SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2294

POLUNADZIRANO UČENJE S VIRTUALNIM NEPRIJATELJSKIM PRIMJERIMA

Andrija Lonza

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2294

POLUNADZIRANO UČENJE S VIRTUALNIM NEPRIJATELJSKIM PRIMJERIMA

Andrija Lonza

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2294

Pristupnik:	**Andrija Lonza (0036487026)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Polunadzirano učenje s virtualnim neprijateljskim primjerima**

Opis zadatka:

Strogo nadzirani diskriminativni modeli glavni su sastojak mnogih praktičnih primjena računalnog vida. U posljednje vrijeme, veliki uspjeh u tom području ostvaruju metode temeljene na konvolucijskim modelima. Međutim, konvolucijski modeli prikladnog kapaciteta zahtijevaju iznimno veliki broj primjera za učenje. Zbog toga su regularizacijske tehnike koje koriste neoznačene podatke vrlo zanimljivo područje istraživanja. U okviru rada, potrebno je istražiti i ukratko opisati postojeće pristupe učenja diskriminativnih modela koje mogu koristiti i neoznačene podatke. Posebno proučiti tehnike koje potiču glatkoću decizijske granice zahtijevanjem da neoznačeni primjeri imaju što sličniju predikciju kao i njihove perturbirane inačice. Razviti postupak polunadziranog učenja klasifikacijskog modela zasnovan na virtualnim neprijateljskim primjerima. Validirati hiperparametre te prikazati i ocijeniti ostvarene rezultate. Predložiti pravce budućeg razvoja. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

Zahvaljujem se poštovanom profesoru Siniši Šegviću na svoj pruženoj pomoći za vrijeme studija, zahvaljujem mu se na svim odgovorenim pitanjima i razjašnjenim dilemama.

Zahvaljujem se obitelji na bezuvjetnoj podršci i brizi. Spomenimo i prijatelje s kojima sam proveo nebrojne sate spremajući ispite.

SADRŽAJ

1. Uvod

Podaci nad kojima pokušavamo provesti učenje modela jako su važan faktor u algoritmima dubokog učenja. Stoga, algoritme dubokog učenja dijelimo prema podacima na nadzirano, nenadzirano i polunadzirano učenje. Nadzirano učenje je ono koje za svaki ulazni primjer poznajemo njegovu izlaznu oznaku. Algoritam uspoređuje predikciju modela i stvarnu oznaku te razliku propšta u nazad. Takav optimizacijski algoritam nazivamo propuštanje gradijenata unazad. Nenadzirano učenje nema oznaka za ulazne podatke te nam služi kako bi naučili nešto o prirodi podataka koji su nam dostupni. Polunadzirano učenje je između ova dva načina te nam pruža prednosti oba. Dakle, u skupu za učenje nalaze se ulazni podaci koji imaju izlaznu oznaku za ulazne primjere i oni koji nemaju. Naš cilj je postići zadovoljavajuću konačnu statistiku uz što manji broj označenih primjera. Sa polunadziranim učenjem pokušat ćemo riješiti problem nadziranog učenja uz pomoć obilježenih primjera nadopunjenih s karakteristikama koje smo dobili iz neoznačenih primjera. Omjer označenih i neoznačenih primjera je jako važan u odlučivanju je li neka metoda valjana za odabir. Upravo se u tome krije jedna od prepreka koju duboko učenje pokušava premostiti: problem označavanja podataka. Zbog uspjeha u području računalnog vida posebno su aktualne metode koje se temelje na konvolucijskim modelima. Kako bismo postigli zadovoljavajuće rezultate potreban nam je prikladan kapacitet konvolucijskog modela koji zahtjeva iznimno veliki broj primjera za učenje. Stoga su nam u podjeli algoritama dubokog učenja prema podacima posebno zanimljiva polunadzirana učenja. Zbog karakteristike polunadziranog učenja da nam ne treba potpuni skup podataka za učenje biti označen, regularizacijske tehnike koje koriste neoznačene podatke vrlo su zanimljivo područje istraživanja.

Naime, podatke označavaju ljudi, što je problematično iz više razloga. Ljudski resurs košta, označavanje zahtjeva vremena te je upitna ljudska procjena u označavanju kao i konzistentnost u procjeni. Stoga je bolje takav faktor zamijeniti metodom ili algoritmom koji će podatke obrađivati u skladu s našim zahtjevima. Naveo bi ovdje primjere iz realnog života kako bi potkrijepio navedene tvrdnje. Primjeri dolaze od ljudi koji su bili zaposleni na označavanju podataka u jednoj firmi koja se bavi ra-

zvojem algoritama dubokog učenja – određivanjem dobi na slici. Zaposlenici su imali zadatak određivanja dobi osoba na slici. Tvrdili su da im nakon nekog vremena dođe ista slika na procjenu te da svaki puta dodjele drugu dob istoj osobi, jer je toga trenutka drukčija osobna procjena procjenjivača. Poslodavci nisu bili zadovoljni s učinkom pojedinih zaposlenika pa su im uveli dnevne kvote za slike koje moraju označiti. Konstantno im je nedostajalo primjera za provođenje eksperimenata. Ovaj primjer je kratka ilustracija realnih problema s kojima se susrela tvrtka u razvijanju projekta. Uzmimo primjer problema semantičke segmentacije u kojem svakom pikselu na slici mora biti određena klasa. Kako bismo provodili nadzirano učenje nad problemima semantičke segmentacije označivači trebaju označiti svaki pojedini piksel na slici. Dakle, više nije problem označavanja generalnog objekta koji slika predstavlja nego svakog pojedinog pa i najmanjeg detalja na slici.

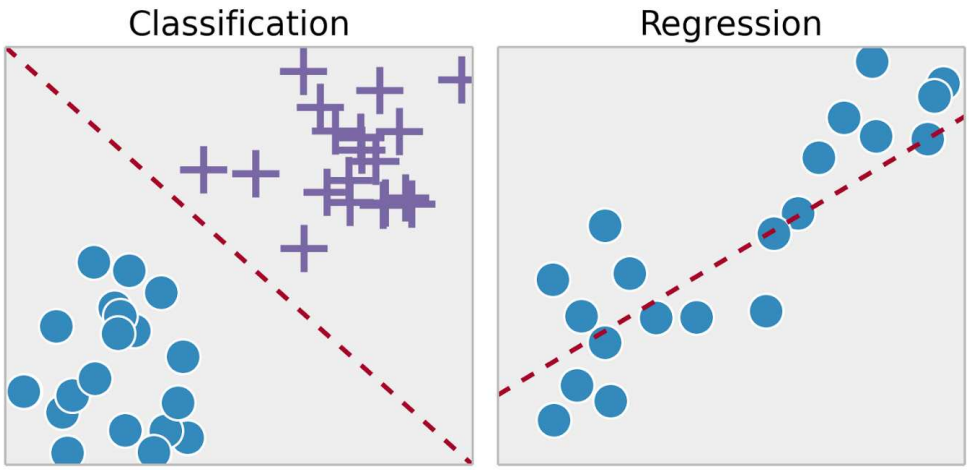
S obzirom da označavanje primjera postaje otežavajuća okolnost napretku rezultata iz već navedenih razloga, algoritmi polunadziranog učenja postaju sve popularniji. Učenje s virtualnim neprijateljskim primjerima pripada skupini polunadziranih algoritama dubokog učenja čija je glavna blagodat poticanje glatkoće decizijske granice. To postižemo tako da neoznačeni primjeri imaju što sličniju predikciju kao i njihove izmijenjene inačice perturbirane u smjeru najveće promjene izlaza. VAT je prikladan za korištenje na problemima kada nemamo oznake klasa za pojedine slike te se smatra regularizacijskom metodom.

Uzevši u obzir ove činjenice možemo doći do zaključka da označavanje podataka postaje osobito skupo, zahtjevno i dugotrajno te napori uloženi u istraživanja tehnika koje ne koriste označene podatke su opravdani.

2. Vrste učenja

Vrste učenja modela definirane su podatcima koje imamo na raspolaganju za rješavanje problema, kao i prirodom problema koju pokušavamo riješiti. Podatci koje koristimo za učenje je skup primjera određene dimenzionalnosti, za koje svaka dimenzija odgovara nekoj veličini s kojom smo odlučili predstaviti ulazne podatke. Npr. ako su ulazni podaci slike, to su vrijednosti pixela (0-255), za tri boje RGB, preko čitave širine i dužine slike. Dakle, ukupno $image\_height \cdot image\_width \cdot 3$. Ako pokušavamo odrediti cijenu nekretnina u obzir možemo uzeti veličine kao što su starost nekretnine i udaljenost od centra grada. U ovom primjeru imamo dvije dimenzije, po jednu dimenziju za svaku od veličina. Ulazni primjeri moraju imati izlaznu vrijednost koju poprimaju kada te vrijednosti uvrstimo u funkciju ili oznaku klase kojoj pripadaju ako podatke uvrštavamo u klasifikator. Ako su nam poznate oznake klasa kojima primjeri pripadaju takav skup primjera nazivamo označenim skupom primjera te nam je omogućeno nadzirano učenje. Ako podaci nisu označeni potrebno ih je označiti što uglavnom zahtjeva ljudski resurs sa znanjima u području problema koji pokušavamo riješiti. Takav način označavanja vrlo često rezultira velikim troškovima. Međutim postoje koncepti poput nenadziranog i polunadziranog učenja koji nam mogu pomoći kod ovakvih problema.

Zadatke koje rješavamo u stojnom ili dubokom učenju općenito možemo podijeliti na klasifikacijske i regresijske. Kod klasifikacijskih problema pokušavamo predvidjeti diskretnu vrijednost, dakle izlaz iz našeg modela bi mogla biti vjerojatnost da pojedini primjer pripada klasi koja je označena s diskretnom vrijednosti. Primjer klasifikacijskih problema bi bile slike rukom napisanih znamenki kao one u skupu primjera MNIST koji se vrlo često koristi u edukativne svrhe za treniranje modela. S obzirom na to da se radi o znamenkama 0-9, naš model bi na izlazu imao 10 klasa, te bi za ulaznu sliku predvidio kojoj klasi ona pripada. Regresijski problemi su kontinuirane prirode. Izlaz iz našeg modela bi bila kontinuirana vrijednost, odnosno aproksimacijska funkcija kojom pokušavamo oponašati ponašanje podataka. Primjer problema kojeg bi mogli riješiti regresijom bi bio predviđanje kretanja količine prihoda u odnosu na starosnu dob.



Slika 2.1: Grafički prikaz klasifikacije i regresije [4]

Slika 2.1 s lijeve strane prikazuje klasifikaciju koja s linearnom granicom odvaja dvije skupine primjera koje su prirodno grupirane na istom području, dok je na desnoj strani primjer regresije koja s afina funkcijom pokušava aproksimirati ponašanje primjera te predvidjeti njihovu vrijednost. Klasifikacijom se može razvrstati primjere u veći broj klasa, kao što regresijom možemo oponašati nelinearne raspodjele primjera kao što su polinomijalne raspodjele. U oba slučaja složenost modela raste.

Podjela učenja s obzirom na podatke [3] [4]:

1. Nadzirano učenje
2. Nenadzirano učenje
3. Polunadzirano učenje
4. Učenje pojačanjima

2.1. Nadzirano učenje

U nadziranom učenju oznaka ulaznih primjera je poznata - poznajemo distribuciju $p(y|x)$ - te naš model nakon učenja mora biti u stanju predvidjeti ispravan izlaz y za ulazni primjer x kojeg nikada nije vidio. Ako model to može nad velikom količinom nepoznatih ulaznih primjera onda kažemo da model dobro generalizira. Na ulaz modela dovodimo podatke te vršimo predikciju, na temelju razlike predikcije modela i stvarno distribucije $p(y|x)$ algoritmom propuštanja gradijenata unazad (engl. backpropagation) ažuriramo parametre modela (težine) te tako učimo model. Nadzirani algoritmi učenja pokušavaju modelirati odnose i ovisnosti između ciljanog izlaza i ulaznih primjera pomoću značajki modela tako da možemo predvidjeti izlazne vrijednosti za nove podatke na temelju onih odnosa koje je naučio iz prethodnih skupova podataka [3].

Dakle, ključan pojam u nadziranom učenju je označen skup podataka, na temelju kojeg možemo vršiti nadzor nad učenjem našega modela. Uspoređujemo odstupanje između predikcije našega modela i stvarne oznake modela.

Algoritmi ovoga tipa:

1. Linearna regresija
2. Stroj potpornih vektora (SVM)
3. Stabla odluke
4. Naivni Bayes
5. Algoritam k - najbližih susjeda

2.2. Nenadzirano učenje

U nenadziranom učenju oznake za dane ulaze su nepoznate [4]. U ovoj vrsti učenja nemamo kako nadzirati proces učenja podataka jer ne poznajemo distribuciju $p(y|x)$. Ovakva vrsta učenja nam donosi nova znanja o podacima nakon što algoritam nauči uzorke u podacima. Ovakvi algoritmi su osobito korisni kada stručnjak ne zna što točno tražiti u podaci i kakva apriorna znanja primijeniti. Ovo su algoritmi strojnog učenja koji se uglavnom koriste u detekciji uzoraka i deskriptivnom modeliranju [3]. Međutim, ovdje nema izlaznih kategorija ili oznaka na temelju kojih algoritam može pokušati modelirati odnose [3]. Ovi algoritmi primjenjuju različite tehnike kako bi dobili smislen uvid u podatke te ih što bolje opisali. Dakle, temeljna karakteristika ovakve vrste učenja jest da baratamo s neoznačenim podacima, već ovim algoritmima pokušavamo doznati više podataka o karakteristikama primjera. Nenadzirano učenje ne uključuje kontrolu procesa učenja. Ako je glavna svrha nadziranog učenja da znate rezultate i trebate razvrstati podatke, tada u slučaju nenadziranih algoritama strojnog učenja željeni rezultati su nepoznati i tek trebaju biti definirani.

Najčešća nenadzirana metoda učenja je analiza klastera, koja se koristi za provođenje analize nad podacima kako bi se pronašli skriveni obrasci ili ustanovilo na koji način bi se podaci trebali grupirati. Klasteri se modeliraju korištenjem mjera sličnosti kao što su euklidska ili vjerojatnosna udaljenost.

Algoritmi ovoga tipa su:

1. Hijerarhijsko grupiranje
2. Gaussova mješavina
3. Skriveni Markovi modeli
4. Generativni modeli - GAN i VAE
5. Normalizacijski tok

Nenadzirane metode učenja koriste se u bioinformatici za analizu sekvenci i genetsko grupiranje te u medicini za semantičku segmentaciju slika [1].

2.3. Polunadzirano učenje

Polunadzirano učenje jest kombinacija nadziranog i nenadziranog učenja te je kao i prijašnje vrste učenja uvelike određeno skupom primjera. Skup podataka se sastoji od skupa označenim i skupa neoznačenih primjera. Strategije polunadziranog učenja se baziraju na proširenjima nenadziranog ili nadziranog učenja kako bi u proces uključile informacije specifične za te paradigme [14]. Primjerice, proces polunadziranog učenja bi mogao ujediniti pronalaženje uzoraka u podacima ili prirodno klasteriranje iz područja nenadziranog učenja s učenjem prave distribucije izlaza u odnosu na primjer na ulazu u mrežu $p(y|x)$. Polunadzirano učenje obuhvaća nekoliko različitih postavki, uključujući [14]:

1. **Polunadzirana klasifikacija:** to je nadogradnja na nadziranu klasifikaciju. Koristi označene i neoznačene skupove podataka. Skup podataka za trening se sastoji od N_l označenih primjera $\{(x_i, y_i)\}_{i=1}^{N_l}$ i N_{ul} neoznačenih primjera $\{(x_j)\}_{j=1}^{N_l+N_{ul}}$. Krenuvši od ovakvog pristupa pokušavamo postići da imamo više neoznačenih primjera nego označenih, a da nam klasifikacija ostane jedna ili bolje nego u slučaju nadziranog učenja. Idealno, želimo da broj neoznačenih primjera bude puno veći nego onih označenih, $N_{ul} >> N_l$. Cilj polunadzirane klasifikacije je istrenirati klasifikator f na primjerima iz oba skupa takav da bude bolji od klasifikatora treniranog na skupu označenih primjera [14].

2. **Ograničeno grupiranje** je proširenje nenadziranog učenja. Informacije s kojima raspolažemo za treniranje modela su: skup primjera za treniranje veličine N koji se sastoji od neoznačenih instanci $\{(x_i)\}_{i=1}^{N}$ te "nadziranih informacija" o grupama. Na primjer takva informacija može biti da primjeri x_i, x_j moraju ili ne moraju biti u istoj grupi. Ovakva ograničenja se nazivaju "ograničeno grupiranje", a cilj ovakvog pristupa je dobivanje boljih grupa nego što ih dobivamo grupiranjem postupkom nenadziranim učenjem.

Proučavanje polunadziranog učenja motivirano je s dva čimbenika: njegovom praktičnom vrijednošću u izgradnji boljih algoritama i njegovom teorijskom vrijednošću u razumijevanju učenja [14]. Polunadzirano učenje ima veliko značenje u praktičnom smislu jer u većini slučajeva imamo manjak označenih podataka. Kao što je već istaknuto u ovom radu, oznaka pripadnosti primjera klasi se može teško dodijeliti što iz razloga troška stručnih ljudskih znanja, tako i zbog sporosti. Iako u nekim područjima možemo prikupiti puno podataka za učenje njih je iz spomenutih razloga nekada nemoguće označiti. Nasuprot ovoj tezi stoji polunadzirano učenje koje nam omogućava

da velik dio skupa za učenje ostane neoznačen. Pristup polunadziranim učenjem je superioran drugim oblicima jer ujedinjuje označene i neoznačene primjere te postigne rezultate jednake ili (nadamo se) bolje od nadziranog učenja, ali s manjim brojem označenih primjera. Dakle, možemo reći da bi s ovim pristupom riješili problem troškova ljudskih resursa. Primjer polunadziranog učenja je VAT - virtualno neprijateljsko učenje.

2.3.1. Tehnike polunadziranog učenja

Ukratko ćemo predstaviti metodu polunadziranog učenja koju nazivamo pseudooznačavanje. Tehnika pseudooznačavanja predstavljena je u radu [8]. Imamo određenu količinu označenih podataka s kojima učimo model, možemo to nazvati prednaučiti model. S naučenom distribucijom $p(y|x, \theta)$ modelu dajemo neoznačeni dio skupa primjera te im dodjeljujemo oznake. Tako dodijeljene oznake nazivamo pseudooznake,a označavamo ih sa $\tilde{y}$. S tako pseudooznačenim primjerima i označenim primjerima provodimo iteraciju učenja te postupak pseudooznačavanja provedemo ponovno. Što više iteracija provedemo sa pseudooznačenim primjerima to model nad njima više uči. Gubitak ovakvog učenja definiramo kao:

$$L = \frac{1}{N_l} \sum_{x_i \in D_l} L(x_i, y_i) + \alpha(t) \cdot \frac{1}{N_{ul}} \sum_{x_j \in D_{ul}} L(x_j, y_j) \tag{2.1}$$

Funkcija se sastoji od gubitka koji računamo nad označenim primjerima te gubitka koji računamo nad neoznačenim primjerima svaki gubitak pridonosi ukupnom gubitku kao prosjek svih primjera. Funkcijom $\alpha(t)$ reguliramo utjecaj neoznačenih primjera na ukupni gubitak. Na početku ne želimo veliki utjecaj pseudogubitka jer model prvo želimo naučiti da ispravno klasificira označene primjere te postoji mogućnost od zapinjanja u lokalnom optimumu. Na ovakav način provodimo proces polunadziranog učenja pseudooznačavanjem. Sličan princip koristi i metoda opisana u ovome radu koja se kao i pseudooznačavanje temelji na pretpostavci o klasterima. Također ćemo računati kojim klasama primjeri pripadaju samo što ćemo koristiti virtualnu neprijateljsku perturbaciju kako bi zagladili decizijsku granicu te poboljšali klasifikaciju.

Tehnika koju valja spomenuti je minimizacija entropije [7]. Ovako metoda temelji se na pretpostavci da ulazni primjeri s malom prostornom udaljenošću pripadaju istoj grupi - što nazivamo pretpostavkom o klasterima. Ova funkcija se dodaje na funkciju gubitka.

$$H(Y|X) = -\sum_{x}^{X} p(x) \sum_{y}^{Y} p(y|x) \cdot log(p(y|x) \tag{2.2}$$

2.4. Učenje pojačanjima

Učenjem s pojačanjima postupak učenja se odvija u okruženje gdje se obučava metodom pokušaja i promašaja [4]. Algoritam uči pomoću prijašnjih iskustava te pokušava postići najveću moguću točnost za donošenje ispravnih odluka na temelju primljenih povratnih informacija. Cilj metode je primjena opažanja prikupljenih iz interakcije s okolinom kako bi se poduzele mjere koje bi maksimizirale nagradu ili minimizirali rizik [3]. Model koji obučavamo učenjem pojačanjima nazivamo agent, a učenje se odvija iterativno pomoću povratnih informacija iz okoline. Na temelju dosadašnji iskustava prikupljenih povratnim informacija iz okoline učimo agenta da odredi najbolje moguće ponašanje u određenom kontekstu. Potrebna je jednostavna povratna informacija kako bi agent mogao naučiti kako se ponašati u okolini u kojoj se nalazi - to je poznato kao signal pojačanja [3].

Koraci algoritma učenja pojačanjima [3]:

1. Agent promatra ulazno stanje
2. Funkcija odlučivanja koristi se da bi agent izvršio akciju
3. Nakon što je akcija izvršena, agent dobiva nagradu ili pojačanje iz okoline
4. Podaci o paru stanja-akcije o nagradi se pohranjuju

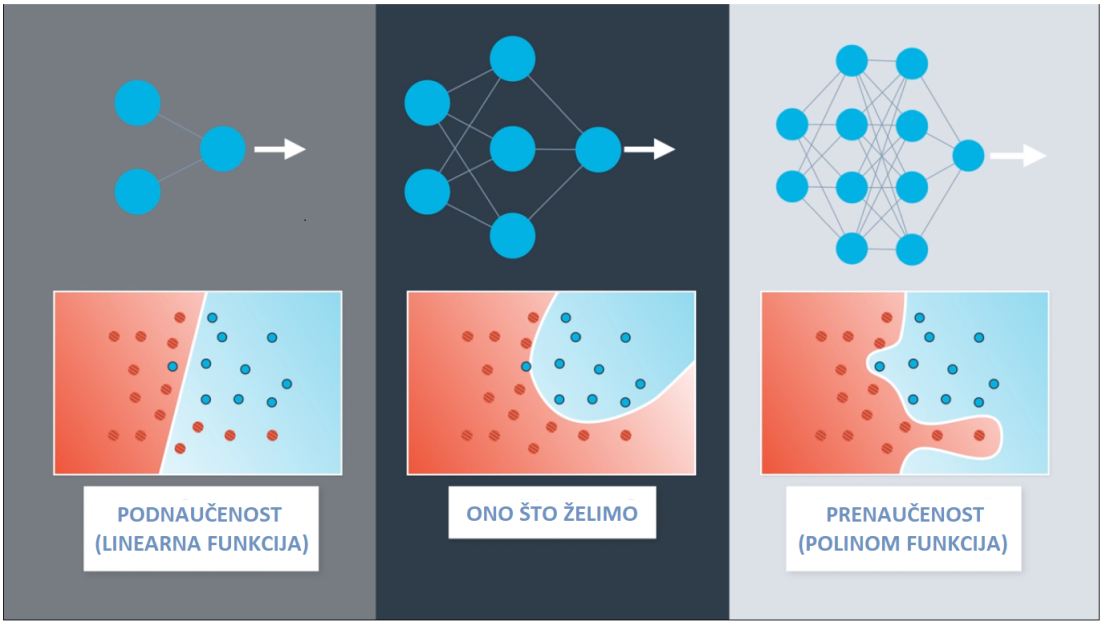
Algoritmi ovoga tipa [3]:

1. Agent promatra ulazno stanje
2. Funkcija odlučivanja koristi se da bi agent izvršio akciju
3. Nakon što je akcija izvršena, agent dobiva nagradu ili pojačanje iz okoline

3. Učenje s virtualnim neprijateljskim primjerima

Učenjem dubokih modela susrećemo se s problemima podnaučenosti i prenaučenosti. Podnaučenost i prenaučenosti su dva suprotstavljena problema. Podnaučenost se događa kada model nije dovoljno naučio iz dostupnih podataka. Uzrok ovoga problema može biti nedovoljna količina podataka - koja za posljedicu ima da nemamo iz čega naučiti generalne uzorke prema kojima se podaci ravnaju, premali broj iteracija učenja modela te mali kapacitet modela. Suprotno ovome problemu, imamo problem prenaučenosti. Prenaučenost se događa uslijed prevelikog prilagođavanja modela skupu za učenje. Uzroke možemo pronaći u prevelikom broju iteracija učenja modela ili prevelikom kapacitetu - odnosno prevelikoj složenosti modela. Velika pogreška na skupu za učenje kao i skupu za testiranje obilježja su podnaučenosti, suprotno tome prenaučenost ima obilježja male pogreške na skupu za učenje, ali velike na skupu za testiranje. Rješavanje problema podnaučenosti je jednostavno, a ono se nalazi u prikupljanju većega skupa podataka za učenje i/ili povećanju kapaciteta mreže. Međutim, sve i da izbjegnemo problem podnaučenosti i dalje nam ostaje problem prenaučenosti. Regularizacija je proces uvođenja dodatnih informacija kako bi se upravljalo ovim neizbježnim jazom između greške u učenju i greške u testiranju [11]. Klasične regularizacije koriste jednostavne norme (npr. $L1$ ili $L2$) kako bi smanjile vrijednosti naučenih parametar - "pritežemo" vrijednosti parametara prema nuli - kako bi smanjili prilagođenost skupu za učenje. Regularizacija koju proučavam u ovom radu identificira smjer u kojem je ponašanje klasifikatora najosjetljivije te je primjenjiva na polunadzirano učenje. To postižemo tako da neoznačeni primjeri imaju što sličniju predikciju kao i njihove izmijenjene inačice perturbirane u smjeru najveće promjene izlaza.

Slika 3.1 podijeljena je u tri dijela. Lijevi dio prikazuje kako bi izgledala decizijska granica kada bi model bio podnaučen, srednji dio prikazuje kako bi željeli da decizijska granica izgleda za prikazane primjere. Dok desni dio prikazuje kako izgleda prenaučenost na skupu za učenje. Možemo vidjeti da je funkcija koja predstavlja



Slika 3.1: Grafički prikaz podnaučenosti, onoga što želimo postići te prenaučenosti [12]

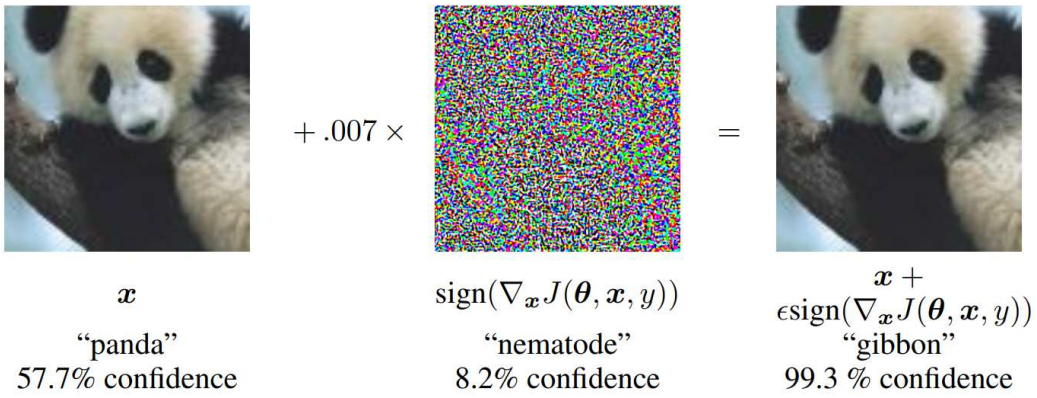
decizijsku granicu u području podnaučenosti linearna što je premala složenost funkcije - greška je jako velika. U području prenaučenosti funkcija za odvajanje primjera je polinomijalna te je greška praktički nula - prevelika prilagođenost primjerima za učenje. Srednje područje je optimalno. Naučili smo na podacima točno onoliko koliko smo htjeli kako bi model mogao dobro generalizirati - greška na testnim primjerima je prihvatljiva. Vidimo na slici da je funkcija koja odvaja primjere parabola, dakle polinom drugog. Dozvoljavanjem male greške na skupu za učenje pokušavamo ostvariti benefite u generalizaciji - smanjenje greške na skupu za testiranje.

Regularizaciju uvodimo u model zbrajanjem regularizacijskog izraza sa funkcijom gubitka. U jednadžbi 2.1 vidimo da izraze $ED(w)$ koji predstavlja funkciju gubitka, $ED(w)$ koji predstavlja regularizacijski izraz, te hiperparametar λ s kojim reguliramo utjecaj regularizacijskog dijela izraza na ukupni gubitak.

$$ED(w) + \lambda * EW(w) \tag{3.1}$$

Prema Bayesovom stajalištu, regularizaciju tumačimo kao usklađivanje s apriornim znanjem o podacima. Popularno apriorno uvjerenje temeljeno na promatranju prirodnih pojava jest da su izlazi većine prirodnih sustava glatki s obzirom na prostorne i/ili vremenske ulaze [11]. Fizički i matematički zakoni često se opisuju glatkim kontinuiranim krivuljama stoga oni u većini slučajeva opravdavaju ova uvjerenja. Razmotrimo model uvjetne vjerojatnosti kakvu konstruiramo prilikom izrade modela dubokog učenja $p(y|x)$ - vjerojatnost izlazne oznake y za zadani ulazni primjer x. Uzevši u ob-

zir prethodno navedeno uvjerenje usmjerit ćemo nastojanja da navedena distribucija $p(y|x)$ bude glatka s obzirom na ulazni primjer x. Zaglađivanjem izlazne distribucije pokušavamo poboljšati rad klasifikatora, što se u praksi često pokazuje istinitom pretpostavkom. Dodjeljivanje oznaka neoznačenim primjerima iz skupa za učenje jedan je od praktičkih primjena zaglađivanja distribucije,a takvim principom se koristi i metoda opisana u ovome radu. Intuitivno nam je jasno te možemo pretpostaviti da ulazni primjeri koji su prostorno blizu jedan drugome pripadaju istoj klasi. Dakle, iako nemamo oznaku ulaznih primjera na temelju njihovog položaja u prostoru te označenim ulaznim primjerima može pretpostaviti njihovu oznaku. Već je dokazano da u neuronskim mrežama možemo poboljšati generalizaciju slučajnim perturbacijama ulaza kako bi generirali umjetne ulazne podatke te prisilili model da pridijeli slične izlaze umjetnim ulazima generiranih iz istih primjera. Umjetni ulazi mogu biti generirani pomakom u nekom smjeru na slici kao i rotacijama slike. Ovakav pristup model štiti od slučajnih i lokalnih perturbacija ulaza te se pokazuje učinkovit u polunadziranom učenju. Prednosti ovakvog pristupa su vidljive, međutim model i dalje ostaje ranjiv na male perturbacije u određenim smjerovima. Ti smjerovi u ulaznom prostoru predstavljaju perturbacije koje kada ih primijenimo na ulazni primjer čine oznaku vjerojatnosti $p(y = k|x)$ izlaza modela najosjetljivijom. Navedene smjerove nazivamo neprijateljskim perturbacijama. Modeli koji koriste standardne tehnike regularizacija, poput $L1$ i $L2$ normi, nisu mogli naučiti kako klasificirati neprijateljske primjere te stoga lako griješe kada na ulazne podatke djelujemo s neprijateljskim perturbacijama [11]. Neprijateljske preturbacije ne moraju sadrzavati vrijednosti vidljive ljudskome oku. Uzmimo za primjer sliku, ako na vrijednosti piksela u slici dodamo neprijateljsku preturbaciju nećemo moći zapaziti osjetniju razliku izmedu originala i promijenjenog primjera. Naravno, neke slike će imati vidljivu zapaženiju razliku u nijansama boja.



Slika 3.2: Prikaz neprijateljske perturbacije te kako djeluje na ulazni primjer [6]

Slika 3.2 prikazuje opisanu situaciju u kojoj neprijateljska perturbacija nije promijenila izgled slike vidljiv ljudskom oku. Vidimo da s do tada naučenim parametrima mreže model je sa 57.7% siguran da se na slici nalazi panda. Međutim nakon dodavanja neprijateljske perturbacije, model primjer svrstava u klasu gibona sa 99.3%. Vrijednost $\epsilon = 0.007$ je hiperparametar metode kojim utječemo u kojoj će mjeri neprijateljska perturbacija djelovati na ulazni primjer.

3.1. Karakteristike virtualnog neprijateljskog učenja

Model treniramo regularizacijskom metodom koja će rezultirati izotropno glatkom funkcijom izlaza oko svakog ulaznog podatka [11]. Zaglađivanje modela provest će se u dijelovima gdje je najosjetljiviji na promjene - dakle u najmanje izotropnim smjerovima. Virtualna neprijateljska perturbacija (engl. virtual adversarial direction) je pojam koji uvodimo kako bi proveli ideju i podrobnije je objasnili. Virtualna neprijateljska perturbacija je onaj smjer koji najsnažnije može promijeniti izlaznu distribuciju u smislu mjere različitosti dvaju distribucija. Djelovanje virtualnim neprijateljskim smjerom na ulazne podatke ima za posljedicu u najvećoj mjeri smanjivanje vjerojatnosti ispravne klasifikacije. Možemo ga definirati i kako smjer u kojem model može najviše odstupati u odnosu na ispravnu oznaku. Naš cilj - način na koji model uči pomoću ove metode - je da smanjimo predikcijsku razliku između ulaznih primjera i njihove izmijenjene inačice perturbirane u smjeru najveće promjene izlaza. Za razliku od neprijateljske perturbacije, virtualnu neprijateljsku perturbaciju možemo definirati i nad neoznačenim primjerima. Ova činjenica ovakvu vrstu treniranja modela svrstava u polunadzirano učenje. Definiranjem virtualne neprijateljske perturbacije na ovaj način možemo kvantificirati lokalnu distribucijsku glatkoću modela bez korištenja nadziranog učenja [11]. Lokalnu distribucijsku glatkoću (engl. local distributional smoothness - LDS) definiramo kao divergenciju između distribucije primjera x i primjera x na koji djelujemo u virtualnom neprijateljskom perturbacijom. Treniranje modela ovim principom nazivamo učenje s virtualnim neprijateljskim primjerima (engl. virtual adversarial training - VAT).

Prednosti VAT regularizacije [11]:

1. Primjenjivost na zadatke koji su povoljni za rješavanje polunadziranim učenjem
2. Primjenjivost na nadzirano i polunadzirano učenje
3. Primjenjivost na bilo koje parametarske modele za koje možemo izračunati gradijent s obzirom na ulaz i parametre
4. Mali broj hiperparametara
5. Zaglađivanje decizijske granice
6. Relativno mali troškovi izračuna [10]
7. "Pametna" (invarijantna) regularizacija s obzirom na parametre.

VAT s obzirom na to da koristi određeni broj označenih primjera, ali i većinu neoznačenih primjera svrstavamo u kategoriju polunadziranog učenja. Međutim on se može koristiti u nadziranom učenju. Za obje vrste učenja ima pozitivnu karakteristiku pomicanja decizijske granice. Široka primjenjivost VAT-a uzevši u razmatranje da je za učenje potrebno računanja gradijenta s obzirom na ulaz, jer tako dobivamo virtualnu neprijateljsku perturbaciju. Razmatrajući neuronske mreže, u kako bismo proveli jednu iteraciju algoritma moramo provesti ne više od 3 para prolazaka unaprijed i unazad (engl. forward and backward propagation) [10]. Prednost je svakako i mali broj hiperparametara, točnije dva - ϵ i ξ.

Promotrimo pobliže treću navedenu prednost. Na prvi pogled možemo pomisliti da algoritam mora riješiti unutarnji optimizacijski problem kako bi pronašao virtualnu neprijateljsku perturbaciju. Međutim pronalaženje virtualne neprijateljske perturbacije zbog specifičnosti problema nije moguće gradijentnim spustom, nego se ono provodi metodom iteracije potencija. Ovakav pristup nam omogućava da troškovi učenja neuronske mreže ne premašuju za više od tri puta standardne troškove učenja bez regularizacije. Ovakav iterativni postupak približavanja željenoj vrijednosti virtualne neprijateljske perturbacije je važan dio VAT algoritma koji ga čini lako prilagodljivim na različite modele.

Za linearne modele, L_p regulacija ima učinak smanjivanja razine prilagođenosti izlaza u odnosu na ulaz - odnosno smanjivanje prenaučenosti, a može se kontrolirati jakost njegovog učinka preko hiperparametra λ. Međutim, standardne regularizacije putem norme djeluju direktno na naučene parametre. VAT koristi posredan pristup preko predikcije koju model pruža u datom trenutku, te parametre korigira prolasku unazad. Upravo ta činjenica ga čini invarijantnom regularizacijom s obzirom na parametre.

3.2. Neprijateljski primjeri

Modeli dubokog učenja postižu odlične rezultate na području obrade teksta, računalnog vida kao i raspoznavanja govora. Neuronske mreže danas pronalaze uzorke u slikama izrazito učinkovito zbog velikog kapaciteta mreže kao i mnogobrojnim nelinearnim transformacijama koje uvodimo aktivacijskim funkcijama. Međutim, kako modele učimo provođenjem automatskim algoritmom prolaska unazad (engl. backpropagation) naučene uzorke može biti teško protumačiti te svojstva mogu biti neintuitivna [13]. neintuitivno svojstvo koje ćemo predstaviti tiče se stabilnosti predikcije neuronskih mreža s obzirom na male perturbacije ulaza. Očekujemo da će modeli koji dobro generaliziraju biti otporni na male perturbacije ulaza, međutim to nije slučaj. Smatrali smo da takve promjene ne mogu objektivno promijeniti klasu u koju slika pripada. Dokazano je da neprijateljska promjena koja nije slučajna utječe na predikciju modela. Takve ne slučajne promjene nazivamo neprijateljskim smjerovima, a primjene takvim perturbacija na ulazne primjere čine ih neprijateljskim primjerima. Ovakve perturbacije računamo optimizirajući izlaz tako da maksimiziramo predikcijsku pogrešku [13].

3.2.1. Linearno objašnjenje neprijateljskih primjera

Promotrimo pobliže postojanje neprijateljskih primjera u linearnim modelima. Jednadžba linearnog modela:

$$y = w^T \cdot x \tag{3.2}$$

Preciznost svojstava koje možemo prikazati kod ulaznih primjera je najčešće limitirana [6]. Konkretan primjer u računalnom vidu je prikaz slike. Slika se u digitalnom obliku prikazuje s 8 bita po pikselu - tako se može prikazati 255 različitih vrijednosti - prema tome se sve informacije ispod $\frac{1}{2^8} = \frac{1}{255}$ ne mogu prikazati. Ovaj problem svrstavamo u problem kvantizacije signala u ovom slučaju slike. Ukoliko bismo uzeli veći broj bita mogli bismo prikazati više vrijednosti, dakle manje razlike u nijansama boja - mogućnosti prikaza većeg broja boja. Zbog manjka preciznosti racionalno je očekivati da će ulazni primjer x prediktor svrstati u istu klasu kao i neprijateljski primjer $\tilde{x} = x + \eta$, uz pretpostavku da je svaka pojedina vrijednost perturbacije η od određene preciznosti - u gore navedenom primjeru vrijedi $\eta[i] < \frac{1}{255}$, gdje i predstavlja indeks svakog pojedinog elementa. Pretpostavka iz perspektive Bayesa izgleda ovako:

$$p(y|x) = p(y|x + \eta) \tag{3.3}$$

Formalno, očekujemo da će prediktor svrstati primjer x i neprijateljski primjer $\tilde{x}$ u istu klasu sve dok je:

$$\|\eta\|_\infty < \epsilon \tag{3.4}$$

Gdje je ϵ dovoljno malen da ga ne možemo prikazati našom preciznošću. Uvrstimo u našu pretpostavku o svrstavanju primjera u istu klasu jednadžbu linearnog modela 3.2:

$$y = \tilde{y} \tag{3.5}$$

$$w^T \cdot x = w^T \cdot \tilde{x} \tag{3.6}$$

Promotrimo sada linearni model te na njegov ulaz dovedimo neprijateljski primjer $\tilde{x}$:

$$w^T \cdot \tilde{x} = w^T \cdot (x + \eta) \tag{3.7}$$

$$w^T \cdot \tilde{x} = w^T \cdot x + w^T \cdot \eta \tag{3.8}$$

Neprijateljska perturbacija u izraz unosi razliku od $w^T * \eta$. Želimo maksimizirati tu razliku stoga uvodimo $\eta = sign(w)$ uz prethodno navedeno ograničenje $\|\eta\|_\infty < \epsilon$. Ako je dimenzionalnost matrice w jednaka n, prosječna apsolutna vrijednost elemenata u matrici je m, onda je unesena razlika u jednadžbu iznositi ϵnm:

$$w^T \cdot \eta = \epsilon nm \tag{3.9}$$

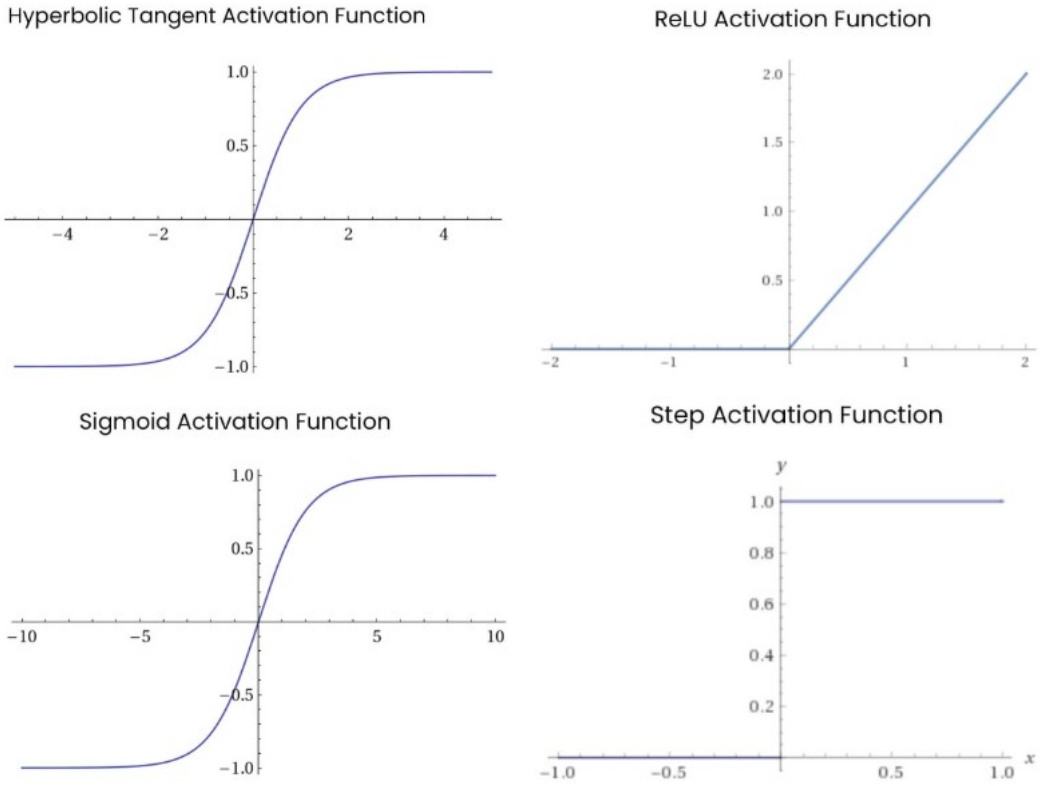
Prema jednadžbi 3.9 vidimo da promjena raste linearno sa n - brojem dimenzija vektora. Dakle za probleme velikih dimenzija možemo napraviti niz infinitezimalnih promjena ulaza koje će rezultirati velikom promjenom izlaza [6]. Iz navedenih zakonitosti te jednadžbe 3.8 vidimo da jednadžbe 3.5 i 3.6 ne mogu vrijediti, nego vrijedi sljedeće:

$$y \neq \tilde{y} \tag{3.10}$$

$$w^T \cdot x \neq w^T \cdot \tilde{x} \tag{3.11}$$

3.2.2. Neprijateljski primjeri u dubokim mrežama

Duboke neuronske mreže su linearno dizajnirane te se nelinearnost pokušava uvesti aktivacijskim funkcijama poput ReLU, tanh ili sigmoida. Na slici 3.3 prikazane su



Slika 3.3: Prikaz aktivacijski funkcija koje se koriste u dubokim neuronskim mrežama [2]

aktivacijske funkcije koje se koriste u neuronskim mrežama. Vidimo da se funkcije izvan područija zasićenja ponašaju poprilično linearno, prema tome iako unose određenu nelinearnost u model nisu otporne na neprijateljske primjere. Već smo dokazali da razlika u predikciji između primjera i njemu srodnog neprijateljskog primjera ovisi o dimenzionalnosti - što je drugi razlog ranjivosti neuronskih mreža kada ih izložimo neprijateljskim primjerima.

Razmotrimo sada generiranje neprijateljskih primjera kroz algoritam propagiranja unazad u neuronskoj mreži [5]. Neka $L(x, y, \theta)$ funkcija gubitka, x predstavlja ulazni primjer, y izlaz modela, θ parametre modela. Za točke x_0 odredimo perturbaciju η koja davanjem na ulazni primjer najviše mijenja funkciju gubitka $L(x, y, \theta)$. Općeniti razvoj funkcije u Taylorov red prvog stupnja:

$$f(x) \approx f(a) + \nabla f(x)(x - a) \tag{3.12}$$

Razvoj funkcije gubitka u Taylorov red prvoga stupnja izgleda ovako:

$$L(x_0 + \eta, y, \theta) \approx L(x_0, y, \theta) + \nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0} \cdot ((x_0 + \eta) - x_0) \tag{3.13}$$

$$L(x_0 + \eta, y, \theta) = L(x_0, y, \theta) + \nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0} \cdot \eta \tag{3.14}$$

Prebacivanjem dijela izraza na lijevu stranu dobivamo:

$$L(x_0 + \eta, y, \theta) - L(x_0, y, \theta) = \nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0} \cdot \eta \tag{3.15}$$

Promjena funkcije gubitka iznosi, a to je ujedno i primjena jednaka onoj u jednadžbi 3.8:

$$\nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0} \cdot \eta \tag{3.16}$$

Ovaj puta je matrica težina zamijenjena sa $\nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0}$ te će upravo tada promjena u jednadžbi 3.8 biti najveća:

$$\eta = \nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0} \tag{3.17}$$

Prema ovom postupku dodavanje ove perturbacije na ulazni primjer formira neprijateljski primjer:

$$x_{adv} = x_0 + \nabla_x L(x_0 + \eta, y, \theta)|_{x=x_0} \tag{3.18}$$

3.3. Opis metode

U ovom paragrafu definirat ćemo skup notacija koje će nam služiti kao norma označavanja u sljedećim poglavlju. Oznake su preuzete iz rada koji je predstavio VAT metodu regularizacije. [11]. Neka $x \in R^I$ predstavlja ulazni primjer gdje je I dimenzija ulaznih podataka, dok će $y \in Q$ označavati oznake pripadnosti ulaznom primjeru - Q predstavlja skup svih oznaka. $p(y|x, \theta)$ predstavlja izlaznu distribuciju parametriziranu sa θ. Sa $\hat{\theta}$ ćemo označavati trenutno naučene parametre modela - parametri modela točno određenoj iteraciji procesa učenja [11]. Označene primjere ćemo opisivati s izrazom:

$$D_l = \{x_l^{(n)}, y_l^{(n)} | n = 1, 2, ..., N_l\} \tag{3.19}$$

Neoznačene primjere opisujemo s izrazom:

$$D_{ul} = \{x_{ul}^{(m)} | m = 1, 2, ..., N_{ul}\} \tag{3.20}$$

N_l predstavlja broj primjera s oznakom, dok N_{ul} broj primjera bez oznake. Model treniramo na skupu označenih primjera D_l i neoznačenih primjera D_{ul}.

3.3.1. Učenje s neprijateljskim primjerima

Promotrimo učenje s virtualnim neprijateljskim primjerima prije definiranja učenja s virtualnim neprijateljskim primjerima. Definirajmo model učenja s neprijateljskim primjerima. Funkcija gubitka modela učenja s neprijateljskim primjerima definirana je kao [6]:

$$L_{adv}(x_l, \theta) := D[q(y|x_l), p(y|x_l + r_{adv}, \theta)] \tag{3.21}$$

Gdje r_{adv} izračunavamo prema sljedećoj formuli:

$$r_{adv} := \underset{r;\|r\| \leq \epsilon}{\operatorname{argmax}} D[q(y|x_l), p(y|x_l + r), \theta)] \tag{3.22}$$

Primijetimo da je neprijateljska perturbacija r_{adv} zapravo η u prošlom poglavlju. Funkcija $D[q, p]$ iz jednadžbe 3.21 je neka funkcija pozitivnih vrijednosti koja mjeri razliku između dvaju distribucija q i p. Stavimo ovakvu funkciju u kontekst neuronski mreža i na koji način prikazuju i interpretiraju izlaz. Neuronske mreže se konstruiraju tako da na izlazu sadrže vektor dimenzija istih broju klasa u koje želimo razvrstati ulazne primjere. Pojedina vrijednost iz izlaznog vektora je vjerojatnost pripadnosti nekoj od klasa. Funkcija D može biti unakrsna entropija:

$$D[q, p] = -\sum_{i}^{K} q_i \cdot log(p_i) \tag{3.23}$$

q i p su vektori vjerojatnosti za koje svaka i-ta pozicija predstavlja vjerojatnost pripadnosti i-toj klasi. Funkcija $q(y|x_l)$ je stvarna distribucija pripadnosti klasi y uz uvjet da smo na ulaz doveli označeni primjer x_l. Ovako definirana funkcija gubitka bolje aproksimira stvarnu distribuciju $q(y|x_l)$, odnosno naša aproksimacija ima točnu predikciju klase y za ulazni primjer x_l. Ovime se postiže da model bude otporan na neprijateljske primjere [6]. Neprijateljski primjeri nastaju dodavanjem neprijateljske perturbacije r_{adv} na ulazni primjer x_l. Stvarnu distribuciju $q(y|x_l)$ pokušavamo aproksimirati učenjem parametara distribucije $p(y|x_l, \theta)$. Npr. distribucija $q(y|x_l)$ može biti aproksimirana vektorom $h(y; y_l)$. $h(y; y_l)$ vektor je definiran tako da na svim indeksima ima vrijednost 0 (nula), osim na mjestu gdje je indeks vektora jednak indeksu klase - na tom indeksu vrijednost poprima vrijednost 1 (jedan) - ovakav vektor nazivamo one-hot vektor.

Formula za izračun r_{adv} (3.22) nije funkcija koja ima rješenje u zatvorenoj formi, što znači da izračun r_{adv} nije moguće provesti analitički. Međutim, r_{adv} možemo izračunati linearnom aproksimacijom funkcije D s obzirom na r:

$$g = \nabla_{x_l} D[h(y; y_l)], p(y|x_l, \theta) \tag{3.24}$$

Aproksimiranu neprijateljsku perturbaciju normaliziramo normom L_2:

$$r_{adv} = \frac{g}{\|g\|_2} \tag{3.25}$$

Kada za normalizaciju koristimo normu L_∞: $r_{adv} = \epsilon \cdot sign(g)$ - što je isto što smo već opisivali u poglavljima vezano za neprijateljske primjere. Ovakav način izračuna r_{adv} možemo efikasno provesti algoritmom prolaska unazad (engl. backpropagation) [6]. Učenjem s neprijateljskim primjerima možemo naučiti model koji će bolje generalizirati [11].

3.3.2. Učenje s virtualnim neprijateljskim primjerima

Učenje na temelju neprijateljskih primjera je uspješna metoda za rješavanje mnogih problema putem nadziranog učenja [11]. Za razliku od neprijateljske perturbacije, virtualnu neprijateljsku perturbaciju možemo definirati i nad neoznačenim primjerima - što ovakvu vrstu učenja modela svrstava u polunadzirano učenje. U uvodu rada spominjao sam poteškoće s označavanjem primjera koje upućuju na to da bi trebali razmisliti o metodama učenja u kojima nam nisu potrebne oznake y_l ulaznih primjera x.

Razmotrimo kako ćemo pretvoriti nadzirano učenje u polunadzirano. Uvedimo novu oznaku za ulazne primjere x_* nad kojima provodimo učenja, a sastoje se od $x_* = \{x_l, x_{ul}\}$. Za razliku od neprijateljskog učenja nemamo informacije o stvarnoj izlaznoj distribuciji $q(y|x)$, stoga moramo pronaći adekvatnu zamjenu kako bi učenje bilo moguće. Promotri što bi se u našem modelu promijenilo kada bismo stvarnu distribuciju zamijenili s njezinom trenutnom aproksimacijom - s onom distribucijom koju smo trenutno naučili naš model $p(y|x, \theta)$. Naravno, trenutna naučena distribucija ovisi o primjerima kojima znamo oznaku. Dakle proces učenja ovisi o broju označenih primjera. Iako nam intuicija govori da bi aproksimacija bila "loša" ili "naivna" to ne mora biti slučaj. Ako imamo veći broj označenih primjera aproksimacija može biti iznimno dobra, a konačna statistika koju daje naučeni model usporediva s modelima koje smo trenirali nadziranim učenjem. Uporabom pretpostavljene distribucije ujedno koristimo pretpostavljene oznake primjera koje su generirane iz dosad naučene distribucije $p(y|x, \theta)$. Tako generirane oznake nazivamo virtualnim oznakama koje potom koristimo umjesto nepoznatih oznaka neoznačenih ulaznih primjera u računanju virtualne neprijateljske perturbacije [11]. Upravo zato što koristimo "pretpostavljene" ili "virtualne" oznake neprijateljsku perturbacije možemo nazvati virtualnom [11]. Uvrstimo opisan postupak u formule za neprijateljsko učenje, gubitak 3.21 i izračun neprijateljske perturbacije 3.22:

$$L_{vadv}(x_*, \theta) := D[p(y|x_*, \hat{\theta}), p(y|x_* + r_{vadv}, \theta)] \tag{3.26}$$

Pri čemu je:

$$r_{vadv} := \underset{r;\|r\|_2 \leq \epsilon}{\operatorname{argmax}} D[p(y|x_*, \hat{\theta}), p(y|x_* + r, \hat{\theta})] \tag{3.27}$$

U formulama 3.26 i 3.27 vidimo dvije oznake koje predstavljaju varijable parametara modela θ i $\hat{\theta}$. Oznaku θ (bez kapice) koristimo kada propuštanjem gradijenata u nazad učimo model, dok $\hat{\theta}$ koristimo samo njihovu trenutnu vrijednost za aproksimaciju distribucije bez ažuriranja gradijenata. Gubitak prikazan u jednadžbi 3.26 predstavlja funkciju lokalne glatkoće LDS (engl. Local Distributional Smoothness) oko ulaznog

podatka x_* za trenutnu aproksimaciju razdiobe - stanje modela. LDS je negativna mjera lokalne glatkoće, što znači da će smanjivanje vrijednosti koje LDS poprima rezultirati većom glatkoćom modela oko ulaznog primjera x_* [11]. 3.27 je virtualna neprijateljska perturbacija koja traži najveću promjenu distribucije oko primjera x_*. Izračun ovih formula provodimo lokalno za svaki primjer što ih čini mjerom lokalne glatkoće distribucije. LDS je mjera koja nam govori koliko je distribucija $p(y|x_*)$ glatka.

Regularizacijski izraz je srednja vrijednost funkcije pogreške nad cijelim skupom ulaznih podataka [11]:

$$R_{vadv}(D_l, D_{ul}, \theta) = \frac{1}{N_l + N_{ul}} \sum_{x_* \in D_l, D_{ul}} L_{vadv}(x_*, \theta) \tag{3.28}$$

Izraz za potpunu funkciju cilja modela kojim učimo VAT regularizacijom:

$$l(D_l, \theta) + \alpha \cdot R_{vadv}(D_l, D_{ul}, \theta) \tag{3.29}$$

Gdje $l(D_l, \theta)$ predstavlja gubitak na skupu označenih ulaznih primjera, a funkcija je negativne log-izglednosti. Prednosti u ovakvom pristupu učenju su to što imamo samo dva hiperparametra, a to su norma ograničenja $\epsilon > 0$ za neprijateljsku perturbaciju te regularizacijski faktor $\alpha > 0$ koji regulira omjer negativne log-izglednosti i regularizacijske funkcije.

3.3.3. Metoda efikasne aproksimacije r_{vadv}

Jednom izračunata virtualna neprijateljska perturbacija r_{vadv} te njenim uvrštavanjem u funkciju lokalne glatkoće $LDS(x_*, \theta)$ ona postaje mjera različitosti ili divergencije između dvije distribucije $p(y|x_*, \hat{\theta})$ i $p(y|x_* + r_{vadv}, \theta)$ [11]. Promotrimo kako izgleda funkcija gubitka s uvrštenom virtualnom neprijateljskom perturbacijom:
$D[p(y|x_*, \hat{\theta}), p(y|x_* + r_{vadv}, \theta)]$. Uzevši u obzir da funkcija mjeri razliku dvaju distribucija te da ta funkcija ujedno predstavlja regularizacijski izraz koji zajedno s funkcijom pogreške čini funkciju cilja koju želimo minimizirati kako bi naš model ispravno radio, te da razliku u distribucijama unosi perturbacija r dolazimo do problematike pronalaženja virtualne neprijateljske perturbacije r_{vadv}. To je optimizacijski problem za koji nije poznato postoji li rješenje u zatvorenoj formi. Problem je u tome što r_{vadv} ne možemo izračunati metodom linearne aproksimacije koju često koristimo za izračunavanje derivacija prvoga reda. Ovakva pojava ima za posljedicu da ne možemo koristiti metodu gradijentnog spusta.

Uvrštavanjem $r = 0$ u funkciju lokalne glatkoće, funkcija poprima vrijednost 0 (nula) - $D[p(y|x_*, \hat{\theta}), p(y|x_* + 0, \theta)] = 0$. Problem je u tome što funkcija poprima vrijednosti lokalnog minimuma, za dani $r = 0$. Postoji opasnost od "zapinjanja" metode gradijentnog spusta u lokalnom minimumu što nam onemogućava pronalaženje prave vrijednosti maksimuma za vrijednost funkcije lokalne glatkoće pa posljedično i prave vrijednosti perturbacije r_{vadv}.

Promotrimo kako ćemo računati r_{vadv}. Jednostavnosti radi $D[p(y|x_*, \hat{\theta}), p(y|x_* + r_{vadv}, \theta)]$ ćemo označiti sa $D(r, x_*, \theta)$. Ako funkciju $D(r, x_*, \hat{\theta})$ promatramo na manjem lokalnom području možemo je aproksimirati u obliku kvadratne forme. Aproksimacija se provodi razvojem u Taylorov red drugog stupnja:

$$f(x) \approx f(a) + \nabla f(a)(x - a) + \frac{1}{2}(x - a)^T H f(a)(x - a) \tag{3.30}$$

Gdje je H Hesseova matrica koja je matrica drugih derivacija. Pretpostavljamo da je distribucija $p(y|x, \theta$ dva puta diferencijabilna po θ i x na cijelom području domene. Uvrstimo sada funkciju $D(r, x_*, \hat{\theta})$ u formulu za razvoj u Taylorov red drugog stupnja oko točke $r = 0$:

$$D(r, x_*, \hat{\theta}) \approx D(0, x_*, \hat{\theta}) + \nabla D(r, x_*, \hat{\theta})|_{r=0} + \frac{1}{2} r^T \cdot H(r, x_*, \hat{\theta})|_{r=0} \cdot r \tag{3.31}$$

$\nabla D(r, x_*, \hat{\theta})$ i $H(r, x_*, \hat{\theta})$ predstavljaju gradijent i Hesseovu matricu funkcije $D(r, x_*, \hat{\theta})$. Funkcija $D(r, x_*, \hat{\theta})$ u $r = 0$ poprima vrijednost 0 (nula) što je lokalni minimum, pretpostavka diferencijabilnosti govori da prva derivacija funkcije u ekstremima poprima

vrijednost nula. Dakle, gradijent $\nabla_r D(r, x_*, \hat{\theta})|_{r=0} = 0$. Zbog svega navedenog u razvoju u Taylorov red drugog stupnja ostaje nam:

$$D(r, x_*, \hat{\theta}) \approx \frac{1}{2} r^T \cdot H(r, x_*, \hat{\theta})|_{r=0} \cdot r \tag{3.32}$$

Gdje je $H(x, \hat{\theta})$ Hesseova matrica koju možemo predstaviti izrazom:

$$H(x, \hat{\theta}) := \nabla\nabla_r D(r, x_*, \hat{\theta})|_{r=0} \tag{3.33}$$

Iz ovih aproksimacija problem pronalaženja neprijateljske perturbacije r_{vadv} svodi se na pronalaženje prvog svojstvenog vektora Hesseove matrice $H(x, \hat{\theta})$, a možemo zapisati sljedećom formulom:

$$r_{vadv} \approx \underset{r;\|r\|_2 \leq \epsilon}{\operatorname{argmax}} \left\{ \frac{1}{2} r^T \cdot H(r, x_*, \hat{\theta})|_{r=0} \cdot r \right\} = \epsilon \cdot \frac{u(x, \hat{\theta})}{\left\| u(x, \hat{\theta}) \right\|_2} \tag{3.34}$$

Ovaj problem rješavamo metodom uzastopnog potenciranja te linearnom aproksimacijom derivacija [11]. d je slučajno uzrokovan vektor koji nije okomit na dominantni svojstveni vektor u matrice H, iterativnim izračunom sljedeće formule d će konvergirati u u:

$$d_n \longleftarrow \frac{H \cdot d_{n-1}}{\|H \cdot d_{n-1}\|_2} \tag{3.35}$$

Problem izračuna svojstvenih vektora Hesseove matrice ima vremensku složenost $O(I^3)$[11]. Kako bismo smanjili računalnu zahtjevnost izračuna Hesseove matrice, koja doista može biti velika s obzirom na njezinu definiciju derivacije gradijenta po svim varijablama, nećemo je računati direktno. Problem računalne složenosti izračunavanja H rješavamo aproksimacijom. Korištenjem metode konačnih razlika derivacija dobivamo:

$$Hd \approx \frac{\nabla_r D(r, x_*, \hat{\theta})|_{r=\xi d} - \nabla_r D(r, x_*, \hat{\theta})|_{r=0}}{\xi} = \frac{\nabla_r D(r, x_*, \hat{\theta})|_{r=\xi d}}{\xi} \tag{3.36}$$

Konačni d možemo izračunati sljedećom formulom:

$$d = \frac{\nabla_r D(r, x_*, \hat{\theta})|_{r=\xi d}}{\left\| \nabla_r D(r, x_*, \hat{\theta})|_{r=\xi d} \right\|_2} \tag{3.37}$$

Izračun d se provodi iterativno ponavljajući opisani postupak K puta. K je broj iteracija kojim provodimo postupak uzastopnog potenciranja te ga možemo smatrati hiperparametrom modela. U pravilu $K = 1$ je dovoljan za provođenje učenja. Za $K > 1$ ne postižu se bolji rezultati, a vrijeme izvođenja raste. Provođenjem opisanog postupka d će biti valjana aproksimacija virtualne neprijateljske perturbacije r_{vadv}.

3.4. Postupak izračuna VAT gubitka

U ovom poglavlju predočit ćemo pseudokod postupka izračuna VAT gubitka. U prošlome poglavlju smo definirali teoretsku podlogu postupka, a sada ćemo definirati kako bi to izgledalo u računalu.

Algorithm 1 Algoritam izračuna VAT gubitka

1: Primjeri: $D = \{\vec{x}_i\}_1^M$
2: Generiraj M slučajnih vektora iz normalne distribucije $\vec{d_i}$
3: **za** $k = 1, k{+}{+}$, **dok** $k < K$:
4: $\quad g_i \leftarrow \nabla_r D[p(y|\vec{x}_*; \hat{\theta}), p(y|\vec{x}_* + \vec{r}; \hat{\theta})]|_{r=\xi d_i}$
5: $\quad \vec{r_i} \leftarrow \frac{\vec{g_i}}{\|\vec{g_i}\|_2}$
6: $\quad d_i \leftarrow r_i$
7: **Vrati** $\nabla_\theta (1/M \sum D[p(y|\vec{x}_*; \hat{\theta}), p(y|\vec{x}_* + \vec{r}; \hat{\theta})])|_{\theta=\hat{\theta}}$

Postupak izračuna VAT gubitka pokreće se u svakoj iteraciji postupka učenja modela, a provodi se nad minigrupom ulaznih primjera D. Veličina minigrupe je M - u mini grupi ima M ulaznih primjera. Potom generiramo M slučajni vektora d - dimenzija jednakih kao ulazni primjeri - za svaki ulazni primjer. Potom ulazimo u petlju po kojoj iteriramo K puta - praktično ostvarivanje metode uzastopnog potenciranja. U jednoj iteraciji petlje računamo gradijent funkcije lokalne glatkoće distribucije s obzirom na virtualnu neprijateljsku perturbaciju r. Potom gradijent normaliziramo te ga pridružujemo r. Ovaj postupak provodimo za svaki ulazni primjer, jer svaki ulazni primjer ima svoju jedinstvenu virtualnu neprijateljsku perturbaciju r. Na temelju izračunatih vrijednosti virtualnih neprijateljskih perturbacija iz opisane procedure vraćamo uprosječeni gradijent funkcije lokalne glatkoće distribucije s obzirom na parametre θ.

3.5. Postupak nadziranog učenja

U ovom poglavlju predočit ćemo pseudokod postupka nadziranog učenja. VAT regularizacija pripada polunadziranim postupcima učenja. Međutim ona se može provoditi i nad označenim skupovima podataka. Ne samo da se može provoditi nego može poboljšati klasifikaciju modela, jer u slučaju nadziranog učenja možemo se osloniti na to da aproksimacija distribucije $p(y|x, \theta)$ nije niti malo naivna s obzirom na velik broj označenih primjera nad kojima model uči.

Algorithm 2 Algoritam nadziranog učenja

1: Skup podataka: $D_l = \{(\vec{x}_i, y_i)\}_1^N$
2: Veličina minigrupe podataka: M
3: **za** $j = 1$, $j{+}{+}$, **dok** $j < broj\_iteracija$:
4: Minigrupa veličine M iz D
5: $L_{ce} \leftarrow 1/M \sum_{i=1}^{M} H(p(y|\vec{x}_i; \theta), y_i)$
6: $L_{vat} \leftarrow 1/M \sum_{i=1}^{M} LDS(\vec{x}_i, \theta)$
7: $L \leftarrow L_{ce} + \alpha * L_{vat}$
8: $\hat{\theta} \leftarrow \hat{\theta} + \nabla_\theta L|_{\theta=\hat{\theta}}$
9: **Vrati** $\hat{\theta}$

U postupku nadziranog učenja cijeli skup ulaznih primjera je označen D_l, odabiremo veličinu minigrupe M. Ulaskom u petlju kojom određujemo broj iteracija koliko će se provoditi postupak učenja. U svakoj iteraciji učenja modela odabiremo minigrupu veličine M iz skupa primjera D. Minigrupu propuštamo kroz mrežu na čijem izlazu dobivamo predikciju modela. Potom nad dobivenim vrijednostima računamo gubitak unakrsne entropije (L_{ce}), te VAT gubitak (L_{vat}). Ukupna funkcija gubitka je zbroj gubitka unakrsne entropije s umnoškom koeficijenta α s funkcijom VAT gubitka. Nad takvom funkcijom gubitka računamo gradijente s obzirom na parametre postupkom prolaska unazad (enlg. backpropagation) te ažuriramo težine našega modela. U ovakvom postupku nismo iskoristili prednosti neoznačavanja primjere, ali smo dobili prednost u tome što model bolje generalizira.

3.6. Postupak polunadziranog učenja

U ovom poglavlju predočit ćemo pseudokod postupka polunadziranog učenja. Već smo definirali da VAT regularizacije pripada polunadziranim postupcima učenja, a sada ćemo definirati kako bi to izgledalo u računalu.

Algorithm 3 Algoritam polunadziranog učenja

1: Označeni skup podataka: $D_l = \{(\vec{x}_i^l, y_i)\}_1^{N_l}$
2: Neoznačeni skup podataka: $D_{ul} = \{\vec{x}_i^{ul}\}_1^{N_{ul}}$
3: Veličina minigrupe označenih podataka: M_l
4: Veličina minigrupe neoznačenih podataka: M_{ul}
5: **za** $j = 1, j{+}{+}$, **dok** $j < broj\_iteracija$:
6: Minigrupa veličine M_l iz D_l
7: Minigrupa veličine M_{ul} iz $D_u l$
8: $L_{ce} \leftarrow 1/M_l \sum_{i=1}^{M_l} H(p(y|\vec{x}_i;\theta), y_i)$
9: $L_{vat} \leftarrow 1/M_{ul} \sum_{i=1}^{M_{ul}} LDS(\vec{x}_i, \theta)$
10: $L \leftarrow L_{ce} + \alpha * L_{vat}$
11: $\hat{\theta} \leftarrow \hat{\theta} + \nabla_\theta L|_{\theta=\hat{\theta}}$
12: **Vrati** $\hat{\theta}$

U postupku polunadziranog učenja skup ulaznih podataka dijelimo na označeni skup podataka D_l i neoznačeni skup podataka D_{ul}. Potom definiramo veličine minigrupe za označene primjere M_l i neoznačene primjere M_{ul}. Ulaskom u petlju kojom određujemo broj iteracija koliko će se provoditi postupak učenja. Za svaku iteraciju postupka uzimamo M_l označenih primjera te M_{ul} neoznačenih primjera. Obje minigrupe propušta kroz mrežu na čijem izlazu dobivamo predikciju modela. Potom računamo gubitak unakrsne entropije (L_{ce}) nad označenom minigrupom, te VAT gubitak (L_{vat}) nad neoznačenim podacima. Ukupna funkcija gubitka je zbroj gubitka unakrsne entropije s umnoškom koeficijenta α s funkcijom VAT gubitka. Nad takvom funkcijom gubitka računamo gradijente s obzirom na parametre postupkom prolaska unazad (enlg. backpropagation) te ažuriramo težine našega modela. Bilo bi poželjno da VAT gubitak računamo i nad označenim minigrupama. Iako primjeri jesu označene model će svejedno učiti iz njih, te provoditi zaglađivanje decizijske granice. Postupak izračuna vat gubitka se provodi već opisanim algoritmom 1. - postupka izračuna VAT gubitka.

4. Vizualizacija procesa učenja s točkama u 2D prostoru s VAT gubitkom

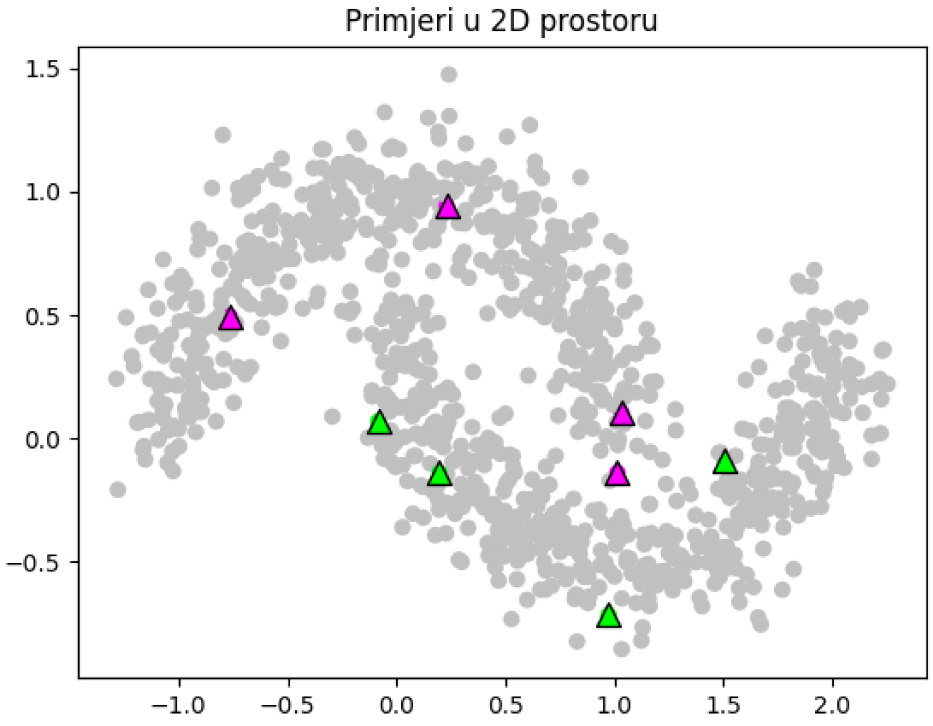
U ovom poglavlju ćemo razmotriti proces učenja s VAT gubitkom na skupu podataka točaka u 2D prostoru. Ova vizualizacija je inspirirana iz znanstvenog rada [11] gdje su već provodili ovakvu ilustraciju procesa učenja. Generirat ćemo dva skupa točaka u 2D prostoru u obliku polumjeseca. Skup primjera će se sastojati od 1000 točaka, a izdvojiti ćemo 8 točaka - po 4 iz svake oznake. Koristiti ćemo jednostavan model neuronske mreže s dva skrivena sloja veličine 100 i 100. Prikazivati ćemo predikciju i VAT gubitak kroz različite iteracije učenja modela. Paralelno ćemo prikazivati model koji učimo s VAT gubitkom i model koji učimo bez VAT gubitka. Ovakav pristup će nam uvelike olakšati da shvatimo i vizualno prikažemo tezu o zaglađivanju decizijske granice između različitih oznaka unutar skupa podataka - blagodat koju nam VAT donosi uz mogućnost korištenja malog broja označenih primjera.

Slika 4.1 prikazuje nasumično generiran skup podataka u obliku dva polumjeseca iz kojeg smo nasumice odabrali 8 označenih točaka, po 4 točke iz svake oznake. Izlaz iz modela će biti vjerojatnost pripadnosti klasi $p(y = 1|x, \hat{\theta}) = 1$. Dvije oznake ćemo u prostoru vizualizirati bojama tako da:

1. $p(y = 1|x, \hat{\theta}) = 1.0$ predstavljat će zelenu boju
2. $p(y = 1|x, \hat{\theta}) = 0.5$ predstavljat će sivu boju
3. $p(y = 1|x, \hat{\theta}) = 0.0$ predstavljat će ljubičastu boju

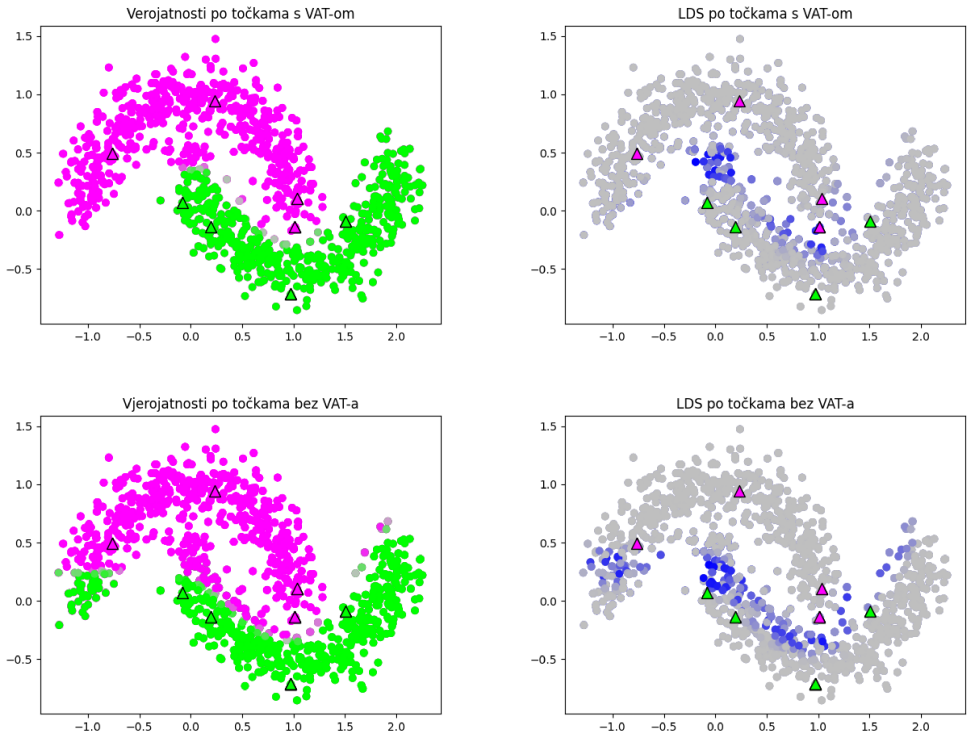
Vjerojatnosti koje će biti između gore napisanih predstavljati ćemo nijansama boja između sive i ljubičaste te sive i zelene. Uz prikaz predikcije spomenuli smo i prikaz VAT gubitka koji ćemo također prikazivati. Regularizacijski izraz prikazivati ćemo različitim nijansama plave boje u mjestima ulaznih podataka. Najplavijom nijansom

će biti označen podatak koji ima najveći VAT gubitak, dok će onaj bez VAT gubitka biti obojan u sivu boju - svjetlije nijanse plave prikazivati će manje vrijednosti VAT gubitka.



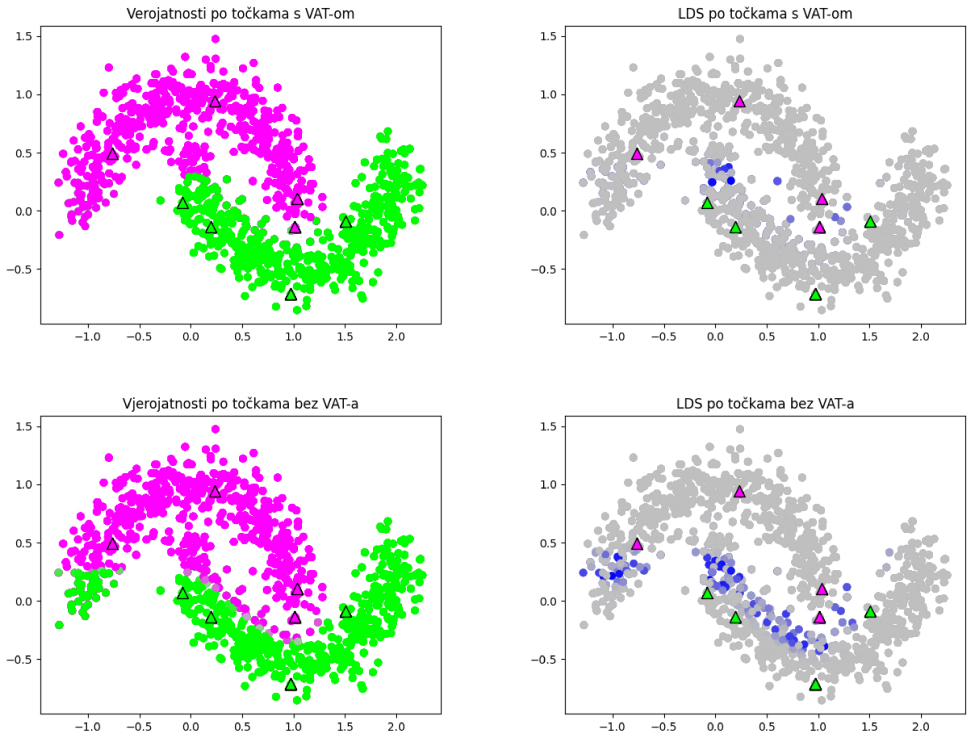
Slika 4.1: Ulazni skup podataka u 2D prostoru

Hiperparametre koje smo koristili za provođenje učenja su: ϵ=0.2, ξ=1e-3, K=1, α=1. Opisane grafove prikazivati ćemo u skupinama po 4 grafa za određene iteracije procesa učenja, kako bismo vidjeli kako napreduje proces učenja sa i bez VAT gubitka. U prvom redu će biti vizualizirane predikcije i pripadajući VAT gubitak u procesu učenja s VAT-om dok će u drugom redu biti prikazan proces učenja bez VAT-a.



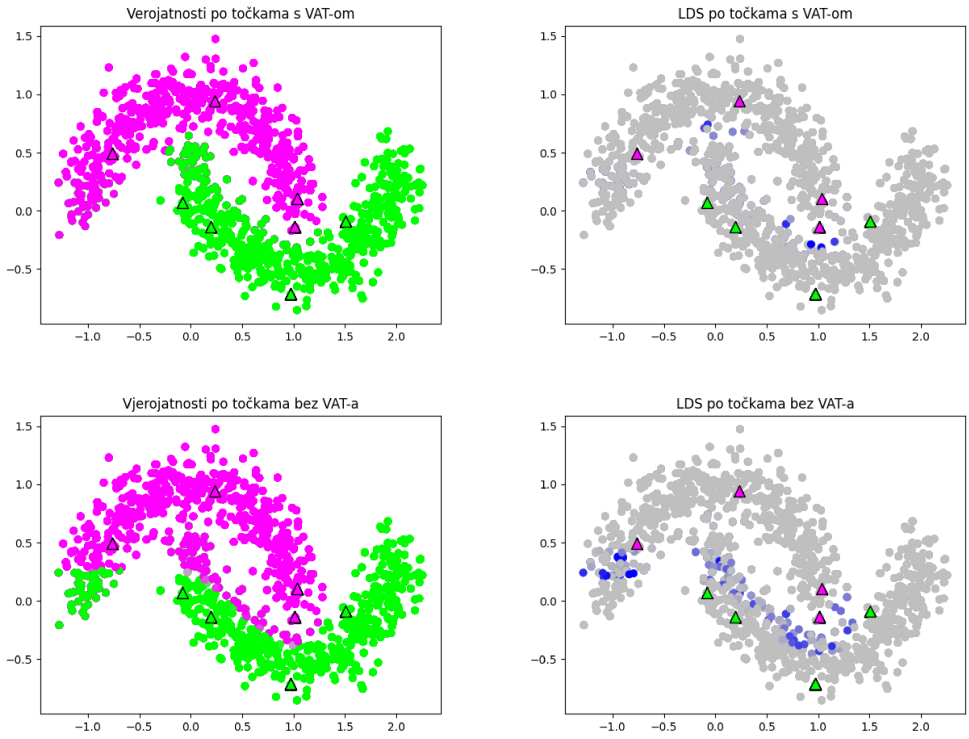
Slika 4.2: Prikaz procesa učenja nakon 100 iteracija

Na slici 4.2 možemo vidjeti znatnu razliku u predikciji između modela u kojem koristimo VAT gubitak - gornji par slika, te modela u kojem ne koristimo VAT gubitak - donji par slika. Na gornjim slikama vidimo da model puno bolje razaznaje dva skupa podataka. Kriva klasifikacija se događa samo na mjestima dodira dvaju različitih oznaka. Pripadajući graf s VAT gubitkom upravo na tim mjestima gdje je potrebno dodatno zaglađivanje decizijske granice postiže najveće vrijednosti. Donji grafovi jednostavno nemaju kako naučiti kako zagladiti granicu pa se kriva klasifikacija rasprostire po cijelom skupu ulaznih podataka. Vrijednosti VAT gubitaka također se rasprostiru po mjestima gdje bi decizijska granica trebala biti glađa - po cijelom skupu ulaznih podataka. U procesu učenja s VAT gubitkom vidimo poboljšanje u odnosu na proces učenja bez VAT-a.



Slika 4.3: Prikaz procesa učenja nakon 1000 iteracija

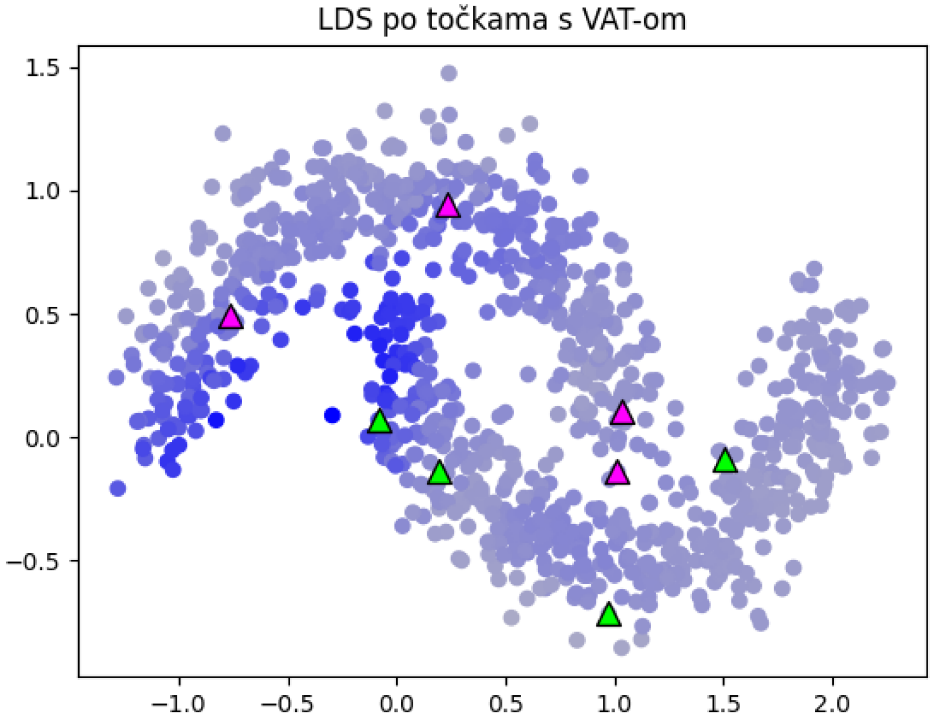
Na slici vidimo blagu promjenu pojedinačnih predikcija u procesu učenja s VAT gubitkom. Prikaz VAT gubitaka se fokusirao na konkretne točke, te se uvelike lokalizirao. Prikaz procesa učenja bez VAT gubitka je ostao približno isti kao u 100. iteraciji.



Slika 4.4: Prikaz procesa učenja nakon 10000 iteracija

Slika 4.4 prikazuje pripadajuće grafove po završetku učenja, točnije nakon 10000 iteracija. Možemo vidjeti da je model koji smo učili s VAT regularizacijom postigao poprilično dobre rezultate, te da je jasno naučio granicu između dvije oznake. Mjera VAT gubitka prikazuje gubitak samo na pojedinim točkama, točno na mjestima gdje se skupovi polumjeseca dodiruju. Model pokušava dodatno zagladiti decizijsku granicu na tim mjestima. Vidimo da predikcija modela koji smo učili nadziranom metodom nije bitno poboljšana, te model nije uspio naučiti ispravno razvrstavati ovaj skup podataka. Pripadajući VAT gubitak je lokaliziran, međutim i dalje nema bitnijeg poboljšanja u predikciji modela.

Ovime smo pokazali blagodat VAT metode da uz relativno mali broj označenih ulaznih primjera možemo provesti proces učenja, kao i zagladiti decizijsku granicu između oznaka poboljšavajući time predikciju modela.



Slika 4.5: Prikaz VAT gubitka po točkama nakon 10. iteracija

Slika 4.5 prikazuje vrijednosti VAT gubitka po ulaznim točkama 2D prostora nakon 10. iteracije učenja modela s VAT gubitkom. Možemo vidjeti da praktički svaka točka ulaznog prostora barem malo pridonosi ukupnom VAT gubitku, odnosno da svaka točka ima određenu vrijednosti VAT gubitka pa time i plavu nijansu. Prikazom VAT gubitka kroz proces učenja vidimo da se VAT gubitak u kasnijim iteracijama lokalizira na mjesta gdje se primjeri različitih oznaka dodiruju, dok na kraju učenja (slika 4.4) imamo samo par točaka koje poprimaju plavu boju u prikazu VAT gubitka.

5. Eksperimenti na skupu podataka MNIST

MNIST (engl. Modified National Institute of Standards and Technology database) je skup monokromatskih slika rukom pisanih znamenki 0-9. Dimenzije ulaznih slika su 28x28 piksela. MNIST se smatra jednostavnim skupom za učenje podataka, stoga se često koristi u dubokom učenju za testiranje metoda i arhitektura. Često se koristi i u edukativne svrhe za provođenje eksperimenata. Sastoji se od velikog broja primjera za učenje, točnije sadrži 60000 primjera za učenje (engl. Training set) i 10000 primjera za testiranje (engl. Test set). Dodatno ćemo iz skupa primjera za učenje izdvojiti skup primjera za validaciju (engl. Validation set). Validacijski skup nam služi kako bi evaluirali hiperparametre modela. Dok ćemo skup primjera za testiranje iskoristiti da provjerimo kvalitetu modela na primjerima koje model nikada nije vidio.

5.1. Postupak nadziranog učenja

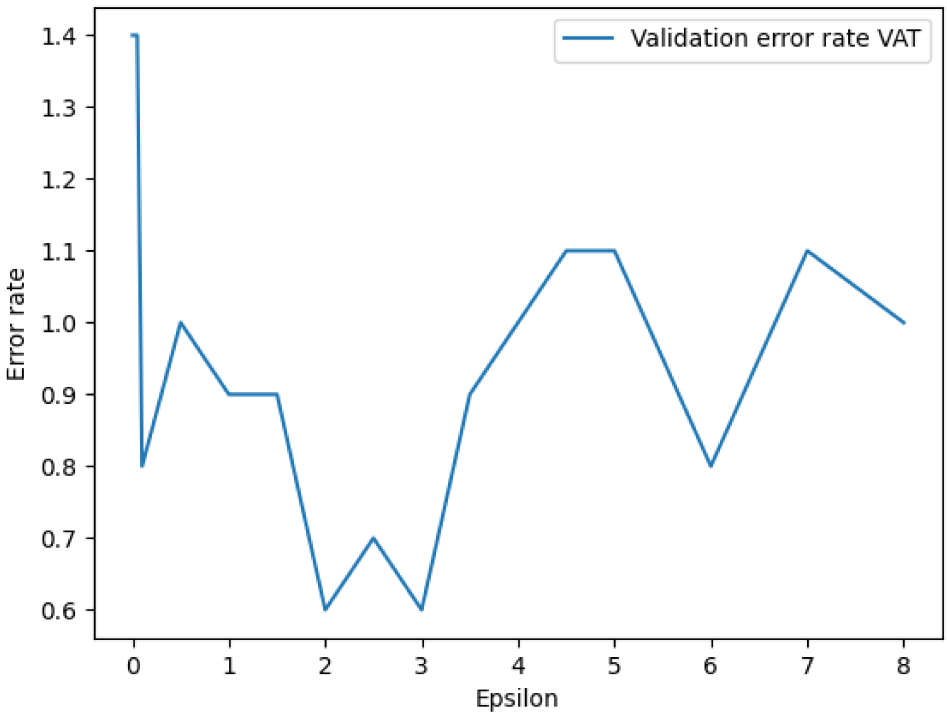
Postupak nadziranog učenja proveo sam nad arhitekturom potpuno povezane neuronske mreže s četiri skrivena sloja veličine 1200, 600, 300 i 150. Na ulaz dovodimo izravnate slike - slike dimenzija 28x28 izravnamo u vektor veličine 784. Značenje nije narušeno, jer svaki puta redove raspoređujemo deterministički na isto mjesto. S obzirom na to da skup sadrži znamenke izlaz našeg modela će imati vektor veličine 10. Na izlazu svakog skrivenog sloja nalazi se normalizacija po grupi, dok se kako aktivacijska funkcija koristi ReLU. Iz skupa primjera za učenje smo izdvojili 1000 slika za validaciju.

5.1.1. Validacija hiperparametra ϵ

Iz skupa za treniranje smo izdvojili 1000 slika u skup za validaciju, te ćemo učenje provoditi na skupu za treniranje veličine 59000 slika. Model nećemo učiti nad skupom za validaciju kako bi bio valjana mjera koja je vrijednost ϵ najbolja. Izdvajanje slika na dva skupa ćemo provesti samo jednom na početku postupka kako bi svaki put kada učimo s različitom vrijednosti hiperparametra ϵ modeli imali iste uvjete za učenje i validaciju. Veličina minigrupe (engl. batch size) postavljena je na 100. Učenja se provode algoritmom Adam, stopa učenja (engl. learning rate) je postavljena na 0.003, te se eksponencijalno smanjivala po stopi 0.6 svakih 5 epoha - 5000 iteracija. Algoritam se izvodio 60000 iteracija, raspodijeljenih u 60 epoha po 1000 iteracija radi izračuna statistike kao i umanjivanju stope učenja. Stopa učenja se počinje smanjivati nakon 30 epoha - 30000 iteracija. Hiperparametri VAT-a koje sam ostavio konstantno na istoj vrijednosti su ξ=1e-6, koeficijent regularizacije α=1, dok je broj uzastopnih iteracija kod izračuna r_{vadv} bio postavljen na K=1. Pogledajmo kako se ponašao model za različite vrijednosti ϵ.

Pogreška za različitu vrijednost ϵ								
ϵ	0.0	0.05	0.1	0.5	1.0	1.5	2.0	2.5
Pogreška (%)	1.4	1.4	0.8	1.0	0.9	0.9	**0.6**	0.7
ϵ	3.0	3.5	4.0	4.5	5.0	6.0	7.0	8.0
Pogreška (%)	**0.6**	0.9	1.0	1.1	1.1	0.8	1.1	1.0

Tablica iznad pokazuje kako se mijenjala pogreška predikcije modela koji smo učili za različite ϵ. Najbolji rezultat je postigao model koji sam učio za vrijednost hiperparametra ϵ=2.0, koji je postigao pogrešku predikcije od 0.6 na validacijskom skupu. Istu vrijednost pogreške predikcije je postignuta i za ϵ=3.0. Prikažimo postignute rezultate i grafički kako bi imali potpuniji uvid u mjerenja koja smo postigli na validacijskom skupu. Na temelju ovih mjerenja odabiremo vrijednost hiperparametra za koji model postiže najbolje rezultate te nad njim provodimo mjerenja na skupu za testiranje. Također, grafove koje ćemo prikazati u nastavku biti će generirani iz postupka učenja modela s tim hiperparametrima.



Slika 5.1: Grafički prikaz promjene pogreške predikcije s obzirom na promjenu ϵ

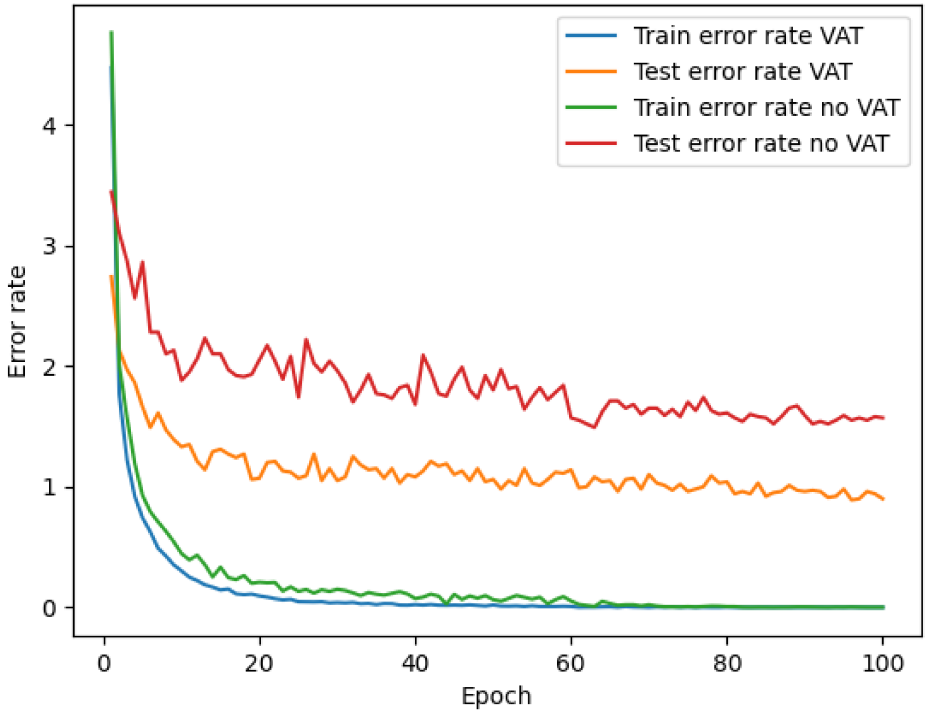
Slika 5.1 prikazuje kako se mijenja pogreška predikcije s obzirom na promjenu hiperparametra ϵ. Ona je zapravo grafička manifestacija vrijednosti prikazanih u tablici, a produkt je dobivenih rezultata generiranih iz provedenih eksperimenata. Vidimo da je model najbolje naučen za vrijednosti ϵ=2.0 i ϵ=3.0.

5.1.2. Rezultati

Prezentirani rezultati dobiveni su na skupu za testiranje od 10000 ulaznih primjera, a učenje je provedeno na sljedeći način. Veličina minigrupe (engl. batch size) postavljen je na 100. Učenja se provode algoritmom Adam. Stopa učenja (engl. learning rate) je postavljena na 0.003, te se eksponencijalno smanjivala po stopi 0.6 svakih 10 epoha - 10000 iteracija. Algoritam se izvodio 100000 iteracija raspodijeljenih u 100 epoha po 1000 iteracija radi izračuna statistike kao i umanjivanju stope učenja. Stopa učenja se počinje smanjivati nakon 50 epoha - 50000 iteracija. Hiperparametri VAT-a koje sam koristio za generiranje grafova i rezultata su ξ=1e-6, koeficijent regularizacije α=1, broj uzastopnih iteracija kod izračuna r_{vadv} bio je postavljen na K=1, dok je ϵ=2.0. Pogledajmo kako se ponašao model za spomenute vrijednosti hiperparametara.

Rezultati	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	1.49
$\epsilon = 2.0$	0.89
Znanstveni rad [11]	0.66

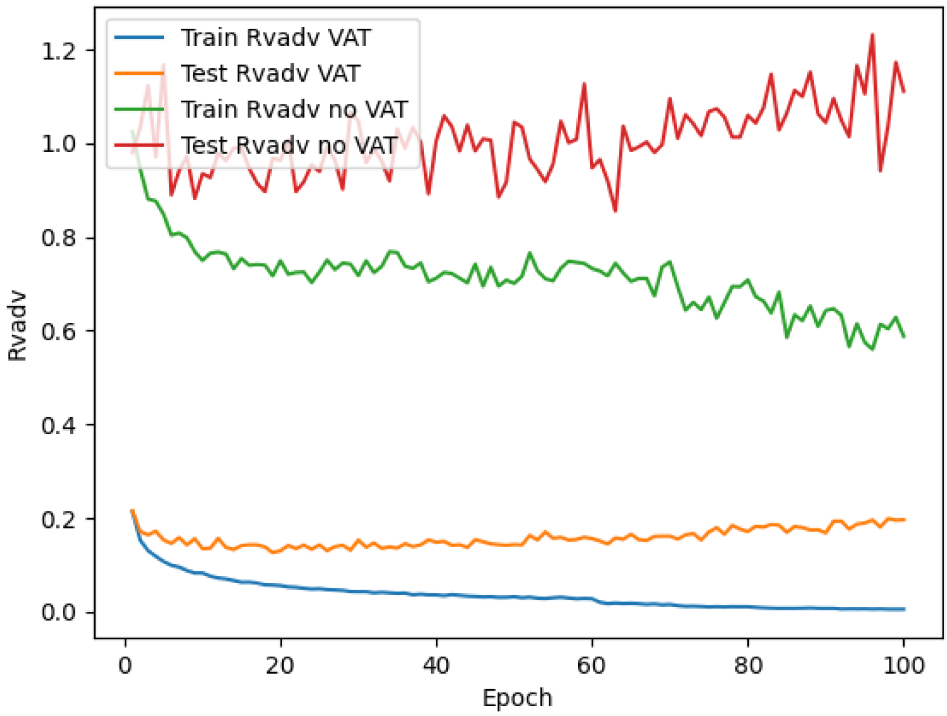
U tablici su prikazane pogreške predikcije koje sam postigao koristeći VAT regularizaciju sa ϵ=2.0 i bez korištenja VAT-a, kao i pogrešku koju su postigli u znanstvenom radu [11]. Vidimo da korištenjem VAT-a u nadziranom postupku učenja i dalje možemo znatno poboljšati predikciju modela, u ovom konkretnom slučaju pogreška je manja za 0.6%. Međutim isto tako možemo vidjeti da su postignuti rezultati lošiji od rezultata iz rada [11].



Slika 5.2: Grafički prikaz promjene pogreške predikcije kroz različite iteracije algoritma

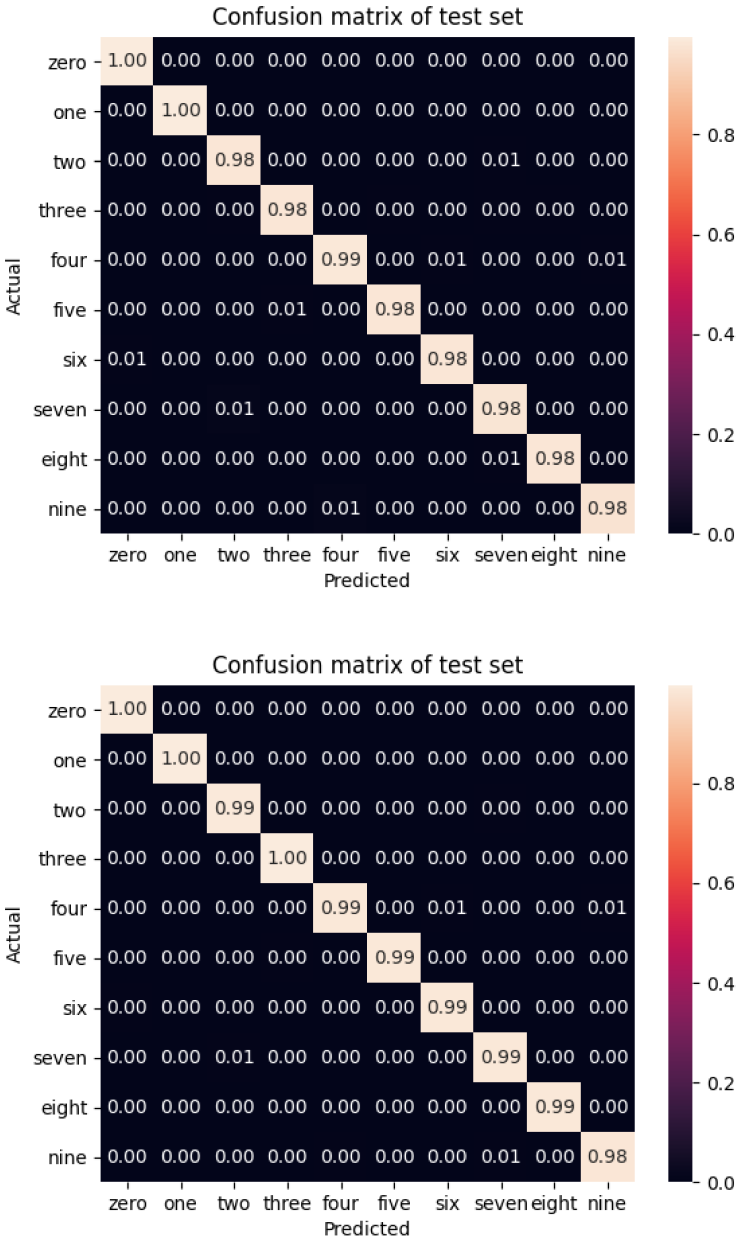
Slika 5.2 prikazuje kako se mijenja pogreška predikcije sa svakom epohom učenja - svako 1000 iteracija. Na slici su prikazane krivulje koje sam izračunao na skupu za učenje te skupu za testiranje sa i bez VAT regularizacije. Očekivano, krivulje koje prikazuje pogrešku predikcije na skupu za učenje padaju puno brže nego one na skupu za testiranje. Možemo vidjeti da se pogreška predikcije na skupu za učenje s VAT-om

brže približava nuli nego krivulja koja ne koristi VAT. Također iz grafa vidimo da je krivulja pogreške predikcije nad skupom za testiranje niže pozicionirana nego ona na kojoj ne koristimo VAT. Možemo zaključiti da VAT nepobitno poboljšava generalizaciju modela.



Slika 5.3: Grafički prikaz promjene VAT gubitka kroz različite iteracije algoritma

Slika 5.3 prikazuje kako se mijenja VAT gubitak kroz proces učenja modela. Vrijednosti su prikazivane nakon svake epohe - 1000 iteracija. Na slici su prikazane krivulje koje sam izračunao na skupu za učenje te skupu za testiranje sa i bez VAT regularizacije. Ukoliko usporedimo krivulje generirane koristeći VAT i one bez VAT-a, možemo primijetiti da VAT neupitno smanjuje virtualni neprijateljski gubitak. Krivulje koje prikazuju postupak koji nije koristio VAT vidimo da poprimaju znatno veće vrijednosti ovog gubitka.



Slika 5.4: Prikaz matrica konfuzije bez VAT regularizacije te s VAT regularizacijom

Slika 5.4 prikazuje matrice konfuzije u kojima nismo koristili VAT regularizaciju (gornja matrica) i ona u kojoj smo koristili (donja matrica). Možemo vidjeti da je VAT doista zagladio decizijsku granicu između pojedinih klasa te poboljšao predikciju. Primjerice model je bolje naučio raspoznavati brojeve 2, 3, 5, 6, 7 i 8.

5.2. Postupak polunadziranog učenja

Postupak se provodi nad modelom iste arhitekture koju sam već opisao u postupku nadziranog učenja. Jedina razlika u slučaju postupka polunadziranog učenja je to što smo dodali Gaussov slučajan šum s očekivanjem 0 i standardnom devijacijom 0.5. Ovakav šum dodajemo zbog stabilizacije rada VAT-a [11]. U polunadziranoj metodi pokušavamo smanjiti broj označenih primjera, a i dalje održati predikcijske rezultate usporedive s onima u nadziranim metodama učenja.

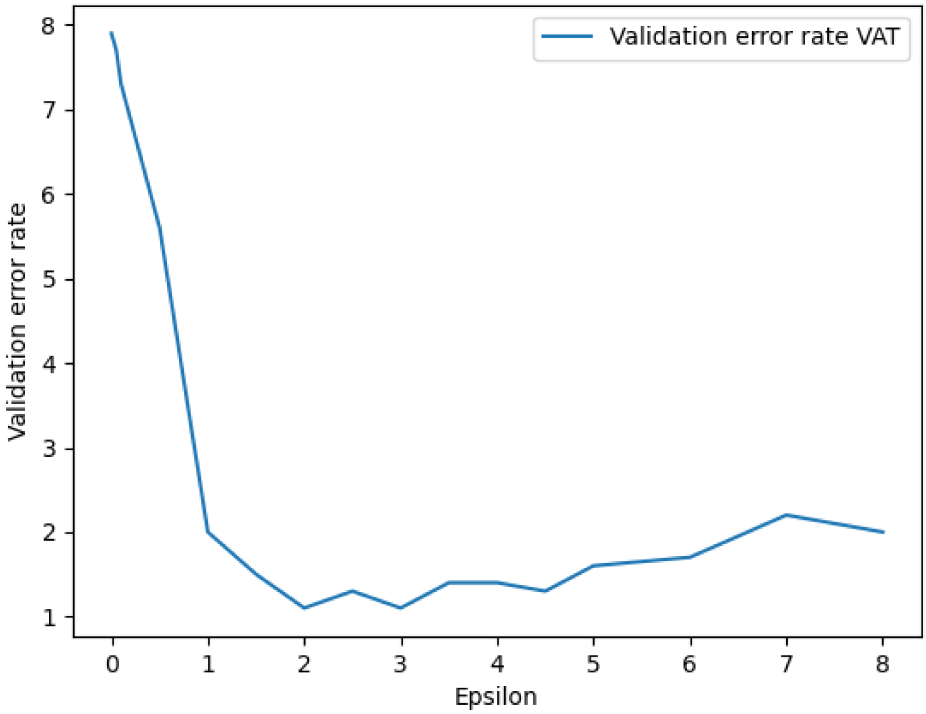
5.2.1. Validacija hiperparametra ϵ s N_l=1000

Iz skupa za treniranje smo izdvojili 1000 slika u skup za validaciju, te ćemo učenje provoditi na skupu za treniranje veličine 59000 slika. Nadalje iz skupa za učenje izdvajamo 1000 primjera kojima ćemo znati oznaku - N_l=1000 označenih primjera. Ostatak od 58000 slika smatrat ćemo neoznačenim primjerima - N_{ul}=58000. Veličina minigrupe za označene primjere bit će 64 slike - nad označenim primjerima računamo gubitak unakrsne entropije, dok će minigrupa nad kojom računamo VAT gubitak biti 256 slika. Skup slika nad kojim računamo VAT gubitak u sebi će sadržavati označene i neoznačene primjere. Učenja se provode algoritmom Adam, stopa učenja (engl. learning rate) je postavljena na 0.003, te se eksponencijalno smanjivala po stopi 0.6 svakih 5 epoha - 5000 iteracija. Algoritam se vrtio 60000 iteracija, raspodijeljenih u 60 epoha po 1000 iteracija radi izračuna statistike kao i umanjivanju stope učenja. Stopa učenja se počinje smanjivati nakon 30 epoha - 30000 iteracija. Hiperparametri VAT-a koje sam ostavio konstantno na istoj vrijednosti su ξ=1e-6, koeficijent regularizacije α=1, dok je broj uzastopnih iteracija kod izračuna r_{vadv} bio postavljen na K=1. Pogledajmo kako se ponašao model za različite vrijednosti ϵ.

Pogreška za različitu vrijednost ϵ								
ϵ	0.0	0.05	0.1	0.5	1.0	1.5	2.0	2.5
Pogreška (%)	7.9	7.7	7.3	5.6	2.0	1.5	**1.1**	1.3
ϵ	3.0	3.5	4.0	4.5	5.0	6.0	7.0	8.0
Pogreška (%)	**1.1**	1.4	1.4	1.3	1.6	1.7	2.2	2.0

Tablica iznad pokazuje kako se mijenjala pogreška predikcije modela koji smo učili za različite ϵ. Najbolji rezultat je postigao model koji sam učio za vrijednost hiperparametra ϵ=2.0 i ϵ=3.0, koji je postigao pogrešku predikcije od 1.1 na validacijskom skupu. Na temelju ovih vrijednosti odabirem hiperparametre za koje provodimo postupak učenja modela, te konačnu mjeru kvalitete naučenog modela računamo na

testnom skupu. Prikažimo postignute rezultate i grafički kako bi imali potpuniji uvid u mjerenja koja smo postigli na validacijskom skupu.



Slika 5.5: Grafički prikaz promjene pogreške predikcije s obzirom na promjenu ϵ

Slika 5.5 prikazuje kako se mijenja pogreška predikcije s obzirom na promjenu hiperparametra ϵ za postupak polunadziranog učenja nad 1000 označenih primjera. Ona je zapravo grafička manifestacija vrijednosti prikazanih u tablici, a produkt je dobivenih rezultata generiranih iz provedenih eksperimenata. Vidimo da je model najbolje naučen za vrijednosti ϵ=2.0 i ϵ=3.0.

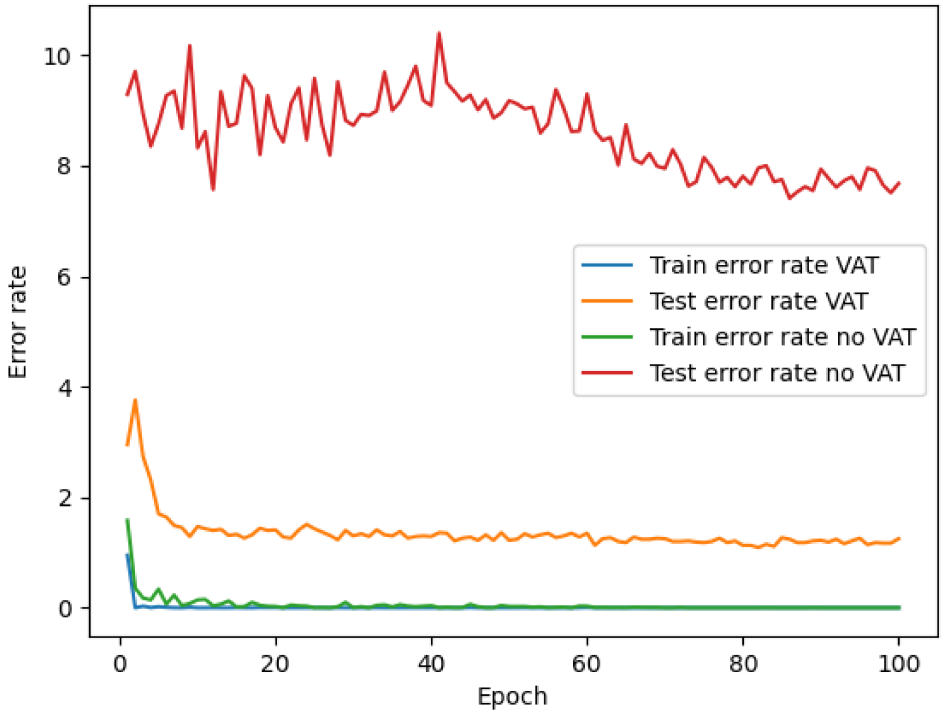
5.2.2. Rezultati s N_l=1000

Prezentirani rezultati dobiveni su na skupu za testiranje od 10000 ulaznih primjera, a učenje je provedeno na sljedeći način. Veličina minigrupe za označene primjere bit će 64 slike - nad označenim primjerima računamo gubitak unakrsne entropije, dok će minigrupa nad kojom računamo VAT gubitak biti 256 slika. Skup slika nad kojim računamo VAT gubitak u sebi će sadržavati označene i neoznačene primjere. Učenje se provodi algoritmom Adam. Stopa učenja (engl. learning rate) je postavljena na 0.003, te se eksponencijalno smanjivala po stopi 0.6 svakih 10 epoha - 10000 iteracija.

Algoritam se vrtio 100000 iteracija raspodijeljenih u 100 epoha po 1000 iteracija radi izračuna statistike kao i umanjivanju stope učenja. Stopa učenja se počinje smanjivati nakon 50 epoha - 50000 iteracija. Hiperparametri VAT-a koje sam ostavio konstantno na istoj vrijednosti su ξ=1e-6, koeficijent regularizacije α=1, dok je broj uzastopnih iteracija kod izračuna r_{vadv} bio postavljen na K=1, a ϵ=2.0 i ϵ=3.0. Pogledajmo kako se ponašao model za spomenute vrijednosti hiperparametara.

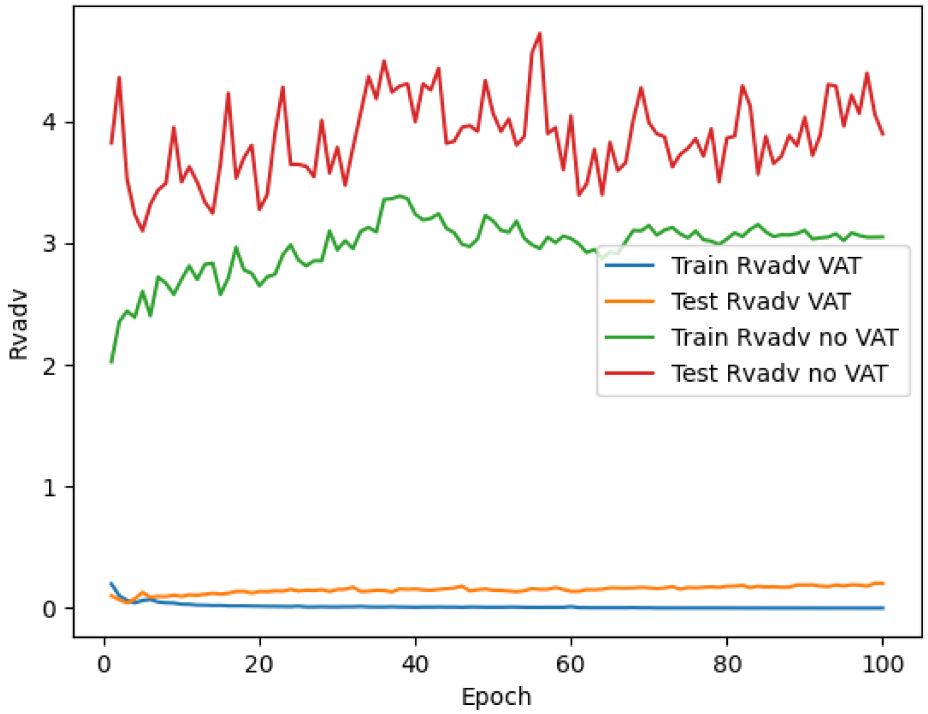
Rezultati N_l=1000	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	7.41
ϵ = 2.0	1.21
ϵ = 3.0	1.17
Znanstveni rad [11]	1.27

U tablici su prikazane pogreške predikcije koje sam postigao koristeći VAT regularizaciju sa ϵ=2.0, ϵ=3.0 i bez korištenja VAT-a, kao i pogrešku koju su postigli u znanstvenom radu [11]. U ovome slučaju sam uspio reproducirati rezultate koje su dobili u radu [11]. Iz tablice neupitno vidimo da VAT poboljšava predikciju modela.



Slika 5.6: Grafički prikaz promjene pogreške predikcije kroz različite iteracije algoritma

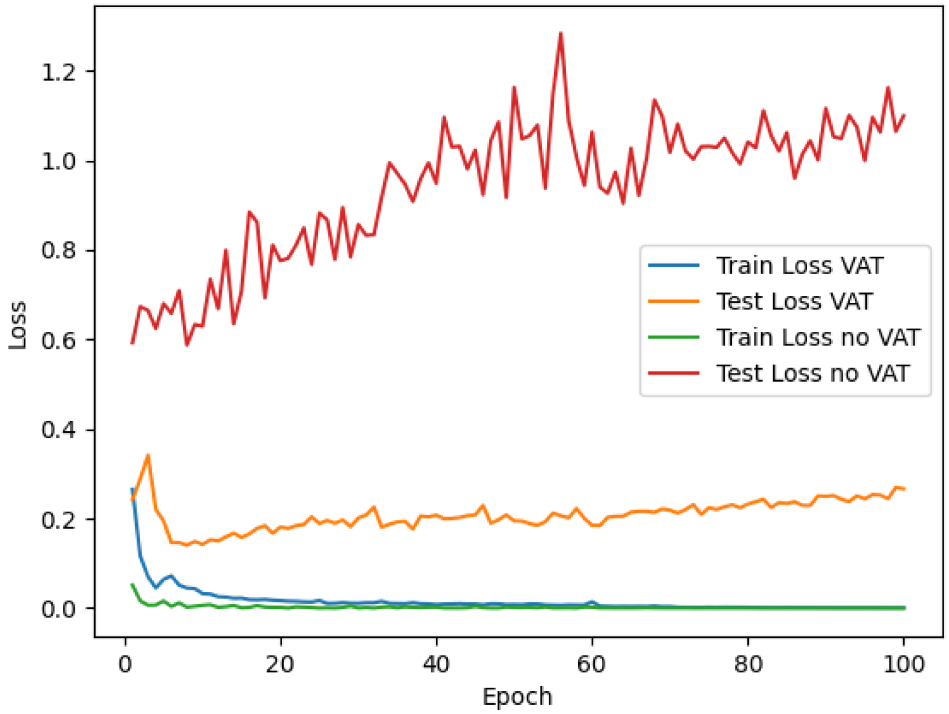
Slika 5.6 prikazuje kako se mijenja pogreška predikcije sa svakom epohom učenja - svako 1000 iteracija. Na slici su prikazane krivulje koje sam izračunao na skupu za učenje te skupu za testiranje sa i bez VAT regularizacije. Očekivano krivulje koje prikazuju pogrešku predikcije na skupu za učenje padaju puno brže nego one na skupu za testiranje. Možemo vidjeti da se pogreška predikcije na skupu za učenje s VAT-om brže približava nuli nego krivulja koja ne koristi VAT. Također iz grafa vidimo da je krivulja pogreške predikcije nad skupom za testiranje niže pozicionirana nego ona na kojoj ne koristimo VAT. Vidimo i da je krivulja dobivena na skupu za testiranje s VAT-om uravnoteženijih vrijednosti od one koja nije koristila VAT. Možemo zaključiti da VAT nepobitno poboljšava generalizaciju modela.



Slika 5.7: Grafički prikaz promjene VAT gubitka kroz različite iteracije algoritma

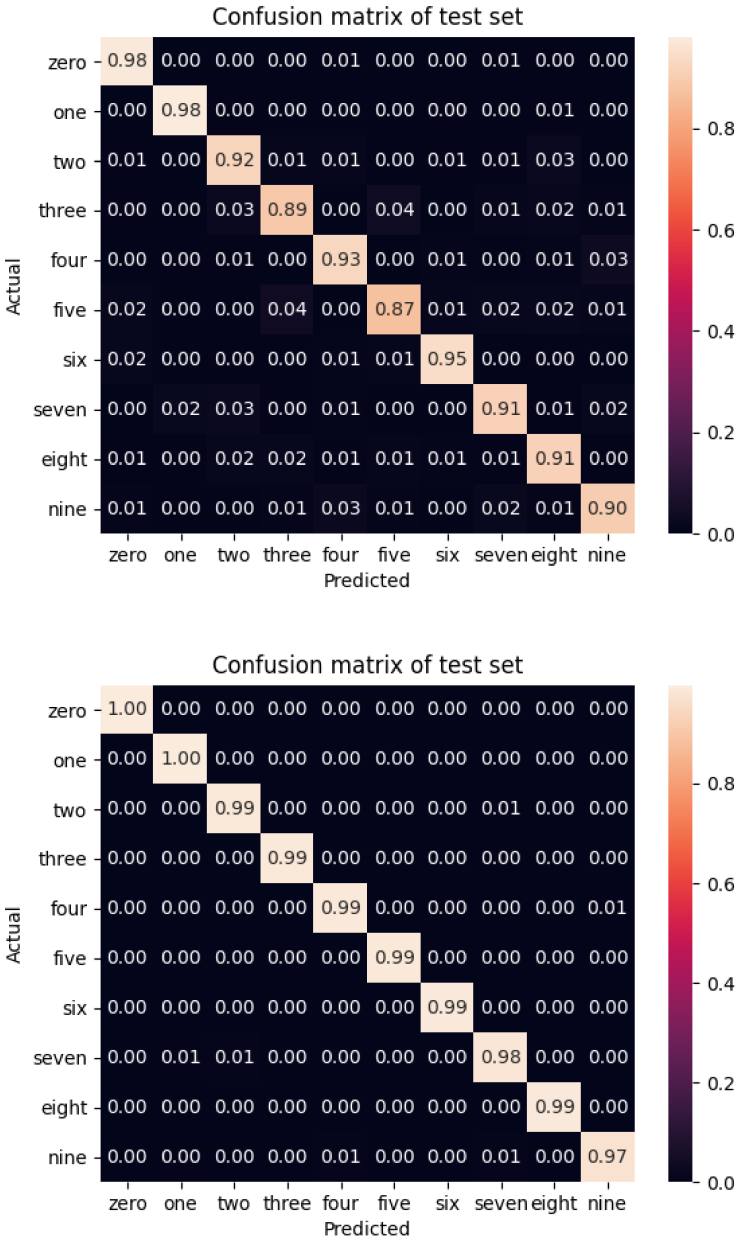
Slika 5.7 prikazuje promjenu VAT gubitka sa svakom epohom - svako 1000 iteracija. Na slici su prikazane krivulje koje sam izračunao na skupu za učenje te skupu za testiranje sa i bez VAT regularizacije. Vidimo kako je VAT gubitak puno veći na krivuljama generiran u postupku učenja bez VAT regularizacije. Upravo to dovodi do velike razlike u pogrešci predikcije između postupka učenja sa i bez VAT-a. Možemo zaključiti kako je distribucija naučena uz VAT gubitak lokalno glađa od one koja nije koristila VAT. Možemo vidjeti i da korištenje VAT gubitka u procesu učenja smanjuje

VAT gubitak i na skupu za testiranje, što nužno upučuje na to da VAT povećava glatkoću naučene distribucije primjera, te time poboljšava predikciju neviđenih primjera.



Slika 5.8: Grafički prikaz promjene funkcije gubitka kroz različite iteracije algoritma

Slika 5.8 prikazuje promjenu ukupnog gubitka kroz epohe učenja - svako 1000 iteracija. Na slici su prikazane krivulje koje sam izračunao na skupu za učenje te skupu za testiranje sa i bez VAT regularizacije. Na početku krivulja skupa za učenje s VAT-a poprima veće vrijednosti, a u kasnijim epohama se praktički izjednači s vrijednostima koje poprima krivulja skupa za učenje bez VAT-a. Razlog je u tome što u postupku učenja s VAT-om postoji dodatna komponenta VAT gubitka, te je potrebno vrijeme da se ona smanji. Vidimo da je krivulja dobivena na skupu za testiranje bez VAT-a i dalje poprima veće vrijednosti od krivulje dobivene na skupu za testiranje s VAT-om. Također možemo primijetiti da je krivulja pogreške na skupu na testiranje bez VAT-a konstantno u blagom trendu porasta vrijednosti.



Slika 5.9: Prikaz matrica konfuzije bez VAT regularizacije te s VAT regularizacijom

Slika 5.9 prikazuje matricu konfuzije dobivene kroz opisani proces učenja. Gornja matrica generirana je iz postupka učenja bez VAT-a, dok je donja matrica generirana putem postupka s VAT regularizacijom uz hiperparametar ϵ=3.0. Iz matrica možemo vidjeti da VAT popravlja predikciju svih klasa. Model koji koristi VAT je puno sigurniji u predikciji nego model bez VAT-a.

5.2.3. Validacija hiperparametra ϵ s N_l=100

Iz skupa za treniranje smo izdvojili 1000 slika u skup za validaciju, te ćemo učenje provoditi na skupu za treniranje veličine 59000 slika. Nadalje iz skupa za učenje izdvajamo 100 primjera kojima ćemo znati oznaku - N_l=100 označenih primjera. Ostatak od 58900 slika smatrat ćemo neoznačenim primjerima - N_{ul}=58900. Veličina minigrupe za označene primjere bit će 64 slike - nad označenim primjerima računamo gubitak unakrsne entropije, dok će minigrupa nad kojom računamo VAT gubitak biti 256 slika. Skup slika nad kojim računamo VAT gubitak u sebi će sadržavati označene i neoznačene primjere. Učenja se provode algoritmom Adam. Stopa učenja (engl. learning rate) je postavljena na 0.002, te se eksponencijalno smanjivala po stopi 0.7 svakih 5 epoha - 5000 iteracija. Algoritam se vrtio 60000 iteracija, raspodijeljenih u 60 epoha po 1000 iteracija radi izračuna statistike kao i umanjivanju stope učenja. Stopa učenja se počinje smanjivati nakon 30 epoha - 30000 iteracija. Hiperparametri VAT-a koje sam ostavio konstantno na istoj vrijednosti su ξ=1e-6, koeficijent regularizacije α=1, dok je broj uzastopnih iteracija kod izračuna r_{vadv} bio postavljen na K=1. Pogledajmo kako se ponašao model za različite vrijednosti ϵ.

Pogreška za različitu vrijednost ϵ								
ϵ	0.0	0.05	0.1	0.5	1.0	2.0	3.0	3.5
Pogreška (%)	20	45	44.7	41.2	56	76.8	61.2	2.3
ϵ	4.0	4.5	5.0	5.5	6.0	6.5	7.0	8.0
Pogreška (%)	1.4	**1.3**	**1.3**	**1.3**	1.7	1.7	2.1	1.9

Tablica prikazuje kako se mijenja pogreška predikcije na skupu za validaciju s obzirom na promjenu hiperparametara ϵ. Vidimo da kroz provedenu evaluaciju model poprima vrijednost pogreške predikcije jednaku 1.3% za čak tri vrijednosti ϵ. Vrijednosti ϵ za koji model poprimam minimum na validacijskom skupu su 4.5, 5.0 i 5.5. Stoga ćemo provesti dodatne testove za sve od navedenih vrijednosti te ih prikazati u grafu.

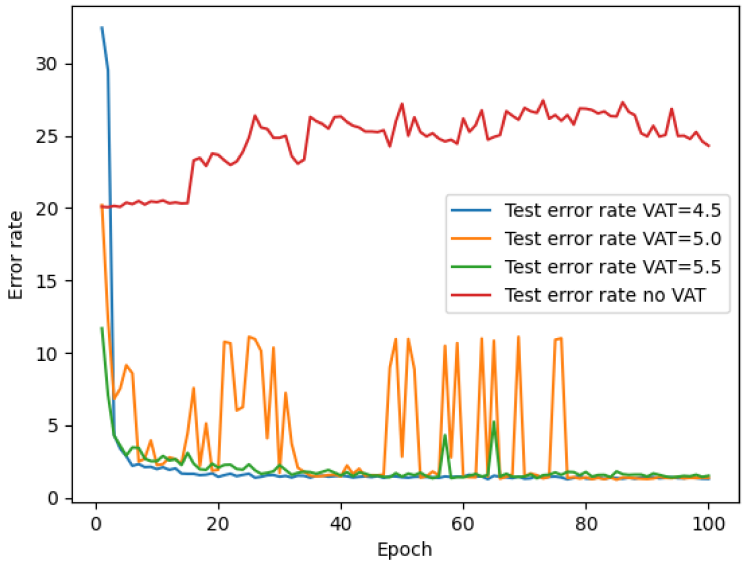
Generalno je poznato da se na validacijskom skupu provodi analiza kako bismo odredili za koji hiperparametar model postiže najbolje vrijednosti predikcije. Međutim na skupu od 100 označenih primjera možemo primijetiti i pojavu prenaučenosti koju nismo imali kod većeg broja označenih primjera, primjerice 1000 označenih primjera. Ovime izbor hiperparametara postaje još važniji. Također vidimo da smanjenjem broja označenih primjera opada i mogućnost modela da generalizira - porast predikcijske pogreške na skupu primjera koje model nikada nije vidio.

5.2.4. Rezultati s N_l=100

Prezentirani rezultati dobiveni su na skupu za testiranje koji se sastoji od 10000 ulaznih primjera. Učenje se provodilo istim optimizacijskim postupkom kao što smo opisali u odjeljku o validaciji hiperparemetara za 100 označenih primjera. Jedina promjena je bila ta što smo postupak učenja provodili 100000 iteracija. Pogledajmo vrijednosti pogrešaka predikcije koje je model postigao za najbolje ϵ.

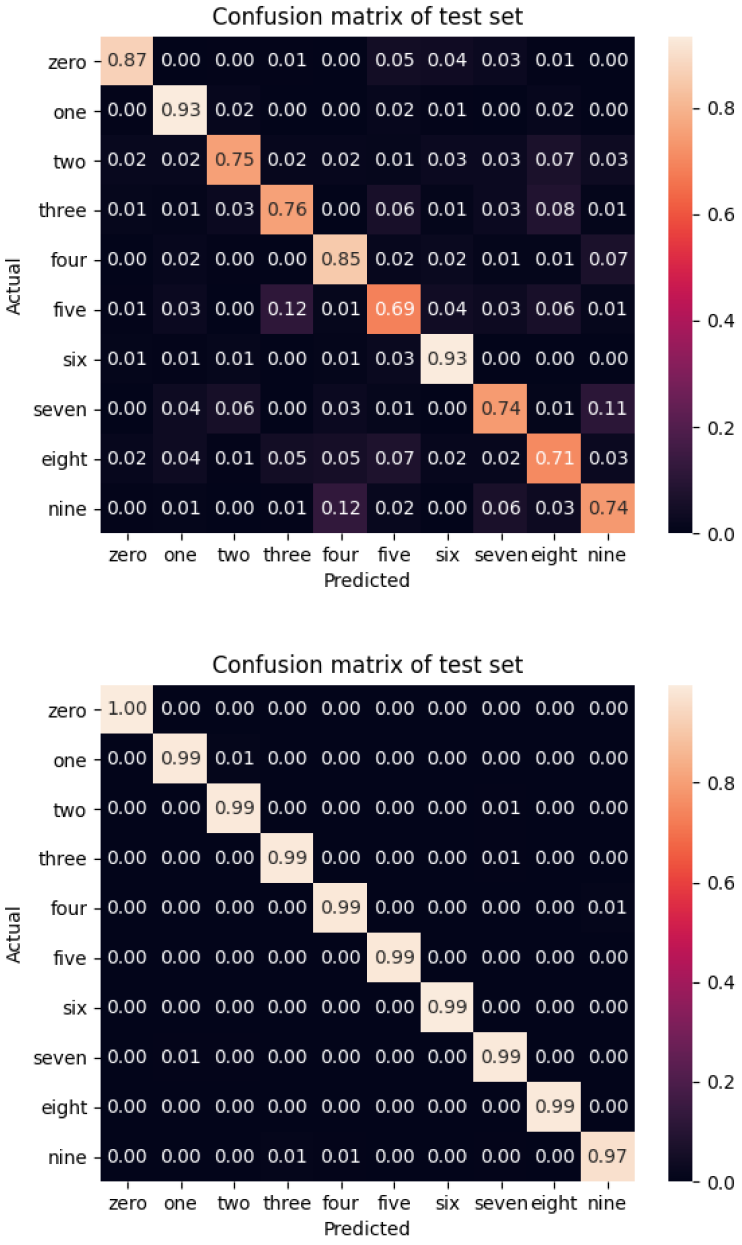
Rezultati N_l=100	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	20.56
$\epsilon = 4.5$	1.28
$\epsilon = 5.0$	1.27
$\epsilon = 5.5$	1.32
Znanstveni rad [11]	1.36

Tablica prikazuje rezultate koje su modeli postigli na skupu za testiranje. Rezultati iz rada su uspješno reproducirani, a iz tablice možemo iščitati da VAT poboljšava predikciju modela jer postupak u kojem nismo koristili VAT značajno zaostaje za onim modelima koje smo učili uz VAT gubitak. To je svakako očekivano s obzirom na to da se nadzirano učenje provodilo nad samo 100 označenih primjera.



Slika 5.10: Grafički prikaz promjene pogreške predikcije kroz različite iteracije algoritma

Slika 5.10 prikazuje promjenu pogreške predikcije koje smo postigli na modelima treniranim s različitim vrijednostima ϵ (4.5, 5.0, 5.5), modela treniranog bez VAT metode. Na slici možemo vidjeti da graf generiran s rezultatima iz modela sa ϵ=5.0 ima poraste u pogrešci predikcije, međutim nakon 75. epohe pravilno konvergira u istu točku kao i ostali postupci. Graf generiran podatcima iz modela koji nije koristio VAT ima znatno lošiju predikciju. To je očekivano, jer se nadzirano učenje provodi na 100 označenih primjera te je model ušao u područje prenaučenosti.



Slika 5.11: Prikaz matrica konfuzije bez VAT regularizacije te s VAT regularizacijom

Slika 5.11 prikazuje matricu konfuzije dobivene kroz opisani proces učenja. Gornja matrica generirana je iz postupka učenja bez VAT-a, dok je donja matrica generirana putem postupka s VAT regularizacijom uz hiperparametar ϵ=5.5. Iz matrica možemo vidjeti da VAT popravlja predikciju svih klasa.

6. Eksperimenti na skupovima podataka SVHN i CIFAR-10

6.1. Skup podataka SVHN

SVHN (engl. The Street View House Numbers) je skup podataka dobiven iz stvarnog svijeta, a služi za provođenje postupaka učenja dubokih modela. Skup podataka je jednostavan jer ne zahtijeva pretprocesiranje i formatiranje slika. Sastoji se od znamenaka skupljenih sa slika kućnih brojeva koje je uslikao Google dok je izrađivao projekt Google Street View. Možemo ga smatrati kao svojevrsna nadogradnja na skup podataka MNIST. SVHN je kompleksniji od MNIST-a jer su slike RGB formata, te su uslikane u stvarnim okruženjima. Sastoji se od 10 klasa, po jedna za svaku znamenku. Sadrži 73257 slika za treniranje, 26032 za testiranje, te 531131 dodatnih slika. Za provođenje eksperimenata u ovome radu nismo koristili dodatne slike. Dimenzije slika su 32x32.

6.2. Skup podataka CIFAR-10

CIFAR (engl. Canadian Institute For Advanced Research) je skup podataka jako popularan u provođenju eksperimenata u području računarnog vida. Sastoji se od 10 klasa koje su redom: zrakoplov, auto, ptica, mačka, jelen, pas, žabe, konji, brodovi i kamioni. Skup se sastoji od 60000 slika podijeljenih u manje skupine po 10000 slika. 5 manjih skupina se koristi za treniranje dok je 1 za testiranje. Dimenzije slika su 32x32. Ovaj skup se smatra kompleksnijim od do sada spomenutih u ovom radu.

6.3. Arhitekture modela

Predstavit ćemo modele koje koristimo u provođenju eksperimenata. Modeli su istovjetni s onim u znanstvenom radu [11] kako bi usporedbu dobivenih rezultata smatrali valjanima.

Conv-Small on SVHN	Conv-Small on CIFAR-10	Conv-Large
32×32 RGB image		
3×3 conv. 64 lReLU 3×3 conv. 64 lReLU 3×3 conv. 64 lReLU	3×3 conv. 96 lReLU 3×3 conv. 96 lReLU 3×3 conv. 96 lReLU	3×3 conv. 128 lReLU 3×3 conv. 128 lReLU 3×3 conv. 128 lReLU
2×2 max-pool, stride 2 dropout, $p = 0.5$		
3×3 conv. 128 lReLU 3×3 conv. 128 lReLU 3×3 conv. 128 lReLU	3×3 conv. 192 lReLU 3×3 conv. 192 lReLU 3×3 conv. 192 lReLU	3×3 conv. 256 lReLU 3×3 conv. 256 lReLU 3×3 conv. 256 lReLU
2×2 max-pool, stride 2 dropout, $p = 0.5$		
3×3 conv. 128 lReLU 1×1 conv. 128 lReLU 1×1 conv. 128 lReLU	3×3 conv. 192 lReLU 1×1 conv. 192 lReLU 1×1 conv. 192 lReLU	3×3 conv. 512 lReLU 1×1 conv. 256 lReLU 1×1 conv. 128 lReLU
global average pool, 6×6 → 1×1		
dense 128 → 10	dense 192→ 10	dense 128→ 10
10-way softmax		

Slika 6.1: Arhitekture modela korištene u eksperimentima na SVHN i CIFAR-10 skupovima

Vidimo da se modeli sastoje od blokova postavljenih po tri u nizu nakon kojih dolaze slojevi sažimanja (max-pool) te dropout s vjerojatnosti odbacivanja 0.5. Nakon zadnjeg takvog bloka se nalazi sloj globalnog sažimanja prosjekom koji se spaja na potpuno povezani sloj. Izlaz našega modela je softmax funkcija.

Blokovi se sastoje od konvolucijskih slojeva različitih vrijednosti ulaznih i izlaznih mapa te jezgara veličine 3x3 i 1x1. Mape značajki iz konvolucijskih slojeva prolaze kroz normalizaciju po grupi, nakon čega prolaze kroz aktivacijsku funkciju LReLU. LReLU je skraćenica za LeakyReLU. Ova funkcija nam služi za unošenje nelinearnosti u model te ju smatramo nadogradnjom na funkciju ReLU. LReLU se u području od $-\infty$ do 0 ponaša kao linearna funkcija određenog nagiba. Nagib je proizvoljan te se zadaje kao parametar - u našem slučaju je 0.1. Na suprotnoj strani imamo funkciju ReLU koja se na istom području ponaša tako da poprima vrijednosti 0. Na području od 0 do $+\infty$ obje funkcije se ponašaju identično, linearno s nagibom 1.

6.4. Eksperimenti SVHN

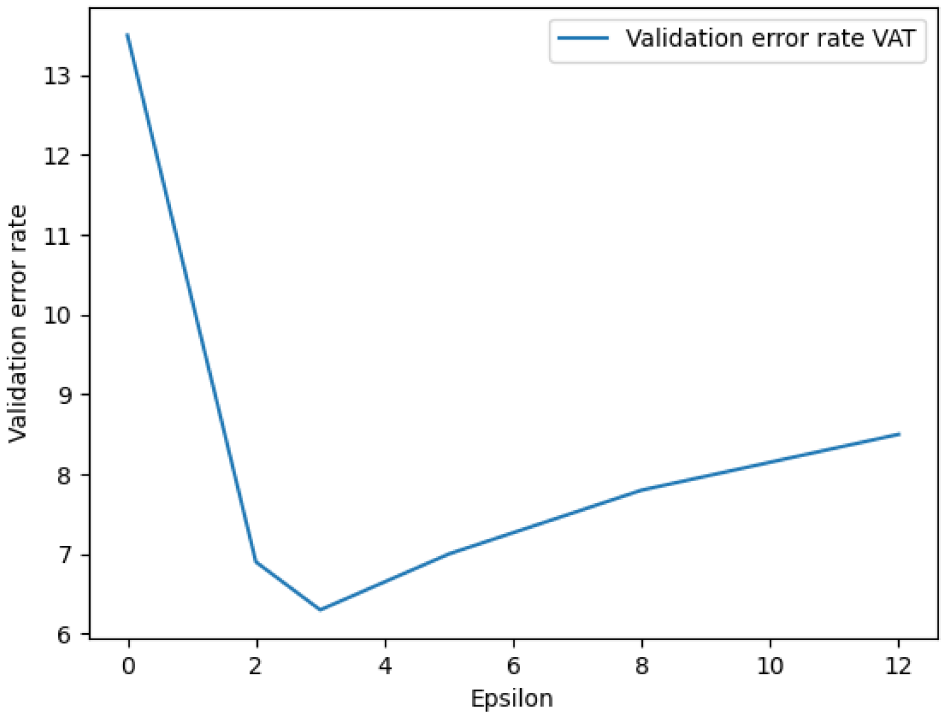
Od 73257 slika iz skupa za treniranje smo odvojili 1000 slika za validaciju. Učenje smo provodili na ostatku od 72257 slika, od čega smo odvojili još 1000 slika u skup označenih dok je ostatak od 71257 slika ostao neoznačen. Veličinu minigrupe označenih primjera postavili smo na 32 dok je veličina minigrupe neoznačenih primjera jednaka 128. VAT gubitak računamo nad označenim i neoznačenim primjerima. Proces učenja modela provodimo algoritmom Adam, stopa učenja je postavljena na 0.001, te se eksponencijalno smanjuje po stopi 0.9 za svaku epohu - 1000 iteracija. Proces učenja smo provodili 48000 epoha podijeljenih u 48 epoha po 1000 iteracija. nakon 32 epohe počinje smanjivanje stope učenja. Hiperparametri VAT-a koje sam ostavio konstantno na istoj vrijednosti su ξ=1e-6, koeficijent regularizacije α=1, dok je broj uzastopnih iteracija kod izračuna r_{vadv} bio postavljen na K=1.

6.4.1. Validacija hiperparematra ϵ

Pogledajmo kako se ponašao model za različite vrijednosti ϵ učenih na Conv-Small modelu sa slike 6.1.

Pogreška za različitu vrijednost ϵ						
ϵ	0.0	2.0	3.0	5.0	8.0	12.0
Pogreška (%)	13.5	6.9	**6.3**	7.0	7.8	8.5

Tablica iznad pokazuje kako se mijenjala pogreška predikcije modela koji smo učili za različite ϵ. Najbolji rezultat je postigao model koji sam učio za vrijednost hiperparametra ϵ=3.0, koji je postigao pogrešku predikcije od 6.3 na validacijskom skupu. Na temelju ovih vrijednosti odabirem hiperparametre za koje provodimo postupak učenja modela, te konačnu mjeru kvalitete naučenog modela računamo na testnom skupu. Prikažimo postignute rezultate i grafički kako bi imali potpuniji uvid u mjerenja koja smo postigli na validacijskom skupu.



Slika 6.2: Grafički prikaz promjene pogreške predikcije s obzirom na promjenu ϵ

Pogledajmo kako se ponašao model za različite vrijednosti ϵ učenih na Conv-Large modelu sa slike 6.1.

Pogreška za različitu vrijednost ϵ						
ϵ	0.0	2.0	3.0	5.0	8.0	12.0
Pogreška (%)	11.6	6.5	**6.0**	6.4	7.4	7.6

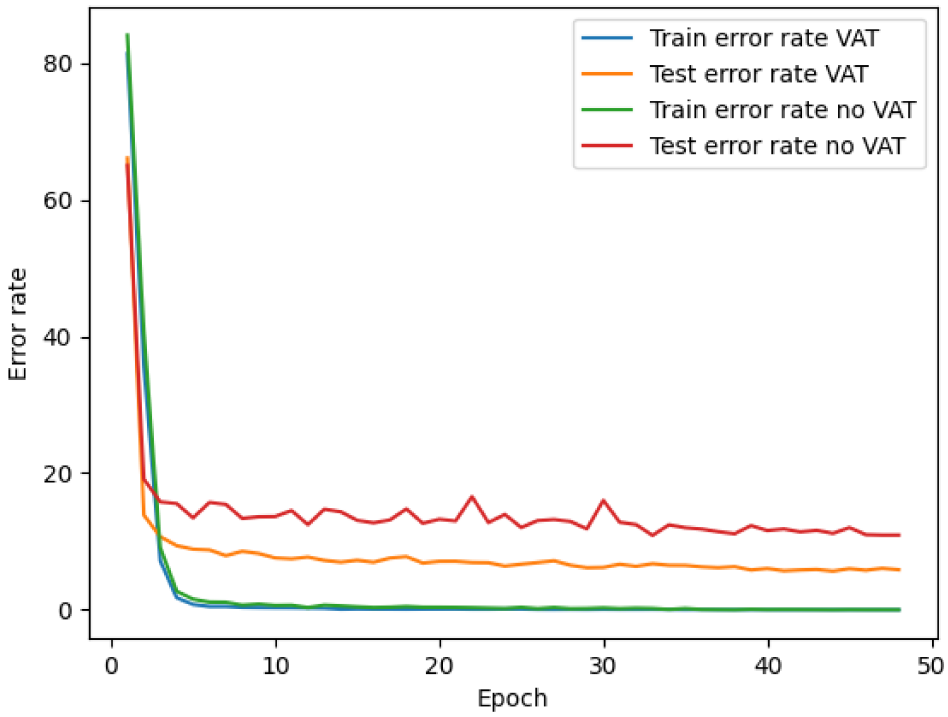
Tablica iznad pokazuje kako se mijenjala pogreška predikcije modela koji smo učili za različite ϵ. Najbolji rezultat je postigao model koji sam učio za vrijednost hiperparametra ϵ=3.0, koji je postigao pogrešku predikcije od 6.0 na validacijskom skupu. Na temelju ovih vrijednosti odabirem hiperparametre za koje provodimo postupak učenja modela, te konačnu mjeru kvalitete naučenog modela računamo na testnom skupu. Vidimo da se u slučaju promjene arhitekture nije promijenila vrijednost ϵ za koji se postižu najbolji rezultati.

6.4.2. Rezultati

Postupak učenja se provodio na isti način kao što je opisano u prethodnom odjeljku vezanom za validaciju hiperparametara. Vrijednosti koje ćemo prezentirati dobivene su na skupu za testiranje veličine 26032 ulaznih primjera. Pogledajmo vrijednosti pogreške predikcije koje je model Conv-Small postigao za najbolji ϵ.

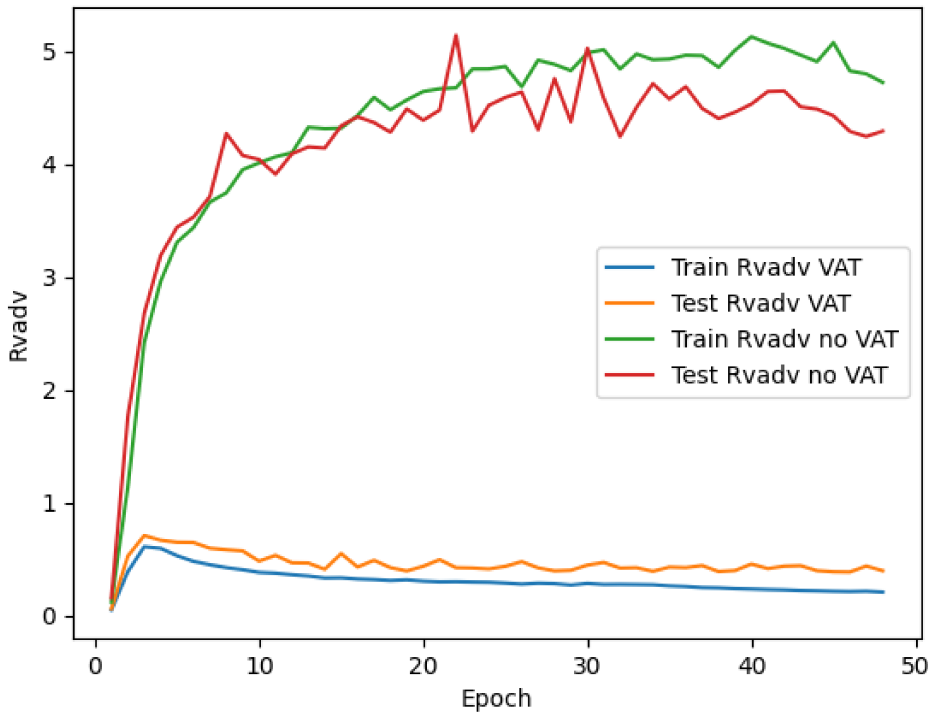
Rezultati Conv-Small N_l=1000	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	13.5
ϵ = 3.0	5.9
Znanstveni rad [11]	6.83

Tablica prikazuje rezultate koje je model Conv-Small postigao na skupu za testiranje. Rezultati iz rada su uspješno reproducirani.



Slika 6.3: Grafički prikaz promjene pogreške predikcije kroz različite iteracije algoritma

Slika 6.3 prikazuje promjenu pogreške predikcije po epohama na skupu za treniranje sa i bez VAT gubitka, te skupu za testiranje sa i bez VAT gubitka. Krivulje se ponašaju očekivano.



Slika 6.4: Grafički prikaz promjene VAT gubitka kroz različite iteracije algoritma

Slika 6.4 prikazuje promjenu VAT gubitka po epohama na skupu za treniranje sa i bez VAT gubitka, te skupu za testiranje sa i bez VAT gubitka. Krivulje koje smo generirali modelom koji smo trenirali s VAT gubitkom se nalaze puno niže na grafu te vidimo da je vrijednost gubitka u konstantnom opadanju. Dok krivulje koje su generirane iz modela koji se nije trenirao VAT metodom zadržavaju konstantni trend rasta. Ako uzmemo u obzir smanjenje pogreške predikcije korištenjem VAT metode i konstantan silazni trend krivulja modela treniranih VAT-om dolazimo do zaključka da metoda poboljšava predikciju.

Pogledajmo vrijednosti pogreške predikcije koje je model Conv-Large postigao.

Rezultati Conv-Large N_l=1000	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	10.6
ϵ = 3.0	5.13
VAT + EntMin	4.57
Znanstveni rad [11]	5.77

Tablica prikazuje rezultate koje je model Conv-Large postigao na skupu za testiranje.

6.5. Eksperimenti CIFAR-10

Za postupak validacije hiperparametara smo iz skupa za treniranje izdojili 1000 slika. Broj označenih primjera je 4000, dakle neoznačenih primjera u skupu za učenje je ostalo 45000. Veličinu minigrupe označenih primjera postavili smo na 32 dok je veličina minigrupe neoznačenih primjera jednaka 100. VAT gubitak računamo nad označenim i neoznačenim primjerima. Proces učenja modela provodimo algoritmom Adam. Stopa učenja je postavljena na 0.001, te se eksponencijalno smanjuje po stopi 0.9 za svaku epohu - 1000 iteracija. Proces učenja smo provodili 100000 iteracija podijeljenih u 100 epoha po 1000 iteracija. Nakon 32 epohe počinje smanjivanje stope učenja. Hiperparametri VAT-a koje sam ostavio konstantno na istoj vrijednosti su ξ=1e-6, koeficijent regularizacije α=1, dok je broj uzastopnih iteracija kod izračuna r_{vadv} bio postavljen na K=1.

6.5.1. Validacija hiperparematra ϵ

Pogledajmo kako se ponašao model za različite vrijednosti ϵ učenih na Conv-Small modelu sa slike 6.1.

Pogreška za različitu vrijednost ϵ					
ϵ	0.0	3.0	5.0	6.0	8.0
Pogreška (%)	33.1	21.5	19.0	19.3	20.9

Tablica iznad pokazuje kako se mijenjala pogreška predikcije modela koji smo učili za različite ϵ. Najbolji rezultat je postigao model koji sam učio za vrijednost hiperparametra ϵ=5.0, koji je postigao pogrešku predikcije od 19.0 na validacijskom skupu.

Pogledajmo kako se ponašao model za različite vrijednosti ϵ učenih na Conv-Large modelu sa slike 6.1.

Pogreška za različitu vrijednost ϵ					
ϵ	0.0	3.0	5.0	6.0	8.0
Pogreška (%)	32.5	20.6	18.1	18.5	19.8

Tablica iznad pokazuje kako se mijenjala pogreška predikcije modela koji smo učili za različite ϵ. Najbolji rezultat je postigao model koji sam učio za vrijednost hiperparametra ϵ=5.0, koji je postigao pogrešku predikcije od 18.1 na validacijskom skupu. Ovim postupkom odabrali smo najbolji hiperparametar ϵ te ćemo s njima dobiti konačnu mjeru kvalitete metoda na skupu za testiranje.

6.5.2. Rezultati

Vrijednosti koje smo koristili za učenje modela iste su kao u prethodnom odjeljku o validaciji hiperparametara. Vrijednosti koje ćemo prezentirati dobivene su na skupu za testiranje veličine 10000 primjera. Učenje se provodimo 200000 iteracija. Pogledajmo vrijednosti pogreške predikcije koje je model Conv-Small postigao za najbolji ϵ.

Rezultati Conv-Small N_l=4000	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	32.07
ϵ = 5.0	15.96
Znanstveni rad [11]	14.87

Tablica prikazuje rezultate koje je model Conv-Small postigao na skupu za testiranje.

Tablica prikazuje rezultate koje je model Conv-Large postigao na skupu za testiranje. Rezultati iz rada su ptibližno reproducirani.

Rezultati Conv-Large N_l=4000	
Metoda	**Progreška predikcije (%)**
Bez VAT-a	31.78
ϵ = 5.0	15.48
Znanstveni rad [11]	14.18

7. Zaključak

Prevedeni eksperimenti u ovome radu potvrdili su nam da je VAT kao regularizacijska metoda utjecala pozitivno na predikcije modela kod nadziranog kao i kod polunadziranog načina učenja. Potvrdila se činjenica o glavnoj blagodati VAT-a, zaglađivanju decizijske granice što za posljedicu ima poboljšanje klasifikacije. Takav učinak poticanja glatkoće decizijske granice postižemo zahtijevanjem da neoznačeni primjeri imaju što sličniju predikciju kao i njihove perturbirane inačice. VAT to postiže kao regularizacijski izraz koji dodajemo na funkciju gubitka, stoga ga smatramo regularizacijskim postupkom.

Polunadzirani algoritmi učenja modela općenito koriste označene i neoznačene podatke. Industrija se susreće upravo s poteškoćama označavanja podataka. Problemi u označavanju podataka mogu biti razni. Primjerice, potreba za specifičnim znanjima u pojedinim područjima, manjak računarnih te financijskih resursa kao i nekonzistentnost u označavanju. Nekonzistentnost se može riješiti donošenjem normi za označavanje na razini tima. Manjak financijskih resursa nije toliko izražen kod informatičkih giganata kako Google, međutim kod malih firmi on je odlučujući faktor. Informatički giganti imaju već ustanovljenu infrastrukturu i financijske resurse za velike timove označivača, dok male firme koje kreću na tržište to ne mogu priuštiti. Upravo u tome vidim drugu blagodat VAT-a. Kroz provedene eksperimente možemo vidjeti da broj neoznačenih primjera uvelike nadmašuje broj označenih primjera koje smo koristili u procesu učenja. Dakle, na označavanje podataka možemo potrošiti manje sredstava i vremena nego da provodimo nadzirano učenje. Ovisno o drugim faktorima, moguće je da za označavanje ne moramo osnovati poseban tim i zaposliti nove ljude nego da to se može odraditi s postojećim kadrom.

Uz smanjenje broja označenih primjera veoma je bitno da točnost predikcije modela kojeg smo učili sa polunadziranim principom ostane usporediv s rezultatima koje postižemo u nadziranim načinima učenja modela. Drugim riječima, da iskoristimo sve navedene blagodati iz prethodna dva paragrafa, pri čemu nećemo izgubiti na točnosti predikcije modela.

Često je kod mnogih modela i metoda izražena problematika njihove primjene zbog broja hiperparametara. Validacija hiperparametara se koristi kako bismo dobili vrijednosti hiperparametara za koje model postiže najveću moguću točnost. Najčešće je potrebna iscrpna pretraga hiperparametara, a nerijetko koristimo metode poput "grid search". Za ovakav pristup potrebna je velika računarska snaga kako bismo proveli iscrpnu pretragu velikog broja hiperparametara. VAT metodu regularizacije možemo svesti na pretragu samo jednog hiperparametra. Naime, pokazalo se da povećanjem broja uzastopnih iteracija kod izračuna virtualne neprijateljske perturbacije K ne pridonosi boljoj predikciji modela. Dakle za sve postupke učenja ga možemo fiksirati na vrijednost 1[11]. Iako se u teoretskoj razradi modela hiperparametri ϵ i ξ predstavljaju kao različiti, oni u implementaciji imaju istu ulogu. Stoga smatramo da se svode na jedan hiperparametar te jedan od njih fiksiramo na konstantnu vrijednost dok nad drugim provodimo validaciju modela. Treća prednost VAT-a je mali broj hiperparametara.

Prilikom validacije modela naišao sam na jednu manjkavost, a to je jako dugo vrijeme učenja svakog pojedinačnog modela, što prilikom validacije hiperparametara dodatno iscrpljuje resurse. Naime, proces učenja modela s VAT-om se produžuje za tri puta u odnosu na proces nadziranog učenja u kojem ne koristimo VAT. Moramo uzeti u obzir da je računarska snaga koja je meni bila na raspolaganju puno manja nego računarska snaga koju na raspolaganju ima firma koja se bavi ovim područjem. Stoga, kako bi utvrdili koliko je ovaj iskaz zaista manjkav moramo uzeti u obzir računarsku snagu koju koristimo za učenje kao i ostale prednosti ove metode.

Primjenom VAT-a na različite arhitekture modela kao i skupove podataka ustvrdili smo da VAT nije podložan arhitekturi ili skupu podataka nad kojim provodimo učenje. Koristili smo od najjednostavnijih potpuno vezanih mrežnih arhitektura pa sve do složenih konvolucijskih modela. Također skupovi podataka nad kojima smo provodili učenje su najjednostavnije točke u prostoru pa sve do slika iz CIFAR-10 skupa podataka.

Iz svega napisanog te iskustva u praksi, smatram da polunadzirane metode općenito, pa tako i VAT, imaju svoju primjenu u industriji. Blagodati koje VAT donosi su daleko veće nego manjkavosti, stoga smatram da može uvelike donijeti poboljšanja poslovnim i inženjerskim procesima unutar firme. VAT metoda se može kombinirati s različitim augmentcijama koje već same po sebi poboljšavaju generalizaciju modela, stoga u tome vidim mogućnost budućeg napretka. U praksi postoje problemi koji se rješavaju u domeni semantičke segmentacije i semantičkog instanciranja. Prema tome, primjenu i daljnji napredak VAT-a najviše vidim upravo u toj domeni. Primjeri semantičke segmentacije u komercijalnoj uporabi može biti automatska navigacija vozila.

LITERATURA

[1] URL https://www.mathworks.com/discovery/unsupervised-learning.html.

[2] About Kris Bolton. A quick introduction to artificial neural networks (part 2). 2018. URL http://krisbolton.com/a-quick-introduction-to-artificial-neural-networks-part-2.

[3] David Fumo. Types of machine learning algorithms you should know. 2017. URL https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[4] Chethan Kumar GN. Machine learning types and algorithms. 2018. URL https://towardsdatascience.com/machine-learning-types-and-algorithms-d8b79545a6ec.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair Aaron Courville, i Yoshua Bengio. Generative adversarial nets. 2014. URL https://arxiv.org/pdf/1406.2661.pdf.

[6] Ian J. Goodfellow, Jonathon Shlens, i Christian Szegedy. Explaining and harnessing adversarial examples. 2015. URL https://arxiv.org/pdf/1412.6572.pdf.

[7] Yves Grandvalet i Yoshua Bengio. Semi-supervised learning by entropy minimization.

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

[9] Sergey Zagoruyko Lee i Nikos Komodakis. Wide residual networks. 2017.

[10] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, i Shin Ishii. *Distributional smoothing with virtual adversarial training*. Graduate School of

Informatics - Kyoto University, 2016. URL https://arxiv.org/pdf/1507.00677.pdf.

[11] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, i Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. 2018. URL https://arxiv.org/pdf/1704.03976.pdf.

[12] Khush Patel. Overfitting vs underfitting in neural network and comparison of error rate with complexity graph. 2019. URL https://towardsdatascience.com/overfitting-vs-underfitting-ddc80c2fc00d.

[13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, i Rob Fergus. Intriguing properties of neural networks. 2014. URL https://arxiv.org/pdf/1312.6199.pdf.

[14] Xiaojin Zhu i Andrew B.Goldberg. Introductionto semi-supervisedlearning. 2009.

Sažetak

Diplomski rad obuhvaća kratku teoretsku razradu vrsta učenja modela u području dubokog učenja, te opsežnu teoretsku razradu metode polunadziranog učenja pod nazivom Polunadzirano učenje s virtualnim neprijateljskim primjerima. Predstavlja regularizacijsku tehniku učenja s virtualnim neprijateljskim primjerima koja potiče glatkoću decizijske granice zahtijevajući da neoznačeni primjeri imaju što sličniju predikciju kao i njihove perturbirane inačice. Predstavlja pojam virtualne neprijateljske perturbacije, te funkciju lokalne glatkoće razdiobe koje predstavljaju ključne pojmove u metodi. Provedeni su eksperimenti nad skupovima podataka MNIST, SVHN te CIFAR-10. Razvijen je postupak polunadziranog učenja klasifikacijskog modela zasnovan na virtualnim neprijateljskim primjerima.

Ključne riječi: virtualna, neprijateljska, perturbacija, VAT, regularizacija, lokalna glatkoća razdiobe, polunadzirana metoda.

Semi-supervised learning with virtual adversarial examples

Abstract

The masters thesis includes a short theoretical elaboration of the types of model training in the field of deep learning and an extensive theoretical elaboration of the method of semi-supervised learning called Semi-supervised learning with virtual adversarial examples. It represents a regularization technique with virtual adversarial examples that encourages the smoothness of the decision boundary by requiring unlabeled examples to have as similar prediction as possible to their perturbed versions. It represents the term of virtual adversarial perturbation and the local smoothness distribution function that represent key terms in the method. Experiments were performed on the data sets MNIST, SVHN and CIFAR-10. A process of semi-supervised training of the classification model based on virtual adversarial examples has been developed.

Keywords: virtual, adversarial, perturbation, VAT, regularization, local smoothness of distribution, semi-supervised method.