SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2008

Učenje s virtualnim neprijateljskim primjerima

Pero Skoko

Zagreb, rujan 2019.

1

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem se prof. Šegviću na velikoj pomoći tijekom diplomskog seminara, projekata, završnog i diplomskog rada te roditeljima na podršci tijekom studija.

SADRŽAJ

1. Uvod **1**

2. Polunadzirano učenje **3**

2.1. Osnovni pojmovi strojnog učenja 3

2.1.1. Vjerojatnosna interpretacija strojnog učenja 4

2.1.2. Duboko učenje 4

2.2. Pretpostavke polunadziranog učenja 5

2.2.1. Glatkoća polunadziranog učenja 6

2.2.2. Pretpostavka o klasterima 6

2.2.3. Pretpostavka o mnogostrukosti 7

2.3. Tehnike polunadziranog učenja 8

2.3.1. Minimizacija entropije 8

2.3.2. Pseudooznačavanje 9

3. Uvod u učenje s virtualnim neprijateljskim primjerima **11**

3.1. Neprijateljski primjeri 11

3.2. Učenje s neprijateljskim primjerima 12

3.2.1. Linearnost i neprijateljski primjeri 12

3.2.2. Neprijateljski primjeri i duboki modeli 13

3.2.3. Učenje s neprijateljskim primjerima 15

3.3. Učenje s virtualnim neprijateljskim primjerima 16

3.3.1. Efikasna aproksimacija virtualnog neprijateljskog vektora smjera 17

3.4. Algoritam za izračun VAT gubitka 19

3.5. Postupak nadziranog učenja s VAT-om 20

3.6. Postupak polunadziranog učenja s VAT-om 20

4. Vizualna demonstracija rada VAT-a **21**

4.1. Skup podataka u obliku polumjeseca 21

4.2. Skup podataka s koncentričnim elipsama 28

5. Eksperimenti na MNIST-u **30**

5.1. Model za MNIST 30

5.2. Nadzirano učenje 30

5.2.1. Validacija hiperparametra ϵ 31

5.2.2. Rezultati nadziranog učenja 31

5.3. Polunadzirano učenje 33

5.3.1. Validacija hiperparametra ϵ 33

5.3.2. Rezultati polunadziranog učenja 36

6. Eksperimenti na CIFAR-u i SVHN-u **40**

6.1. CIFAR i SVHN 40

6.2. Modeli za CIFAR i SVHN 40

6.3. Učenje modela 42

6.4. Rezultati na SVHN-u 42

6.5. Rezultati na CIFAR-u 44

7. Zaključak **45**

Literatura **47**

1. Uvod

U ovom diplomskome radu proučavat ćemo primjenu učenja s virtualnim neprijateljskim primjerima [8] na polunadzirano učenje.

Jedan od najtežih zadataka u strojnom učenju jest pribaviti označene podatke. S druge strane do neoznačenih podataka je u pravilu vrlo lako doći. Zbog toga polunadzirano učenje može imati veliku praktičnu primjenu jer se ono koristi i označenim i neoznačenim podacima, što je glavna motivacija za njegovo proučavanje u ovom radu.

Učenje s virtualnim neprijateljskim primjerima je metoda regularizacije utemeljena na ideji povećavanja glatkoće lokalne distribucije oznaka ulaznih primjera, što potiče glatkoću decizijske granice između razreda. Zbog načina na koji je ona definirana, ona ne zahtijeva poznavanje pravih oznaka ulaznih primjera što znači da se može i primijeniti na neoznačene primjene. To omogućuje da učenje s virtualnim neprijateljskim primjerima koristi kao tehnika polunadziranog učenja što nam daje motivaciju da tu primjenu bolje istražimo.

Obradu teme rada započet ćemo s uvodom u teoriju polunadziranog učenja i prikazom njegovih metoda koje su bliske učenju s neprijateljskim primjerima koje ćemo zatim teorijski objasniti u trećem poglavlju. Nakon njegovog teorijskog izvoda, do kojeg ćemo doći preko neprijateljskih primjera i učenja s neprijateljskim primjerima, objasnit ćemo kako ga koristiti pri nadziranom i nenadziranom učenju.

Da bismo bolje shvatili kako učenje s virtualnim neprijateljskim primjerima djeluje, generirat ćemo dvodimenzionalni skup podataka s dva moguća razreda na kojem ćemo učiti jednoslojni model. Kroz vizualizaciju oznaka skupa podataka u različitim iteracijama postupka učenja, upoznat ćemo se načinom na koji učenje s virtualnim neprijateljskim primjerima propagira oznake preko neoznačenih primjera.

U zadnjem dijelu rada provodit ćemo eksperimente na skupovima podataka MNIST, CIFAR i SVHN. Cilj toga je provjeriti kako učenje s virtualnim neprijateljskim primjerima radi u praksi. Eksperimente na MNIST-u ćemo provoditi s potpupno povezanim modelom, dok ćemo za CIFAR i SVHN koristiti puno složenije konvolucijske modele. Konvolucijski modeli imaju veliku primjenu u računalnom vidu pa je bitno na njima

evaluirati učenje s virtualnim neprijateljskim primjerima. S druge strane MNIST je bitno manje zahtjevan skup podataka od CIFAR-a i SVHNA-a pa ima smisla započeti prve prave eksperimente upravo na njemu.

2. Polunadzirano učenje

Glavni cilj ovoga rada je primijeniti učenje s virtualnim neprijateljskim primjerima na polunadzirano učenje. Da bismo to mogli staviti u kontekst područja strojnog i dubokog učenja, pogotovo dijela koji se bavi polunadziranim učenjem, u ovom ćemo poglavlju dati kratki prikaz općenitog polunadziranog učenja i nekih njegovih metoda.

Započet ćemo opisom osnovnih pojmova strojnog i dubokog učenja te notacije vezane za njih koju ćemo koristiti u ostatku rada. Nastavit ćemo s općenitim opisom problema polunadziranog učenja gdje ćemo prikazati tri temeljne pretpostavke koje su u pozadini velike većine njegovih metoda. Na kraju ćemo navesti i objasniti neke njegove metode koje su usporedive s učenjem s virtualnim neprijateljskim primjerima.

2.1. Osnovni pojmovi strojnog učenja

Strojno učenje je područje računarske znanosti u kojem se problemima pristupa tako da se iz određenog skupa podataka (npr. slike, rečenice, itd.) tih problema pokušavaju naučiti pravila koja ih rješavaju.

Primjer problema koji rješavamo strojnim učenjem jest prepoznavanje slika u kojemu je cilj neku sliku svrstati u neki razred ovisno o tome što se na njoj nalazi (npr. razredi slika koje prikazuju auta, avione, ljude, itd.). Slike nam u tom slučaju predstavljaju pojedinačne ulazne primjere koje ćemo označavati s $\vec{x}$, a skup svih mogućih ulaznih primjera s $\mathcal{X}$. Moguće razrede tih slika ćemo označavati s y, a skup tih oznaka s $\mathcal{Y}$. Broj razreda će biti C, a $y^{(j)}$ će predstavljati oznaku za j-ti razred. y_i će biti oznaka i-tog ulaznog primjera $\vec{x}_i$. S $h : \mathcal{X} \to \mathcal{Y}$ označavamo neku funkciju koja svakom ulaznom primjeru dodjeljuje neki razred. Takvo preslikavanje nazivamo hipoteza. S $\mathcal{D}$ označavat ćemo skup primjera za učenje.

Osnovni radni okvir strojnog učenja čine: model, funkcija gubitka i optimizacijski postupak. Model je skup iz kojeg biramo hipoteze. Pojedina hipoteza iz modela je određena parametrima toga modela koje ćemo označavati s θ. Funkcija gubitka mjeri pogrešku na jednom primjeru i označavat ćemo je s $L(\vec{x}, y, \theta)$. Postupak opti-

mizacije nam govori kako da smanjimo ukupnu grešku na primjerima za učenje. Da bismo procijenili koliko dobro model generalizira, odnosno kakvu performansu ima na neviđenim podacima, mjerimo njegovu performansu na skupu za testiranje čije primjere model nije vidio tijekom učenja.

Općenito u strojnom učenju razlikujemo nadzirano, nenadzirano i polunadzirano učenje. U nadziranom učenju za sve podatke u skupu za učenje nam je poznata njihova prava oznaka te u samom postupku optimiziranja modela koristimo i ulazne primjere i njihove oznake.

U nenadziranom učenju imamo dostupe samo podatke, bez njihovih oznaka, te učenjem pokušavamo rekonstruirati što više informacija o njihovoj razdiobi.

Polunadzirano učenje se nalazi između nadziranog i nenadziranog, dostupni su nam i označeni podaci $\{(\vec{x}_i, y_i)\}$ i neoznačeni podaci $\{\vec{x}_j\}$

2.1.1. Vjerojatnosna interpretacija strojnog učenja

Moderni pristup strojnom učenju temelji se teoriji vjerojatnosti tj. vjerojatnosnim interpretacijama njegovih osnovnih pojmova. Tako skupu ulaznih primjera pridružujemo slučajnu varijablu X koja nam govori o vjerojatnostima pojavljivanja određenih ulaznih primjera, te vjerojatnosti označavamo s $p(\vec{x})$. Slično, skupu mogućih razreda $\mathcal{Y}$ pridružujemo slučajnu varijablu Y, a odgovarajuće vjerojatnosti označavamo s $p(y)$.

Problem klasifikacije ulaznih primjera u razrede sada možemo interpretirati kao pokušaj učenja uvjetne distribucije oznaka y u uz poznate ulazne primjere $\vec{x}$, tj. $p(y|\vec{x})$. Ako imamo model čije parametre označavamo s θ i ako nam je iz konteksta jasno o kojem se modelu radi, onda ćemo njegovo preslikavanje ulaznih primjera u razrede označavati kao $p(y|\vec{x}, \theta)$. U ovom radu bavit ćemo se isključivo diskriminativnim modelima, tj. onima koji distribuciju $p(y|x)$ uče eksplicitno, a elemente izlaznog vektora tih modela interpretiramo kao vjerojatnosti pripadnosti određenim razredima, pa ima smisla takve modele u teorijskim razmatranjima poistovjetiti s njihovom uvjetnom distribucijom.

Detaljniji opis osnovnih pojmova strojnog učenja nalazi se u [1] i [12]

2.1.2. Duboko učenje

Duboko učenje je područje strojnog učena koje se bavi dubokim neuronskim mrežama. One danas imaju široku primjenu u područjima poput računalnog vida, prirodne obrade jezika, bioinformatike i ostalim. Njihovi osnovni gradivni elementi su slojevi različitog tipa koji predstavljaju razne matematičke operacije nad podacima. Kombiniranje

tih slojeva je, matematički gledano, kompozicija funkcija, pa su duboke neuronske mreže prikladne za probleme koji imaju kompozicijsku strukturu. A i takva struktura omogućuje efikasno računanje eksplicitnih gradijenata parametara mreže unatražnim postupkom (engl. *Backpropagation*)[3][12]. Ujedno, karakterizira ih veliki broj parametera i velika računska složenst, stoga se one optimiraju postupcima koji se temelje na gradijentnom spustu kao što je ADAM [6].

2.2. Pretpostavke polunadziranog učenja

Kao što smo već rekli, u polunadziranom učenju uz označene primjere u skupu za učenje, dostupni su nam i ulazni primjeri bez oznake. Njegov osnovni problem jest kako pri učenju iskoristiti te primjere za postizanje veće generalizacijske performanse. No prije svega postavlja se pitanje je li to uopće moguće, odnosno može li se iz neoznačenih primjera izvući neka informacija koja bi mogla poboljšati točnost predikcije modela.

Očito je da za pozitivan odgovor na to pitanje moraju vrijediti neke pretpostavke, odnosno informacija o $p(\vec{x})$ koju dobijemo iz neoznačenih primjera iz skupa za učenje $\mathcal{D}_{ul}$ nam mora davati nekakvu informaciju o $p(y|x)$. [2]

U Chapelle et al. [2] su navedene tri pretpostavke koje su u pozadini većine metoda polunadziranog učenju i koje ćemo u narednom tekstu opisati. Ali prije toga objasniti terminologiju koja se koristi u iskazivanju njih.

Kod nekih modela ulazne primjere i izlaze možemo promatrati kao vektore. Npr. sliku veličine 200×300 piksela u RGB formatu vidimo kao tenzor oblika $200 \times 300 \times 3$ koji možemo promatrati kao jedan 180000 dimenzionalni vektor. Izlazi iz naših diskriminativnih modela su također vektori, čije su komponente vjerojatnosti pripadnosti pojedinim razredima. Dakle skupovi ulaznih primjera i skupovi izlaza iz modela čine vektorske prostore, s tim da ulazni prostor može imati jako puno dimenzija, reda veličine sto tisuća kao što vidimo na gornjem primjeru. Dimenzije izlaznih prostora su puno manje, obično između 10 i 1000 i jednaki su broju razreda. Te prostore ćemo promatrati kao klasične euklidske prostore tj. normirane prostore s L_2 normom. U tom smislu, primjeri će biti blizu ako im je udaljenost izračunata preko L_2 norme dovoljno mala. Reći ćemo da je neki podskup ulaznog prostora regija visoke gustoće, ako primjeri u njemu imaju veliku vjerojatnost $p(\vec{x})$.

2.2.1. Glatkoća polunadziranog učenja

Prva pretpostavka koju ćemo navesti je generalizacija slične pretpostavke za nadzirano učenje:

> *Ako su dva ulazna primjera x_1 i x_2 blizu u ulaznom prostoru, onda su njihove odgovarajuće oznake y_1 i y_2 također blizu.*

Bez gornje pretpostavke bilo nemoguće dobiti naučiti model koji dobro generalizira. Naime, u skupu za učenje imamo samo konačan broj podataka, dok je cijeli prostor ulaznih primjera beskonačan, stoga nam ova pretpostavka omogućuje da taj beskonačni prostor aproksimiramo konačnim skupom za učenje.

Za polunadzirano učenje, tu pretpostavku ćemo izmijeniti tako da ćemo uzimati u obzir i gustoću regije u kojoj se primjeri nalaze:

> *Ako su dva ulazna primjera x_1 i x_2 blizu u regiji visoke gustoće ulaznoga prostora, onda su njihove odgovarajuće oznake y_1 i y_2 također blizu.*

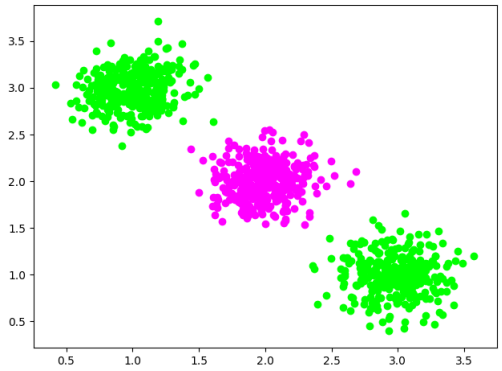
Ova pretpostavka u praksi govori da je funkcija koja ulaznim primjerima pridjeljuje oznake u izlaznom prostoru glatka u regijama visoke gustoće.

2.2.2. Pretpostavka o klasterima

U ovoj pretpostavci govori se o primjerima koji se nalaze u klasterima. Kažemo da je grupa primjera formira klaster u ulaznom prostoru, ako ti primjeri imaju malu međusobnu udaljenost. Za primjere koji su istom klasteru, čini se razumno pretpostaviti da su i u istom razredu, što je bio temelj za razvoj nekih od prvih algoritama polunadziranog i nenadziranog učenja. U njihovoj pozadini stoji pretpostavka:

> *Ako su primjeri unutar istoga klastera, vjerojatno je da su u istom razredu.*

Iz ove pretpostavke ne slijedi da se primjeri iz svakog razreda nalaze u jednome klasteru toga razreda, već samo da je malo vjerojatno da se u istome klasteru nalaze primjeri iz dva različita razreda.

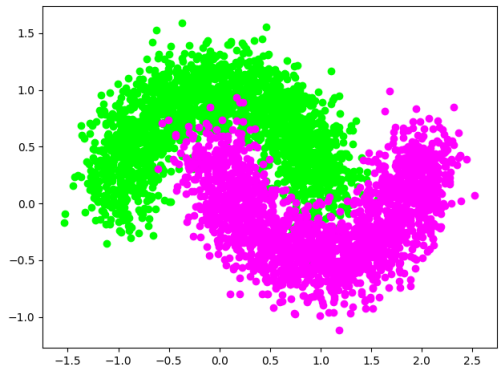


Slika 2.1: Primjer problema u kojem nisu svi podaci iz istog razreda u jednom klasteru

Pretpostavka o klasterima se može izreći i na drugi ekvivalentan način:

Decizijska granica bi se trebala nalaziti u područjima niske gustoće.

Ako bi decizijska granica prolazila kroz područje visoke gustoće, pretpostavka o klasterima ne bi vrijedila, jer područje visoke gustoće predstavlja jedan klaster primjera u kojem bismo imali primjere koji nisu u istom razredu. Iz toga zaključujemo da su te pretpostavke ekvivalentne.



Slika 2.2: Primjer problema u kojem se decizijska granica nalazi u regijama niske gustoće

2.2.3. Pretpostavka o mnogostrukosti

Ulazni primjeri se obično nalaze u nekim visokodimenzionalnim vektorskim prostorima. Unutar tih visokodimenzionalnih prostori, mogu postojati njihovi podskupovi koji su lokalno glatki tj. lokalno "izgledaju" kao vektorski prostori niže dimenzije. Primjer toga bi bila sfera u trodimenzionalnom prostoru koja je dvodimenzionalna ploha. Naime, ako promatramo neku točku na sferi, neka njezina dovoljno mala

okolina će za nju izgledati kao ravnina po kojoj se može kretati u dva smjera. Takvi objekti se općenito zovu mnogostrukostima. A sljedeća pretpostavka govori upravo o njima:

> *Visokodimenzionalni podaci se približno nalaze na niskodimenzionalnoj mnogostrukosti.*

Ovom pretptpostavkom može se ublažiti problem prokletstva dimenzionalnosti (engl. *Curse of Dimensionality*) tj. rasta volumena eksponencijalno s rastom broja dimenzija. On je posebno izražen kod generativnih modela koji se bave procjenom distribucije ulaznih podataka. Međutim, ako ova je pretpostavka zadovoljena, ti modeli mogu efektivno djelovati na niskodimenzionalnom prostoru gdje je prokletstvo dimenzionalnosti manje izraženo.

2.3. Tehnike polunadziranog učenja

U ovom radu, radit ćemo polunadzirano učenje na diskriminativnim dubokim modelima pa ćemo se osvrnuti na polunadzirane tehnike koje koriste takve modele.

2.3.1. Minimizacija entropije

Minimizacija entropije (engl. *Entropy minimization*) je primjer metode polunadziranog učenja koja se temelji na pretpostavci o klasterima (2.1), prvi put je opisana u radu Grandvalet i Bengio [5]. Ona funkciji gubitka dodaje još jedan gubitak čija je svrha potaknuti mrežu da za sve primjere, i označene i neoznačene, daje predviđanja s visokom pouzdanosti.

Gubitak minimizacije entropije će biti uvjetna entropija $H(Y|X)$ oznaka y uvjetovanih na ulazne primjere x koja je definirana kao:

$$H(Y|X) = -\sum_{x \in \mathcal{X}, Y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}$$

Ona izražava količinu informacije koja je potrebna za odrediti oznaku y ako nam je poznat ulazni primjer x. Koristeći Bayesovu formulu $p(x,y) = p(y|x) \cdot p(x)$ možemo je zapisati kao:

$$H(Y|X) = -\sum_{x \in \mathcal{X}} p(x) \sum_{Y \in \mathcal{Y}} p(y|x) \log p(y|x) \tag{2.1}$$

Znamo da se kod diskriminativnih modela eksplicitno uči preslikavanje iz ulaznih primjera x u vjerojatnosti pripadnosti razredima y, tj. $p(y|x)$ pa u gornju formulu

možemo direktno uvrstiti njegove izlaze $p(y^{(j)}|x_i, \theta)$. Ako imamo N primjera u skupu za učenje, stavit ćemo $p(x_i) = \frac{1}{N}$ jer pretpostavljamo da su primjeri iz skupa za učenje reprezentativni uzorak svih mogućih primjera iz $\mathcal{X}$.

$$\mathcal{R}_{cent} = -\frac{1}{N} \sum_{\vec{x}_i \in \mathcal{D}} \sum_{j=0}^{C-1} p(y^{(j)}|\vec{x}_i, \theta) \log p(y^{(j)}|\vec{x}_i, \theta)$$

2.3.2. Pseudooznačavanje

Pseudooznačavnje je tehnika koja se također temelji na pretpostavci o klasterima, ekvivalentna je minimizaciji entropije, ali je formulirana na drugačiji način. Opisana je u Lee [7].

Da bismo je primijenili, prvo moramo mrežu istrenirati nad označenim podacima koristeći nadzirano učenje. Zatim se neoznačenim podacima pridijele oznake pomoću toga istreniranog modela tako da se svrstaju u razrede koji imaju najveću vjerojatnost $p(y^{(j)}|x, \hat{\theta})$ u izlazu toga modela. To nisu prave oznake, već oznake na temelju onoga što je model do toga trenutka naučio pa ćemo ih zvati pseudooznake i označavati s $\widetilde{y}$. Od toga i dolazi naziv pseudooznačavanje. To se može zapisati kao:

$$\widetilde{y} = \arg\max_{y^{(j)}} p(y^{(j)}|x, \hat{\theta})$$

Zatim ćemo te neoznačene primjere i njihove pseudooznake iskoristiti u jednoj iteraciji nadziranog učenja modela, računat ćemo i na njima vrijednost funkcije gubitka kao da se radi o označenim primjerima i njihovim pravim oznaka. Taj postupak pridjeljivanja pseudooznaka neoznačenim primjerima i učenje modela s njima ćemo ponavljati određen broj iteracija i tako učiti model s polunadziranim podacima. S rastom broja iteracija, povećavat ćemo utjecaj gubitka neoznačenih primjera. Gubitak u jednoj iteraciji toga postupka možemo zapisati kao:

$$L = \frac{1}{N_l} \sum_{\vec{x}_i in \mathcal{D}_l} L(y_i, \vec{x}_i) + \alpha(t) \frac{1}{N_{ul}} \sum_{\vec{x}_i in \mathcal{D}_{ul}} L(\widetilde{y}_i, \vec{x}_i)$$

Utjecaj neoznačenih primjera na postupka učenja u ovisnosti o broju iteracija $\alpha(t)$ može biti bilo koja rastuća funkcija. Na početku, dok model još nije naučio ispravno označavati označene primjer, bitno je da bude jako malen ili čak 0, kako bi se izbjeglo zapinjanje u lokalnim minimumima, kasnije se njegov utjecaj povećava, do određene granice, kako bi i neoznačeni primjeri imali utjecaj na postupak učenja.

Ovakav postupak optimizacije potiče model da primjerima pridjeljuje razrede sa što većom vjerojatnošću, tj. on smanjuje entropiju oznaka, zbog čega je ekvivalentan postupku minimizacije entropije.

3. Uvod u učenje s virtualnim neprijateljskim primjerima

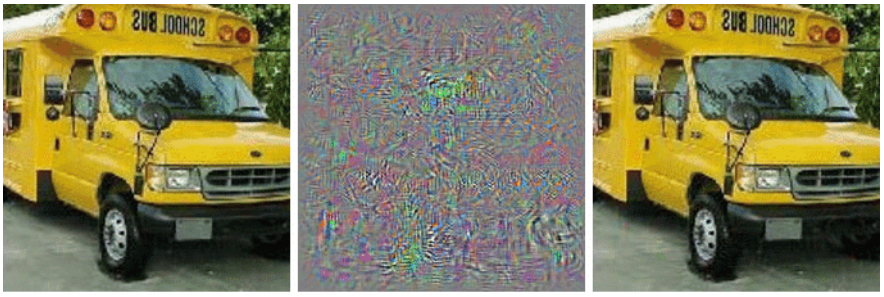
Tema ovoga rada je učenje s virtualnim neprijateljskim primjerima (engl. *Virtual adversarial training*) i njegova primjena na polunadzirano učenje. Učenje s virtualnim neprijateljskim primjerima, u daljnjem tekstu VAT, je relativno nova regularizacijska tehnika koja postiže jako dobre rezultate u nadziranom i polunadziranom učenju. Glavna motivacija za njegovo proučavanje i primjenu jest njegova jednostavnost: ima malen broj hiperparametara, lako ga je iskoristiti pri optimizaciji modela i njegovo djelovanje je objašnjivo i intuitivno.

Cilj ovoga poglavlja je objasniti i prikazati učenje s virtualnim neprijateljskim primjerima. Konceptualno, ono se razvilo iz učenja s neprijateljskim primjerima (engl. *adversarial training*) kojemu je prethodilo otkrivanje tzv. neprijateljskih primjera, pa ćemo objašnjavanje VAT-a započeti s njima.

3.1. Neprijateljski primjeri

Duboke neuronske mreže postižu vrlo visoku točnost na mnogim problemima raznih područja poput računalnog vida, raspoznavanja govora, obrade prirodnog jezika te ostalih. Primjer jednog takvog zadatka je klasifikacija objekata na slikama u razrede gdje duboki modeli postižu točnost koja je usporediva s onom koju postižu ljudi[10]. Razlog za to je njihova velika izražajnost koja im, kroz velik broj parametara i nelinearnih transformacija, omogućuje da nauče proizvoljna preslikavanja. Zbog toga naučene duboke modele nije lako interpretirati te mogu imati neočekivana svojstva, a jedno od njih je pojava neprijateljskih primjera.[11]

Intuitivno, kod klasifikacije slika, očekivali bismo da dodavanje maloga, ljudskom oku neprimjetnog, šuma na originalnu sliku ne može promijeniti rezultat klasifikacije modela koji ima visoku točnost. No u Szegedy et al. [11] je razvijena tehnika koja na proizvoljnom dubokom modelu može generirati takve slike. Općenito takvi primjeri



Slika 3.1: Primjer neprijateljskog primjera na alexnet-91, preuzeto iz Szegedy et al. [11]
Lijevo - orignalna slika označena kao autobus
Desno - neprijateljski primjer označen kao noj
Sredina - razlika između slika uvećana 10 puta

se nazivaju neprijateljski primjeri (engl. *adversarial examples*).

3.2. Učenje s neprijateljskim primjerima

Na prvi pogled pojava neprijateljskih primjera bi se mogla pripisati velikoj nelinearnosti dubokih modela, no u Goodfellow et al. [4] se kreće od suprotne pretpostavke i pokazuje da je to posljedica njihove prevelike linearnosti. Duboke mreže su općenito dizajnirane tako da se mnogi njihovi dijelovi ponašaju linearno. Čak i nelinearne funkcije poput ReLU i sigmoide se dijelom ponašaju linearno, posebice u područjima van njihova zasićenja. Zbog toga neprijateljski primjeri u dubokim modelima mogu biti posljedica neprijateljskih primjera kod linearnih modela.

3.2.1. Linearnost i neprijateljski primjeri

Promatrajmo jednostavni linearni model koji ima samo vektor težina $\mathbf{w}$, bez pomaka. Neka je ulaz u njega vektor $\vec{x}$ pa je on definiran kao:

$$y = \mathbf{w}^T \cdot \vec{x} \tag{3.1}$$

Promatrajmo sada vektor šuma $\vec{\eta}$ koji je iste veličine kao $\vec{x}$. Želimo da dodavanje toga vektora šuma na $\vec{x}$ bude neprimjetno ljudskom oku ili nekom senzoru što ćemo kvantificirati s parametrom ϵ, tako da šum smatramo neprimjetnim ako mu je svaka komponenta manja od ϵ. To možemo pomoću norme beskonačno:

$$\|\vec{\eta}\|_\infty \leq \epsilon \tag{3.2}$$

Stavimo $\widetilde{x} = \vec{x} + \vec{\eta}$. Ako je ϵ dovoljno malen za područje primjene, onda je razlika između $\vec{x}$ i $\widetilde{x}$ nezamjetna. Stvar je u tome da, ovisno o području primjene, podaci imaju konačnu preciznost. Npr. kod slika koje su pohranjene kao 8-bitni pikseli, sva razlika između sva informacija koja je ispod $\frac{1}{255}$ dinamičkog raspona slike se odbacuje zbog njegove kvantizacije na 255 mogućih vrijednosti. I ako bismo vrijednosti piksela te slike normalizirali na interval $[0, 1]$ i onda koristili kao ulaz modela, bilo bi razumno očekivati da se njegov izlaz neće značajno promijeniti ako na taj ulaz dodamo šum čije su komponente manje od $\frac{1}{255}$.

Promatrajmo linearni model i na njegov ulaz stavimo $\widetilde{x}$, njegov izlaz će biti:

$$\mathbf{w}^T \cdot \widetilde{x} = \mathbf{w}^T \cdot \vec{x} + \mathbf{w}^T \cdot \vec{\eta} \tag{3.3}$$

Dakle razlika u izlazu je $\mathbf{w}^T \cdot \vec{\eta}$. Cilj nam je tu razliku maksimizirati uz ograničenje $\|\vec{\eta}\|_\infty \leq \epsilon$, a to ćemo postići tako da stavimo:

$$\vec{\eta} = \epsilon \cdot sign(\mathbf{w}) \tag{3.4}$$

Ako je n dimenzija vektora $\vec{x}$ i $\mathbf{w}$, a m prosječna apsolutna vrijednost komponenti $\mathbf{w}$ tj. prosječna apsolutna vrijednost težina u našem linearnom modelu, onda za tu razliku izlazima vrijedi:

$$\mathbf{w}^T \cdot \vec{\eta} = \epsilon nm \tag{3.5}$$

Iz toga vidimo da ta maksimalna razlika raste linearno s n, brojem dimenzija ulaznog vektora. To znači da se kod problema s velikim brojem dimenzije može dogoditi da niz malih promjena u ulaznom primjeru proizvede jednu veliku promjenu na izlazu modela. Dakle i linearni model može imati neprijateljske primjere ako ima dovoljno veliku dimenzionalnost.

Mali šum $\vec{\eta}$ koji dodajemo na ulazni primjer ćemo u daljnjem tekstu nazivati perturbacija, a perturbacije koje su dobivene linearizacijom modela zvat ćemo linearnima. Sada ćemo ove zaključke dobivene na linearnom modelu pokušati primijeniti na općeniti duboki diskriminativni model.

3.2.2. Neprijateljski primjeri i duboki modeli

Kao što smo već zaključili, duboki modeli u sebi inherentno sadrže dosta linearnosti pa zbog toga ima smisla zaključke iz prethodnog potpoglavlja primijeniti na njih. Tu se linearnost modela odnosila na linearnu ovisnost izlaza o ulaznom vektoru $\vec{x}$, pa ćemo krenuti od linearizacije općenitog dubokok modela oko njegova ulaza $\vec{x}$ preko njegove funkcija gubitka.

Neka θ predstavlja parametre dubokog modela, $\vec{x}$ ulaz u njega, a y njegov željeni izlaz te $L(\vec{x}, y, \theta)$ funkcija gubitka toga modela. Fiksirajmo neki ulazni primjer $\vec{x_0}$ i za njega odredimo perturbaciju čije će dodavanje najviše promijeniti vrijednost funkcije gubitka, naravno uz ograničenje njezine veličine: $\|\vec{\eta}\|_\infty \leq \epsilon$. Razvijmo funkciju gubitka $L(\vec{x}, y, \theta)$ u Taylorov red prvog stupnja oko točke $\vec{x_0}$:

$$L(\vec{x_0} + \vec{\eta}, y, \theta) = L(\vec{x_0}, y, \theta) + \nabla_{\vec{x}} L(\vec{x}, y, \theta)|_{\vec{x}=\vec{x_0}} \cdot \vec{\eta}$$

Vrijedi:

$$L(\vec{x_0} + \vec{\eta}, y, \theta) - L(\vec{x_0}, y, \theta) = \nabla_{\vec{x}} L(\vec{x}, y, \theta)|_{\vec{x}=\vec{x_0}} \cdot \vec{\eta}$$

Dakle promjena funkcija gubitka za perturbacije $\vec{\eta}$ iznosi:

$$\nabla_{\vec{x}} L(\vec{x}, y, \theta)|_{\vec{x}=\vec{x_0}} \cdot \vec{\eta}$$

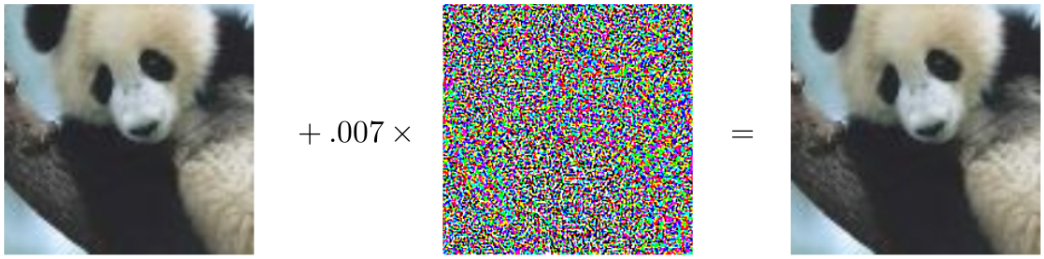
Ta promjena je prikazana preko linearne funkcije kao u 3.2.1. U ovom slučaju uloga vektora težina ima gradijent funkcije gubitka oko $\vec{x_0}$ tj. $\mathbf{w} = \nabla_{\vec{x}} L(\vec{x}, y, \theta)|_{\vec{x}=\vec{x_0}}$. Iz toga znamo da će, uz dano ograničenje $\|\vec{\eta}\|_\infty \leq \epsilon$ ta promjena biti najveća kada stavimo:

$$\vec{\eta} = \epsilon \cdot sign(\mathbf{w}) = \epsilon \cdot \nabla_{\vec{x}} L(\vec{x}, y, \theta)|_{\vec{x}=\vec{x_0}}$$

Dodavanjem te perturbacije na originalni vektor $\vec{x_0}$ dobili smo njegov odgovarajući neprijateljski primjer:

$$\vec{x}_{adv} = \vec{x_0} + \vec{\eta} = \vec{x_0} + \epsilon \cdot sign(\mathbf{w}) = \vec{x_0} + \epsilon \cdot \nabla_{\vec{x}} L(\vec{x}, y, \theta)|_{\vec{x}=\vec{x_0}}$$

Bitno je primijetiti da ova procjena perturbacije s najvećom promjenom funkcije gubitka nije egzaktna, ne mora značiti da će ona zaista i dati najbolju takvu perturbaciju. Međutim, zbog već utvrđene inherentne prisutnosti linearnosti kod dubokih modela imamo razloga pretpostavljati da ćemo s ovakvom procjenom dobiti zadovoljavajuće neprijateljske primjere.



Slika 3.2: Primjer neprijateljskog primjera na GoogLeNet, preuzeto iz Goodfellow et al. [4]
Lijevo - orignalna slika označena kao panda
Desno - neprijateljski primjer označen kao gibon
Sredina - predznaci gradijenta modela za sliku lijevo

I praksi se pokazalo da je ova metoda dosta dobra za generiranje neprijateljskih primjera, naziva se FGSM (engl. *Fast gradient sign method*). Prvi put se pojavila u radu *GoodfellowSS14* u kojem je na temelju nje razvijeno tzv. učenje s neprijateljskim primjerima (engl. *adversarial training*). Upravo iz tog učenja dobit će se učenje s virtualnim neprijateljskim primjerima koje je tema ovoga rada.

3.2.3. Učenje s neprijateljskim primjerima

U učenju s neprijateljskim primjerima ideja je tijekom postupka učenja u skup podataka konstantno ubacivati neprijateljske primjere prema trenutnom stanju modela. To ćemo napraviti preko funkcije gubitka $L(\vec{x}, y, \theta)$. Ako imamo primjer $\vec{x}$ s oznakom y, iz prošlog potpoglavlja znamo da odgovarajući neprijateljski primjer dobivamo kao: $\vec{x} + \epsilon \cdot \nabla_{\vec{x}} L(\vec{x}, y, \theta)$. I na tome neprijateljskom primjeru ćemo računati vrijednost funkcije gubitka: $L(\vec{x} + \epsilon \cdot \nabla_{\vec{x}} L(\vec{x}, y, \theta), y, \theta)$ koju ćemo dodati na originalnu funkciju gubitka:

$$\widetilde{L}(\vec{x}, y, \theta) = L(\vec{x}, y, \theta) + \alpha L(\vec{x} + \epsilon \cdot \nabla_{\vec{x}} L(\vec{x}, y, \theta), y, \theta) \tag{3.6}$$

Korištenje te modificirane funkcije gubitka $\widetilde{L}(\vec{x}, y, \theta)$ je učenje s neprijateljskim primjerima, a α je njezin hiperparametar.

3.3. Učenje s virtualnim neprijateljskim primjerima

Da bismo iz učenja s neprijateljskim primjerima, dobili učenje s virtualnim neprijateljskim primjerima, prvo ćemo gubitak iz napisati na malo drugačiji način. Budući da radimo s problemima klasifikacije znamo funkcija gubitka L je ustvari unakrsna entropija između izlaza modela i vektora sa stvarnom oznakom ulaznog primjera. Taj vektor ima jedinicu na komponenti koja odgovara razredu ulaznog primjera, a na svim ostalim nula. To možemo interpretirati kao stvarnu uvjetnu distribuciju oznaka y uvjetno na ulazne primjere $\vec{x}$, i nju ćemo, da bi notacija bila jasnija, označiti s $q(y|\vec{x})$. Dakle funkciju gubitka možemo napisati kao:

$$L(\vec{x},y,\theta) = H\left[q(y|\vec{x}), p(y|\vec{x},\theta)\right] \tag{3.7}$$

Perturbaciju $\vec{\eta}$ i prethodnih potpoglavlja ćemo zvati neprijateljski smjer i označavati s $\vec{r}_{nep}$. Također umjesto znake unakrsne entropije H stavit ćemo D kao oznaku za općenitu udaljenost između dvije distribucije. Pa se funkcija gubitka učenja s neprijateljskim primjerima može napisati kao:

$$L_{nep}(\vec{x},y,\theta) = D\left[q(y|\vec{x}), p(y|\vec{x}+\vec{r}_{nep},\theta)\right] \tag{3.8}$$

$\vec{r}_{nep}$ odnosno $\vec{\eta}$ iz prethodnog poglavlja smo dobili tako da smo tražili perturbaciju u kojoj dolazi do najveće promjene funkcije gubitka. Sada ćemo to reformulirati tako da ćemo tražiti smjer u kojem je najveća promjena izlazne **distribucije**, pa to možemo izraziti kao:

$$\vec{r}_{nep} := \underset{r;\|r\|_2\leq\epsilon}{\arg\max}\, D\left[q(y|\vec{x}), p(q|\vec{x}+\vec{r}_{nep},\hat{\theta})\right] \tag{3.9}$$

I umjesto norme beskonačno, koristimo L_2 normu za postavljanje ograničenja.

U (3.8) i (3.9) vidimo da učenje s neprijateljskim primjerima ovisi o točnoj distribuciji oznaka ovisno o ulaznim, dakle za njega je potrebno poznavati oznake ulaznih primjera pa se zbog toga ne može koristiti u polunadziranom učenju. Zato ćemo u tim formulama, umjesto stvarne uvjetne distribucije $q(y|x)$ staviti trenutnu uvjetnu distribuciju modela $p(y|\vec{x},\theta)$. Tu vidimo analogiju s prethodno opisanim metodama polunadziranog učenja 2.3.2 i 2.3.1. U njima isto koristimo ono što model u određenom trenutku "vjeruje" za uključivanje neoznačenih primjera u proces učenja. Samo što ovdje to kombiniramo s neprijateljskim učenjem da bismo dobili tzv. virtualne neprijateljske primjere odnosno virtualni neprijateljski smjer $\vec{r}_{vnep}$.

Da bi ostatak izvoda bio jasniji, malo ćemo promijeniti notaciju. S $\vec{x}_\star$ ćemo označavati primjer za koji računamo virtualni neprijateljski primjer. Kada parametre modela

θ promatramo kao varijable po kojima se računa gradijent pri optimizaciji modela, označavat ćemo ih bez "kapice" tj. θ, a s druge strane kada koristimo samo njihovu trenutnu vrijednost i ne gledamo ih varijable za optimizaciju modela, označavat ćemo ih s "kapicom" tj. $\hat{\theta}$.

Sada na temelju zamjene distribucija definiramo sljedeću mjeru:

$$L_{vnep}(x_*, \theta) := D\left[p(y|\vec{x}_\star, \hat{\theta}), p(y|\vec{x}_\star + \vec{r}_{vnep}, \theta)\right] \tag{3.10}$$

Gdje je:

$$\vec{r}_{vnep} := \underset{r;\|r\|_2 \leq \epsilon}{\arg\max} D\left[p(y|x_\star, \hat{\theta}), p(y|x_* + r, \hat{\theta})\right] \tag{3.11}$$

Gubitak iz (3.10) je ustvari negativna mjera lokalne glatkoće distribucije (engl. *Local Distributional Smoothness*) oko točke $\vec{x}_\star$ za trenutno stanje modela. Pod negativna mjera, mislimo na to da što je njezina vrijednost manja, lokalna glatkoća distribucije oko nekog primjera je veća. U (3.11) tražimo, u nekoj maloj okolini primjera $\vec{x}_\star$ smjer u kojem se distribucija oznaka, uvjetno na taj primjer, najviše mijenja, i onda u (3.10) i računamo kolika je ta promjena u distribuciji. Ona nam zapravo govori koliko je ta uvjetna distribucija $p(y|\vec{x}_\star)$ glatka, a zbog toga što to u maloj okolini primjer $\vec{x}_\star$, to je mjera lokalne glatkoće distribucije.

U daljnjem tekstu ćemo za učenje s virtualnim neprijateljskim primjerima koristiti kraticu VAT.

3.3.1. Efikasna aproksimacija virtualnog neprijateljskog vektora smjera

Računanje virtualnog neprijatljeskog gubitka temelji se na poznavanju virtualnog neprijateljskog vektora smjera $\vec{r}_{vsup}$ za dani ulazni primjer $\vec{x}_\star$. Po definiciji, to je smjer koji uzrokuje najveću promjenu distribucije izlaza danog primjera uz ograničenje njegove L_2 norme: $\|\vec{r}_{vsup}\|_2 \leq \epsilon$.

To je optimizacijski problem s eksplicitnim ograničenjem za koji nije poznato postoji li rješenje u zatvorenoj formi. Definirajmo:

$$D(\vec{r}, \vec{x}_\star, \hat{\theta}) := D\left[p(y|\vec{x}_\star, \hat{\theta}), p(y|\vec{x}_\star + \vec{r}, \hat{\theta})\right] \tag{3.12}$$

$D(\vec{r}, \vec{x}_\star, \hat{\theta})$ ćemo promatrati kao funkciju koja ovisi o $\vec{r}$ s fiksnim parametrima $\vec{x}_\star$ i $\hat{\theta}$. U prvom koraku ćemo ga razviti oko točke $\vec{r} = 0$ u Taylorov red drugog stupnja.

$$D(\vec{r}, \vec{x}_\star, \hat{\theta}) \approx D(0, \vec{x}_\star, \hat{\theta}) + \nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=0} + \frac{1}{2}\vec{r}^\top \cdot H(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=0} \cdot \vec{r} \tag{3.13}$$

$\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})$ i $H(\vec{r}, \vec{x}_\star, \hat{\theta})$ označavaju odgovarajući gradijent i Hesseovu matricu od $D(\vec{r}, \vec{x}_\star, \hat{\theta})$. Formalna matematička pretpostavka za ovakav razvoj je da je $p(y|\vec{x}, \theta)$ dva puta diferencijabilna kao funkcija po varijabli $\vec{x}$. Za $\vec{r} = \vec{0}$ je $D(\vec{0}, \vec{x}_\star, \hat{\theta}) = 0$ zbog:

$$D(\vec{0}, \vec{x}_\star, \hat{\theta}) = D\left[p(y|\vec{x}_\star, \hat{\theta}), p(y|\vec{x}_\star, \hat{\theta}) = 0\right]$$

Tada je to udaljenost između dvije iste distribucije pa mora biti 0. Također, to je ujedno i njezin minimum jer udaljenost između distribucija ne može biti manja od 0 (odnosno veća od 0 ako nam D vraća negativne vrijednosti pa je to maksimum). Zbog toga je "prva derivacija" odnosno gradijent jednak 0:

$$\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\vec{0}} = 0 \tag{3.14}$$

pa je konačni izraz:

$$D(\vec{r}, \vec{x}_\star, \hat{\theta}) \approx \frac{1}{2}\vec{r}^\top \cdot H(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\vec{0}} \cdot \vec{r} \tag{3.15}$$

Iz toga slijedi da optimizacijski problem možemo zapisati kao (konstantu $\frac{1}{2}$ možemo zanemariti jer utječe samo na vrijednost optimuma, ne i točku u kojoj se postiže) :

$$\vec{r}_{vnep} := \underset{\vec{r};\|r\|_2 \leq \epsilon}{\arg\max} \left\{\vec{r}^\top H(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\vec{0}} \cdot \vec{r}\right\} \tag{3.16}$$

Rješavanje gornjeg optimizacijskog problema svodi se na pronalaženje svojstvenog vektora matrice $H(\vec{0}, \vec{x}_\star, \hat{\theta})$ koji ima najveću odgovarajuću svojstvenu vrijednosti. Tada taj vektor samo normaliziramo tako da bude duljine ϵ. Tj. ako je $u(\vec{x}_\star, \hat{\theta})$ takav svojstveni vektor rješenje optimizacijskog problema je:

$$\vec{r}_{vnep} = \epsilon \frac{u(\vec{x}_\star, \hat{\theta})}{\left\|u(\vec{x}_\star, \hat{\theta})\right\|_2} \tag{3.17}$$

U sljedećemu koraku cilj nam je aproksimirati dominantni svojstveni vektor u za što ćemo iskoristiti metodu iteracije potencija. Naime, ako je d bilo kakav vektor koji nije okomit na dominantni svojstveni vektor u matrice H, sljedeći niz vektora će konvergirati prema u:

$$\vec{b_0} = \vec{d} \tag{3.18}$$

$$\vec{b}_{n+1} = \frac{H\vec{b}_n}{\left\|H\vec{b}_n\right\|_2} \tag{3.19}$$

Broj iteracija koje će se raditi u gornjoj procjeni vektora u označavat ćemo s K i on je hiperparametar algoritma.

Na kraju moramo još aproksimirati računanje vektora Hd jer bi izravno računanje Hesseove matrice zbog velikog broja parametara bilo praktički nemoguće u kontekstu dubokog učenja. Iskoristit ćemo činjenicu da je u našem slučaju ta matrica gradijent gradijenta tj:

$$H(\vec{0}, \vec{x}_\star, \hat{\theta}) = \nabla\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\vec{0}} \tag{3.20}$$

Pa $H(\vec{0}, \vec{x}_\star, \hat{\theta})d$ možemo aproksimirati po definiciji preko konačne razlike gradijenata:

$$H(\vec{0}, \vec{x}_\star, \hat{\theta})\vec{d} = \frac{\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\xi\vec{d}} - \nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\vec{0}}}{\xi} \tag{3.21}$$

Opet iskorištavamo činjenicu da je u $\vec{r} = \vec{0}$ optimum i $\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\vec{0}}$ pa je:

$$H(\vec{0}, \vec{x}_\star, \hat{\theta})\vec{d} = \frac{\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\xi\vec{d}}}{\xi} \tag{3.22}$$

I jednu iteraciju tog postupka možemo zapisati kao:

$$\vec{d} \longleftarrow \frac{\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\xi\vec{d}}}{\left\|\nabla D(\vec{r}, \vec{x}_\star, \hat{\theta})|_{\vec{r}=\xi\vec{d}}\right\|} \tag{3.23}$$

Nakon provođenja željenog broja iteracija, d će biti aproksimacija virtualnog suparničkog vektora. U implementaciji, vrijednost za računanje končane razlike će biti $\xi = 10^{-6}$.

3.4. Algoritam za izračun VAT gubitka

Podatci: $\mathcal{D} = \{(\vec{x}_i)\}_1^M$ - grupa od M ulaznih primjera
Generiraj M slučajnih jediničnih vektora iz normalne distribucije $\vec{d}_i$
Pridruži $\vec{r}_i = \xi\vec{d}_i$ za $i = 1 \dots M$
za $k \leftarrow 1$ **do** K **čini**
  izračunaj gradijente $\vec{g}_i$ s obzirom na $\vec{r}$ u točki $\vec{r}_i$ za $i = 1 \dots M$:
  $\vec{g}_i \leftarrow \nabla_{\vec{r}} D\left[p(y|\vec{x}_\star, \hat{\theta}), p(y|\vec{x}_\star + \vec{r}, \hat{\theta})\right] |_{\vec{r}} = \vec{r}_i$
  $\vec{r}_i \leftarrow \frac{\vec{g}_i}{\|\vec{g}_i\|_2}$
kraj
vrati $\nabla_\theta \left(\frac{1}{M}\sum_{i=1}^{M} D\left[p(y|\vec{x}_i, \hat{\theta}), p(y|\vec{x}_i + \vec{r}_i, \theta)\right]\right) |_{\theta=\hat{\theta}}$

Algoritam 1: Izračun gradijenta gubitka učenja s virtualnim neprijateljskim primjerima na grupi

U 1 je okvirno ovisan postupak računanja gradijenta kod VAT gubitka, pri učenju na grupi od M primjera.

3.5. Postupak nadziranog učenja s VAT-om

U nadziranom učenju VAT ima regularizacijsku ulogu, on samo dodaje novi član na već postojeću funkciju gubitka. U svakoj iteraciji odaberemo grupu podataka (engl. *batch*) za koju računamo i standardni gubitak unakrsne entropije, i VAT gubitak.

Podatci: $D = \{(\vec{x}_i, y_i)\}_1^N$
za $i \leftarrow 1$ **do** *broj iteracija* **čini**
  Odaberi grupu od M podataka $\{(\vec{x}_i, y_i)\}_{i=1}^M$
  $L_{ce} \leftarrow \frac{1}{M} \sum_{i=1}^M H(p(y|\vec{x_i}; \theta), y_i)$
  $L_{vat} \leftarrow \frac{1}{M} \sum_{i=1}^M LDS(\vec{x}_i, \theta)$
  $L \leftarrow L_{ce} + \alpha \cdot L_{vat}$
  izračunaj $\nabla_\theta L|_{\theta=\hat{\theta}}$, za $\nabla_\theta L_{vat}|_{\theta=\hat{\theta}}$ iskoristi postupak 1
  Ažuriraj $\hat{\theta}$ uz pomoć $\nabla_\theta L|_{\theta=\hat{\theta}}$
kraj

Algoritam 2: Okvirni postupak nadziranog učenja uz pomoć VAT-a

3.6. Postupak polunadziranog učenja s VAT-om

U polunadziranom učenju s VAT-om u svakoj iteraciji imamo dvije grupe podataka. Prva dolazi iz skupa označenih podataka i na njoj računamo standardni gubitak unakrsne entropije. Drugu grupu dobivamo iz cijelog skupa podataka, dakle i označenih i neoznačenih primjera, i na njima računamo VAT gubitak i gradijent s obzirom na njega.

Podatci: $D_l = \{(\vec{x}_i^l, y_i)\}_1^{N_l}$, $D_{ul} = \{\vec{x}_i^{ul}\}_1^{N_{ul}}$
za $i \leftarrow 1$ **do** *broj iteracija* **čini**
  Odaberi grupu od M_l podataka $\{(\vec{x}_i^l, y_i)\}_{i=1}^{M_l}$ iz D_l
  Odaberi grupu od M_{vat} podataka $\{\vec{x}_i\}_{i=1}^{M_{vat}}$ iz D_l i D_{ul}
  $L_{ce} \leftarrow \frac{1}{M_l} \sum_{i=1}^{M_l} H(p(y|\vec{x_i}^l; \theta), y_i)$
  $L_{vat} \leftarrow \frac{1}{M_{vat}} \sum_{i=1}^{M_{vat}} LDS(\vec{x}_i, \theta)$
  $L \leftarrow L_{ce} + \alpha \cdot L_{vat}$
  izračunaj $\nabla_\theta L|_{\theta=\hat{\theta}}$, za $\nabla_\theta L_{vat}|_{\theta=\hat{\theta}}$ iskoristiti postupak 1
  Ažuriraj $\hat{\theta}$ uz pomoć $\nabla_\theta L|_{\theta=\hat{\theta}}$
kraj

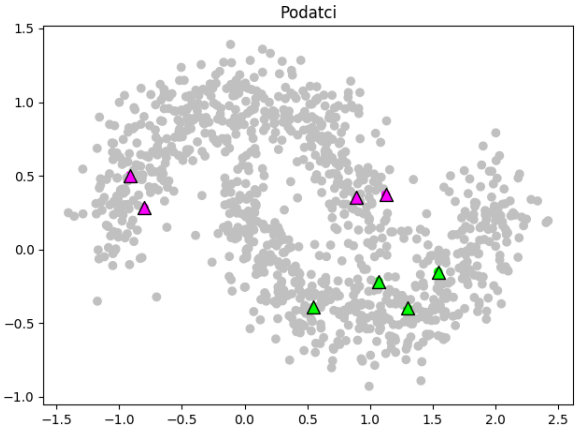
Algoritam 3: Okvirni postupak polunadziranog učenja uz pomoć VAT-a

4. Vizualna demonstracija rada VAT-a

U 3.3 smo zaključili da VAT povećava glatkoću decizijske granice između razreda u ulaznom prostoru tako da povećava lokalnu distribucijsku glatkoću. Da bismo bolje mogli razumjeti kako to praksi izgleda, a i da bismo se uvjerili da VAT zaista može imati pozitivan utjecaj na učenje modela, provest ćemo dva jednostavna eksperimenta u kojima ćemo klasificirati dvodimenzionalne podatke u dva razreda. Tako ćemo moći jednostavno vizualizirati vjerojatnosti pripadnosti pojedinih podataka određenom razredu, i vrijednosti njihovih VAT gubitaka.

4.1. Skup podataka u obliku polumjeseca

Prvi eksperiment ćemo provesti na podatcima generiranima u dva polukruga koji se međusobno preklapaju, a od kojih svaki predstavlja jedan razred. Na slučajan način ćemo odabrati po četiri primjera iz oba razreda za koje će nam pri postupku učenja biti poznata oznaka, i onda na temelju njih pokušati naučiti model a klasificira sve podatke iz skupa.



Slika 4.1: Generirani skup podataka

Model će nam biti neuronska mreža s jednim skrivenim slojem od 80 neurona, izlaz

iz nje će biti vjerojatnost da određeni primjer pripada drugom razredu tj. $p(y = 1|\vec{x}, \hat{\theta})$. To ćemo vizualizirati tako da ćemo podatke obojiti na sljedeći način:

- $p(y = 1|\vec{x}, \hat{\theta}) = 0$ - ljubičasta boja
- $p(y = 1|\vec{x}, \hat{\theta}) = 0.5$ - siva boja
- $p(y = 1|\vec{x}, \hat{\theta}) = 1$ - zelena boja
- boje ostalih vjerojatnosti ćemo interpolirati između prošle tri

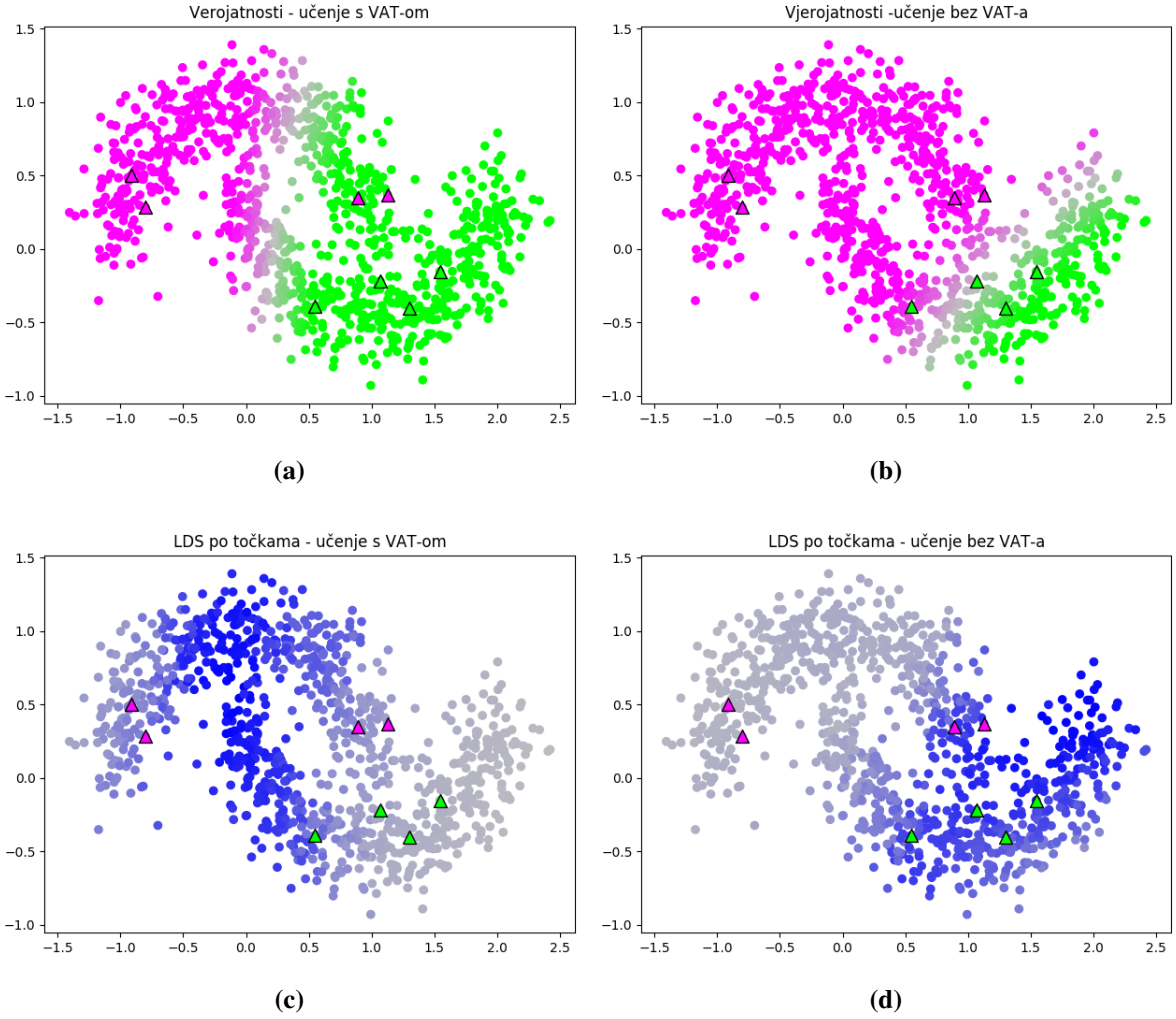
Ljubičastim i zelenim trokutićima prikazat ćemo primjere za koje nam je poznata oznaka pri postupku učenja. Generirali smo tisuću primjera pomoću funkcije *make_moons* iz biblioteke *scikit-learn* s vrijednosti šuma 0.175.

Provodit ćemo polunadzirani postupak učenja opisan u 3, s tim da će grupe označenih i neoznačenih podataka sadržavati sve podatke. Zbog male veličine podataka, svi mogu lako stati u memoriju pa nema potrebe za podjelom podataka u manje grupe. Stopa učenja će biti 0.1, a vrijednosti hiperparametara VAT-a su: $\epsilon = 0.2$, broj iteracije $K = 1$ i $\alpha = 1$.

Pored vjerojatnosti pripadnosti drugom razredu, vizualizirat ćemo i vrijednosti VAT gubitka za pojedine primjere, s rasponom od sive, za primjere s najmanjim VAT gubitkom, do plave za primjere s najvećim VAT gubitkom.

Paralelno ćemo učiti dva modela s istom arhitekturom, za jedan ćemo koristiti VAT gubitak, a za drugi ne kako bismo ih mogli usporediti i odrediti utjecaj VAT-a.Na slici 4.1 je prikazan generirani skup podataka.

Stanje nakon 1. iteracije učenja



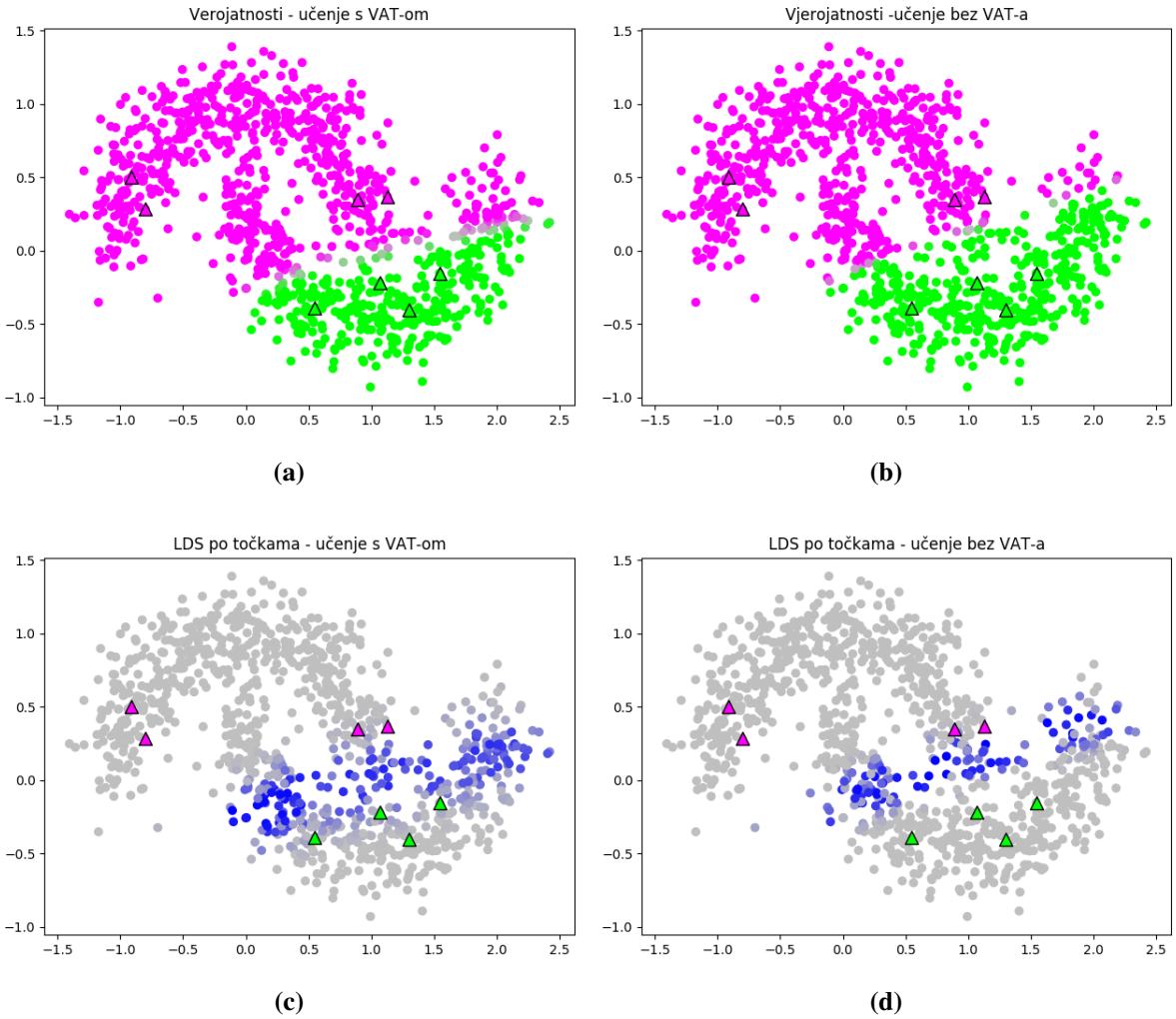
Slika 4.2: Vizualizacija nakon prve iteracije

Na slici 4.2 vidimo vizualizaciju nakon jedne iteracije učenja. S obzirom na to da se radi o jednoj iteraciji, ne možemo još sa sigurnošću analizirati razliku u klasifikaciji između modela s VAT-om i bez VAT-a jer bi ona mogla biti posljedica različite inicijalizacije težina modela. No možemo primijetiti da je VAT gubitak najveći oko područja u okolini granice između prvog i drugog razreda.

Na slici 4.2a koja prikazuje vjerojatnosti kod učenja s VAT-om, vidimo da se dva podatka iz prvog nalaze duboko u području drugog razreda, a na clici 4.2c se vidi da VAT gubitak u tom području nije najveći. To je zato što je za primjere s oznakom zadužen gubitak logističke regresije, on treba osigurati da takve primjere model ispravno označava. A VAT gubitak ima svrhu poboljšati glatkoću lokalne distribucije

i tako osigurati da model bolje klasificira neoznačene primjere propagirajući oznake onih koji su označeni. S obzirom na to da je cijela okolina ta dva primjera s oznakom označena kao drugi razred, VAT gubitak ne može biti velik.

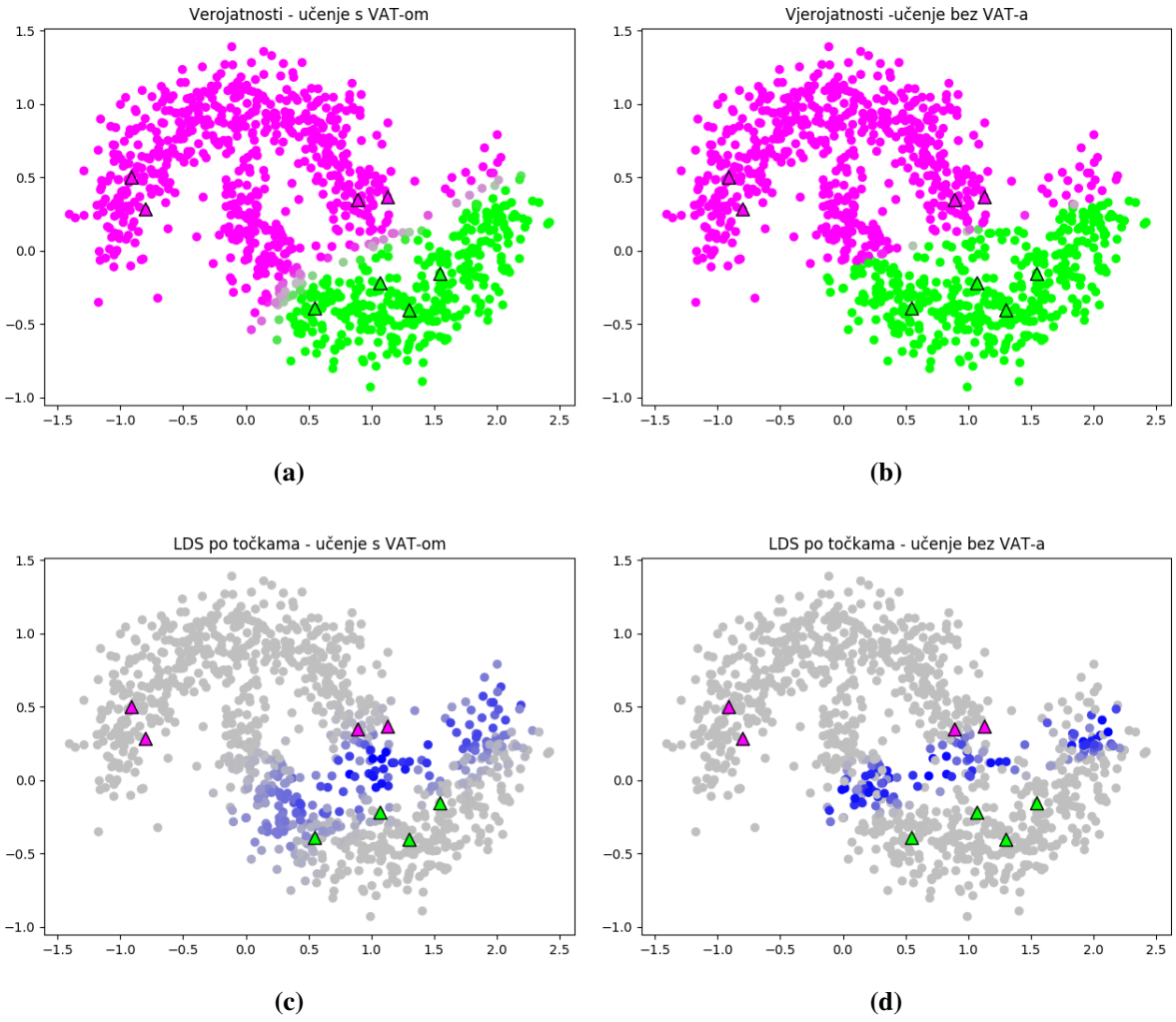
Stanje nakon 10 iteracija učenja



Slika 4.3: Vizualizacija nakon 10. iteracije

Nakon 10 iteracija (4.3) oba modela su naučila ispravno klasificirati primjere s oznakom, i kod oba je granica razgraničenja između dva razreda praktički linearna. Kod modela bez VAT-a granica je oštrija nego kod onog s VAT-om. To se na slici VAT gubitka očituje tako da je na 4.3c područje povećanog VAT gubitka oko decizijske granice šire nego na 4.3d.

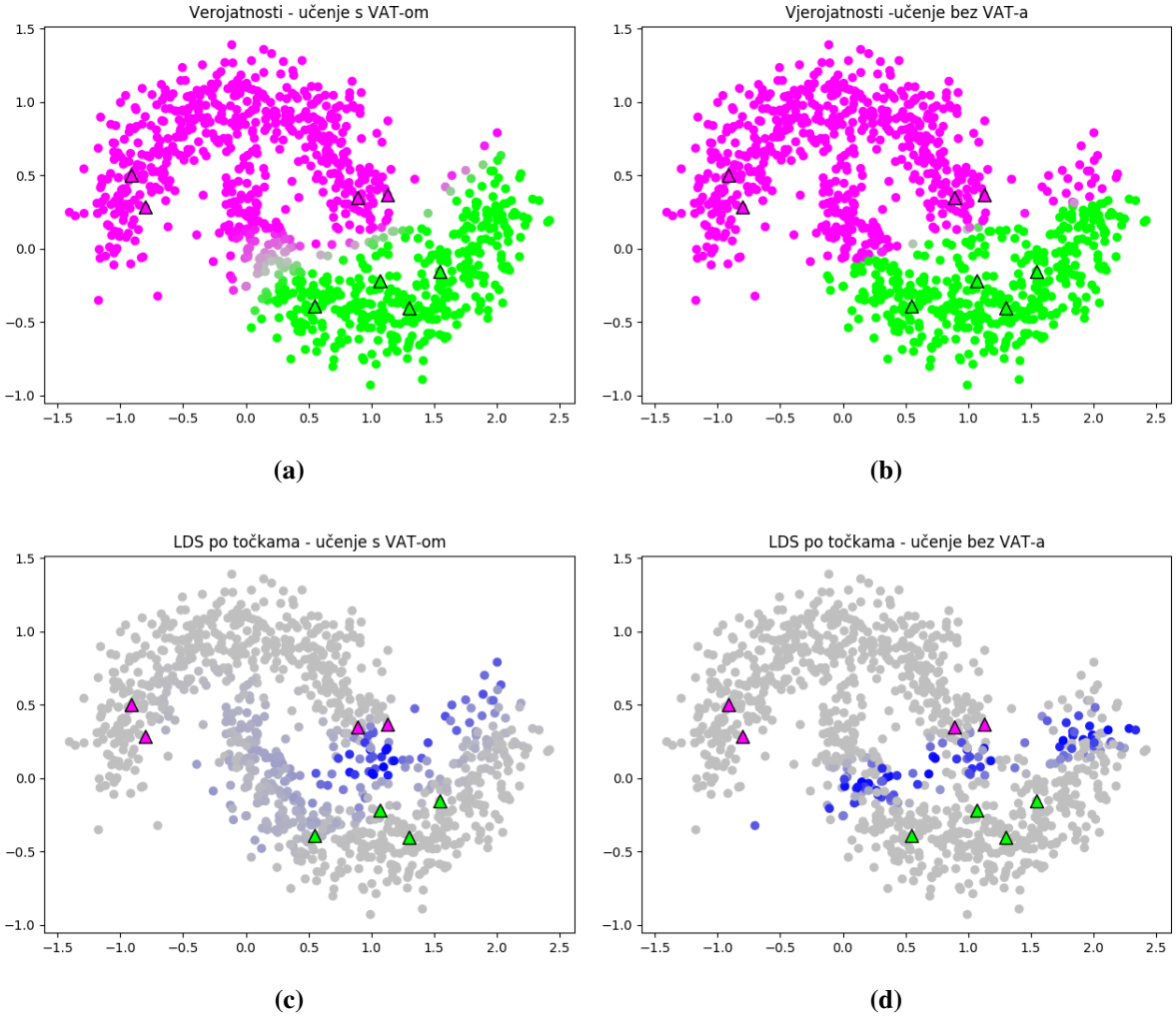
Stanje nakon 100 iteracija učenja



Slika 4.4: Vizualizacija nakon 100. iteracije

Nakon 100 iteracija (4.4), nije došlo do većih promjena u odnosu na 10. iteraciju, granice su još uvijek linearne, s tim da je kod modela s VAT-om ta linearna granica postigla malo veći nagib, te relativni iznos VAT gubitka na lijevoj i desnoj strani donje polukružnice smanjio, dok je onaj, između njih, na desnoj strani gornje polukružnice ostao relativno velik.

Stanje nakon 500 iteracija učenja

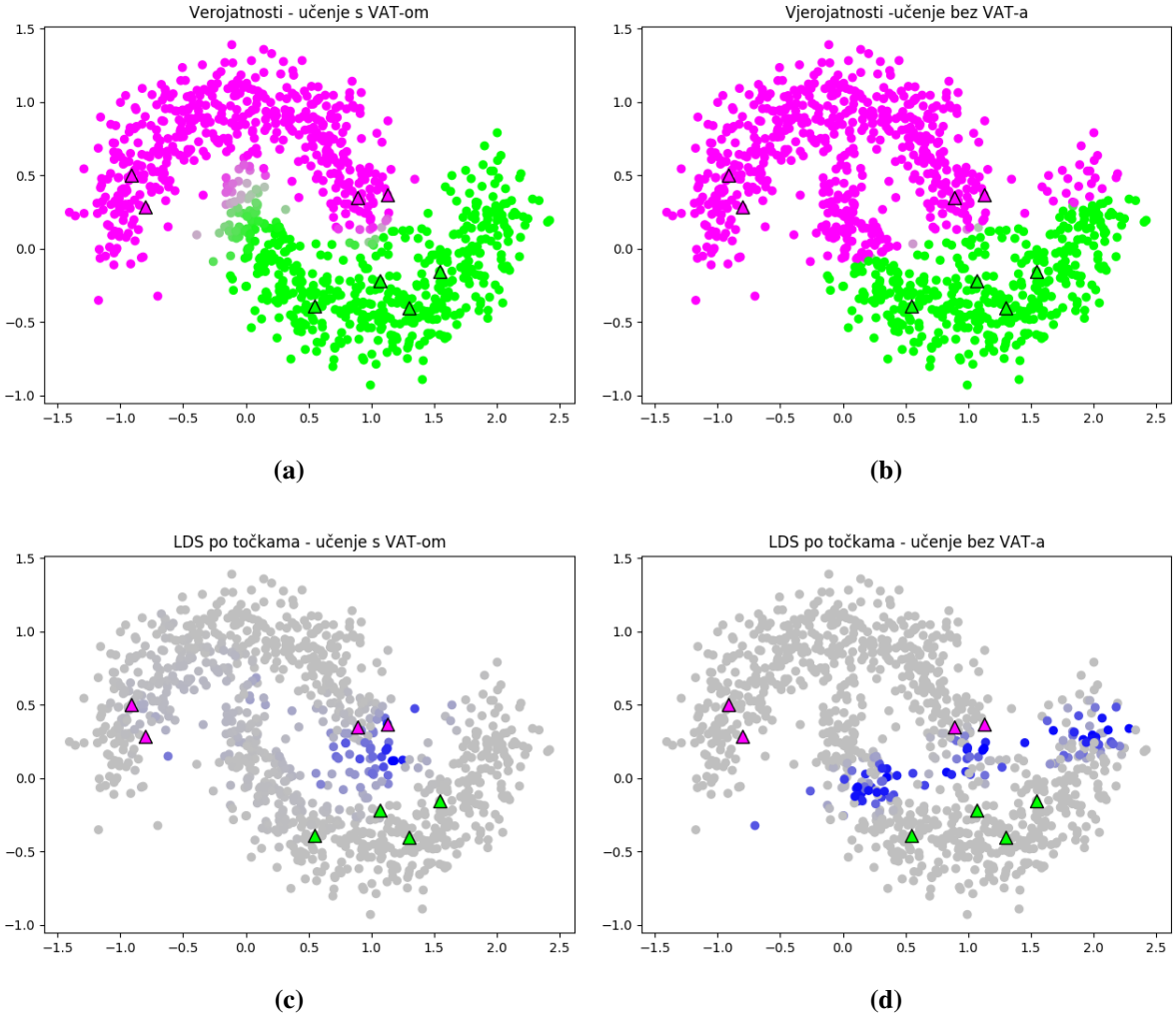


Slika 4.5: Vizualizacija nakon 500. iteracije

Nakon 500 iteracija već se vidi da model s VAT-om počinje polagano učiti polukružni oblik drugog razreda, njegova decizijska granica se pomiče prema njegovim rubovima, što se posebno vidi na desnom kraku. Relativni iznos VAT gubitka na mjestima opisanim kod opisa 100. iteracije se još više smanjio.

S druge strane, na modelu bez VAT-a granica je još uvijek linearna, zbog samog rasporeda označenih primjera, takav model nema nikakvih informacija da bi ta granica mogla biti drugačija. U donjem lijevom dijelu 4.5d vidimo jednu točku koja ima velik iznos VAT gubitka što je posljedica toga da se nalazi blizu njegove linearne decizijske granice. Na 4.5c ta točka, a ni ostale u njezinoj blizini nemaju visok VAT gubitak pa je jasno da decizijska granica modela s VAT-om ne prolazi tuda.

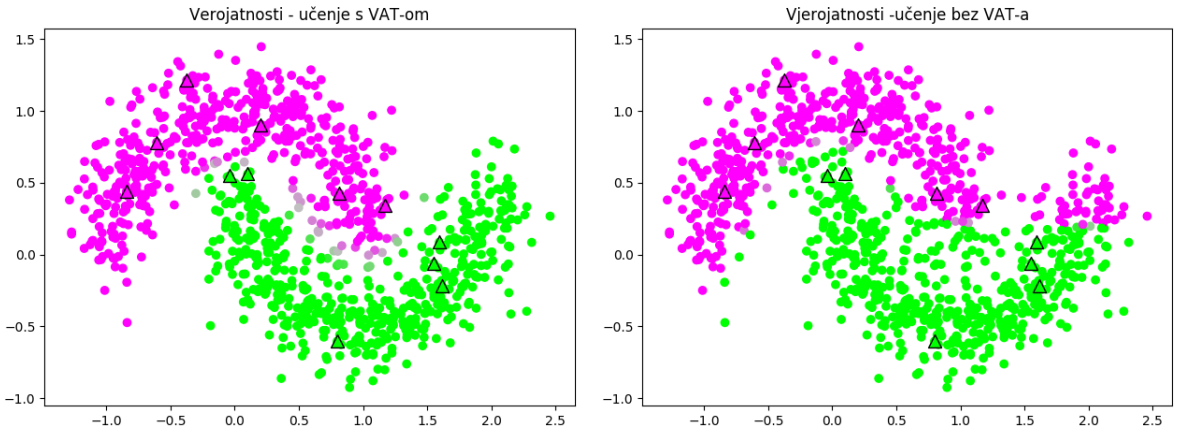
Stanje nakon 1000 iteracija učenja



Slika 4.6: Vizualizacija nakon 1000. iteracije

Nakon 1000 iteracija, model s VAT-om je poprilično dobro uspio naučiti polukružni oblik obaju razreda. Promatrajmo donji polukrug koji predstavlja donji razred. Sva četiri označena primjera se nalaze u sredini polukruga, i iz njih nema nikakvih naznaka da bi taj razred mogao imati polukružni oblik, pa ga stoga model bez VAT-a ne može ni naučiti. No model s VAT-om to uspijeva, i tu riječ *virtualni* iz učenja s virtualnim neprijateljskim primjerima dolazi do izražaja. Iz primjera s oznakama čija se ispravna klasifikacija nauči uz pomoć logističke regresije, uz pomoć virtualnih neprijateljskih primjera, mu omoguće da tu ispravnu klasifikaciju "širi" prema krakovima polukružnice upravo preko neoznačenih primjera. A zbog toga što su ti primjeri neoznačeni govorimo o **virtualnim** neprijateljskim primjerima.

Bitno je naglasiti da je, u ovom slučaju, za demonstraciju djelovanja VAT-a važan raspored označenih primjera. Kad bi označeni primjeri bili ravnomjernije raspoređeni u svim dijelovima obaju razreda, model bez VAT-a bi također naučio nelinearnu granicu, kao na 4.7. No tu također vidimo da je model s VAT-om bolji.



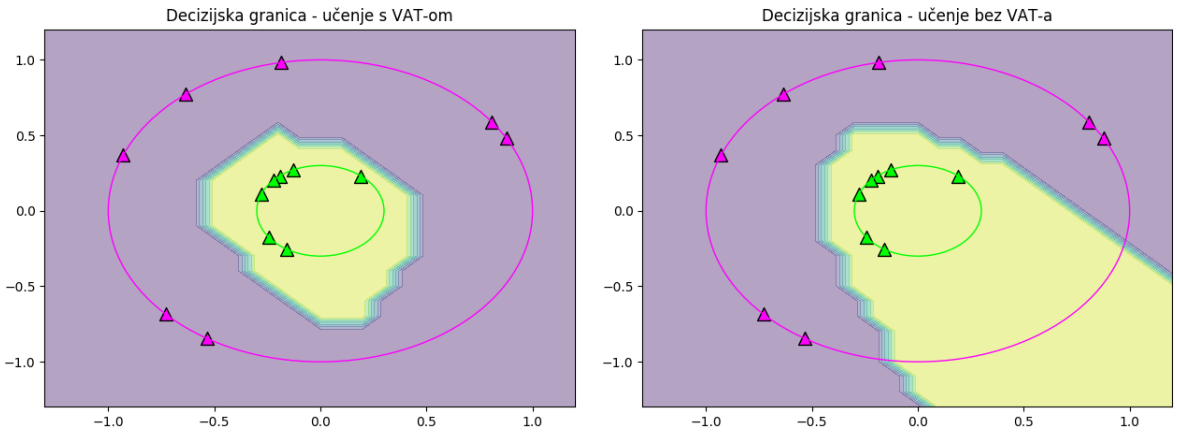
(a) **(b)**

Slika 4.7: Vizualizacija nakon 1000 iteracija

4.2. Skup podataka s koncentričnim elipsama

Sada ćemo koristiti skup podataka sličan onom u prošlom potpoglavlju, ali umjesto pomiješanih polukrugova, imat će dvije koncentrične elipse, i pri tome ćemo koristiti samo označene podatke. Model s VAT-om će pored gubitka logističke regresije, računati i VAT gubitak nad tim istim označenim primjerima, dok će model bez VAT-a imati samo gubitak unakrsne entropije. Dakle ovdje će oba modela koristiti vidjeti iste podatke, i ovdje je cilj vidjeti da VAT može imati i regularizacijski učinak.

Model je isti kao i u prethodnom potpoglavlju, jedan skriveni sloj s 80 neurona, stopa učenja je 0.01, a parametar ϵ je postavljen na 10.0.



(a) **(b)**

Slika 4.8: Vizualizacija decizijske granice kod koncentričnih elipsi

Pomoću funkcije *make_circles* iz biblioteke *scikit-learn* generirali smo skup podataka, po sedam primjera iz vanjske i unutarnje elipse i na njima učili modele. Na slici 4.8 se vidi rezultat vizualizacije, ljubičasto područje označava točke koje bi model svrstao u razred vanjske elipse, a žuto područje one točke koje bi svrstao u razred unutarnje elipse. Učenje bez VAT-a je cijelo područje od sredine unutarnje elipse pa prema donjem desnom kutu označilo kao razred unutarnje elipse. Kod učenja s VAT-om, ljubičasti trokuti su uspjeli preko VAT gubitka uspjeli potaknuti model da cijelu vanjsku elipsu stavi u ljubičasti razred.

5. Eksperimenti na MNIST-u

MNIST (engl. *Modified National Institute of Standards and Technology database*) je skup podataka koji se sastoji od rukom pisanih znamenki od nula do devet. Slike znamenki su crnobijele i fiksne veličine 28×28 piksela i ima ih 70 tisuća. Od toga 60 tisuća slika čine skup za učenje, a 10 tisuća skup za testiranje. MNIST je jedan od najčešće korištenih skupova za učenje i evaluaciju raznih metoda računalnog vida i strojnog učenja jer je relativno jednostavan, sastoji se od malih monokromatskih slika koje imaju 10 razreda, i dovoljno velik. Zato ćemo u ovome radu početi s eksperimentima na njemu.

5.1. Model za MNIST

Model koji ćemo koristiti za eksperimente na MNIST-u bit će potpuno povezan, neće u sebi imati konvolucije. Ulaz u njega će biti slike koje ćemo iz dvodimenzionalnog tenzora 28×28 prikazati kao vektor sa 784 elementa što će biti ulaz u prvi potpuno povezani sloj, a zadnji potpuno povezani sloj će imati 10 neurona jer vršimo klasifikaciju u 10 mogućih razreda. Nakon njega slijedi softmax koji će izlaz pretvoriti u vjerojatnosti pripadnosti određenim razredima.

Model će imati četiri skrivena sloja, čije su veličine redom: 1200, 600, 300, 150.

Kao funkciju nelinearnosti koristit će se rektificirana linearna funkcija tj. *ReLU* poslije svakog potpuno povezanog sloja. Uz to, prije Relu će se polije svakog potpuno povezanog sloja, osim zadnjeg, dodati i normalizacija po grupama (engl. *Batchnorm*).

5.2. Nadzirano učenje

U nadziranom učenju će za sve primjere iz skupa za učenje biti poznate odgovarajuće oznake koje će se koristiti pri optimizaciji modela. Pri tome će funkcija gubitka biti unakrsna entropija između izlaza modela i stvarnih oznaka pretvorenih *one-hot* kodiranjem u vektore. Kao model

Iz teorijskog izvoda u 3.3 možemo pretpostaviti da bi hiperparametar ϵ trebao imati najveći utjecaj na rezultate učenja pa ćemo započeti s njegovom validacijom.

5.2.1. Validacija hiperparametra ϵ

Da bismo odredili optimalnu vrijednost nekog hiperparametra, trebamo trenirati model s različitim vrijednostima toga hiperparametra i gledati za koju vrijednost on postiže najbolje rezultate. Za računanje tih rezultata ne smijemo koristiti ni skup za testiranje ni skup za treniranje, već ćemo koristiti poseban skup za validaciju. Taj skup za validaciju ćemo dobiti tako da ćemo iz skupa za treniranje na slučajan način izdvojiti 1000 slika, pazeći da su svi razredi slika jednako zastupljeni, koje nećemo koristiti pri učenju.

Na tablici 5.1 se nalazi postotak pogrešno označenih primjera na validacijskom skupu uz različite vrijednosti hiperparametra ϵ nakon učenja.

ϵ	Bez VAT-a	0.05	0.1	0.5	1.0	**2.0**	3.0
Pogreška	1%	1%	0.8%	0.6%	0.6%	**0.5%**	0.7%
ϵ	4.0	5.0	6.0	7.0	8.0	9.0	10.0
Pogreška	0.7%	0.7%	0.6%	0.9%	0.9%	1%	0.8%

Tablica 5.1: Pogreška predviđanja na validacijskom skupu pri nadziranom učenju

Najmanja pogreška na validacijskom skupu je 0.5% i postiže se za $\epsilon = 2.0$.

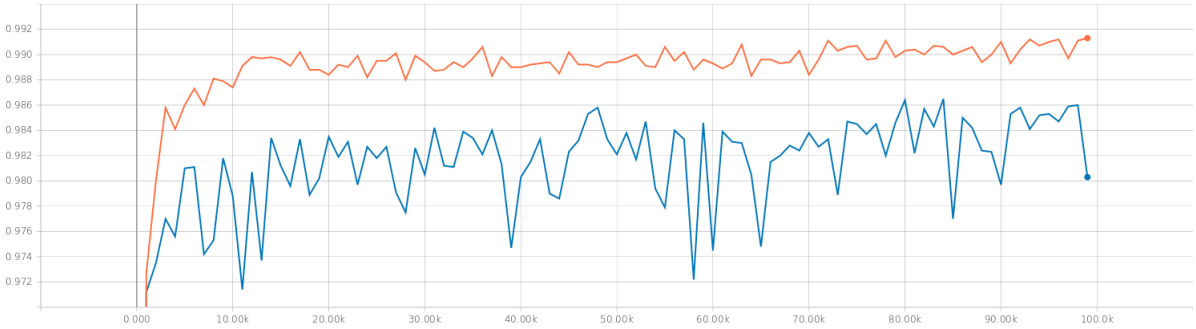
Učenje se provodilo 50 tisuća iteracija, optimizacijski algoritam je bio ADAM uz stopu početnu stopu učenja 0.002. Nakon 25 tisuća iteracija, stopu učenja smo svakih šest tisuća iteracija množili sa 0.7 čime se postigli njezino eksponencijalno smanjivanje.

5.2.2. Rezultati nadziranog učenja

Nakon što smo utvrdili da je za $\epsilon = 2.0$ najmanja pogreška na validacijskom skupu, s njim smo učili model na cijelom skupu za učenje i evaluirali ga na skupu za testiranje. Postigli smo sljedeće rezultate:

Metoda	Pogreška
Bez VAT-a	1.32%
$\epsilon = 2.0$	0.87%
Miyato et al. [8]	0.66%

Tablica 5.2: Pogreška na testnom skupu MNIST-a za nadzirano učenje

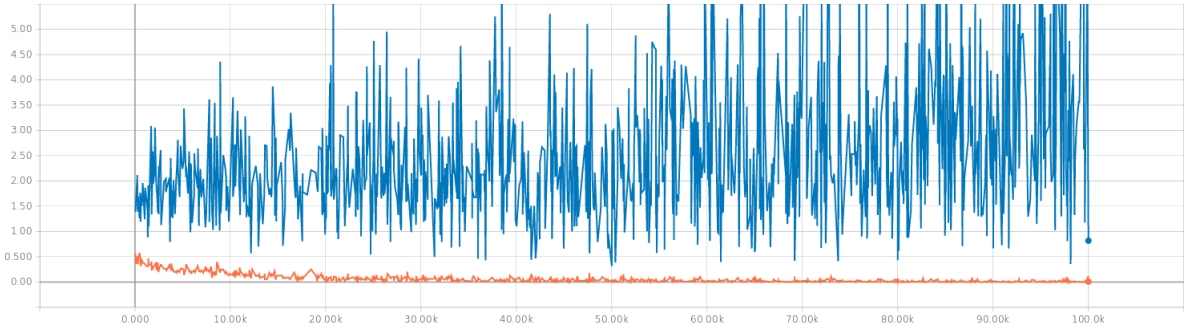


Slika 5.1: Točnost na MNIST-u na testnom skupu tijekom nadziranog učenja

Narančasto - $\epsilon = 2.0$

Plavo - bez VAT-a

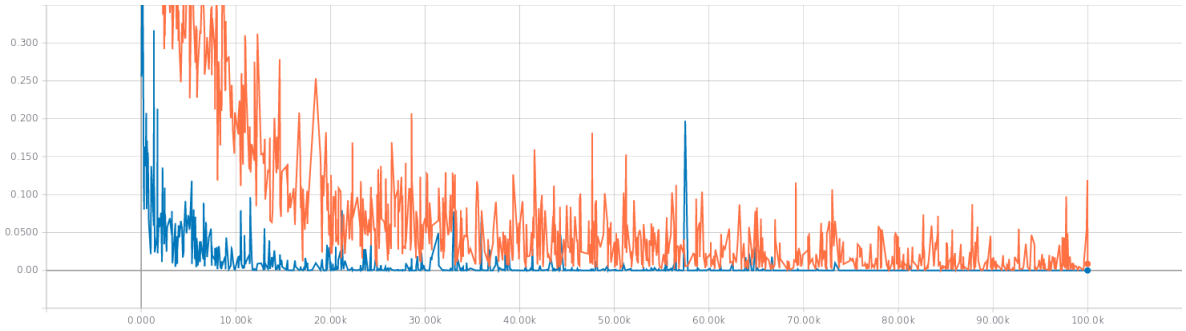
Na slikama 5.2 i 5.3 vidimo VAT i ukupni gubitak tijekom učenja za oba modela.



Slika 5.2: VAT gubitak na skupu za učenje tijekom nadziranog učenja na MNIST-u

Narančasto - $\epsilon = 2.0$

Plavo - bez VAT-a



Slika 5.3: Ukupni gubitak na skupu za učenje tijekom nadziranog učenja na MNIST-u
Narančasto - $\epsilon = 2.0$
Plavo - bez VAT-a

5.3. Polunadzirano učenje

U polunadziranom učenju koristit će se cijeli skup za učenje, ali za razliku od nadziranog, za većinu primjera neće biti poznate oznake. Na slučajan način odabrat ćemo određen broj slika iz skupa za učenje čije ćemo oznake koristiti pri učenju modela tako da ćemo na njima računati gubitak unakrsne entropije. Tome gubitku ćemo pridodati VAT gubitak za čije ćemo računanje koristiti sve slike iz skupa za učenje, i označene i neoznačene.

Da bismo stabilizirali učenje polunadziranog modela, na izlaz svakog skrivenog potpuno povezanog sloja osim zadnjeg dodavat ćemo Gaussov slučajni šum s očekivanjem nula i standardnom devijacijom 0.5 kao i u Miyato et al. [8]. To dodavanje je implementirano kao sloj neuronske mreže.

5.3.1. Validacija hiperparametra ϵ

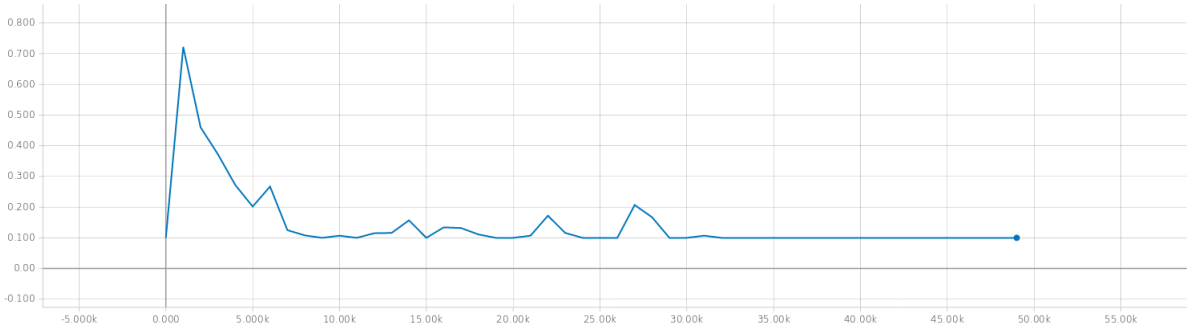
100 označenih primjera

U tablici 5.3 se nalaze postotak pogrešno klasificiranih slika iz skupa za validaciju u eksperimentima sa 100 označenih primjera, za različite vrijednosti hiperparametra ϵ u rasponu od 0.05 do 10.0.

Prvo što možemo uočiti na tablici je da za vrijednosti ϵ manje ili jednake 1.0, dolazi do velike degradacije točnosti u odnosu na učenje bez VAT-a.

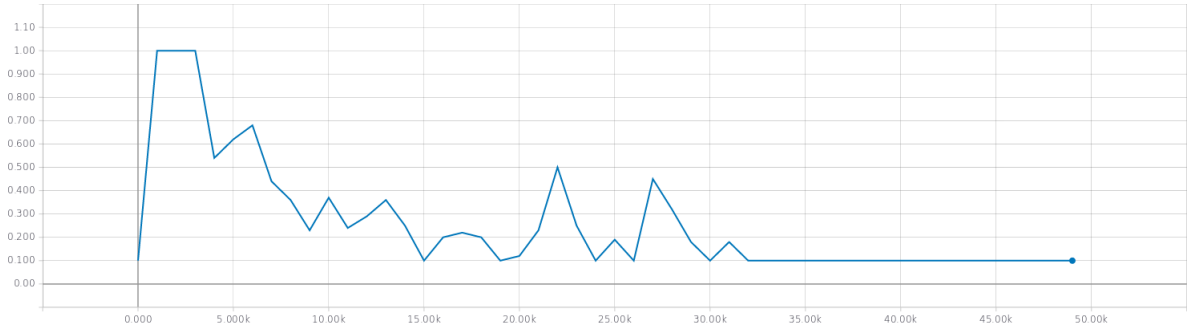
ϵ	Bez VAT-a	0.05	0.1	0.5	1.0	**2.0**	**3.0**
Pogreška	17.0%	35.9%	27.8%	36%	54.9%	**1.4%**	**1.4%**
ϵ	4.0	5.0	6.0	7.0	8.0	9.0	10.0
Pogreška	2.4%	1.5%	2.1%	1.6%	3%	4.8%	3.8%

Tablica 5.3: Pogreška predviđanja na validacijskom skupu za prvi model za 100 označenih primjera



Slika 5.4: Točnost na skupu za validaciju tijekom učenja uz $\epsilon = 0.1$ uz 100 označenih primjera

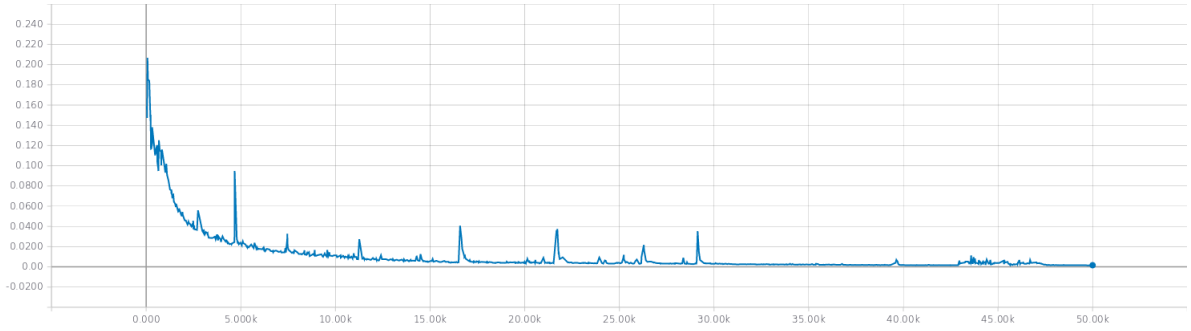
Na grafu 5.4 vidimo kako se točnost na validacijskom skupu mijenja tijekom postupka učenja uz $\epsilon = 0.1$. Uočavamo da na početku učenja postiže najveća točnost, koja se kasnije, uz manje oscilacije, smanjuje na 10%. Iz grafa 5.5 uočavamo da se isto događa i s točnosti na označenim slikama iz skupa za učenje



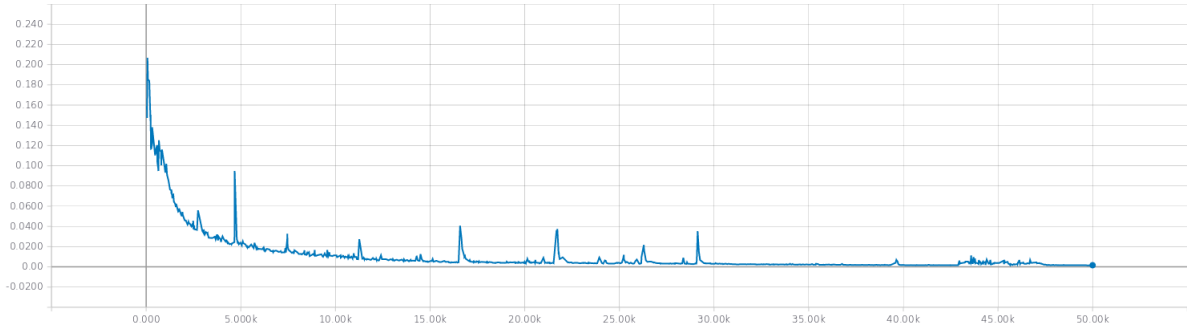
Slika 5.5: Točnost na označenim slikama iz skupa za učenje tijekom učenja uz $\epsilon = 0.1$ uz 100 označenih primjera na prvom modelu

Kako bismo mogli takvo ponašanje mogli objasniti malenom veličinom hiperparametra ϵ, moramo se uvjeriti da je model zaista konvergirao, i da se VAT gubitak,

zajedno s gubitkom unakrsne entropije, smanjio. Kretanje tih gubitaka možemo vidjeti na grafovima 5.6 i 5.7.



Slika 5.6: VAT gubitak na skupu za učenje tijekom učenja uz $\epsilon = 0.1$ uz 100 označenih primjera na prvom modelu



Slika 5.7: Gubitak unakrsne entropije na označenim slikama iz skupa za učenje tijekom učenja uz $\epsilon = 0.1$ uz 100 označenih primjera na prvom modelu

Iz njih vidimo da su se gubitak unakrsne entropije s označenih slika i VAT gubitak konvergirali i približili se nuli. Samim time se isto dogodilo i s ukupnim gubitkom učenja koji je suma njih dvoje. Iz toga možemo zaključiti da premala vrijednost hiperparametra ϵ može negativno utjecati na točnost modela kod polunadziranog učenja.

1000 označenih primjera

Na tablici 5.4 vidimo rezultate validacije za učenje s 1000 označenih primjera.

ϵ	Bez VAT-a	0.05	0.1	0.5	**1.0**	2.0	3.0
Pogreška	7%	6.9%	6.5%	4.7%	**1.2%**	1.3%	1.5%
ϵ	4.0	5.0	6.0	7.0	8.0	9.0	10.0
Pogreška	1.3%	1.4%	1.6%	1.8%	1.8%	2%	2%

Tablica 5.4: Pogreška predviđanja s 1000 označenih primjera na validacijskom skupu za prvi model

Najmanja pogreška od 1.2% se postiže za $\epsilon = 1.0$.

5.3.2. Rezultati polunadziranog učenja

Nakon što smo utvrdili vrijednosti hiperparametra ϵ na kojima se postiže najmanja pogreška na skupu za validaciju, sada ćemo s tim vrijednostima učiti modele na kojima ćemo evaluirati skup za testiranje. Da bismo se mogli uvjeriti da VAT stvarno djeluje, učit ćemo modele na nadzirani način samo nad označenim primjerima i usporediti te rezultate s učenjem s VAT-om i neoznačenim primjerima.

Sva učenja smo proveli s ADAM-om uz početnu stopu učenja od 0.003, nakon 50 tisuća iteracija, počeli smo eksponencijalno smanjivati stopu učenja s faktorom 0.7 svakih deset tisuća iteracija. Sva učenja su imala ukupno 100 tisuća iteracija.

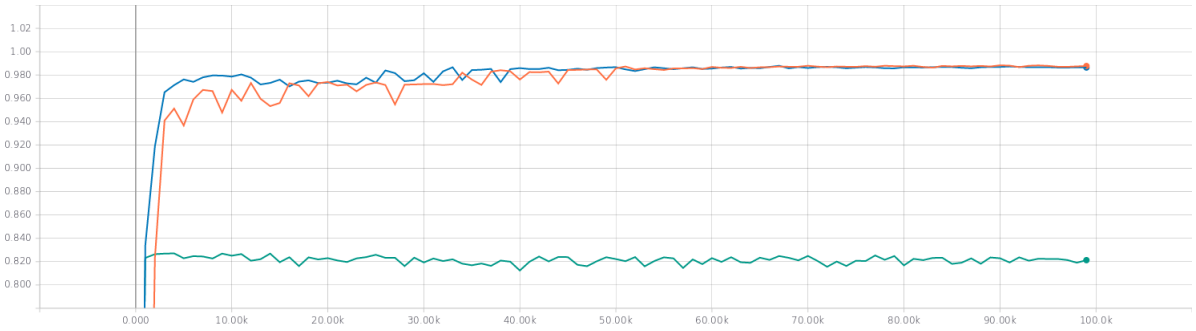
100 označenih primjera

Za 100 označenih primjera utvrdili smo da se najmanja pogreška na validacijskom skupu od 1.4% postiže za vrijednosti $\epsilon = 2.0$ i $\epsilon = 3.0$. Postigli smo sljedeće rezultate:

Metoda	Pogreška
Prvi model, bez VAT-a	17.21%
Prvi model, $\epsilon = 2.0$	1.17%
Prvi model, $\epsilon = 3.0$	1.21%
Miyato et al. [8]	1.36%

Tablica 5.5: Pogreška na testnom skupu MNIST-a za 100 označenih primjera

Na 5.9 vidimo VAT gubitak za modele s VAT-om i model bez VAT-a tijekom učenja kroz iteracije. Može se uočiti kako VAT gubitak, kod modela bez njegove optimizacije,



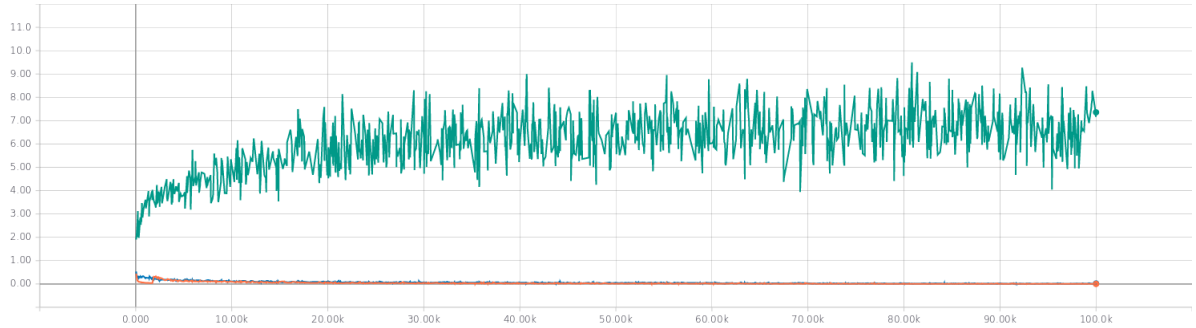
Slika 5.8: Točnost na skupu za testiranje tijekom učenja za 100 označenih primjera

Plavo - Prvi model s $\epsilon = 3.0$

Narančasto - Prvi model s $\epsilon = 2.0$

Zeleno - Prvi model bez VAT-a.

u prosjeku raste tijekom učenja, a kod modela s njegovom optimizacijom on je praktički zanemariv. Na 5.10 je prikazan VAT gubitak samo kod modela s njegovom optimizacijom, kako bi se moglo bolje uočiti njegovo opadanje tijekom postupka učenja.



Slika 5.9: VAT gubitak tijekom učenja za 100 označenih primjera

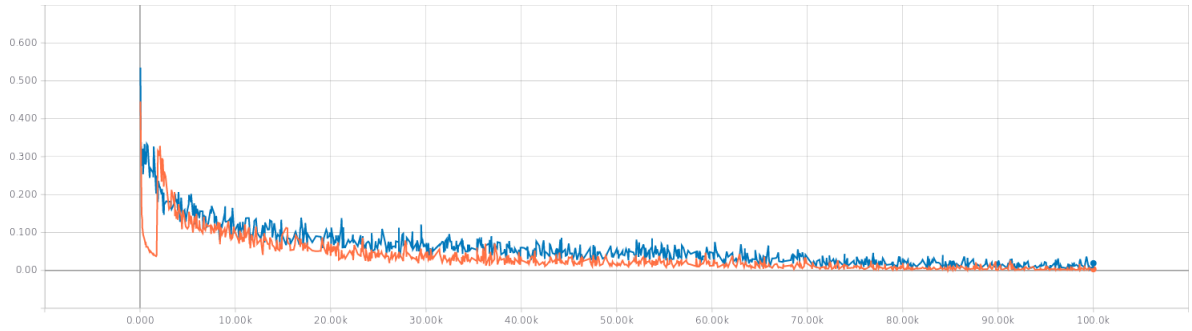
Plavo - Prvi model s $\epsilon = 3.0$

Narančasto - Prvi model s $\epsilon = 2.0$

Zeleno - Prvi model bez VAT-a.

1000 označenih primjera

Za 1000 označenih primjera utvrdili smo da se najmanja pogreška na validacijskom od 1.2% postiže za vrijednost $\epsilon = 1.0$. Postignuti su sljedeći rezultati:



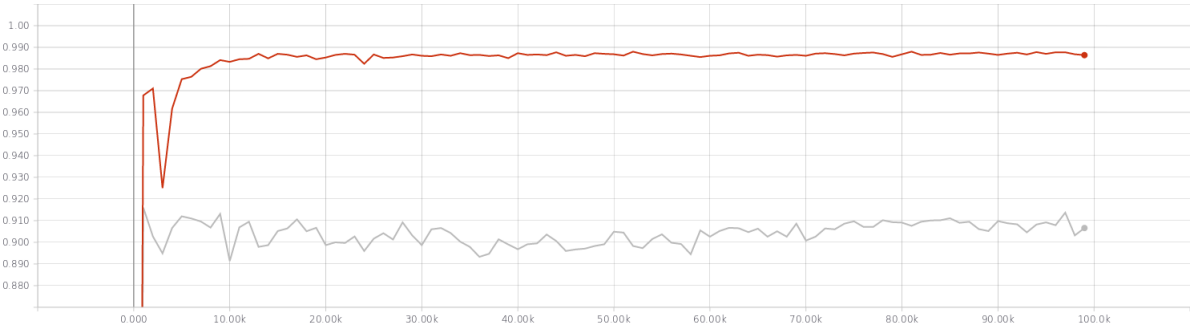
Slika 5.10: VAT gubitak tijekom učenja za 100 označenih primjera

Plavo - Prvi model s $\epsilon = 3.0$

Narančasto - Prvi model s $\epsilon = 2.0$

Metoda	Pogreška
Prvi model, bez VAT-a	8.63%
Prvi model, $\epsilon = 1.0$	1.20%
Miyato et al. [8]	1.27%

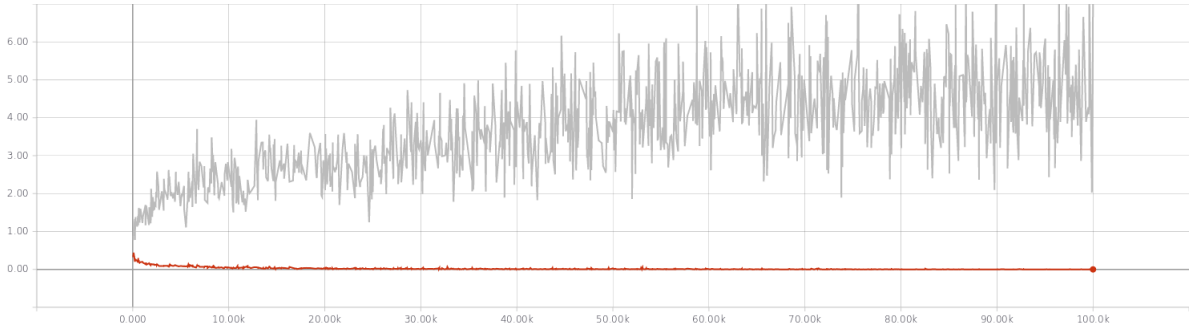
Tablica 5.6: Pogreška na testnom skupu MNIST-a za 1000 označenih primjera



Slika 5.11: Točnost na skupu za testiranje tijekom učenja za 1000 označenih primjera

Crveno - Prvi model s $\epsilon = 1.0$

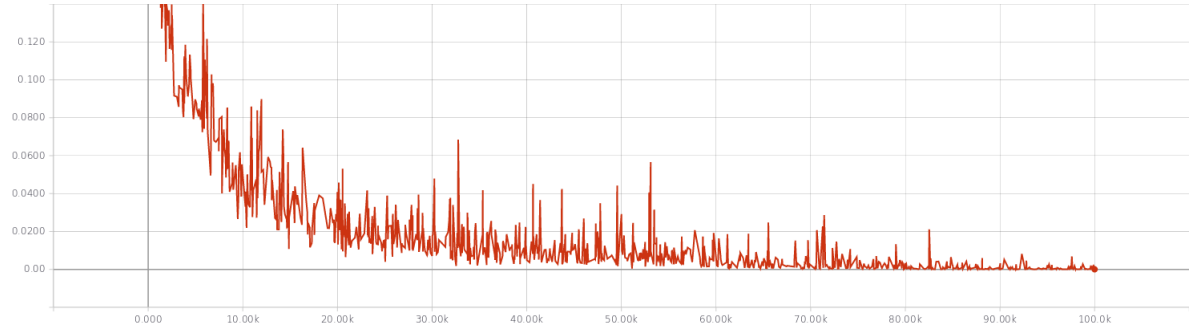
Sivo - Prvi model bez VAT-a.



Slika 5.12: VAT gubitak tijekom učenja za 1000 označenih primjera

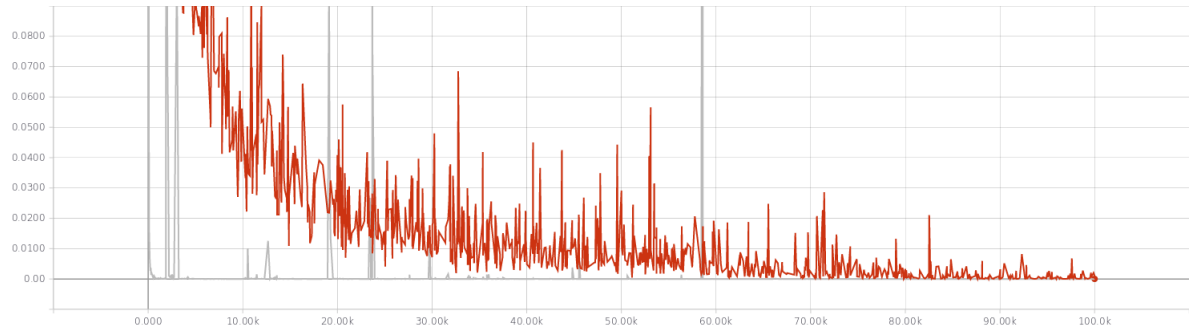
Crveno - Prvi model s $\epsilon = 1.0$

Sivo - Prvi model bez VAT-a.



Slika 5.13: VAT gubitak tijekom učenja za 1000 označenih primjera

Crveno - Prvi model s $\epsilon = 1.0$



Slika 5.14: Ukupni gubitak tijekom učenja za 1000 označenih primjera

Crveno - Prvi model s $\epsilon = 1.0$

Sivo - Prvi model bez VAT-a.

6. Eksperimenti na CIFAR-u i SVHN-u

Nakon što smo u prošlom poglavlju radili eksperimente na MNIST-u, koji je dosta jednostavan skup podataka, sada ćemo ih provoditi na složenijim skupovima podataka, SVHN-u i CIFAR-u, koji dolaze iz stvarnog svijeta. Uz to, za njih ćemo koristiti i konvolucijske modele koji su ključni za rješavanje problema računalnog vida. Ti eksperimenti su prvi korak k primjeni učenja s virtualnim neprijateljskim primjerima na stvarne probleme.

6.1. CIFAR i SVHN

SVHN (engl. *Street View House Numbers*) je skup podataka koji sadrži stvarne slike kućnih brojeva dobivenih iz *Google Street View*. [9] Kao i MNIST, ima deset mogućih razreda jer predstavlja znamenke od nula do devet, no on je bitno zahtjevniji njega zbog svoje velike raznolikosti.

CIFAR je skup podataka koji se sastoji od 60 tisuća RGB slika veličine 32×32 iz deset različitih razreda. Od toga 50 tisuća čine skup za učenje, a ostalih 10 tisuća su skup za testiranje. Za razliku od MNIST-a i SVHN-a, na njemu se ne nalaze znamenke, već objekti iz stvarnog svijeta.

6.2. Modeli za CIFAR i SVHN

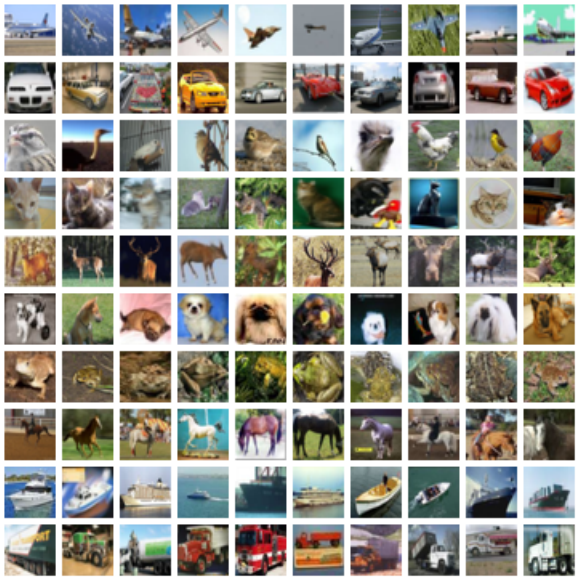
Modeli za CIFAR i SVHN će, za razliku od modela za MNIST, biti konvolucijski. Za oba ćemo imati po dva različita modela koje ćemo zvati *Conv-Small* i *Conv-Large*. Na 6.1 prikazani su njihovi slojevi.

Slika 6.1: Primjer slika iz SVHN-a
preuzeto s http://ufldl.stanford.edu/housenumbers/

Conv-Small za SVHN	*Conv-Small za CIFAR*	*Conv-Large*
Ulazna 32×32 RGB slika		
3×3 konvolucija, 64 filtera	3×3 konvolucija, 96 filtera	3×3 konvolucija, 128 filtera
3×3 konvolucija, 64 filtera	3×3 konvolucija, 96 filtera	3×3 konvolucija, 128 filtera
3×3 konvolucija, 64 filtera	3×3 konvolucija, 96 filtera	3×3 konvolucija, 128 filtera
2×2 *Max pooling*, korak 2		
Dropout, $p = 0.5$		
3×3 konvolucija, 128 filtera	3×3 konvolucija, 192 filtera	3×3 konvolucija, 256 filtera
3×3 konvolucija, 128 filtera	3×3 konvolucija, 192 filtera	3×3 konvolucija, 256 filtera
3×3 konvolucija, 128 filtera	3×3 konvolucija, 192 filtera	3×3 konvolucija, 256 filtera
2×2 *Max pooling*, korak 2		
Dropout, $p = 0.5$		
3×3 konvolucija, 128 filtera	3×3 konvolucija, 192 filtera	3×3 konvolucija, 512 filtera
1×1 konvolucija, 128 filtera	1×1 konvolucija, 192 filtera	1×1 konvolucija, 256 filtera
1×1 konvolucija, 128 filtera	1×1 konvolucija, 192 filtera	1×1 konvolucija, 128 filtera
Global average pooling, $6 \times 6 \rightarrow 1 \times 1$		
Potpuno povezani sloj $128 \rightarrow 10$	Potpuno povezani sloj $192 \rightarrow 10$	Potpuno povezani sloj $128 \rightarrow 10$
Softmax		

Tablica 6.1: Slojevi modela za CIFAR i SVHN

Poslije svakog konvolucijskog sloja slijedi nelinearnost *LReLU* s nagibom 0.1. Ta funkcija je slična funkciji *ReLU*, no razlikuje se za vrijednosti manje od nula. Tada je vrijednost *ReLU* konstantna i jednaka nuli, a *LReLU* je tada linearna funkcija s nagibom koji joj se zadaje kao parameter. A između nelinearnosti i konvolucija se nalazi normalizacija po grupi (engl. *batchnorm*), kao i poslije zadnjeg potpuno povezang



Slika 6.2: Primjer slika iz CIFAR-a
preuzeto s https://www.cs.toronto.edu/ kriz/cifar.html/housenumbers/

sloja. Model *Conv-Small* se razlikuje ovisno za SVHN i MNIST, dok je *Conv-Large* isti za oba.

6.3. Učenje modela

Sve modele u ovome u poglavlju smo učili u sto tisuća iteracija s algoritmom ADAM. Početna stopa za učenje je bila 0.003 koju smo tijekom posljednjih 50 tisuća iteracija eksponencijalno smanjivali s faktorom 0.9.

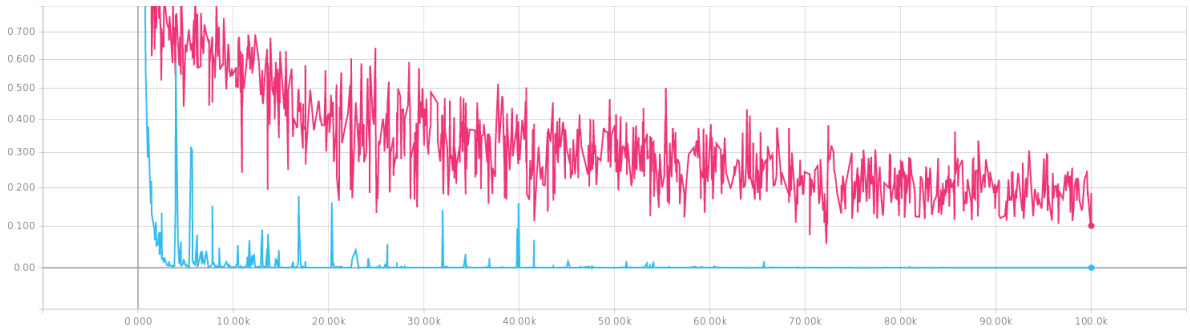
6.4. Rezultati na SVHN-u

Trenirali smo modele *Conv-Small* i *Conv-Large* s 1000 označenih primjera s VAT-om i bez VAT-a. Za hiperparametar ϵ koristili smo $\epsilon = 2.5$ kod *Conv-Small*, i $\epsilon = 3.5$ kod *Conv-Large*. Dobili smo sljedeće rezultate:

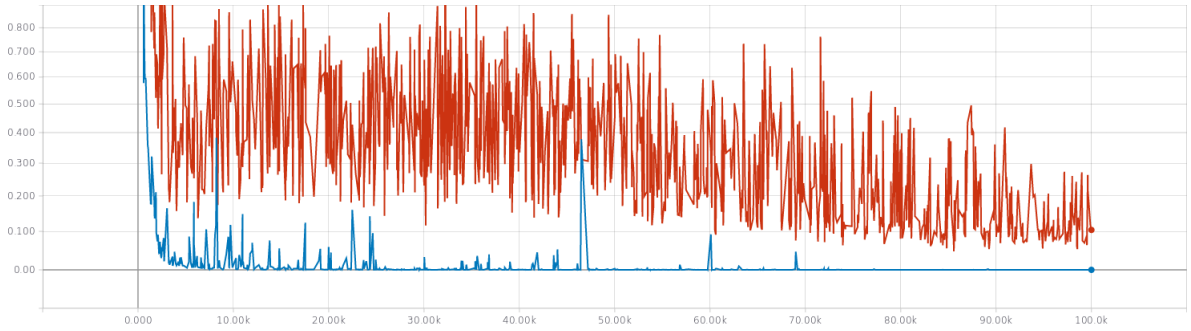
Metoda	Pogreška
Conv-Small, Bez VAT-a	21%
Conv-Small,$\epsilon = 2.5$	11.42%
Conv-Large, Bez VAT-a	20.77%
Conv-Large,$\epsilon = 3.5$	12.68%
Conv-Small,Miyato et al. [8]	8.41%
Conv-Large,Miyato et al. [8]	5.43%

Tablica 6.2: Pogreška na SVHN skupu podataka za 1000 označenih primjera

Iz tablice vidimo da VAT na oba modela smanjuje pogrešku za oko $8 - 9\%$. Nismo uspjeli reproducirati rezultate iz Miyato et al. [8] što je vjerojatno posljedica razlika u implementaciji.



Slika 6.3: Ukupni gubitak tijekom učenja na *Conv-Small* za SVHN
Ljubičasto - $\epsilon = 2.5$
Plavo - Bez VAT-a.



Slika 6.4: Ukupni gubitak tijekom učenja na *Conv-Large* za SVHN

Crveno - $\epsilon = 3.5$

Plavo - Bez VAT-a.

6.5. Rezultati na CIFAR-u

Učeli smo modele *Conv-Small* i *Conv-Large* s 4000 označenih primjera s VAT-om i bez VAT-a. Za hiperparametar ϵ koristili smo $\epsilon = 8.0$ kod *Conv-Small*, i $\epsilon = 10.0$ kod *Conv-Large*. Dobili smo sljedeće rezultate:

Metoda	Pogreška
Conv-Small, bez VAT-a	27.32%
Conv-Small,$\epsilon = 8.0$	20.43%
Conv-Large, bez VAT-a	26.42%
Conv-Large, $\epsilon = 10.0$	19.83%
Conv-Small,Miyato et al. [8]	14.87%
Conv-Large,Miyato et al. [8]	13.15%

Tablica 6.3: Pogreška na CIFAR skupu podataka za 4000 označenih primjera

Kod oba modela, VAT je uspio značajno smanjiti pogrešku na skupu za testiranje. Nismo uspjeli reproducirati rezultate iz Miyato et al. [8] što je vjerojatno posljedica lošije implementacije.

7. Zaključak

U ovom radu proučavali smo primjenu učenja s virtualnim neprijateljskim primjerima na polunadzirano učenje. Na početku smo kratko prikazali općenitu teoriju polunadziranog učenja, te u sklopu toga objasnili dvije njegove metode bliske učenju s virtualnim neprijateljskim primjerima.

U daljnjem dijelu smo prikazali razvoj učenja s neprijateljskim primjerima i preko njega došli do učenja s virtualnim neprijateljskim primjerima. Iz njegova izvoda smo teorijski zaključili da on djeluje tako da poboljšava lokalnu glatkoću uvjetne distribucije oznaka ulaznih primjera $p(y|\vec{x})$ što potiče glatkoću decizijske granice. Vidjeli smo da ima malen broj hiperparametara i da ima relativno malu računsku složenost, za njegov izračun su potrebna dva dodatna unatražna prolaza kroz mrežu.

Da bismo se bolje upoznali s djelovanjem VAT-a, generirali smo dvodimenzionalni skup podataka s dva razreda i provodili učenje na njemu s jednostavnim jednoslojnim modelom. Pri tom smo taj skup vizualizirali tijekom više različitih iteracija procesa učenja, te smo vidjeli kako VAT pomiče oznake razreda kroz neoznačene primjere.

Nakon toga proveli smo eksperimente s njim na skupu podataka MNIST. U nadziranom učenju on je imao ulogu regularizacijske metode te smo njegovim korištenjem smanjili evaluacijsku pogrešku s 1.32% na 0.87%. U polunadziranom učenju provodili smo eksperimente sa 100 i 1000 označenih primjera te smo, u odnosu na modele učene samo na označenim primjerima, smanjili evaluacijsku grešku s 17.21% na 1.17% odnosno 8.63% na 1.27%. S 1000 označenih primjera dobili smo malo lošiji rezultat u odnosu na 100 označenih primjera, što je posljedica varijance rezultata modela. Praćenjem VAT gubitka kroz proces učenja vidjeli smo da kod učenja s VAT-om on smanjuje, za razliku od učenja bez VAT-a, te time potvrdili da je poboljšanje evaluacijske performanse posljedica učenja s virtualnim neprijateljskim primjerima. I u nadziranim i nenadziranim eksperimentima vrijednosti hiperparametra ϵ odabrali smo validacijom na validacijskom skupu.

U zadnjem poglavlju smo prikazali rezultate eksperimenata na skupovima podataka koji dolaze iz stvarnog svijeta, SVHN-u i CIFAR-u. Oni su puno zahtjevniji od MNIST-

a, te smo za njih koristili konvolucijske modele koji danas vrlo široku primjenu. Korištenjem VAT-a uspjeli smo povećati točnost na skupu za testiranje u polunadziranom učenju. Iz toga zaključujemo da je VAT moguće primijeniti i na stvarne probleme.

Budući pravci razvoja išli bi u smjeru primjene učenja s virtualnim neprijateljskim primjerima na semantičku segmentaciju. To je vrlo važan problem u računalnom vidu u kojem se svaki piksel slike klasificira u neki razred. Za nadzirano učenje nužno je označiti svaki piksel slike što dosta otežava dobavljanje označenog skupa za učenje u testiranje zbog čega bi polunadzirano učenje tu moglo imati važnu primjenu. U ovom radu uvjerili smo se VAT može značajno poboljšati rezultate polunadziranog učenja stoga ima smisla proučavati njegov utjecaj na polunadziranu semantičku segmentaciju.

LITERATURA

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[2] Olivier Chapelle, Bernhard Schölkopf, i Alexander Zien, urednici. *Semi-Supervised Learning*. The MIT Press, 2006. ISBN 9780262033589. doi: 10.7551/mitpress/9780262033589.001.0001. URL https://doi.org/10.7551/mitpress/9780262033589.001.0001.

[3] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] Ian J. Goodfellow, Jonathon Shlens, i Christian Szegedy. Explaining and harnessing adversarial examples. U Yoshua Bengio i Yann LeCun, urednici, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

[5] Yves Grandvalet i Yoshua Bengio. Semi-supervised learning by entropy minimization. U François Denis, urednik, *Actes de CAP 05, Conférence francophone sur l'apprentissage automatique - 2005, Nice, France, du 31 mai au 3 juin 2005*, stranice 281–296. PUG, 2005.

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

[7] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. 2013.

[8] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, i Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised

learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1979–1993, 2019. doi: 10.1109/TPAMI.2018.2858821. URL https://doi.org/10.1109/TPAMI.2018.2858821.

[9] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, i Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. U *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, i Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

[11] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, i Rob Fergus. Intriguing properties of neural networks. U Yoshua Bengio i Yann LeCun, urednici, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

[12] Siniša Šegvić. Predavanja iz predmeta duboko učenje. URL http://www.zemris.fer.hr/~ssegvic/du/.