Tampereen yliopisto

Joni Pesonen

KONEOPPIMISESSA KOHDATTAVAT PEUKALOIDUT SYÖTTEET

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Tammikuu 2020

TIIVISTELMÄ

Joni Pesonen: Koneoppimisessa kohdattavat peukaloidut syötteet
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Tammikuu 2020

Koneoppimisen saralla on tapahtunut huomattavaa kehitystä, minkä vuoksi sitä on alettu hyödyntää turvallisuuskriittisissäkin ympäristöissä. Viime aikoina on kuitenkin huomattu, että koneoppiminen on altis hyvin suunnitelluille peukaloiduille syötteille. Monesti kyseisiä syötteitä on muokattu niin hienovaraisesti, että ihminen ei näitä muutoksia kykene edes havaitsemaan. Koneoppimisesta löydetyt tietoturvapuutteet ovatkin herättäneet alalla keskustelua, minkä vuoksi peukaloidut syötteet ovat aktiivisen tutkimuksen kohteena.

Tämän tutkielman tarkoituksena on antaa lukijalle hyvät lähtökohdat peukaloitujen syötteiden ymmärtämiseen. Lisäksi esittelen muutamia peukaloitujen syötteiden mahdollistajia ja tutkijoiden suosimia puolustuksia.

Avainsanat: koneoppiminen, peukaloidut syötteet, tietoturva, tietokonenäkö

1 Johdanto

Koneoppimisen saralla on tapahtunut huomattavaa kehitystä, minkä vuoksi sitä on alettu hyödyntää muun muassa kuvan-, puheen- ja haittaohjelmien tunnistuksen tehtävissä. Onnistumisista huolimatta on huomattu, että koneoppiminen on herkkä hyvin suunnitelluille peukaloiduille syötteille [Kurakin *et al.* 2018, Yuan *et al.* 2019]. Monesti kyseisiä syötteitä on muokattu niin hienovaraisesti, että ihminen ei näitä muutoksia kykene edes havaitsemaan, mutta siitä huolimatta koneoppimismalli tekee virheen syötettä tunnistaessaan.

Koneoppimista hyödynnetään tai suunnitellaan hyödynnettäväksi lukuisissa eri fyysisen maailman käyttökohteissa, joihin sisältyvät myös sellaiset ympäristöt, joissa turvallisuus on suurimpia prioriteetteja. Viime aikoina on kuitenkin huomattu [Kurakin *et al.* 2018, Yuan *et al.* 2019], että syötteiden peukalointi toimii myös sellaisia koneoppimismalleja vastaan, jotka toimivat fyysisessä maailmassa ja saavat syötteensä sensoreidensa, kuten kameran tai mikrofonin, kautta, vaikka kyseinen syötetapa ei olekaan yhtä tarkka kuin puhtaasti digitaalinen syöte. Hyökkääjä voi esimerkiksi peukaloida liikennemerkkiä, jolloin itsestään ajava auto luokittelee sen väärin [Kurakin *et al.* 2016], tai peukaloida syötettä siten, että kuvan osiin jakamisesta vastaava neuroverkko jakaa kuvan osiin virheellisesti [Xie *et al.* 2017].

Koneoppiminen perustuu monimutkaisiin tekniikoihin, eikä sen tekemät päätelmät ole ihmiselle helposti ymmärrettävissä, minkä vuoksi sitä on vaikeaa teorisoida. Tästä johtuen kaikkia niitä syitä, minkä vuoksi koneoppiminen on altis peukaloiduille syötteille, ei ole vielä täysin onnistuttu löytämään. Tutkijat ovat kuitenkin löytäneet joitain mahdollistajia, jotka altistavat koneoppimismalleja peukaloiduille syötteille, ja erinäisten puolustusten kehittäminen peukaloituja syötteitä vastaan on tutkimuksen kohteena alalla.

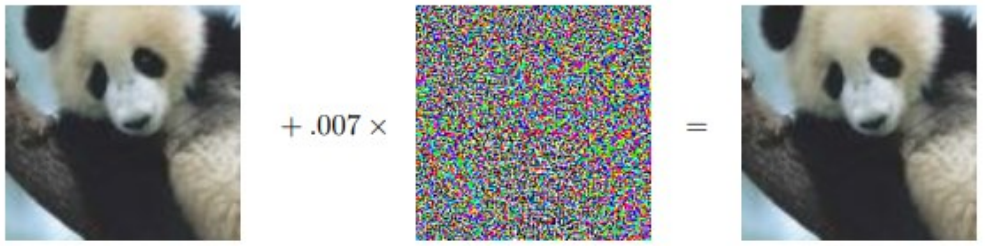
Monet viime aikaisista koneoppimisen saralla tapahtuneista kehitysaskelista ovat perustuneet ohjattuun oppimiseen, ja eritoten tietokonenäköön perustuviin tehtäviin. Sen vuoksi monet peukaloiduista syötteistä on kohdistettu nimenomaan tietokonenäköä hyödyntäviin, luokittelutehtävissä toimiviin malleihin. Vaikka peukaloituja syötteitä kohdistetaan muissakin käyttökohteissa ja tehtävissä toimiviin malleihin, aihealueen laajuuden ja työn suppeuden vuoksi tarkastelen peukaloituja syötteitä lähinnä kuvantunnistuksen näkökulmasta.

Työn tarkoituksena on antaa lukijalle hyvät lähtökohdat peukaloitujen syötteiden ymmärtämiseen luvussa 2 ja tutkia luvussa 3 niitä syitä, jotka mahdollistavat peukaloitujen syötteiden olemassaolon. Lisäksi esittelen luvussa 4 muutamia alan tutkimuksissa esitettyjä ratkaisuja peukaloitujen syötteiden torjumiseksi.

2 Peukaloidut syötteet

Peukaloidut syötteet (adversarial examples) ovat koneoppimismalleille annettavia syötteitä, joita on tarkoituksenmukaisesti muokattu, jotta kyseinen malli antaisi virheellisen tulosteen. Mikäli syötettä ei ole peukaloitu ja se on sellaisenaan normaalisti esiintyvä syöte, on kyseessä *puhdas syöte* (clean example). Mikäli hyökkääjä on muokannut alkuperäisesti puhdasta syötettä tarkoituksenaan saada virheellinen tuloste, on kyseessä peukaloitu syöte. Hyökkäyksellä tarkoitetaan sitä, kun hyökkääjä antaa koneoppimismallille syötteenä peukaloidun syötteen.

Peukaloidut syötteet koostuvat puhtaista syötteistä, joita on muokattu lisäämällä niihin *poikkeutusta* (perturbation). Kuvassa 1 on esitetty esimerkki peukaloidun syötteen muodostamisesta yhdistämällä vasemmalla oleva alkuperäinen puhdas syöte keskellä olevan pienen kertoimen omaavan poikkeutuksen kanssa keskenään. Lopputuloksena saadaan kuvassa oikealla oleva peukaloitu syöte. Poikkeutuksen lisäämisen johdosta malli on vaihtanut luokittelua alkuperäisestä pandasta virheelliseksi gibboniksi.



Kuva 1. Esimerkki peukaloidun syötteen muodostamisesta. [Goodfellow *et al.* 2014]

Poikkeutus eroaa satunnaisesta kohinasta siten, että poikkeutus on tarkoituksenmukaisesti muodostettu. Sen vuoksi poikkeutus saa koneoppimismallin käyttäytymään eri tavalla satunnaiseen kohinaan verrattuna. Poikkeutuksen muodostaminen voidaan kuvata seuraavanlaisena rajoitettuna optimointitehtävänä

$$\begin{aligned} &\min_{x'} \|x' - x\| \\ &\text{s.t. } f(x') = l' \\ &f(x) = l \\ &l \neq l' \\ &x' \in [0, 1] \quad , \end{aligned} \tag{1}$$

jossa x vastaa puhdasta syötettä, x' peukaloitua syötettä, sekä l ja l' vastaavista syötteistä tehtyä luokittelua puhtaan syötteen toimiessa rajoittavana tekijänä ja kuvan pikselien väriarvojen ollessa normalisoituna välille [0, 1] [Yuan *et al.* 2019]. Toisin

sanoen poikkeutuksia muodostaessa yritetään minimoida muutos alkuperäiseen puhtaaseen syötteeseen verrattuna kuitenkin sellaisella tavalla, että poikkeutuksen avulla muodostettu peukaloitu syöte luokitellaan väärin. Poikkeutukset muodostetaan yleensä jonkin algoritmin avulla ratkaisuna kaavan (1) määrittelemään tai siitä johdettuun optimointitehtävään. Muutoksen minimoimisesta johtuen ihminen ei yleensä kykene erottamaan puhdasta syötettä peukaloidusta, mutta koneoppimismalli luokittelee saman peukaloidun syötteen virheellisesti. Kuvantunnistuksen yhteydessä puhtaan syötteen sekä poikkeutuksen yhdistelmää onkin kuvattu ”kuin optiseksi illuusioksi, mutta tietokonenäölle”. On kuitenkin muistettava, että hyökkääjä ei välttämättä onnistu tavoitteissaan, jolloin malli luokittelee peukaloidunkin syötteen oikein. Siksi on tärkeää tarkastella eri mallien tarkkuuksia niiden käsitellessä peukaloituja syötteitä.

Peukaloituja syötteitä voidaan kategorisoida erinäisten ominaisuuksien mukaan esimerkiksi seuraavalla neljällä tavalla:

2.1 Hyökkääjän haluama lopputulos

Hyökkääjän haluama lopputulos viittaa siihen, millä tavalla hyökkääjä haluaa saada koneoppimismallin luokittelemaan sille annetun syötteen väärin. Tämä kategoria voidaan jakaa edelleen kahteen osaan: *kohdistamattomiin* (non-targeted attack) ja *kohdistettuihin* (targeted attack) hyökkäyksiin. Kohdistamattomassa hyökkäyksessä ei ole väliä, miksi luokaksi malli luokittelee syötteen, kunhan se on virheellinen, kun taas kohdistetussa hyökkäyksessä hyökkääjän tavoitteena on saada koneoppimismalli luokittelemaan annettu syöte joksikin tietyksi virheelliseksi luokaksi. [Kurakin *et al.* 2018, Yuan *et al.* 2019]

Kohdistamattomat hyökkäykset ovat helpompia toteuttaa siinä mielessä, että niissä on enemmän mahdollisuuksia onnistua suuremmasta kohdeluokkien lukumäärästä johtuen. Kohdistettuja hyökkäyksiä hyödynnetään, kun halutaan, että koneoppimismalli tunnistaa syötteen joksikin tietyksi virheelliseksi luokaksi, esimerkiksi kasvojentunnistuksen yhteydessä. [Kurakin *et al.* 2018, Yuan *et al.* 2019]

2.2 Hyökkääjän tietämys kohdemallista

Hyökkääjän tietämys kohdemallista viittaa siihen, mitä tietoja kohdemallista hyökkääjällä on. Tällaisia tietoja ovat muun muassa mallin arkkitehtuuri, mallin opetusvaiheessa käytetty data, mallin parametrit sekä mallissa käytetyt painokertoimet. *Lasilaatikolla* (white box) tarkoitetaan sitä, että hyökkääjä tietää kaiken hyökkäyksen kohteena olevasta mallista. *Mustalla laatikolla* (black box) tarkoitetaan sitä, että hyökkääjällä on hyvin vähän tai ei ollenkaan tietoa hyökkäyksen kohteena olevasta mallista. Usein hyökkääjä voi kuitenkin antaa mallille syötteitä ja tarkkailla mallin antamia tulosteita parantaen hyökkäyksiään saatujen tulosteiden perusteella. Hyökkääjä saattaa esimerkiksi saada syötteelleensä tulosteena joko jokaisen mahdollisen luokan toden-

näköisyyden erikseen tai vain todennäköisimmäksi valitun luokan nimen. Mikäli hyökkääjä ei voi tarkkailla syötteistään saamiaan tulosteita, hänen on muodostettava sellaisia peukaloituja syötteitä, jotka onnistuvat huijaamaan useimpia koneoppimismalleja. [Kurakin *et al.* 2018, Yuan *et al.* 2019]

Useimpiin verkossa oleviin koneoppimismalleihin pätee mustien laatikoiden tapaukset. Vaikka useimmat peukaloidut syötteet kehitetäänkin lasilaatikkotapauksissa, voidaan niitä monesti hyödyntää myös mustien laatikoiden tapauksissa *siirrettävyyden* (transferability) johdosta. Siirrettävyyden vuoksi tietyssä tehtävässä toimivaa mallia vastaan kehitetty peukaloitu syöte onnistuu ainakin kohtuullisella todennäköisyydellä hyökkäyksessä myös toista samassa tehtävässä toimivaa mallia vastaan. [Yuan *et al.* 2019, Izmailov *et al.* 2018]

2.3 Syötetapa

Syötetapa voidaan jakaa *digitaalisiin* (digital attack) ja *fyysisiin* (physical attack) hyökkäyksiin. Digitaalisessa hyökkäyksessä hyökkääjä pääsee suoraan syöttämään malliin menevän datan. Tällaisia tapauksia ovat esimerkiksi hyökkääjän peukaloima kuva, jonka hän pääsee suoraan syöttämään digitaalisessa muodossa koneoppimismallille luokiteltavaksi. Fyysisen hyökkäyksen tapauksessa hyökkääjä ei pääse suoraan syöttämään malliin menevää digitaalista versiota peukaloidusta syötteestään. Sen sijaan malliin menevä syöte saadaan esimerkiksi sensorien, kuten kameran tai mikrofonin, kautta. Tämän vuoksi fyysisessä hyökkäyksessä hyökkääjä ei pysty muotoilemaan syötteitään yhtä tarkasti kuin digitaalisessa hyökkäyksessä, sillä samasta fyysisestä syötteestä muodostettu digitaalinen versio saattaa hieman vaihdella esimerkiksi kameran kulman tai valaistuksen seurauksena. [Kurakin *et al.* 2018, Yuan *et al.* 2019]

2.4 Hyökkäysten lukumäärä

Hyökkäysten lukumäärä voidaan jakaa *kertaluonteisiin* (one-time attack) ja *iteratiivisiin* (iterative attack) hyökkäyksiin. Kertaluonteisessa hyökkäyksessä koneoppimismalliin syötetään peukaloitu syöte, joka on jo valmiiksi optimoitu. Peukaloitu syöte syötetään vain kerran, eikä sitä päivitetä enää syöttämisen jälkeen. Iteratiivisessa hyökkäyksessä koneoppimismalliin syötetään peukaloitu syöte, jonka jälkeen kyseistä peukaloitua syötettä päivitetään saatujen tulosteiden perusteella. Tätä toistetaan, kunnes saavutetaan haluttu lopputulos. [Yuan *et al.* 2019]

Kertaluonteisiin hyökkäyksiin verrattuna iteratiivisella hyökkäyksellä saadaan yleensä parempia peukaloituja syötteitä aikaiseksi, mutta iteratiiviset hyökkäykset vaativat enemmän laskenta-aikaa ja enemmän vuorovaikutusta kohdemallin kanssa, mikä ei ole aina mahdollista oikean elämän tilanteissa. Iteratiivisen hyökkäyksen vaatima laskenta-aika saattaakin nousta liian suureksi, jolloin kertaluonteinen hyökkäys voi olla ainoa varteenotettava vaihtoehto. [Yuan *et al.* 2019]

3 Peukaloitujen syötteiden toimintaperiaatteet

Peukaloitujen syötteiden olemassaolon syitä ei vielä täysin tunneta, eivätkä alan tutkijat ole tällä hetkellä yhtä mieltä eri tutkimuksissa esitetyistä teorioista. [Akhtar ja Mian 2019, Zhang ja Li 2019, Yuan *et al.* 2019] Tästä johtuen sen sijaan, että keskittyisin eri tutkimusten esittämiin syihin, käsittelen hyvin yleisellä tasolla niitä asioita, joiden tiedetään liittyvän peukaloitujen syötteiden olemassaoloon. Lisäksi esittelen muutamia peukaloitujen syötteiden mahdollistajia, jotka tulee ottaa huomioon mallia kehitettäessä, sillä hyökkääjät voivat hyväksikäyttää näitä mahdollistajia hyökkäyksiä kehittäessään.

Peukaloitujen syötteiden olemassaolo perustuu yhden koneoppimisen perusperiaatteen rikkomiseen; olettamukseen siitä, että testausvaiheessa käytettävä testausaineisto on peräisin samanlaisesta tilastollisesta jakaumasta kuin opetusvaiheessakin käytetty opetusaineisto [Izmailov *et al.* 2018, Cao *et al.* 2018]. Toisin sanoen opetusaineiston tulee olla tarpeeksi kattava ja monipuolinen otos kaikista mahdollisista vastaantulevista syötteistä. Peukaloituja syötteitä muodostetaankin luomalla tahallisia jakauman muutoksia näiden kahden aineiston välille [Izmailov *et al.* 2018].

Ohjatussa oppimisessa koneoppimismalli opetetaan tehtäväänsä antamalla sille opetusvaiheessa erittäin suuri määrä eri syötteitä opetusaineistona. Siihen on kullekin syötteelle merkitty myös se tulos, eli tässä tapauksessa se luokka, joka kyseisestä syötteestä tulisi seurata. Jokaisen aineistossa olevan syötteen jälkeen malli päivittyy, ja tämän prosessin iteratiivisen luonteen vuoksi mallin sisäiset riippuvuussuhteet vahvistuvat tai heikentyvät vähitellen. Tästä johtuen mitä suurempi ja monipuolisempi opetusaineisto on, sitä tarkempi koneoppimismallista yleensä tulee. [McDaniel *et al.* 2016] Tämä poikkeaa sääntöpohjaisista lähestymistavoista yleistämiskyvyllään; hyvä malli kykenee luokittelemaan opetussyötteisiin kuulumattomatkin syötteet oikein.

Mallin opettamisessa on kuitenkin varottava yli- ja alisovittamista. Ylisovitettu malli mukailee liiankin tarkasti opetusaineistoa; se saattaa mallintaa opetusaineiston erityisen hyvin, mutta yleistää huonosti uusiin ja ennalta näkemättömiin testaussyötteisiin. Ylisovittaminen johtuu siitä, että malli keskittyy liikaa opetusaineiston yksityiskohtiin, mikä heikentää mallin yleistämiskykyä. Tällöin malli voi ottaa esimerkiksi aineistossa esiintyvän kohinan huomioon mallinnettavaa ilmiötä selittävänä tekijänä. Alisovittaminen on ylisovittamisen vastakohta; alisovitettu malli ei keskity tarpeeksi opetusaineiston yksityiskohtiin, jolloin se ei selitä mallinnettavaa ilmiötä riittävän hyvin. Onkin mallin kehittäjien vastuulla pyrkiä löytämään sopiva tasapaino näiden kahden vastakohdan välillä, jotta mallissa esiintyisi mahdollisimman vähän yli- ja alisovittamista. [Czajkowski ja Kretowski 2019]

Mikäli opetusvaiheessa käytetty aineisto ei sisällä kaikkia mahdollisia syötteitä ja niistä seuraavaa tulosta, mallilla ei voi täydellisesti kuvata kuvattavaa ilmiötä. Koska syöte koostuu useasta eri *piirteestä* (feature), jotka voidaan kustakin syötteestä

havainnoida, ei opetusaineisto käytännössä katsoen missään tilanteessa pysty kattamaan jokaista mahdollista syötteiden sisältämien piirteiden yhdistelmää niiden epäkäytännöllisen suuren lukumäärän vuoksi. Tästä johtuen malli on aina approksimaatio kuvattavasta ilmiöstä. Tämä ei yleensä ole ongelma ympäristössä, jossa peukaloituja syötteitä ei esiinny. Silloin tarpeeksi kattava ja odotetusta jakaumasta peräisin oleva opetusaineisto riittää mallin opettamiseksi, sillä testausvaiheessa vastaantulevat syötteet ovat riittävän samankaltaisia opetusaineistoon verrattuna, jotta malli pystyy niiden perusteella päättelemään, millaisia tuloksia uudenlaisilla ja ennalta näkemättömillä syötteillä tulisi saada. [McDaniel *et al.* 2016]

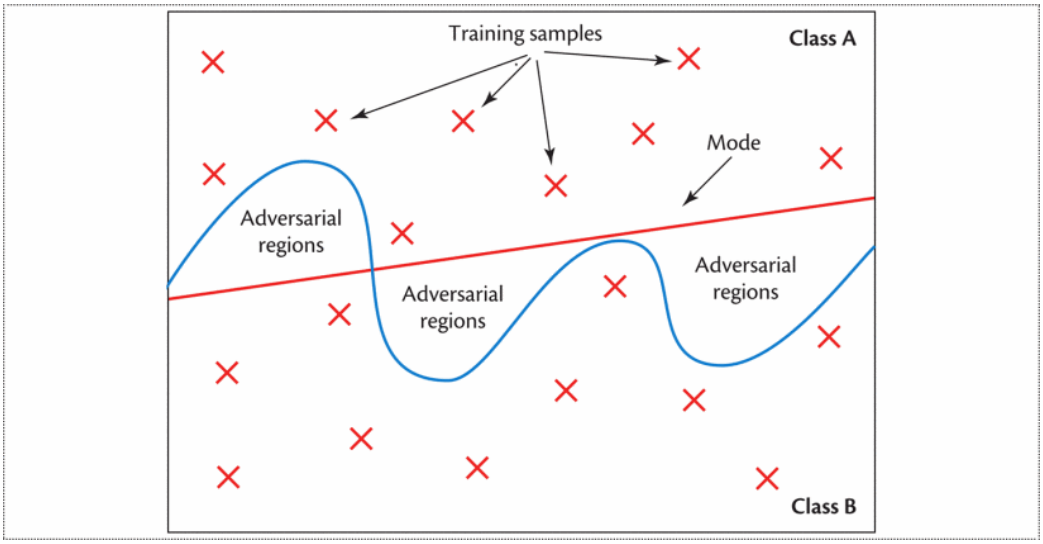
Ongelma syntyy, kun hyökkääjät antavat syötteitä, jotka eivät ole peräisin odotetusta jakaumasta. He etsivät niitä kohtia, joissa malli on hieman epätarkka approksimoivan luonteensa vuoksi. Usein etsiminen tapahtuu jonkin algoritmin avulla, mutta hyökkääjä voi hyödyntää myös yksinkertaista yritys ja erehdys -periaatetta muokkaamalla hieman jotain syötteen piirrettä, tarkkailemalla muutoksen vaikutuksia mallin tekemässä luokittelussa, ja toistamalla tätä niin kauan kunnes saavutetaan haluttu lopputulos tai siirrytään seuraavan syötteen peukalointiin edellisen epäonnistuttua. Yritys ja erehdys -periaatteen hyödyntäminen on kuitenkin harvinaisempaa sen työläyden ja tehottomuuden vuoksi. [McDaniel *et al.* 2016]

Kuvassa 2 kuvataan mallin opetusta ja käyttöä. Kaksi luokkaa, A ja B, ovat syöteavaruuden alueita, ja ne on erotettu toisistaan sinisellä käyrällä. Kaikki sinisen käyrän yläpuolella olevat syötteet kuuluvat luokkaan A, kun taas alapuolella olevat kuuluvat luokkaan B. Tätä käyrää kutsutaan mallin *oikeaksi päätösrajaksi* (real decision boundary). Malli on opetettu syötteillä, jotka on kuvattu merkillä X. Näiden opetussyötteiden perusteella opetuksessa käytetty algoritmi on approksimoinut luokat erilleen punaisella viivalla, *mallin päätösrajalla* (model decision boundary). Oikean ja mallin päätösrajan välisiä alueita kutsutaan *mallin virheeksi* (model error) tai *peukaloitujen syötteiden alueiksi* (adversarial regions). [McDaniel *et al.* 2016]

On huomattava, että esimerkissä kyseessä on pätevä ja tarkka malli, jossa jokainen syöte on luokiteltu oikein. Virhettä esiintyy luonnollisesti oikean päätösrajan approksimoinnin vuoksi. Oikea päätösraja myös muuttuu sitä monimutkaisemmaksi, mitä monipuolisempi kuvattava ilmiö on, ja mitä laajemmaksi ilmiön piirre- ja dimensioavaruus kasvaa. [McDaniel *et al.* 2016]

Toisin sanoen hyökkääjä yrittää poikkeutuksen avulla ”liikuttaa” alkuperäisesti puhtaan syötteen mallin ”harmaille alueille”, joissa malli on hieman epätarkka. Hyökkääjä voi myös hyödyntää mahdollista tietoaan kohdemallista peukaloitujen syötteiden kehittämisessä, ja tällaisissa lasilaatikkotapauksissa hyökkääjä kykeneekin parhaiten kehittämään peukaloituja syötteitä.

Mallin kehityksessä tuleekin hyödyntää *piirrevalintaa* (feature selection). Luokitte-



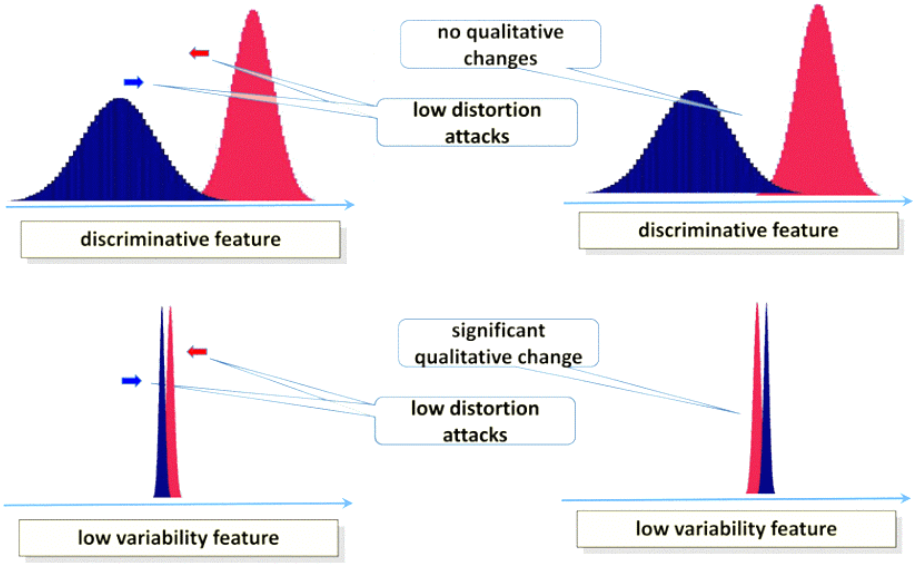
Kuva 2. Mallin opetus ja käyttö. [McDaniel *et al.* 2016]

lussa tarkastellaan samanaikaisesta montaa eri piirrettä, ja yhden syötteen koostuessa useasta eri piirteen yhdistelmästä on erittäin tärkeää yrittää valita juuri ne piirteet, jotka antavat eniten informaatiota tunnistettavasta kohteesta. Informaatiolla voidaan ilmaista, kuinka paljon jokin piirre auttaa luokittelun tekemisessä. Mikäli ei tarkastella tarpeeksi montaa eri piirrettä, oikean luokittelun tekeminen on vaikeaa tai mahdotonta vajavaisen informaation vuoksi. Mikäli tarkastellaan liian montaa eri piirrettä voivat hyökkääjät hyväksikäyttää mahdollisia pienen informaation omaavia piirteitä.

Pienen vaihteluvälin piirteet (low variability features) ovat piirteitä, jotka eivät muutu paljoa syötteestä riippumatta. Täten pienikin muutos tällaisten piirteiden arvoihin voi vaikuttaa suurestikin mallin tekemään luokitteluun, sillä tällaisissa tapauksissa arvon muutos on suhteellisesti suuri muissa syötteissä tavattuihin kyseisen piirteen arvoihin verrattuna. Esimerkiksi kirjoitettaessa vapaasti käsin 8x8-ruudukkoon jokin numero, ovat kaikissa nurkissa olevat ruudut usein tyhjiä ja täten hyvin samanlaisia kirjoitetusta numerosta riippumatta. Tästä ei juurikaan ole haittaa ympäristössä, jossa peukalointia ei esiinny, sillä niissä ympäristöissä tällaiset piirteet käyttäytyvät liki vakionomaisesti. Siitä huolimatta ne saattavat omata nollasta poikkeavia painokertoimia niiden hyvin vähäisestä informaatiosta huolimatta, jolloin hyökkääjä voi nopeastikin muuttaa mallin tekemää luokittelua tällaisia pienen vaihteluvälin piirteitä hyväksikäyttämällä. [Izmailov *et al*. 2018]

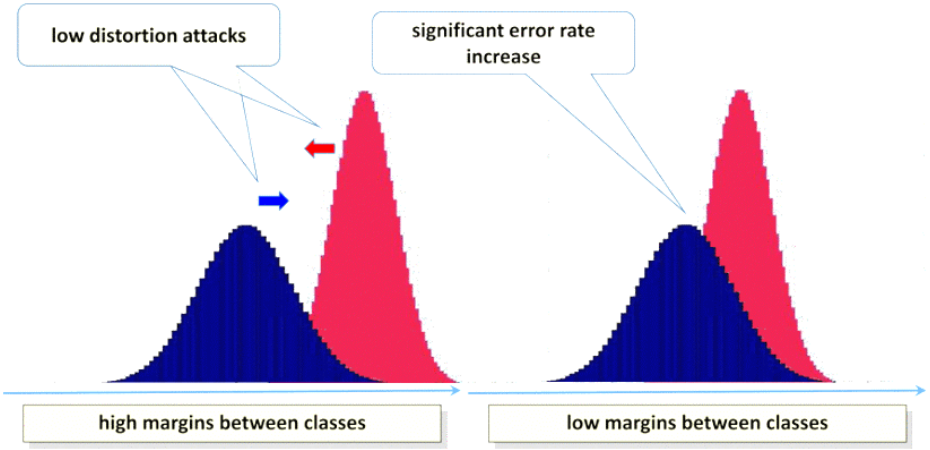
Piirteet ovat ani harvoin yksilöiviä, mistä johtuen yksi piirre voi esiintyä useassakin eri luokan ilmentymässä. Täten kukin piirre on jakautunut omalla tavallaan kaikkien niiden eri luokkien välille, joissa se esiintyy. Kuvassa 3 vertaillaan kuvan ylälaidassa olevaa selkeästi erottuvaa piirrettä alalaidassa olevaan pienen vaihteluvälin piirteeseen kahden luokan tapauksessa. Kuvan vasemmassa laidassa on piirteiden luokkien jakaumat puhtailla syötteillä, kun taas oikeassa laidassa on samojen piirteiden luokkien jakaumat peukaloiduilla syötteillä. Sininen jakauma kuvaa luokkaa A ja punainen

luokkaa B. Kuvasta ilmenee, miten poikkeutuksilla luotu pieni jakauman muutos saa aikaan suuren eron pienen vaihteluvälin piirteen luokittelussa. [Izmailov *et al.* 2018]



Kuva 3. Selkeästi erottuva piirre ja pienen vaihteluvälin piirre. [Izmailov *et al.* 2018]

Sellaisia piirteitä, joiden luokkien väliset jakaumat ovat osittain päällekkäin, kutsutaan *matalan keskinäisinformaation piirteiksi* (low mutual information feature). Matalan keskinäisinformaation piirteet altistavat mallia peukaloiduille syötteille, sillä lisäämällä poikkeutusta kyseisiin piirteisiin näitä jakaumia on mahdollista siirtää entistä enemmän toistensa päälle kuvan 4 mukaisesti. Tästä johtuen piirteen eri luokat erottuvat entistä vähemmän toisistaan, mikä vaikeuttaa syötteen oikein luokittelemista piirteestä saatavan informaation vähentyessä entisestään. Mikäli nämä piirteen eri luokkien jakaumat ovat matalan keskinäisinformaation lisäksi pienen vaihteluvälin omaavia, mahdollistaa se tällaisten piirteiden hyväksikäyttöä entisestään. [Izmailov *et al.* 2018]



Kuva 4. Matalan keskinäisinformaation piirre ja sen kahden esiintymisluokan jakaumat puhtailla syötteillä ja peukaloiduilla syötteillä. [Izmailov *et al.* 2018]

4 Puolustukset

Luvussa 3 käsittelin muutamia peukaloitujen syötteiden mahdollistajia, joita olivat yli- ja alisovittaminen, ja erinäiset piirteiden jakaumien ominaisuudet, kuten piirteen yhden tai useamman esiintymisluokan pieni vaihteluväli, ja piirteen esiintymisluokkien jakaumien läheisyys. Onkin mallin kehittäjien vastuulla suorittaa mahdollisimman onnistunut piirrevalinta ja pyrkiä minimoimaan mallin yli- ja alisovittaminen, sillä ne ovat asioita, joita hyökkääjät voivat hyödyntää hyökkäyksiä kehittäessään.

Koska peukaloitujen syötteiden olemassaolon syistä ei olla vielä täysin varmoja, tutkijoilla ei tällä hetkellä ole ratkaisua asian korjaamiseksi. Tästä johtuen tutkijat ovat alkaneet kehittää puolustuksia peukaloitujen syötteiden torjumiseksi. Ideaalisesti puolustukset olisivat toimintavarmoja kaikissa tilanteissa, mutta koneoppimista on vaikea teorisoida siinä käytettävien monimutkaisten tekniikoiden vuoksi, eikä kaikissa tilanteissa toimivaa puolustusta ole vielä onnistuttu löytämään. Tutkimuksissa esitetyt puolustukset eivät myöskään yleisty kovin hyvin tällä hetkellä, vaan ne toimivat vain tietynlaista hyökkäystä vastaan [Zhang ja Li 2019]. Tästä johtuen hyökkäysten ja puolustusten kehittäminen on tällä hetkellä kilpavarustelua: kun hyökkäykseen kehitetään vahva puolustus torjumaan sitä, hyökkääjät kiertävät kyseisen puolustuksen uudenlaisella hyökkäyksellä, jota puolustaja ei osannut odottaa [Yuan *et al.* 2019].

Puolustukset voidaan jakaa esimerkiksi kolmeen eri kategoriaan: opetusvaiheen tai syötteiden muuttamiseen, mallin muuttamiseen, ja mallin lisäosiin. Opetusvaiheen tai syötteiden muuttamisessa joko koneoppimismallin opetusvaihetta muutetaan tai mallin testausvaiheessa saamia syötteitä muutetaan ennen niiden käsittelyä. Esimerkiksi tiedonpakkaamiseen perustuvat puolustukset, joissa syöte pakataan ennen sen käsittelyä, ovat syötteen muuttamiseen perustuvia puolustuksia. Mallin muuttamisessa muutetaan itse mallia ja sen rakennetta, kun taas mallin lisäosia hyödyntävät puolustukset eivät muuta itse kohdemallia, vaan lisäävät kohdemallin avuksi erillisiä ulkoisia malleja. Kategoriasta riippumatta kaikille puolustuksille yhteistä on se, että niissä pyritään minimoimaan muutokset mallin arkkitehtuuriin sekä säilyttämään mallin nopeus ja tarkkuus. Toisin sanoen puolustuksen lisäämisen ei pitäisi heikentää mallin suorituskykyä, esiintyipä ympäristössä peukalointia tai ei. [Akhtar ja Mian 2019, Zhang ja Li 2019] Eri tutkimuksissa ehdotettujen puolustusten runsaan lukumäärän vuoksi esittelen yleisellä tasolla tarkastelluista tutkimuksista jokaisen kategorian yleisimmin mainitun puolustuksen.

4.1 Opetusvaiheen tai syötteiden muuttaminen

Peukaloinnin opettamisessa (adversarial (re)training) opetusaineistoon lisätään peukaloituja syötteitä, ja niitä luodaan lisää sitä mukaa, kun opetus etenee. Peukaloinnin opettamisen esitti ensimmäisenä Goodfellow *et al.* [2014], ja sen tavoitteena on saada malli luokittelemaan sille annettu syöte oikein, vaikka sitä olisikin peukaloitu. Koska

ihminen ei usein kykene erottamaan puhdasta syötettä peukaloidusta, on puolustajan itse kehitettävä omat peukaloidut syötteensä. Opetusaineistoon on mahdollista sisällyttää useampaakin eri mallia vastaan kehitettyjä peukaloituja syötteitä, mikä tarjoaa monipuolisempia peukaloituja syötteitä ja parantaa mallin yleistämiskykyä *yleisrasitteen* (overhead) nousemisen kustannuksella. [Zhang ja Li 2019] Yleisrasitteella viitataan kaikkiin niihin tehtävässä käytettyihin ylimääräisiin resursseihin, joita tehtävän suorittaminen ei itsessään vaatisi. Tällaisia resursseja ovat esimerkiksi laskenta-aika ja muistinkäyttö.

Peukaloinnin opettaminen on verrattain yksinkertaista ja voi huomattavastikin parantaa mallin robustisuutta etenkin kertaluonteisia ja lasilaatikkohyökkäyksiä vastaan, vaikka malli jääkin haavoittuvaiseksi iteratiivisia ja mustan laatikon hyökkäyksiä vastaan [Akhtar ja Mian 2019, Zhang ja Li 2019]. Robustisuudella tarkoitetaan sitä, että malli ei ole herkkä poikkeaville havaintoarvoille, eli tässä tapauksessa peukaloiduille syötteille. Mikäli mallissa tapahtuu ylisovittamista, peukaloinnin opetus myös regularisoi mallia vähentäen tätä ylisovittamista, mikä puolestaan parantaa yleisellä tasolla mallin robustisuutta peukalointia vastaan. Mallilla saa parhaat tulokset, kun peukaloitujen syötteiden lukumäärä opetusvaiheessa on yhtä suuri kuin puhtaiden syötteiden lukumäärä. [Akhtar ja Mian 2019, Zhang ja Li 2019] Tästä johtuen puolustuksen yksinkertaisuudesta huolimatta peukaloinnin opettaminen lisää suurta yleisrasitetta mallin opetukseen, sillä peukaloitujen syötteiden kehittämiseen menee omat resurssinsa. Vaikka peukaloinnin opettaminen parantaakin mallin robustisuutta, se ei ole mukautuva puolustus. Toisin sanoen se pitää aina kohdistaa jotain tiettyä hyökkäystä vastaan. Vaikka opetuksessa on mahdollista hyödyntää useampaakin eri mallia vastaan kehitettyjä peukaloituja syötteitä ja näin lisätä mallin robustisuutta mustan laatikon hyökkäyksiä vastaan, on vaikeaa teorisoida, mitä nimenomaisia hyökkäyksiä vastaan kehitetyt peukaloidut syötteet ovat parhaita mallin robustisuuden lisäämiseksi. [Akhtar ja Mian 2019, Zhang ja Li 2019]

4.2 Mallin muuttaminen

Mallin tislauksessa (defensive distillation) luodaan pienempi uusi malli jäljittelemään aiemmin luotua isompaa ja laskennallisesti intensiivisempää mallia ilman, että tämän uuden mallin tarkkuus kärsii merkittävällä tasolla. Mallin tislauksen puolustustarkoituksessa esitti ensimmäisenä Papernot *et al.* [2016], ja sen tavoitteena on vähentää mallin herkkyyttä syötteen poikkeutuksille tasoittamalla mallia ja sen luokittimia opetusvaiheen aikana, mikä parantaa mallin sietokykyä peukaloituja syötteitä vastaan. Mallin tislauksessa siirretään isomman ja monimutkaisemman mallin jo oppima tieto pienempään ja yksinkertaisempaan malliin. Tämä toimii käyttämällä isoa mallia luokittelemaan pienemmän mallin opetusaineisto, joka on sama molemmilla malleilla. Pienemmän mallin opetusaineisto poikkeaa normaalista kuitenkin siten, että kutakin syötettä

vastaava tulos onkin isomman mallin antamat eri luokkien todennäköisyydet kyseiselle syötteelle. Hyödyntämällä näitä kutakin syötettä vastaavien luokkien todennäköisyyksiä saadaan pienemmälle mallille välitettyä enemmän tietoa verrattuna pelkän yhden oikean luokan käyttämiseen. Näin isomman mallin jo oppima tieto siirtyy pienemmälle mallille, joka yhdistää vanhan tiedon opetussyötteistä itse oppimaansa tietoon. [Papernot *et al.* 2016, Zhang ja Li 2019] Tislausprosessia voidaan myös toistaa monta kertaa ja näin yhdistää useiden eri mallien oppima tieto toisiinsa [Yuan *et al.* 2019].

Uudella tislatulla mallilla on pienempi koko, parempi robustisuus, ja pienempi laskennallinen vaativuus [Yuan *et al.* 2019]. Mallin tislaus parantaa sen yleistämiskykyä, eikä se lisää opetus- tai testivaiheeseen merkittävää yleisrasitetta. Vaikka mallin tislauksen toteuttaminen on monimutkaista ja hyvin mallista riippuvaista, se parantaa mallin sietokykyä pieniä poikkeutuksia vastaan. [Akhtar ja Mian 2019, Zhang ja Li 2019]

4.3 Mallin lisäosat

Peukaloinnin tunnistamisessa (adversarial detecting) pyritään tunnistamaan, onko mallille annettu syöte puhdas vai peukaloitu. Peukaloinnin tunnistaminen voidaan myös sen toteutustavasta riippuen sisällyttää mallin muuttamiseen, mutta MagNet on Mengin ja Chenin [2017] esittämä mallin lisäosa, joka koostuu yhdestä tai useammasta tunnistinmallista sekä yhdestä syötteen uudelleenrakentamiseen käytettävästä mallista. Opetusvaiheessa MagNet pyrkii oppimaan puhtaiden syötteiden *moniston* (manifold), joka koostuu kunkin opetussyötteen lokaalista ympäristöstä. Testausvaiheessa tästä monistosta kaukana olevat syötteet tunnistetaan peukaloiduiksi ja hylätään, koska tällöin kyseessä on hyvin poikkeuksellinen syöte, jota ei pitäisi tulla vastaan kyseisessä tehtävässä. Syötteet, jotka ovat lähellä monistoa, mutta eivät tarkalleen kuulu siihen, uudelleenrakennetaan siten, että ne kuuluvat monistoon, ja kohdemallin luokittimelle välitetään vain tämä uudelleenrakennettu syöte. Tällä pyritään poistamaan peukalointi syötteestä, jotta kohdemalli voi hoitaa luokittelun normaalisti. [Meng ja Chen 2017, Akhtar ja Mian 2019]

Yleisesti ottaen peukaloinnin tunnistimien ei usein tarvitse muuttaa itse kohdemallia, joten ne ovat verrattain yksinkertaisia. Kohdemallin suorituskyky riippuu kuitenkin suuresti tunnistamiseen valitusta toteutustavasta, sillä tutkimuksissa on esitetty lukuisia toisistaan hyvinkin paljon eroavia toteutustapoja. On myös huomattava, että peukaloitujen syötteiden tunnistamisessa keskitytään vain tunnistamaan vastaantulevat peukaloidut syötteet eikä luokittelemaan niitä oikein. Syötteen varsinainen luokittelu jätetään aina kohdemallille, johon tunnistin on liitetty, mistä johtuen pelkkä peukaloitujen syötteiden tunnistaminen ei paranna itse kohdemallin robustisuutta. [Zhang ja Li 2019] Esimerkiksi itseajavia autoja ajatellen on tärkeää, että yhtään ympäristöstä saatavaa syötettä ei jätetä luokittelematta pelkästään sen vuoksi, että sitä on peukaloitu.

5 Yhteenveto ja pohdinnat

Tässä työssä keskityin peukaloitujen syötteiden perusteisiin ja kuvailin eri tapoja kategorisoida niitä esimerkkitilanteiden kera. Lisäksi esittelin suppeasti koneoppimisen ja peukaloitujen syötteiden teoriaa, ja muutamia tutkimuksissa esitettyjä tapoja puolustautua niiltä.

Koneoppiminen ja sitä kautta myös tietokonenäkö ovat ja tulevat olemaan yhä kasvavissa määrin tärkeä osa yhteiskuntaa, minkä vuoksi niistä löydetyt tietoturvapuutteet ovat herättäneet keskustelua. On kuitenkin muistettava, että koneoppiminen on suuri ja tärkeä kehitysaskel, jonka avulla olemme jo nyt saaneet huomattavaa hyötyä esimerkiksi lääketieteen, tietokonenäön ja robotiikan osa-alueilla. Koneoppimisen käyttökohteet ovatkin huomattavasti monipuolisemmat ja sopeutuvammat kuin aiemmat, puhtaasti eri sääntöihin pohjautuneet lähestymistavat. Tämän vuoksi on tärkeämpää keskittyä korjaamaan vikoja, kuin kokonaan hylätä koneoppiminen konseptina.

Peukaloidut syötteet ja niiltä puolustautuminen ovatkin erittäin aktiivisen tutkimuksen kohteena alalla tällä hetkellä. Koska peukaloitujen syötteiden toimintaperiaatteita ei vielä täysin tunneta, niiden löytäminen on tärkeänä jatkotutkimuksen aiheena. Toimintaperiaatteiden täsmällisten syiden löydyttyä puolustuksista saa helposti vahvempia ja niiden kehittäminen on huomattavasti vaivattomampaa. Tutkijat toivovat, että se myös mahdollistaisi lopullisen ratkaisun löytämisen peukaloituihin syötteisiin. Tulevaisuuden tavoitteena onkin kaikissa tilanteissa toimintavarmojen koneoppimismallien kehittäminen.

Viiteluettelo

Akhtar, Naveed and Ajmal Mian. 2019. Threat of adversarial attacks on deep learning in computer vision: a survey. *IEEE Access*. 6, 14410-14430. doi: 10.1109/ACCESS.2018.2807385

Cao, Ning, Guofu Li, Pengjia Zhu, Qian Sun, Yingying Wang, Jing Li, ... and Yongbin Zhao. 2018. Handling the adversarial attacks. *Journal of Ambient Intelligence and Humanized Computing*. 10 (8), 2929-2943. doi: 10.1007/s12652-018-0714-6

Czajkowski, Marcin, and Marek Kretowski. 2019. Decision tree underfitting in mining of gene expression data. An evolutionary multi-test tree approach. *Expert Systems with Applications*. 137, 392-404. doi: 10.1016/j.eswa.2019.07.019

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. Haettu 27.10.2019. *https://arxiv.org/abs/1412.6572*

Izmailov, Rauf, Shridatt Sugrim, Ritu Chadha, Patrick McDaniel, and Ananthram Swami. 2018. Enablers of adversarial attacks in machine learning. *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, Los Angeles, CA, USA, October 29-31, 2018. doi: 10.1109/MILCOM.2018.8599715

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. Haettu 27.10.2019. *https://arxiv.org/abs/1607.02533*

Kurakin, Alexey, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, ... and Motoki Abe. 2018. Adversarial attacks and defences competition. Haettu 27.10.2019. *https://arxiv.org/abs/1804.00097*

McDaniel, Patrick, Nicolas Papernot, and Z. Berkay Celik. 2016. Machine learning in adversarial settings. *IEEE Security & Privacy*. 40 (3), 68-72. doi: 10.1109/MSP.2016.51

Meng, Dongyu, and Hao Chen. 2017. MagNet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 135-147. doi: 10.1145/3133956.3134057

Papernot, Nicolas, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May 22-26, 2016. doi: 10.1109/SP.2016.41

Xie, Cihang, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. 2017. Adversarial examples for semantic segmentation and object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, October 22-29, 2017. doi: 10.1109/ICCV.2017.153

Yuan, Xiaoyong, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: attacks and defences for deep learning. *IEEE Transactions on Neural Networks and Learning Systems.* 30 (9), 2805-2824. doi: 10.1109/TNNLS.2018.2886017

Zhang, Jiliang and Chen Li. 2019. Adversarial examples: opportunites and challenges. *IEEE Transactions on Neural Networks and Learning Systems*. doi: 10.1109/TNNLS.2019.2933524