálózatokat három nagy szerkezeti részre osztjuk, az első a kódoló (encoder) rész, ez a hálózat első pár rétegét jelenti. Ennek a célja a bemeneti kép dimenziójának redukálása, ezzel egy alacsonyabb dimenziójú reprezentáció létrehozása a képhez. A második rész a neurális háló középső pár blokkja, a belső reprezentációs blokk (bottleneck). Ennek főbb jellemzője, hogy a bemeneti és kimeneti adatainak dimenziója azonos. Az utolsó blokk a dekódoló (decoder), ez a rész gyakorlatilag a belső reprezentációs blokk kimenetét méretezi fel a kívánt kimeneti méretre.



Az szemantikai szegmentáláshoz használt autoencoderek főbb jellemzője, hogy kódoló részük általánosságban egy előtanított klasszifikációs háló rétegeiből áll. Ez lehet például egy Resnet50, VGG16 vagy egyéb más klasszifikációs háló részhálója is. Ez a hálórész lényegében egy előtanított információ feldolgozóként funkcionál a szegmentáló hálón belül. Ez sok szempontból kedvezően hat a tanulásra, gyorsítja a háló konvergálását az optimális megoldáshoz, redukálja a szükséges adatmennyiséget, általánosságban jobb megoldást hoz létre és csökkenti a túltanulás valószínűségét.

3.2 Szegmentációs neurális háló formalizálása

A neurális hálózat által kapott bemeneti kép legyen $X \in \mathbb{R}^{3 \times n \times n}$ és adott neurális hálózat súlyainak jele legyen θ . A kódoló, belső reprezentációs blokk, és dekódoló blokkok súlyai legyenek rendre $\theta^e, \theta^b, \theta^d$, ahol $\theta = (\theta^e, \theta^b, \theta^d)$. A neurális hálózat lehetséges súlyainak halmaza legyen W , tehát $\theta \in W$. A neurális hálózat egymást követő blokkjait tekintsük egymásba ágyazott függvényeknek a következőképpen:

Kódoló(Encoder) :

$$l_1 = E(\theta^e, X)$$

Belső reprezentációs blokk(Bottleneck) :

$$l_2 = B(\theta^b, l_1)$$

Dekódoló(Decoder) :

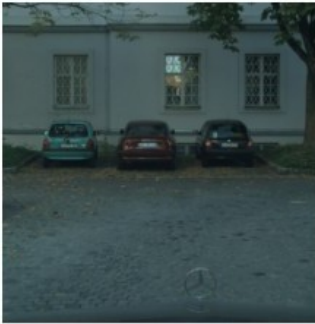
$$l_3 = D(\theta^d, l_2)$$

Ebből következően definiálhatjuk a modellünket az egyszerűbb hivatkozhatóság végett.

$$M(\theta, X) = D(\theta^d, B(\theta^b, E(\theta^e, X)))$$

A modellekről tudjuk, hogy összetett függvényként differenciálhatók, illetve hogy X mentén parciálisan is differenciálhatók. A neurális hálózat által prediktált kimenet legyen $P \in \mathbb{R}^{c \times n \times n}$. A neurális háló által prediktált végső szemantikai maszk, ami csak a minden pixelhez nézett legvalószínűbb osztály indexét tartalmazza legyen $P^* \in \mathbb{Z}^{n \times n}$. Az adott képhez tartozó célmaszk amely csupán az adott pixelhez nézett célosztály indexeit tartalmazza legyen $Y^* \in \mathbb{Z}^{n \times n}$. Illetve a minden pixelhez nézve az összes osztályra nézett valószínűségek értékeit tartalmazó mátrix legyen $Y \in \mathbb{R}^{c \times n \times n}$ gyakorlatilag ez a tanítás során a hálózattól elvárt célérték.

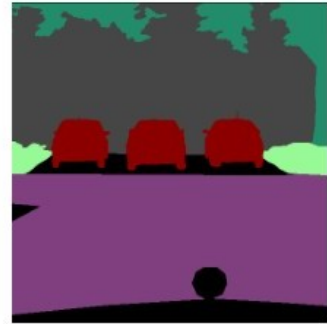
Bemeneti kép : $X \in \mathbb{R}^{3 \times n \times n}$



Prediktált maszk : $P^ \in \mathbb{Z}^{n \times n}$*



Cél maszk : $Y^ \in \mathbb{Z}^{n \times n}$*



3.3 Veszteségfüggvény formalizálása

A $J(M, \theta, D)$, veszteségfüggvény egy olyan differenciálható függvény, amelynek paraméterei sorrendben a modell, a modell paraméterei és az adatbázis. A veszteségfüggvény megadja, hogy az adott modellünk a bemenő képek alapján mekkora pontossággal tudja megbecsülni a kívánt célmaszkokat. A D adatbázis melynek m darab eleme van a következőképpen adjuk meg:

$$D = \{(X_1, Y_1), (X_2, Y_2), \dots (X_m, Y_m)\}$$

A J veszteségfüggvény a mi esetünkben mindig a cross-entropy veszteségfüggvényt fogja jelölni, ami egy igen gyakran használt veszteségfüggvény a szemantikai szegmentálásban. Legyen Y_{ij} az adathalmaz i -edik képének j -edik osztályhoz tartozó valószínűségek értékeit tartalmazó mátrix. Az $M(\theta, X_i)_j$ legyen a modell által az adatbázis i -edik képéhez becsült maszk j -edik osztályhoz tartozó valószínűségeket tartalmazó mátrix. Így a veszteségfüggvény a következő:

$$\sigma(P) = \frac{e^{P_j}}{\sum_{j=1}^c e^{P_j}}$$

$$J(M, \theta, D) = \frac{1}{f} \sum_{i=1}^f \left(- \sum_{j=1}^c Y_{ij} \log(\sigma(M(\theta, X_i)_j)) \right)$$

3.4 Tanítás minimalizálási problémaként

A hálózatok tanítása során a célunk az, hogy egy olyan θ paramétert találjunk a hálózatnak, mely az adathalmazon a hibafüggvény szerinti minimális hibaértéket éri el. Ebből következően tudjuk definiálni a tanítást, mint egy minimalizálási problémát:

$$\min_{\theta \in W} J(M, \theta, D)$$

A tanítás során az adott halmazon iterálunk végig és mindig az adott képre prediktált maszk és célmaszk különbség alapján megváltoztatjuk a neurális hálózat paramétereit. Mindezt tesszük a batch gradiens módszer szerint, ahol a gradienst a backpropagation algoritmussal számoljuk ki [37].

3.5 Metrikák formalizálása

A neurális hálózat eredményeinek kiértékeléséhez a hibafüggvényen túlmenően több másik metrikát is szokás használni. Az első és egyben az egyik legnépszerűbb metrika az *accuracy* vagy jelen esetben pixel szintű pontosság. A pixel szintű pontosság gyakorlatilag a cél és a prediktált maszkon megegyező pixelekhez tartozó címkék száma osztva a célmaszk pixeleinek számával. Fontos megjegyzés, hogy ebben a bekezdésben P^* és Y^* mátrixokat itt a mátrixok (koordinátával ellátott) elemeinek halmazaként értelmezzük:

$$ACC(P^*, Y^*) = \frac{|P^* \cap Y^*|}{|Y^*|}$$

A második metrika az mIOU érték, amely a legelterjedtebb módja a szegmentáló neurális hálózatok kiértékelésének. Az mIOU érték megértéshez először definiáljuk a szimpla IOU értéket. Legyen P_k^* az a prediktált maszk, amit úgy definiálunk, hogy ahol a háló a k-adik osztályt prediktálja, ott 1 az értéke, mindenhol máshol pedig 0. Ugyanígy legyen Y_k^* az a maszk, melyben minden pixelnél, ahol az eredeti maszkon a k-adik osztály volt, az értéket 1-re állítjuk, mindenhol máshol pedig 0-ra. Így a IOU érték kiszámítása a k-adik osztályra nézve a következő:

$$IOU(P_k^*, Y_k^*) = \frac{|P_k^* \cap Y_k^*|}{|P_k^* \cup Y_k^*|}$$

ahol a metszet és unió az 1 értékű pixelek halmazán értendő. Ebből következően az mIOU érték nem más, mint az összes osztályra kiszámított IOU értékek átlaga:

$$mIOU(P^*, Y^*) = \frac{1}{c} \sum_{i=1}^c IOU(P_i^*, Y_i^*)$$

3.6 Az Ellenséges példák formalizálása

Ahhoz hogy egy X képhez tartozó X' képet ellenséges példának nevezzünk, két feltételnek kell teljesülnie. Az első, hogy adott L^1, L^2, L^∞ távolság metrika mentén a (mi esetünkben ez mindig L^∞ lesz) az adott X' kép az X kép ϵ sugarú környezetén belül található. Vagyis ez formálisan annyit jelent hogy:

$$\|X - X'\|_\infty < \epsilon$$

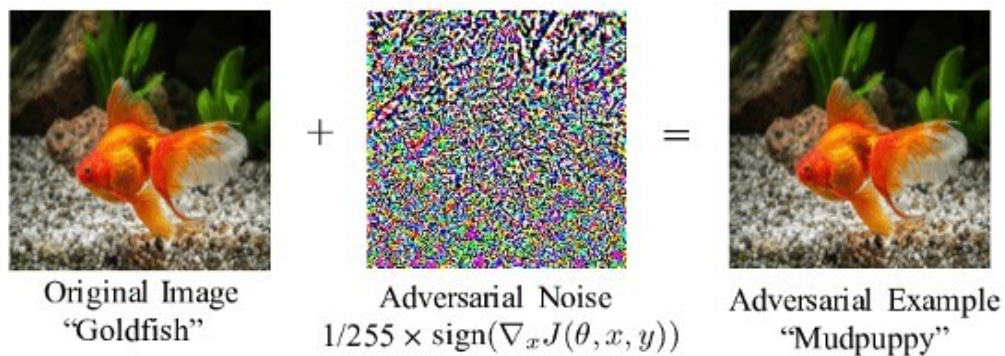
A második, hogy a pixel szintű különbség a célmaszk és a prediktált maszk között egy adott κ szám alatti legyen:

$$ACC(P^*, Y^*) < \kappa$$

Ez a kritérium azért szükséges, hogy egy bizonyos pontosság érték változáshoz tudjuk kötni azt, hogy egy példa ellenséges vagy sem. A klasszifikációs problémáknál sokkal egyértelműbb, hogy egy példa ellenséges vagy sem, hiszen ha a modell a perturbált képre más osztályt becsül, mint az eredeti esetben, akkor az a példa ellenséges. A szegmentálásnál ez nem ilyen egyértelmű, hiszen attól, hogy az új perturbált képre adott prediktált maszk pár pixelben eltér az eredetitől, a maszk még mindig lehet az alkalmazás szempontjából teljesen jó. Egy maszk akkor válik az alkalmazás szempontjából rossz, ha a pixelek kellően nagy része átfordul, melyre jelenleg a szakirodalomban nincsen konszenzusos érték. Ezért annak mérésére, hogy egy támadás mennyire hatékony az mIOU értéket használjuk mely egy 0 és 1 közötti valós érték lehet.

3.7 Az ellenséges példák jelensége

Az ellenséges példák eléggé látványos jelenségek és a szemléltetésük viszonylag egyszerű. A példa kedvéért vegyünk egy egyszerű klasszifikációs hálót, amelynek célja, hogy állatokat azonosítson egy képen. Alapjáraton a háló remekül végzi el a munkát, azonban csupán a pixelek egy árnyalattal való megváltoztatása után, a háló az eredetivel szemben radikálisan különböző eredményt ad, az aranyhalból kutya lesz. Holott szemmel láthatóan nincsen különbség a képek között.



<https://bit.ly/3hKicPP>

3.8 Ellenséges példák szegmentáláshoz

Az ellenséges példák jelensége talán a szemantikai szegmentálásnál még látványosabb. Vegyünk egy másik, szintén egyszerű példát. Egy szemantikai szegmentáló háló célja, hogy egy önvezető járművön belül a kamerától kapott képek alapján azonosítsa az úton lévő objektumokat. A standard objektum detektáláson túlmenően ez a rendszer szemantikai szegmentálást is végez. Kezdetben a rendszer helyesen működik: a validációs halmazon is magas mIOU értéket ér el, és szép, egyértelmű maszkokat hoz létre:



Azonban, ha minimális ellenséges perturbációt helyezünk el a képeken, akkor az eredmény egy radikálisan különböző maszk lesz. Az eredeti, illetve perturbált kép közötti különbség szabad szemmel láthatatlan.



Ez a jelenség már önmagában is veszélyes, hisz teljesen összezavarhatja egy ilyen képfeldolgozó rendszer eredményeit. Ráadásul ellenséges példákkal nem csupán ilyen teljesen random maszkokat tudunk létrehozni. Képesek vagyunk célzottan egy általunk specifikált maszkot is létrehozni kimenetként csupán egy minimális perturbációval. Vegyünk egy másik képhez tartozó célmaszkot:



A legelső képekhez lehetséges olyan perturbációt készíteni amelyekre a második kép célmaszkjához nagyon hasonló kimenetet ad az adott neurális háló. Mégis a perturbált kép szemmel láthatóan nem különbözik az eredeti képtől.



Ez gyakorlatilag annyit jelent, hogy egy nem robusztus szegmentáló modellhez egy adott kép esetén közel bármilyen kimenetet létre lehet hozni úgy, hogy az eredeti és a perturbált képet

szemmel nem lehet megkülönböztetni. Itt rövid megjegyzésként hozzá szeretném tenni, hogy a szegmentálási példákban lévő képek nem csupán illusztrációs jellegűek, hanem az általunk létrehozott ellenséges példák, amelyek valóban működnek az adott neuronhálón.

3.9 Robusztusság formalizálása

A robusztus neuronháló, illetve maga a robusztusság kulcskérdése ennek a dolgozatnak. Ebből adódóan szükségszerű hogy konkretizáljuk, hogy mit is értünk ezalatt. Egy neurális hálózatot robusztusnak tekintünk egy adott D adathalmazon amennyiben minden adathalmazbeli képre teljesül, hogy egy adott ϵ környezetű terében nem lehet hozzá ellenséges példákat létrehozni. Itt felmerülhet a kérdés, hogy azokat a rossz maszkokat, melyeket a neuronháló eleve rosszul szegmentál, ellenséges példáknak tekintjük-e vagy sem. Itt ezeket is annak fogjuk tekinteni, bár ezen kérdéskör tárgyalása túlmutat a dolgozat keretein [38].

3.10 Robusztus tanítás

A robusztus tanítás a jelenleg ismert leghatásosabb módja egy neurális hálózat robusztussá tételének. A standard tanítással szemben ebben az esetben az adott modellet ellenséges példákon tanítjuk. Általánosságban a robusztus tanítást olcsó és könnyen iterálható algoritmusokkal végzik, amelyeket alacsony iterációs számmal használnak. Ilyen támadások a például az fgsm, vagy a bim.

A robusztus tanítási folyamat lépésekre osztva a következő. Első lépésben az adott adathalmazból veszünk egy képet és ehhez létrehozunk egy perturbációt egy támadó algoritmussal. Második lépésben átadjuk a perturbált képet a modellnek amely ez alapján prediktál egy maszkot. Harmadik lépésben a célmaszkot összehasonlítva a prediktált maszkkal kiszámítjuk a veszteségfüggvény értékét. Ezt követően a negyedik lépésben a back propagation segítségével módosítjuk a modell paramétereit és folytatjuk a következő képpel a modell tanítását. A robusztus tanítás komoly hátránya, hogy erőforrás- és időigényesebb, mint a normál tanítás.

A robusztus tanítás magasabb erőforrásigénye három forrásból adódik. Az első az ellenséges példák létrehozásához szükséges plusz memória. A második a robusztus tanításból adódó magasabb tanítási iterációs szám, és a harmadik az ellenséges példák generálásának és magának a tanításnak a szekvenciális jellege. Utóbbi gyakorlatilag annyit jelent, hogy a hálónak a tanulás során mindig várnia kell az adott ellenséges képek legenerálására, épp ezért a tanítási folyamat időben jelentősen kitolódik.

4. Egyszerű támadások formalizálása

4.1 FGSM

Az *fgsm* (fast gradient sign method) támadás az egyik legegyszerűbb nem iteratív gradiens alapú támadás, amelynek létezik célzott és célzatlan verziója is. A támadást gyakorlatilag az ellenséges példák témakörében az alap algoritmusnak tekintik.

4.1.1 Célzatlan támadás

A támadás definiálása előtt szükségszerű, hogy definiáljuk a *sign* (előjel) függvényt ugyanis ez része az algoritmusnak. Először definiáljuk a *sign* függvényt x valós számokra:

$$\text{sgn } x := \begin{cases} -1 & \text{ha } x < 0, \\ 0 & \text{ha } x = 0, \\ 1 & \text{ha } x > 0. \end{cases}$$

A $\text{Sign}(X)$ jelölje a *sign* függvény elvégzését az adott X mátrix összes elemére. Ezután már definiálhatjuk az *fgsm* célzatlan változatát. A célzatlan esetben a célunk az, hogy olyan perturbációt hozzunk létre, amely a célmaszktól veszteségfüggvény szerint a lehető legmesszebb viszi a prediktált maszkt. Célzatlan esetre a következőképpen írható fel:

$$X' = X + \varepsilon \cdot \nabla_X \text{Sign}(J(M, \theta, X, Y))$$

4.1.2 Célzott támadás

A célzott verzió esetén a célunk az, hogy olyan perturbációt hozzunk létre, amely a prediktált maszkot egy általunk megadott $Y' \in \mathbb{R}^{c \times n \times n}$ cél maszkhoz viszi a lehető legközelebb.

$$X' = X - \varepsilon \cdot \nabla_X \text{Sign}(J(M, \theta, X, Y'))$$

A két algoritmus nagyon hasonló, az eltérés lényegében a gradiens előjelében és a célmaszkban van. Az első esetben távolodni szeretnénk egy adott célmaszktól, míg a második esetben közelíteni szeretnénk felé. Ennek megfelelően választunk, hogy hozzáadjuk vagy kivonjuk a gradiens előjelét. Célzott esetben kivonjuk, célzatlanban pedig hozzáadjuk. A *Sign* függvény és az *é* lépésköz garantálja a lehetséges maximális távolságot az eredeti és a perturbált kép között azon feltétel mellett, hogy az eredeti, illetve a perturbált kép végtelen normával nézett távolsága egy adott ε számnál ne legyen nagyobb.

4.2 BIM

Az fgsm-nél hatásosabb támadás a *bim* (basic iterative method) ami az fgsm-mel szemben egy iteratív támadás. Ugyanúgy gradiens alapú, mint az fgsm és szintén létezik célzott és célzatlan változata. A sok iterációból adódóan itt nem tudjuk a gradiens skálázásával garantálni a maximális ε távolságot. Ezért be kell vezetnünk egy visszavágó függvényt. Legyen a $\text{Clip}(X, X', \varepsilon)$ függvény az adott X' értékeinek visszavágása az X végtelen norma szerint nézett ε sugarú környezetébe. Ez annyit tesz, hogy a művelet elvégzése után X' minden eleme benne lesz az $[X - \varepsilon, X + \varepsilon]$ intervallumban.

4.2.1 Célzatlan támadás

Ezután már egy adott maximális I_{max} iteráció mellett definiálhatjuk a *bim* támadást célzatlan esetre a következőképpen:

```

1 Set  $i = 1, X^1 = X$ 
2 while  $i < I_{max}$  do
3    $X^{i+1} = Clip(X, X^i + \alpha \cdot Sign(\nabla_{X^i} J(M, \theta, X^i, Y)), \varepsilon)$ 
4    $i = i + 1$ 
5 end
6 Set  $X' = X^i$ 

```

Az α érték a gradiens vektor és ezzel a lépésköz skálázására használt paraméter. Ennek az értéke helyzetfüggő, és a megfelelő érték megválasztása jelentős hatással lehet a támadás sikerességére.

4.2.2 Célzott támadás

A *bim* támadás esetén ugyanúgy, mint az *fgsm* támadás esetén lehetséges célzott támadást definiálni. Ehhez ugyanúgy csupán egy célmaszkra lesz szükségünk, illetve ebben az esetben a gradienst ki kell vonnunk.

```

1 Set  $i = 1, X^1 = X$ 
2 while  $i < I_{max}$  do
3    $X^{i+1} = Clip(X, X^i - \alpha \cdot Sign(\nabla_{X^i} J(M, \theta, X^i, Y')), \varepsilon)$ 
4    $i = i + 1$ 
5 end
6 Set  $X' = X^i$ 

```

A *bim* gyakorlatilag a legegyszerűbb iteratív gradiens alapú támadás, amit használhatunk, mind klasszifikációs, mind szegmentálási problémák esetén.

4.3 ADAM PGD

Az ellenséges példák létrehozásánál a neurális hálózatok tanításához hasonlóan sok esetben alkalmazunk optimalizálókat. Ezeknek a megoldásoknak a célja, hogy segítsék az optimalizáció során a konvergálást az optimumhoz. Általában ezt a gradiens vektor

hosszának skálázásával teszik vagy az úgynevezett momentum módszerrel. A momentum módszer során gyakorlatilag az új gradiens vektor értéke mindig az adott iteráció beli gradiens érték és az öt megelőző gradiens érték valahányad részének összegeként áll össze. Ennek a lényege, hogy érzéketlenné tegye az optimalizációt a hirtelen változó gradiensre, ezáltal javítsa a globális optimumhoz való konvergálást. Az egyik leghatékonyabb és legelterjedtebb optimalizáló algoritmus az *Adam*. Alapvetően ez az algoritmus főként a gradiens vektor hosszának skálázásával foglalkozik. Ezt egy saját belső optimalizáció keretein belül teszi. Az *adam pgd* a standard *pgd* algoritmus kiegészítése egy *adam* optimalizáló algoritmussal. A *pgd* algoritmus gyakorlatilag egy másik megnevezése a *bim* támadásnak. Definiáljuk az *adam* optimalizáló algoritmust a következőképpen. Legyen $Adam(\theta^o, G)$ az optimalizáló algoritmus ahol θ^o az optimalizáló paraméter és G egy adott gradiens vektor. Az optimalizáló minden lépésben megadja az adott gradiens új értékét, illetve az optimalizáció új paramétereit.

$$\theta^{new}, G^{new} = Adam(\theta^o, G)$$

4.3.1 Célzatlan támadás

Mindezek után már képesek vagyunk definiálni az *adam pgd* támadást I_{max} maximális iterációs lépés és θ_{init}^o optimalizáló kezdő paramétereit mellett.

```

1 Set  $i = 1, X^1 = X, \theta^1 = \theta_{init}^o$ 
2 while  $i < I_{max}$  do
3    $G = \nabla_{X^i} J(M, \theta, X^i, Y)$ 
4    $\theta^{i+1}, G^* = Adam(\theta^i, G)$ 
5    $X^i = X^i + G^*$ 
6    $X^{i+1} = Clip(X, X^i, \varepsilon)$ 
7    $i = i + 1$ 
8 end
9 Set  $X' = X^i$ 
```

4.3.2 Célzott támadás

Az előzőekhez hasonlóan itt is tudunk definiálni célzatlan támadást ugyanúgy, mint a korábbiakban, szükségünk van egy célmaszkra, illetve itt is ki kell vonnunk majd a gradienst.

```
1 Set  $i = 1$ ,  $X^1 = X$ ,  $\theta^1 = \theta_{init}^o$ 
2 while  $i < I_{max}$  do
3    $G = \nabla_{X^i} J(M, \theta, X^i, Y')$ 
4    $\theta^{i+1}, G^* = Adam(\theta^i, G)$ 
5    $X^i = X^i - G^*$ 
6    $X^{i+1} = Clip(X, X^i, \varepsilon)$ 
7    $i = i + 1$ 
8 end
9 Set  $X' = X^i$ 
```

A korábban definiált *bim* támadásnál az *adam pgd* hatékonyabb támadás, illetve az *adam* optimalizálónak köszönhetően sokkal kevésbé érzékeny a gyorsan változó gradiensekre.

5. Saját támadások formalizálása

Ebben a részben szeretném részletezni és formalizálni azokat az algoritmusokat, melyeket a kutatás során létrehoztam. Továbbá be szeretném mutatni, milyen előnyökkel és hátrányokkal rendelkeznek ezek az algoritmusok, illetve ellenséges példákat is szeretnék bemutatni, amelyeket ezekkel az algoritmusokkal hoztam létre.

5.1 Cosine Inner Representation Attack (CIRA)

5.1.1 Veszteségfüggvény részhálózhoz

Ez a támadás a korábban tárgyalt algoritmusokhoz képest két dologban tér el. Az első, hogy a célunk közvetlenül nem a modell kimenetének megváltoztatása, hanem az adott

képhez szeretnénk egy olyan perturbációt készíteni, amely megváltoztatja a modell belső reprezentációját. Ez csupán annyit jelent, hogy a korábban megadott módszerekkel ellentétben nem a J veszteségfüggvény differenciálásval fogjuk meghatározni az adott perturbációt. Emellett nem is az egész modellt hanem csupán a dekódoló és a belső reprezentációs blokkot fogjuk használni. Ehhez először definiáljunk egy függvényt, amely megadja az adott X képhez tartozó kimenetet a belső reprezentációs rétegben:

Legyen $l_k^{(2)}$ a belső reprezentációs blokk kimenete az adott X^k képhez, ez a következőképpen kapható meg:

$$B^*(\theta^e, \theta^b, X) = B(\theta^b, E(\theta^e, X))$$

$$l_k^{(2)} = B^*(\theta^e, \theta^b, X^k)$$

A támadások definiálásához szükségünk lesz egy a belső reprezentációk összehasonlítására alkalmas veszteségfüggvényre. Ennek alapjául a koszinusz hasonlóság szolgál, amelyet a következőképpen kaphatunk meg:

$$CosSim(l_k^{(2)}, l_z^{(2)}) = \frac{l_k^{(2)} \cdot l_z^{(2)}}{||l_k^{(2)}||_2 \cdot ||l_z^{(2)}||_2}$$

A koszinusz távolság mentén már definiálhatjuk a veszteség függvényt a következőképpen:

$$CosDiff(l_k^{(2)}, l_z^{(2)}) = 1 - CosSim(l_k^{(2)}, l_z^{(2)})$$

Fontos hozzátenni, hogy a koszinusz hasonlóság és a veszteségfüggvényünk is egy differenciálható függvény.

5.1.2 Célzatlan támadás

A *cira* támadás a *pgd adam* algoritmushoz hasonlóan szintén *adam* optimalizáló algoritmust használ. Ennek az oka, hogy a belső reprezentációk esetén az optimális lépésköz meghatározása nehézkes, és egy adaptív megoldás sokat javít a hatékonyságon. Az algoritmus működésének alapja a belső reprezentációk eltérítése. Nem feltétlenül tűnhet egyértelműnek, hogy ha a hálózat belső reprezentációját eltérítjük, akkor a tényleges kimenete is más lesz a hálónak. Ezt a hipotézisünket a későbbiekben bemutatott eredményeink igazolják.

Az algoritmus definiálásához szükséges a cél aktiváció, hiszen az optimalizálás során definiálnunk kell azt az aktivációt, melytől távolodni szeretnénk a perturbáció létrehozásával. A belső aktivációt bármilyen X képhez megkaphatjuk a reprezentációs blokk aktivációjának elmentésével. Ha egy adott X^k -hez szeretnénk ellenséges példát létrehozni céltalan támadással, akkor az adott kép $l_k^{(2)}$ belső reprezentációjától szeretnénk távolodni egy X' perturbált kép segítségével. Az első iterációs lépésben az X^k és az X' képek azonosak, hiszen még nem adtunk az adott képhez semmilyen perturbációt, így módon a hozzájuk tartozó belső reprezentációk megegyeznek. Amennyiben adott X' -nél a belső reprezentációk koszinusz távolságának gradiense nulla, a gradiens módszer nem tud elindulni. Ennek kiküszöbölésére az adott $l_k^{(2)}$ kezdeti aktiváció elemeihez mesterségesen egy adott δ számot adunk mellyel célunk az, hogy az első lépésben, ahol az eredeti kép és a perturbált kép még azonos, nullától különböző gradiens értéket kapjunk a belső reprezentációk eltéréséhez. Mindezek után már képesek vagyunk definiálni a támadást a következőképpen:

```

1 Set  $i = 1$ ,  $X^1 = X$ ,  $\theta^1 = \theta_{init}^o$ ,  $l_t^{(2)} = l^{(2)} + \delta$ 
2 while  $i < I_{max}$  do
3    $l_i^{(2)} = B^*(\theta^e, \theta^b, X^i)$ 
4    $G = \nabla_{X^i} CosDiff(l_i^{(2)}, l_t^{(2)})$ 
5    $\theta^{i+1}, G^* = Adam(\theta^i, G)$ 
6    $X^i = X^i + G^*$ 
7    $X^{i+1} = Clip(X, X^i, \varepsilon)$ 
8   if  $i = 1$  then
9      $l_t^{(2)} = l^{(2)}$ 
10  end
11   $i = i + 1$ 
12 end
13 Set  $X' = X^i$ 

```

5.1.3 Céltott Támadás

A támadás definiálása céltott esetre egy kicsivel bonyolultabb, mint a korábbi esetekben. Ugyanis itt nem tudjuk figyelembe venni az adott képhez tartozó maszkot, mivel nem ismert a hozzá tartozó belső aktiváció. Emiatt csak egy másik kép belső reprezentációját tudjuk közelíteni. Más szóval, a neurális háló által az adott X -hez prediktált maszkot tudjuk közelíteni egy X_t képhez tartozó prediktált maszkhoz. Ezt úgy tehetjük meg hogy definiálunk

egy $l_k^{(2)}$ célaktivációt, amely a neurális hálózat által az X_t -hez tartozó belső aktivációval lesz egyenlő.

A *cira* támadás célzott esetében az előző algoritmusokhoz hasonlóan szintén ki kell vonunk a gradienst. Ezáltal a perturbált képhez tartozó belső reprezentáció egyre jobban fog hasonlítani az adott célképhez tartozóhoz, így a kimeneti maszk is egyre hasonlóbb lesz.

```

1 Set  $i = 1$ ,  $X^1 = X$ ,  $\theta^1 = \theta_{init}^o$ ,  $l_t^{(2)} = B^*(\theta^e, \theta^b, X_t)$ 
2 while  $i < I_{max}$  do
3    $l_i^{(2)} = B^*(\theta^e, \theta^b, X^i)$ 
4    $G = \nabla_{X^i} CosDiff(l_i^{(2)}, l_t^{(2)})$ 
5    $\theta^{i+1}, G^* = Adam(\theta^i, G)$ 
6    $X^i = X^i - G^*$ 
7    $X^{i+1} = Clip(X, X^i, \varepsilon)$ 
8    $i = i + 1$ 
9 end
10 Set  $X' = X^i$ 

```

5.1.4 Előnyök és Hátrányok

A támadásom a többi szakirodalomban lévő és korábban említett támadásokhoz képest eléggé szokatlan, mivel csupán az eredeti hálózat egy részhálózatával foglalkozik és egyáltalán nem veszi figyelembe az adott neurális hálózat kimenetét. Ebből adódóan rendelkezik néhány előnnyel és korláttal is.

Az egyik, és talán legnyilvánvalóbb előnye a kisebb számítási igény. Mivel az eredeti hálózatnak körülbelül csak a felét használjuk, ezért a szükséges számítási kapacitás és memóriaigény iterációnként körülbelül a fele az *adam pgd* értékeinek. A támadás további előnye még, hogy nincsen szüksége címkére ahhoz, hogy ellenséges példát hozzon létre. Tehát amennyiben a célunk, hogy adott képhez tartozó kimenet csupán eltérő legyen az eredetitől és nem áll rendelkezésünkre célmaszk, ez egy hatékony és gyors megoldás lehet.

Az algoritmus hátrányos tulajdonságai közé tartozik, hogy, mivel közvetlenül nem foglalkozik a veszteségfüggvénnyel, nem tudja garantálni, hogy bizonyos hibás pixelszám felett is ront a neurális hálózat eredményein. Ezen túlmenően a célzott támadásoknál csupán egy konkrét példa prediktált maszkját tudjuk közelíteni. Tetszőlegesen adott célmaszkok

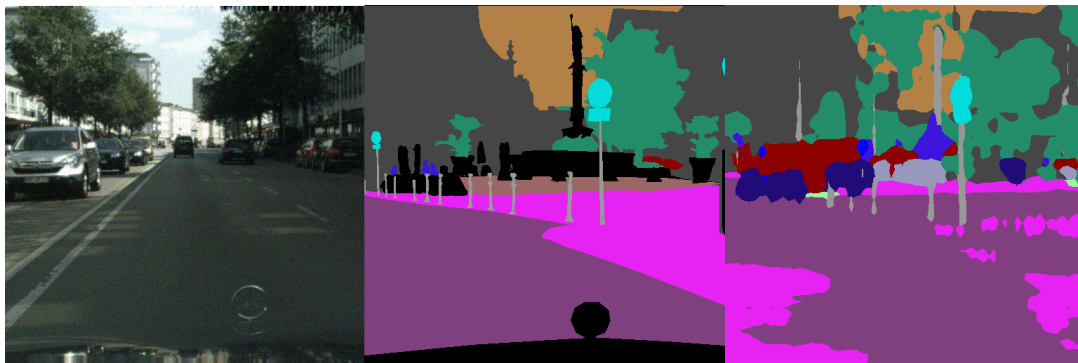
közvetlen céljai nem lehetnek a támadásnak. Ebből adódóan célzott támadások esetén jóval pontatlanabb a támadásom, mint bármelyik a korábbi támadások közül.

5.1.5 Példák

Ebben a bekezdésben be szeretnék mutatni néhány ellenséges példát, amelyet a *cira* algoritmussal hoztunk létre. Először a célzatlan támadásra nézünk egy példát:



A következő példán egy célzott támadást láthatunk. A képek sorrendben balról jobbra az ellenséges példa, a cél maszk, és a prediktált maszk:



Alapvetően az első célzatlan támadás teljesen sikeres, gyakorlatilag láthatatlan perturbációt hoz létre. A második esetben a célzott támadásnál az algoritmus nem annyira sikeres, a prediktált maszk nagyban eltér a kívánt célmaszktól. Ez alapvetően az algoritmus belső reprezentációkra való korlátozottságából adódik, hiszen csak a belső reprezentációt, és nem a tényleges kimenetet próbáltuk illeszteni.

5.2 Cosine Inner Representation Attack + Adam PGD(CIRA+AP)

A második támadás amit létrehoztunk az előző támadás kiegészítése az *adam pgd* támadással. Ez a megoldás a *cira* támadás azon hibájának kiküszöbölésére szolgál, hogy nem tudja figyelembe venni az adott képhez tartozó célmaszkot. A támadás módosításával természetesen elveszítjük a részhaló alkalmazásnak számítási kapacitás tekintetében előnyös tulajdonságait, hiszen a *adam pgd* miatt kénytelenek vagyunk az egész háló gradiensét kiszámolni. Ugyanakkor lehetővé válik a hatékony célzott támadás is.

5.2.1 Célzatlan támadás

Fontos megjegyeznünk, hogy mivel a módszerbe integráltam egy *adam pgd* algoritmus is, ezért a δ paraméter alkalmazása a továbbiakban szükségtelen a támadás elindításához. Emellett bevezetünk két paramétert: az α_1 -et és az α_2 -őt a gradiensek skálázásához a koszinusz távolság, illetve a cross entropy esetén:

```
1 Set  $i = 1$ ,  $X^1 = X$ ,  $\theta^1 = \theta_{init}^o$ ,  $l_t^{(2)} = l^{(2)}$ 
2 while  $i < I_{max}$  do
3    $l_i^{(2)} = B^*(\theta^e, \theta^b, X^i)$ 
4    $G_1 = \alpha_1 \cdot \nabla_{X^i} CosDiff(l_i^{(2)}, l_t^{(2)})$ 
5    $G_2 = \alpha_2 \cdot \nabla_{X^i} J(M, \theta, X^i, Y')$ 
6    $G = G_1 + G_2$ 
7    $\theta^{i+1}, G^* = Adam(\theta^i, G)$ 
8    $X^i = X^i + G^*$ 
9    $X^{i+1} = Clip(X, X^i, \varepsilon)$ 
10   $i = i + 1$ 
11 end
12 Set  $X' = X^i$ 
```

A támadás az *adam pdg* algoritmus gradiensének és a *cira* algoritmus gradiensének összegével dolgozik. Itt meg kell jegyeznünk, hogy az adott α_1 és az α_2 paraméterek értékének megválasztása fontos lehet, hiszen a két támadás gradiensének értéke nagyságrendekben eltérhet. Ugyanakkor tapasztalataink szerint alacsony (3-10) iterációs

lépésszámon belül mind a két paraméter azonos értékre való beállítása hatékony megoldást ad. Ugyanakkor magasabb iterációs szám esetén (10-től felfelé) a *cira* gradiensének felfelé való skálázása szükséges.

5.2.2 Célzott támadás

Az előzőekben megfogalmazottak alapján definiálhatjuk a célzott verzióját is az algoritmusnak. Ebben a verzióban a célunk, hogy a neuronháló kimenete a célmaszkhoz közeledjen, illetve hogy a neuronháló belső reprezentációja az adott cél képhez tartozó belső reprezentációhoz közeledjen. Emiatt, a *cira*-hoz hasonlóan, itt is kelleni fog a célmaszkhoz tartozó kép belső reprezentációja:

```

1 Set  $i = 1$ ,  $X^1 = X$ ,  $\theta^1 = \theta_{init}^o$ ,  $l_t^{(2)} = B^*(\theta^e, \theta^b, X)$ 
2 while  $i < I_{max}$  do
3    $l_i^{(2)} = B^*(\theta^e, \theta^b, X^i)$ 
4    $G_1 = \alpha_1 \cdot \nabla_{X^i} CosDiff(l_i^{(2)}, l_t^{(2)})$ 
5    $G_2 = \alpha_2 \cdot \nabla_{X^i} J(M, \theta, X^i, Y')$ 
6    $G = G_1 + G_2$ 
7    $\theta^{i+1}, G^* = Adam(\theta^i, G)$ 
8    $X^i = X^i - G^*$ 
9    $X^{i+1} = Clip(X, X^i, \varepsilon)$ 
10   $i = i + 1$ 
11 end
12 Set  $X' = X^i$ 

```

Ebben az esetben ismét szükséges megjegyeznünk, hogy az adott α_1 és az α_2 paraméterek megfelelő megválasztása fontos. Ebben az esetben a tapasztalataink pont fordítottak, mint az előző esetben. Itt általánosságban a *adam pgd* gradiensének adaptív skálázása bizonyult szükségesnek.

5.2.3 Előnyök és hátrányok

A támadás előnyei közé tartozik, hogy az ebben az írásban tárgyalt algoritmusok közül ez a leghatékonyabb. Emellett, ahogy a későbbiekben az eredmények tárgyalásánál látni fogjuk, egy erős és olcsó eszköz a gradiens obfuszikáció leleplezésére.

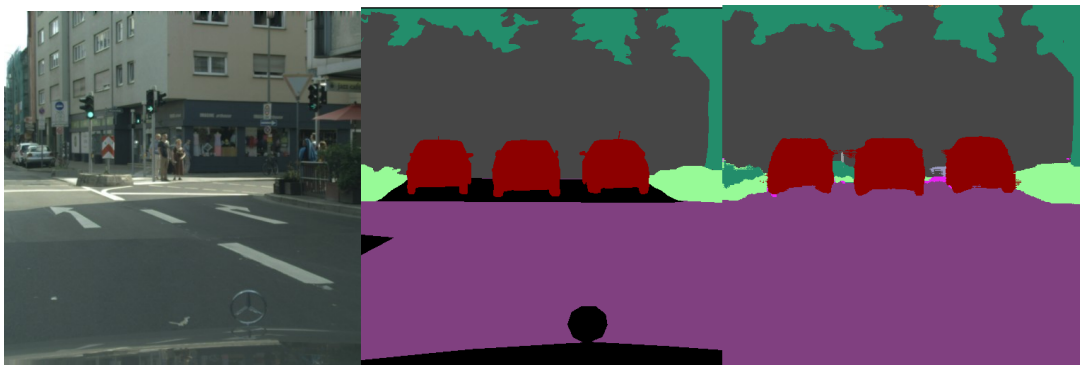
Hátrányai leginkább az erőforrásigényeiben vannak, hiszen mind a *cira* mind az *adam* *pgd* algoritmust alkalmazzuk a támadás során. Ugyanakkor a gradienst itt is egy lépésben ki tudjuk számolni.

5.2.4 Példák

Ebben a bekezdésben *cira+ap*-vel szeretnék példákat mutatni mind célzatlan, mind célzott támadásara. Először nézzünk egy célzatlan támadást:



A következő eset ugyanehhez a képhez egy célzott támadás megvalósítása. A képek sorrendben balról jobbra ellenséges példa, célmaszk, prediktált maszk.



A példákon jól látható, hogy ezzel a támadással hatékonyan lehet létrehozni ellenséges példákat mind célzott, mind célzatlan esethez.

5.3 Auto attack szegmentáláshoz

Mint ahogy ennek a dolgozatnak az elején olvasható, jelenleg nem létezik Auto Attack szemantikai szegmentáláshoz. Éppen ezért a kutatás során célunk volt a támadások implementációjának egy egységes csomagban való összegzése. Ennek a rendszernek a létrehozása során arra törekedtünk, hogy igazodjunk a neurális hálózatok témakörében domináns technológiai trendekhez. Ezért ez egy Python 3.0, Pytorch alapú nyitott forráskódú rendszert lett.

Alapvetően ez a támadó csomag lehetővé teszi, hogy szegmentáló neurális hálózatokat teszteljünk a CityScapes adathalmazon, ami jelenleg a legnépszerűbb teszt adathalmaz a szegmentáló hálók témakörében. A tesztelésre több lehetőségünk van, lehetséges a négy támadás a *bim*, *pgd adam*, *cira*, *cira+ap* -vel való külön-külön tesztelés. Illetve lehetséges *auto attack* tesztet csinálni ami során mind a négy algoritmus lefut és minden képhez végeredményképpen a legrosszabb eredmény kerül mentésre.

Ez a támadó csomag reményeim szerint hiánypótló megoldás lesz a szemantikai szegmentáló neurális hálózatok robusztuságának témakörén belül. Mindemellett bízom benne, hogy a program szabad hozzáférhetősége serkentően hat majd az ebben a témakörben megjelenő rendszerek számát illetően.

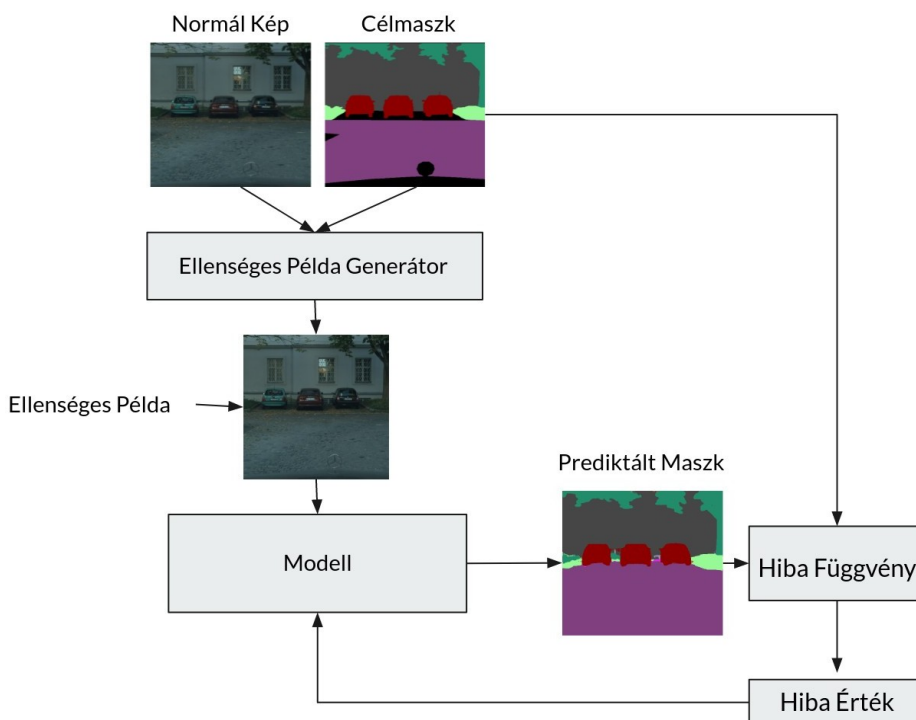
A repository elérhető a következő linken:

<https://github.com/HalmosiL/Segmentation-Segmentation-Adversarial-Tool-Box-AUTO-ATTACK.git>

6. A robusztus tanítás párhuzamosítása

A robusztus neurális hálózatok tanításának legnagyobb időbeli korlátja a tanítási folyamat szerkezetében keresendő. A neurális háló a nem párhuzamosított tanítási folyamat során folyamatosan az ellenséges példák legenerálására vár. Ez alapvetően a tanulás során jelentős időbeli növekedést jelent a normál tanítással szemben.

Standard Robusztus Tanítás

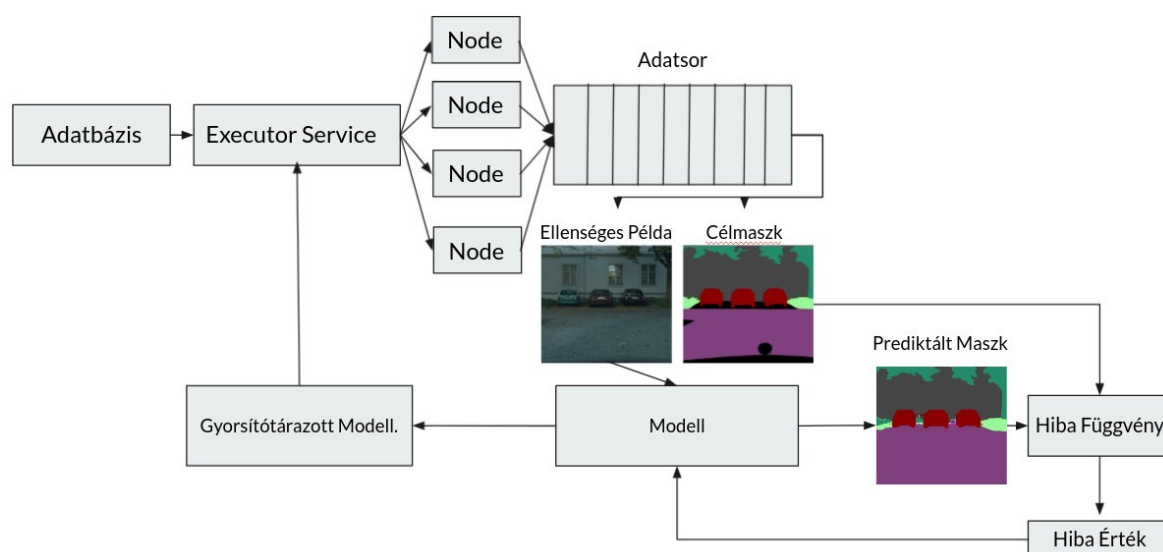


A kutatás során mi az általános megoldással szemben egy párhuzamosított tanító környezetet szeretünk volna létrehozni, melyben az ellenséges példa generálása párhuzamosan történik a háló tanításával. Ennek a megoldásnak a megvalósításához egy erős elméleti feltevésre volt szükségünk, mégpedig, hogy egy tanulás során egy adott iterációnál lévő neurális hálózathoz generált ellenséges példák hasonlóan működni fognak egy pár iterációval későbbi változatánál az adott hálónak. Ez csupán annyit tesz, hogy az ellenséges példák egy tanulás során pár iterációval későbbre is transzferálódnak. Noha ez az állítás nem feltétlen tűnik triviálisnak, az ezzel a módszerrel tanított modelljeink eredményessége igazolja ezt a hipotézist.

A megvalósításunk az előzőektől eltérően tehát egy párhuzamos megoldás, ahol az ellenséges képek generálásához az adott modellt nem használjuk, hanem egy korábbi iterációban lévő elmentett verzióját. Erre a következőkben gyorsítótárazott modellként fogok hivatkozni. Mivel a generáló modell és az éppen tanított modell nem azonos, ezért párhuzamosan futtatható az ellenséges példákon való tanítása és a példák generálása. A szoftveren belül a gyorsítótárazott modellel az ellenséges példákat generáló párhuzamos

processzusokat *node*-oknak nevezzük. Ezek vezérléséért egy úgynevezett *executor service* felel. Ennek célja, hogy egyfajta terheléselosztóként figyelje, hogy az egyes *node*-ok hogyan végeznek a generálással, és új képet osszon ki nekik. A *node*-ok, amint bejezik egy példa generálását egy adatsorba, mentik az így létrehozott ellenséges példát ahonnan a modell folyamatosan tudja olvasni a képeket a tanulás során. A sor idő szerint szerveződik vagyis a később elmentett kép később is kerül ki olvasásra.

Párhuzamosított Robusztus Tanítás



6.1 Implementáció és felhasznált technológiák

A rendszer fejlesztése során arra törekedtünk, hogy igazodjon a neurális hálók tanításához alkalmazott népszerű technológiai megvalósításokhoz. Éppen ezért ez egy Python3.0 alapú rendszer lett amelyben a neurális hálók implemetálásához a Pytorch frameworkot használtuk. Ezzel a részprojekttel egy mindenki számára elérhető eszközt akartunk létrehozni. Éppen ezért ez egy nyílt forráskódú szoftver lett, mely reményeink szerint egy népszerű és sokak által használt rendszer lesz a jövőben.

A repository elérhető a következő linken:

https://github.com/HalmosiL/Semantik_Segmentation_Adversarial_Tranig_CIRA-PDG.git

6.2 Párhuzamosítás Előnyei és Hátrányai

Előny, hogy ez a megvalósítás önmagában lehetővé teszi, hogy megfelelően sok node esetén a tanítás gyakorlatilag a normál tanítás idejével azonos idő alatt fusson le, hiszen a tanító rendszernek csupán be kell olvasnia a képeket a sorból, és ez alapján elvégezni a tanítást. Ez pedig a normál tanítás idejével azonos idő alatt le tud futni. A tanításhoz nyilvánvalóan nagyobb memóriára van szüksége a folyamatnak, hiszen több modellt kell betöltve tartani a memóriában. De ebben a tanítási rendszerben a sebesség kellően nagy rendelkezésre álló memória esetén a normál tanítás idejéig fokozható.

Előnyös tulajdonság továbbá az, hogy a rendszerünk önmagában véve nem egy egységes programként fut, minden egyes node különálló rendszerfolyamatként működik. Ebből adódóan a generálás párhuzamosítása ténylegesen különálló folyamatként futhat, különálló videokártyákon vagy egyéb generáló egységeken. Ezáltal lehetséges a tanításhoz is létrehozni egy több videokártyán működő generáló rendszert.

A rendszernek hátránya, hogy az ellenséges példák generálása nem közvetlenül az adott tanítás alatt lévő modellhez történik, hanem egy pár iterációval korábbi változatához. Ebből adódóan az ellenséges példák hatékonysága a tanított modellenél nem garantált. Bár itt fontos megjegyeznünk, hogy a későbbi fejezetekben tárgyalt eredmények alapján a rendszerrel lehetséges hatékony robusztus modell létrehozása. Ebből adódóan a megközelítés empirikusan igazolható.

7. Kísérletek

Ebben a fejezetben szeretném részletezni a kutatáshoz kötődő kísérleteket, illetve magyarázatot adni arra, hogy miért volt szükséges ezeknek a méréseknek az elvégzésére. Mindemellett ki szeretnék térni arra, hogy a jelenlegi legmagasabb robusztusságú szemantikai szegmentáló háló validálása során mik voltak azok a kardinális nem figyelembe vett kulcskérdések, amelyek megértése elengedhetetlenek a helyes validáló módszertan megalkotásához.

7.1 Hálók robusztuságának vizsgálata

Az első számú kísérlet alapvetően két mérésre osztható ezekre az (7.1.1) és (7.1.2) esetként fogok hivatkozni fejezetszám szerint. Alapvetően az első kísérletben a jelenlegi legnagyobb robusztusságú modellek eredményeit szeretük volna lemérni a saját rendszerünkkel. Ezek mind a *ddc-at* és *bim* algoritmusokkal betanított modellek voltak [11].

7.1.1 PSNET hálók robusztuságának vizsgálata

Az első kísérlet (7.1.1) esetében a *ddc-at* algoritmus repozitóriumban elérhető előtanított PSNET modelleket támadtuk meg [39]. Ezek a modellek mind a Cityscapes adathalmazon tanított modellek voltak. Mindegyik modell rendelkezett előtanított Resnet50-es részhálóval, amit korábban az Image-Net adathalmazon tanítottak elő több osztályos klasszifikációs probléma megoldására. A modelleket két módon tanították. Az első esetben standard robusztus tanítást végeztek el. A standard robusztus tanítás a *bim* algoritmussal való tanítást jelenti. Ebben az esetben minden egyes kép esetén 3 iterációt végeztek el a *bim* algoritmussal. Az ellenséges képeknek az eredeti képhez viszonyított végtelen norma szerinti maximális távolsága $\epsilon = 0.03$ ami gyakorlatilag 7 pixel árnyalatnyi különbséget enged meg pixelenként egy támadásnál. A modelleket 500 epochig tanították, ami a teljes tanító adathalmazon való 500-szori iterációt jelenti. A tanításhoz mindkét esetben SGD optimalizálót használtak dinamikus lépésköz skálázással, ami a gradiens vektor hosszának dinamikus csökkentését jelenti, a kezdeti lépésköz értéke 0.01 volt. A tanító csomag (batch) mérete 16 volt. A második modell az elsővel teljesen azonos módon volt tanítva, annyi különbséggel, hogy a *bim* algoritmus helyett a *ddc-at* algoritmus volt használva a tanításhoz. A *bim* algoritmussal tanított modellre a későbbiekben PSNET-SAT néven fogok hivatkozni a *ddc-at* -vel tanított modellre pedig PSNET-DDCAT néven.

A célunk ebben a kísérletben a PSNET-SAT és a PSNET-DDCAT modellek tesztelése volt az általunk definiált *auto attack* modullal. Mindegyik támadást 120 iterációval futtattuk le, ugyanis igazolni akartuk, hogy valójában ezek a modellek nem robusztusok, csupán gradiens obfuszkációt csinálnak, amit a magasabb iterációs szám leleplezhet. Ennek a hipotézisnek az alapja az a tény hogy a *ddc-at* cikkben használt legerősebb támadás a 6

iterációs *bim* volt, amely még klasszifikációs támadások esetén is egy nagyon gyenge támadásnak és kevés iterációnak tekinthető. Mindegyik támadás esetén a megengedett maximális perturbáció mértéke végtelen norma szerint $\epsilon = 0.03$ ami kb 7 árnyalatnyi különbséget enged meg pixel szinten.

7.1.2 DeepLabV3 hálók robusztuságának vizsgálata

Ebben a kísérletben azt a hipotézist akartuk igazolni miszerint valójában azok a jelenségek amelyeket a 7.1.1 kísérlet mérései alapján megfigyeltünk (ezeket a következő fejezetben részletezem) más típusú neurális hálózatok esetén is jelen vannak. Éppen ezért meg akartuk vizsgálni, hogy a *ddc-at* cikkben említett DEEPLABV3 architektúra hogyan viselkedik a mi *auto attack* modulunkkal szemben.

A kísérlet paraméterei gyakorlatilag megegyeznek a 7.1.1 kísérlet paramétereivel csupán a modellek mások. A DEEPLABV3 modellek ugyanúgy voltak tanítva, mint a korábbi kísérletekben és ezeket szintén a cikkhez tartozó repozitóriumból töltöttük le. A DEEPLABV3 architektúrának itt is volt a *bim* algoritmussal tanított változata, illetve a *ddc-at* algoritmussal tanított változata is. Ezekre az előzőekhez hasonlóan DEEPLABV3-SAT és DEEPLABV3-DDCAT néven fogok hivatkozni.

7.2 Saját hálók robusztussága

A meglévő modellek tesztelésén túlmenően saját modelleket is tanítottunk a kutatás során. A modelleket a saját tanító környezetünkön belül tanítottuk, mindegyik modell PSNET típusú architektúra volt. A kísérlet összes körülménye azonos volt a korábban a hivatkozott cikkben említett beállításokkal, kivéve a következőket. Mindegyik modellhez a *cira+ap* algoritmussal generáltunk ellenséges példákat, a *cira+ap* két lépéses, illetve három lépéses verzióját alkalmaztuk. A két lépéses esetben a modellt az előzőekkel ellentétben 1000 teljes iterációs lépéssel (epoch) tanítottuk. Három lépéses esetben, a modell a korábbiakhoz hasonlóan 500 teljes iterációs lépéssel volt tanítva. Az 500 teljes iterációs, háromlépéses modellre PSNET-CIRA+AP(3)-500 néven fogok hivatkozni, a másik, 1000 teljes iterációs, kétlépéses modellre pedig PSNET-CIRA+AP(2)-1000 néven.

A tanító környezetünkből adódóan, mivel az ellenséges példák generálása és tanítása egyszerre zajlik, gyorsítótáraznunk kellett a modelleket. A mentési ablak mérete azt határozza meg, hogy hány darab iterációs lépés után mentjük el a modellt és osztjuk újra ki a node-oknak. Ez mind a két tanítás során 10 volt, ami annyit jelent, hogy minden tizedik tizenhatos kép csomag után mentjük el a modellt. A hálózat betanítása után mindegyiket validáltuk az *auto attack* modulunkkal. A beállítások azonosak voltak az előző kísérletek beállításával.

8. Eredmények

8.1 PSNET hálók robusztuságának vizsgálata

Alapvetően az első és egyik fő eredményem annak a bizonyítása, hogy a vizsgált cikkben lévő modellek, amelyek a jelenleg a szakirodalomban ismert legrobusztusabb modellek, valójában sokkal kevésbé robusztusak mint amit a cikkben közölt mérések sugallnak. Az eredetileg a cikkben mért robusztussági értékek a *bim* támadás 6 lépéses verziójával szemben a PSNET-SAT modell esetén mIOU érték szerint 25.8 volt. Sikertült igazolni hogy a *bim* támadással is radikálisan körülbelül a felére lehet csökkenteni a robusztuságot mIOU érték szerint 14.66 volt a 120 iterációs esetben. Abszolút értelemben véve ez jelentős különbség és igen nagy eltérés a cikk által becsült robusztusság értékkel szemben. Megjegyzés a táblázatban olvasható adatok a standard mIOU érték százalékos értékei ezért 0-1 intervallum helyett az értékek 0-100 intervallumon értelmezendők, ahol 0 a teljesen pontatlan maszkok esete, a 100 pedig a tökéletesen prediktált maszkoké.

*A modellek eredményei a normál támadás
nélküli esetben, és a 6 és 120 lépéses bim esetén.*

Modellek	Normal	BIM (6 steps)	BIM (120 steps)
PSNET DDCAT	71.70	28.70	14.28
PSNET SAT	69.00	25.80	14.66

Ennek a kísérletnek a legjelentősebb eredményei azonban a második táblázatban olvashatóak. A második táblázatban láthatóak a modellek robusztussági értékei az *adam pgd*, a *cira* és a *cira+ap* támadásokkal szemben. Az eredményekből kiderül hogy a *bim* 120 lépéses értékeihez nagyon hasonló eredményeket lehet elérni az *adam pgd*-vel, bár az eredmények minimális romlását PSNET-DDCAT esetén el lehet vele érni. Az igazán izgalmas és fontos eredmények a *cira* és *cira+ap* esetén voltak. Ugyanis a *cira* képes volt csökkenteni PSNET-DDCAT esetén a mIOU értéket az eredetileg a cikkben említett robusztusság értékhez képest, ami mIOU szerint 28.71 volt, kicsit több, mint harmadára 10.88-ra. A legfontosabb eredmény a táblázat utolsó sorában olvasható a *cira+ap* értékei. Ebben az esetben mindkét modell pontosságát gyakorlatilag nullára sikerült redukálni. Ezáltal sikerült bizonyítani, hogy ezek a modellek egyáltalán nem robusztusak és így megcáfolni a cikk azon állítását, hogy ezekkel a módszerekkel robusztussá lehet tenni a modelleket.

*A modellek eredményei a 120 lépéses
Adam PGD, Cira és Cira+AP algoritmusokkal szemben.*

Modellek	Adam PGD (120 steps)	CIRA (120 steps)	CIRA+AP (120 steps)	MIN
PSNET- DDCAT	12.96	10.89	0.53	0.51
PSNET-SAT	14.83	14.38	0.72	0.68

A táblázatok alaposabb megvizsgálása után egy újabb, a cikkben elhangzó állításra találhatunk cáfolatot. Ugyanis az állítás miszerint a *ddc-at* algoritmus robusztusabbá tenné a

neurális hálókat, mint a *bim*-mel való tanítás, megkérdőjelezhető, mivel a táblázat első két oszlopától eltekintve a PSNET-SAT mindig jobban teljesít, mint a PSNET-DDCAT és a minimális robusztussága is nagyobb.

8.2 Hálók robusztusságának vizsgálata

Az ebben a kísérletben elért eredmények a *bim* támadás esetén nagyon hasonlóak. Szintén látványos eredmény látszik a 6 lépéses *bim* és a 120 lépéses között az iterációs szám növelésével körülbelül a felére csökkenthető az mIOU érték.

A modellek eredményei a normál támadás nélküli esetben, és a 6 és 120 lépéses bim esetén.

Modellek	Normal	BIM (6 steps)	BIM (120 steps)
DeepLab-DDCAT	71.30	29.00	14.19
DeepLab-SAT	69.40	26.20	13.94

A második táblázatban látható, hogy a 120 lépéses *adam pgd* nagyon hasonló eredményt ért el, mint a 120 lépéses *bim*. A *cira* támadás arányosítva és abszolút értelemben is hatékonyabbnak bizonyult DEEPLABV3 modelleknél, mint a PSNET típusú modellek esetén. A *cira+ap* az előző kísérlethez hasonlóan elég hatásos eredményt hozott a DEEPLABV3 modelleknél, mind a kettő esetben gyakorlatilag nullára tudta redukálni a modellek pontosságát ezáltal bebizonyítva, hogy ezek valójában nem robusztusak csupán gradienst obfuszkálnak, és robusztusan csupán gyenge támadások esetén viselkedtek.

*A modellek eredményei a 120 lépéses
Adam PGD, Cira és Cira+AP algoritmusokkal szemben.*

Modellek	PGD (120 steps)	CIRA (120 steps)	CIRA+AP (120 steps)	MIN
DeepLab-DDCAT	12.75	7.49	0.52	0.50
DeepLab-SAT	13.54	11.50	0.34	0.32

Az előző ettől eltérően a DEEPLABV3 modellek robusztusság szempontjából kiegyenlítettebben viselkedtek az eredmények alapján mind a DEEPLABV3-SAT, mind a DEEPLABV3-DDCAT nagyon hasonló mértékben tudott ellenállni a támadásoknak.

8.3 Saját hálók robusztussága

A saját modellek tanításának eredményei felülmúlják az általunk a szakirodalomban talált legjobb eredményeket. Azon túlmenően, hogy robusztusak maradtak a *bim*, *adam pgd* és *cira* algoritmusokkal szemben, emellett még robusztusak lettek a *cira+ap*-vel szemben is.

Modellek	Normal	BIM (120 steps)	Adam PDG (120 steps)
PSNET-CIRA+AP(3)-500	54.88	11.04	10.02
PSNET-CIRA+AP(2)-1000	57.53	6.43	6.27

Az első táblázatban látható eredmények alapján a modellek kevésbé robusztusak a *bim*, illetve az *adam pgd* támadásokkal szemben, mint a *bim*-mel vagy a *ddc-at*-al tanított modellek. Ugyanakkor mégis ellenállónak tekinthetőek a támadásokkal szemben. A normál pontosságuk a modelleknek jelentősen csökkent. Ugyanakkor ez még az alkalmazások

céljából használható pontosságnak tekinthető. Körülbelül az 54.88-as mIOU érték 91%-os pixel szintű pontosságot jelent. A pontosság csökkenés fő oka a megfigyelések alapján a tanított modellek alacsony kapacitása és rövid ideig való tanítása, aminek oka a rendelkezésünkre álló korlátozott számítási kapacitás. Mindemellett a táblázatokból látható, hogy lehetséges átmenetet képezni a normál pontosság és a robusztus pontosság között. A támadás iterációs számának csökkentése esetén a modell normál pontossága nő, robusztus pontossága csökken. Az iterációs szám növelésével az összefüggés fordított.

Modellek	CIRA (120 steps)	CIRA+AP (120 steps)	MIN
PSNET-CIRA+AP(3)-500	25.66	12.27	9.60
PSNET-CIRA+AP(2)-1000	19.41	9.91	5.15

A második táblázat eredményei alapján látható, hogy a modellek robusztusak voltak a *cira* és a *cira+ap* támadással szemben. A minimális robusztusság értékei több, mint egy nagyságrenddel jobbak, mint a *bim*-mel vagy *ddc-at*-al tanított modellek esetén. Ez egy jelentős robusztusság növekedést jelent az eddigi megoldásokhoz képest. Az adatok alapján kijelenthető, hogy az általunk tanított modellek az előzőekkel ellentétben rendelkeznek robusztussággal az összes támadás esetén.

9. Konklúzió

A kutatás során a robusztus szegmentálás minden jelentősebb részterületét vizsgáltuk. A főbb kérdések azonban a neurális hálózatok támadására és a robusztus tanításra fókuszáltak. A kutatás kezdeti hipotézisei alapján sikerült bebizonyítani, hogy az eddigiekben létrehozott legrobusztusabbnak tekintett szemantikai szegmentáló neurális hálózatok valójában nem robusztusak, csupán gradiens obfuszkációt végeznek. Ebből következően sikerült igazolni azt is, hogy az adott cikkben említett megoldásokkal nem lehet robusztus szegmentáló hálót létrehozni. A kutatás során megalkottunk két új, a belső reprezentációt célzó támadást, amelyek hatékonyságát igazoltuk a szemantikai szegmentáló neurális hálók

esetén. Továbbá létrehoztunk egy hatékony és hatásos *auto attack* modult, amely reményeink szerint egy hiánypótló és elterjedt megoldás lesz a robusztus szegmentáló hálók teszteléshez. Betanítottunk kettő robusztus szegmentáló hálót, amik a jelenlegi legmagasabb robusztusságú modelleknek tekinthetők a témán belül. Ezen hálók eredményeit egy hatékony validálási eljárással szemben igazoltuk. Mindemellett kifejlesztettünk egy tanító környezetet, amely a transzferálódás jelenségére alapozva lehetővé teszi az ellenséges példák generálásának párhuzamosítását a tanítással. Ezen megoldás alkalmazhatóságát tényleges modellek tanításával igazoltuk. Összességében kijelenthető, hogy létrehoztunk egy igazolhatóan hatékony módszertani rendszert, amely a robusztus hálók tanításától a validálásig minden feladatrészt lefed. Emellett minden egyes részfeladathoz, kész bárki számára elérhető szoftvereket is tartalmaz. Úgy gondoljuk, hogy a kutatásban bemutatott tanítások és tesztelések összeszedett példákat mutatva, segítenek majd a jövőbeni kutatások felépítésében. Reményeink szerint ezzel a munkával képesek leszünk a módszertani kohézió megteremtésére a robusztus szegmentáláson belül.

10. Köszönetnyilvánítás

Köszönet Jelasity Márknak a témavezetőmnnek a szakmai támogatásért és a kutatásba ölt energiáért. Továbbá köszönet Varga Márknak a szöveg korrigálásáért. A kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében.

11. Hivatkozások

- [1] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014).
- [2] Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018. 99-112.
- [3] Fidel, Gil, Ron Bitton, and Asaf Shabtai. "When explainability meets adversarial learning: Detecting adversarial examples using shap signatures." *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020.
- [4] Montavon, Grégoire, et al. "Layer-wise relevance propagation: an overview." *Explainable AI: interpreting, explaining and visualizing deep learning* (2019): 193-209.
- [5] Croce, Francesco, and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks." *International conference on machine learning*. PMLR, 2020.
- [6] Croce, Francesco, et al. "Robustbench: a standardized adversarial robustness benchmark." *arXiv preprint arXiv:2010.09670* (2020).
- [7] Hendrik Metzen, Jan, et al. "Universal adversarial perturbations against semantic image segmentation." *Proceedings of the IEEE international conference on computer vision*. 2017.
- [8] Treu, Marc, et al. "Fashion-guided adversarial attack on person segmentation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.
- [9] Moeskops, Pim, et al. "Adversarial training and dilated convolutions for brain MRI segmentation." *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, Cham, 2017. 56-64.
- [10] Arnab, Anurag, Ondrej Miksik, and Philip HS Torr. "On the robustness of semantic segmentation models to adversarial attacks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- [11] Xu, Xiaogang, Hengshuang Zhao, and Jiaya Jia. "Dynamic divide-and-conquer adversarial training for robust semantic segmentation." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

- [12] Fezza, Sid Ahmed, et al. "Perceptual evaluation of adversarial attacks for CNN-based image classification." *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2019.
- [13] Liu, Xin, et al. "Dpatch: An adversarial patch attack on object detectors." *arXiv preprint arXiv:1806.02299* (2018).
- [14] Xie, Cihang, et al. "Adversarial examples for semantic segmentation and object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.
- [15] Boloor, Adith, et al. "Simple physical adversarial examples against end-to-end autonomous driving models." *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*. IEEE, 2019.
- [16] Asgari Taghanaki, Saeid, Arkadeep Das, and Ghassan Hamarneh. "Vulnerability analysis of chest X-ray image classification against adversarial attacks." *Understanding and interpreting machine learning in medical image computing applications*. Springer, Cham, 2018. 87-94.
- [17] Garg, Siddhant, and Goutham Ramakrishnan. "Bae: Bert-based adversarial examples for text classification." *arXiv preprint arXiv:2004.01970* (2020).
- [18] Li, Jinfeng, et al. "Textbugger: Generating adversarial text against real-world applications." *arXiv preprint arXiv:1812.05271* (2018).
- [19] Lin, Yen-Chen, et al. "Tactics of adversarial attack on deep reinforcement learning agents." *arXiv preprint arXiv:1703.06748* (2017).
- [20] Xie, Yi, et al. "Real-time, universal, and robust adversarial attacks against speaker recognition systems." *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2020.
- [21] Lin, Chen, et al. "Attacking recommender systems with augmented user profiles." *Proceedings of the 29th ACM international conference on information & knowledge management*. 2020.
- [22] Chen, Hongge, et al. "Robust decision trees against adversarial examples." *International Conference on Machine Learning*. PMLR, 2019.
- [23] Zhou, Yan, et al. "Adversarial support vector machine learning." *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012.
- [24] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).

- [25] Croce, Francesco, and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks." *International conference on machine learning*. PMLR, 2020.
- [26] Croce, Francesco, and Matthias Hein. "Minimally distorted adversarial examples with a fast adaptive boundary attack." *International Conference on Machine Learning*. PMLR, 2020.
- [27] Ilyas, Andrew, et al. "Black-box adversarial attacks with limited queries and information." *International Conference on Machine Learning*. PMLR, 2018.
- [28] Chen, Jinyin, et al. "POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm." *Computers & Security* 85 (2019): 89-106.
- [29] Wang, Ruibin, Yibo Yang, and Dacheng Tao. "ART-Point: Improving Rotation Robustness of Point Cloud Classifiers via Adversarial Rotation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.
- [30] Xie, Cihang, et al. "Improving transferability of adversarial examples with input diversity." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
- [31] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
- [32] Gowal, Sven, et al. "Uncovering the limits of adversarial training against norm-bounded adversarial examples." *arXiv preprint arXiv:2010.03593* (2020).
- [33] Deng, Congyue, and Yi Tian. "Towards Understanding the Trade-off Between Accuracy and Adversarial Robustness."
- [34] Athalye, Anish, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." *International conference on machine learning*. PMLR, 2018.
- [35] Rony, Jérôme, Jean-Christophe Pesquet, and Ismail Ben Ayed. "Proximal Splitting Adversarial Attacks for Semantic Segmentation." *arXiv preprint arXiv:2206.07179* (2022).
- [36] Gu, Jindong, et al. "Adversarial examples on segmentation models can be easy to transfer." *arXiv preprint arXiv:2111.11368* (2021).
- [37] Amari, Shun-ichi. "Backpropagation and stochastic gradient descent method." *Neurocomputing* 5.4-5 (1993): 185-196.

[38] Dreossi, Tommaso, et al. "A formalization of robustness for deep neural networks." *arXiv preprint arXiv:1903.10033* (2019).

[39] <https://github.com/dvlab-research/Robust-Semantic-Segmentation>