SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 872

VARIJACIJSKO UČENJE NA ZAŠUMLJENIM OZNAKAMA

Dominik Jambrović

Zagreb, lipanj, 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

DIPLOMSKI ZADATAK br. 872

Pristupnik:	**Dominik Jambrović (0036534818)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Varijacijsko učenje na zašumljenim oznakama**

Opis zadatka:

Raspoznavanje slika važan je problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme stanje tehnike postižu duboki modeli zasnovani na konvolucijama i slojevima pažnje. Međutim, standardni postupci teško se nose sa zašumljenim oznakama.. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje tenzorima i slikama. Proučiti i ukratko opisati postojeće duboke arhitekture za raspoznavanje slika s posebnim naglaskom na prednaučene samonadzirane modele. Odabrati slobodno dostupne skupove slika te oblikovati podskupove za učenje, validaciju i testiranje. Formulirati optimizacijski cilj s latentnim predikcijama čistih razreda te predložiti rješenje utemeljeno na varijacijskoj aproksimaciji te maksimiziranju očekivanja. Komentirati učinkovitost učenja i zaključivanja. Predložiti pravce za budući rad. 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: 4. srpnja 2025.

Zahvale!

Sadržaj

1. Uvod

Duboki modeli koriste se u brojnim aspektima naše svakodnevice. Pri razvoju i učenju modela, pažnju prije svega posvećujemo performansama na neviđenim podatcima - želimo naučiti modele koji dobro generaliziraju. Drugim riječima, želimo da modeli daju ispravna predviđanja za viđene, ali i za neviđene podatke. Ovime osiguravamo da naša rješenja imaju primjenu i van laboratorijskih uvjeta u kojima se uče.

U procesu razvoja modela za određeni zadatak strojnog učenja, osim odabira arhitekture, algoritma učenja i hiperparametara, veliku ulogu igraju podatci na kojima učimo. Općenito govoreći, prikupljanje i označavanje podataka jedan je od najskupljih dijelova procesa razvoja rješenja za nekih problem. Važno je da prikupljeni podatci što realističnije predstavljaju stvarne situacije s kojima će se naš model susretati tj. da distribucija podataka odgovara stvarnoj distribuciji situacija koje prikazuju. Dodatno, pokazuje se da duboki modeli uz dovoljan kapacitet mogu naučiti ispravno predviđati oznake čak i za nasumično označene podatke [1], tako da je veoma važno da su prikupljeni podatci što točnije označeni.

Područje računalnog vida [2] bavi se razvojem algoritama i modela za brojne zadatke raspoznavanja i razumijevanja slika. Najčešći zadatak je klasifikacija slika - model na ulazu dobiva sliku, a na izlazu treba predvidjeti razred koji odgovara ulaznom primjeru. Iako postoje brojni skupovi slikovnih podataka koji se mogu koristiti za učenje i evaluaciju modela, za konkretne zadatke u većini slučajeva trebamo prikupiti i označiti vlastite slike. Pritom postoji nekoliko čestih opasnosti: napadi umetanjem stražnjih vrata [3] ili prisutnost zašumljenih oznaka [4].

Kada govorimo o napadima umetanjem stražnjih vrata (engl. *backdoor attack*), maliciozni agent u skup podataka dodaje zatrovane podatke s ciljem manipulacije izlaza naučenog modela za određene ulaze. S druge strane, anotator podataka bez zlih namjera određenim podatcima može pridijeliti netočne oznake, time dodajući podatke sa zašumljenim oznakama u skup. Kroz vrijeme, razvili su se brojni algoritmi za obranu modela od napada umetanjem stražnjih vrata [5, 6, 7], kao i za učenje na zašumljenim oznakama [8, 9]. Ipak, većina radova se fokusira na samo jedan od ovih problema, a ne na razvoj algoritma koji se može nositi s oba problema.

Cilj ovog rada je reproducirati i poboljšati rezultate okvira za obranu od napada umetanjem stražnjih vrata imena VIBE (engl. *Variational inference for backdoor elimination*) [10]. Osim ovoga, cilj je i primijeniti VIBE na problem zašumljenih oznaka. Pritom VIBE evaluiramo na nekoliko čestih vrsta napada odnosno metoda zašumljivanja oznaka kako bi se osigurala robusnost okvira. Dodatno, cilj je usporediti VIBE sa stanjem tehnike (engl. *state of the art - SotA*) za problem zašumljenih oznaka.

2. Napadi umetanjem stražnjih vrata

Cilj napada umetanjem stražnjih vrata [3] je dodavanjem zatrovanih podataka ugraditi stražnja vrata u naučeni model. Ako napad uspije, napadač može kontrolirati izlaz modela koristeći suptilne izmjene ulaznog primjera. Općenito govoreći, napad umetanjem stražnjih vrata podrazumijeva dodavanje vizualnog okidača na ulazni primjer, kao i prikladnu izmjenu oznaka. Pritom napadač radi izmjenu određenog udjela podataka, dok preostali podatci ostaju neizmjenjeni. Hiperparametar koji opisuje udio zatrovanih podataka zvat ćemo stopom trovanja (engl. *poisoning rate*). Pojedini napadi razlikuju se po načinu dodavanja okidača tj. načinu izmjene ulaznih primjera, kao i po načinu izmjene oznaka.

Kada govorimo o načinu izmjene ulaznih primjera, možemo napraviti podjelu na lokalne i globalne izmjene primjera. Kod lokalnih izmjena, mijenja se samo određeno područje slike, najčešće dodavanjem zadanog okidača na to područje [11]. S druge strane, kod globalnih izmjena se mijenja cijela slika koristeći različite tehnike poput miješanja slike s okidačem [12] ili transformiranja slike na temelju zadanog deformacijskog polja [13]. Osim korištenja jednog okidača za sve zatrovane podatke, određeni napadi koriste okidače specifične za pojedini uzorak [14].

Većinu napada možemo svrstati u jedan od dva načina izmjena oznaka: *all-to-one* i *all-to-all* izmjena oznaka [15]. Kod *all-to-one* metode, primjeri dobivaju zatrovanu oznaku jednog proizvoljno odabranog razreda neovisno o originalnim oznakama pojedinih primjera. S druge strane, kod *all-to-all* metode, primjeri dobivaju zasebne zatrovane oznake ovisno o originalnim oznakama. Određeni napadi uopće ne mijenjaju oznake zatrovanih primjera, već se oslanjaju isključivo na jače izmjene ulaznih primjera. Ovakve napade zovemo napadi s čistim oznakama (engl. *clean-label attacks*) [16].

Uspješnost pojedinog napada mjerimo metrikom imena stopa uspješnosti napada (engl. *attack success rate* - ASR). Ovu mjeru definiramo kao točnost modela mjerenu isključivo na zatrovanim primjerima. Cilj algoritama za obranu od napada umetanjem stražnjih vrata je naučiti čisti model na zatrovanom skupu. Drugim riječima, glavni cilj obrane je naučiti model koji ima izvrsne performanse na čistim podatcima, ali i što nižu stopu uspješnosti napada.

2.1. Implementacije napada

U ovome radu, fokusiramo se na tri napada umetanjem stražnjih vrata: napade BadNets [11], Blend [12] te WaNet [13].

2.1.1. BadNets

Napad BadNets uobičajeno dodaje jedan zadani okidač na svaki odabrani ulazni primjer. Okidač možemo shvatiti kao uzorak piksela koji se dodaje na specifično mjesto na slici. Na primjer, okidač može biti bijeli pravokutnik pozicioniran u donjem lijevom kutu slike. Naravno, korišteni uzorak može biti proizvoljne kompleksnosti i veličine. Kod napada BadNets, izmjene oznaka su najčešće tipa *all-to-one*, ali česte su i izmjene tipa *all-to-all*.

Slika 2.1. Primjer primjene napada BadNets. Izvornoj slici (lijevo) dodaje se okidač kako bi nastala zatrovana slika (desno).

2.1.2. Blend

Napad Blend provodi miješanje zadanog okidača sa svakim odabranim ulaznim primjerom. Pritom je jačina napada određena hiperparametrom α koji nazivamo jačina miješanja (engl. *blending strength*). Primjenu napada Blend možemo prikazati jednadžbom:

$$\tilde{\boldsymbol{x}} = (1 - \alpha) \cdot \boldsymbol{x} + \alpha \cdot \boldsymbol{t} \tag{2.1}$$

Pri čemu $\boldsymbol{x}$ označava ulazni primjer, $\boldsymbol{t}$ okidač, a $\tilde{\boldsymbol{x}}$ zatrovani primjer. Kod napada Blend, izmjene oznaka su uobičajeno tipa *all-to-one*.

Slika 2.2. Primjer primjene napada Blend. Izvorna slika (lijevo) miješa se s okidačem uz $\alpha = 0.2$ kako bi nastala zatrovana slika (desno).

2.1.3. WaNet

Napad WaNet provodi geometrijsku transformaciju svakog odabranog ulaznog primjera koristeći nasumično generirano deformacijsko polje. Deformacijsko polje svakom pikselu odredišne slike dodjeljuje vektor pomaka prema pikselu izvorne slike. Pritom hiperparametar k određuje veličinu nasumično generiranog polja šuma na temelju kojeg se skaliranjem i interpolacijom dobiva konačno deformacijsko polje, a hiperparametar s određuje jačinu deformacije. Primjenu napada WaNet možemo definirati jednadžbom:

$$\tilde{\boldsymbol{x}} = \mathcal{W}(\boldsymbol{x}, \boldsymbol{M}(k, s)) \tag{2.2}$$

Pri čemu $\boldsymbol{x}$ označava ulazni primjer, $\boldsymbol{M}$ deformacijsko polje generirano uz hiperparametre k i s, $\mathcal{W}$ primjenu deformacijskog polja na ulazni primjer, a $\tilde{\boldsymbol{x}}$ zatrovani primjer. Kao i kod napada Blend, kod napada WaNet su izmjene oznaka uobičajeno tipa *all-to-one*.

Slika 2.3. Primjer primjene napada WaNet. Izvorna slika (lijevo) transformira se koristeći deformacijsko polje uz $k = 8$ i $s = 4$ kako bi nastala zatrovana slika (desno). Hiperparametri k i s su uvećani kako bi učinak napada bio uočljiviji.

2.2. Obrane od napada

U ovome radu, rezultate okvira VIBE uspoređujemo s rezultatima triju obrana od napada umetanjem stražnjih vrata: *Anti-backdoor learning* (ABL) [5], *Decoupling based defense* (DBD) [6] te *Adaptively splitting dataset-based defense* (ASD) [7].

2.2.1. ABL

Algoritam *Anti-backdoor learning* (ABL) sastoji se od dva glavna koraka: izoliranje stražnjih vrata (engl. *backdoor isolation*) i odučavanje stražnjih vrata (engl. *backdoor unlearning*). Osnovna ideja ove obrane je da se nakon određenog broja epoha učenja uz posebno definiran gubitak izolira određeni broj primjera za koje se smatra da su zatrovani. Nakon prvog koraka, ti se primjeri koriste za odučavanje stražnjih vrata, dok se preostali primjeri koriste za standardno učenje.

Konkretno, cilj prvog koraka je zadržati vrijednost gubitka svakog pojedinog primjera oko praga γ. Kako bi ovo postigli, autori predlažu korištenje gradijentnog uspona u slučaju da gubitak primjera padne ispod praga, dok se inače koristi gradijentni spust. Gubitak u prvom koraku možemo prikazati jednadžbom:

$$\mathcal{L}_1 = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\operatorname{sign}\left(\ell(f_\theta(\boldsymbol{x}),y)-\gamma\right)\cdot\ell(f_\theta(\boldsymbol{x}),y)\right] \tag{2.3}$$

Pritom $(\boldsymbol{x},y)\sim\mathcal{D}$ označava primjer $\boldsymbol{x}$ s pripadnom oznakom y iz skupa podataka $\mathcal{D}$, $f_\theta(\boldsymbol{x})$ izlaz modela s parametrima θ, $\ell(f_\theta(\boldsymbol{x}),y)$ gubitak za izlaz modela i stvarnu oznaku y, a sign operaciju signum.

Ideja je da će gubitak za zatrovane primjere veoma brzo pasti ispod praga te će se za njih često aktivirati gradijentni uspon, dok će gubitak čistih primjera sporije padati i stabilizirati se oko praga. Nakon zadanog broja epoha, izolira se udio p primjera s najnižim gubitkom i proglašava potencijalnim zatrovanim skupom. Važno je napomenuti da bismo prvi korak ABL-a mogli zamijeniti proizvoljnim algoritmom detekcije zatrovanih primjera.

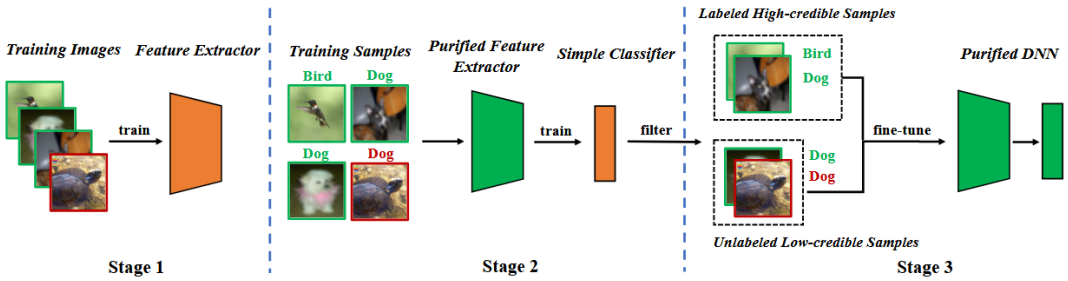
U drugom koraku, učenje se u svakoj epohi provodi zasebno za procijenjeni čisti odnosno zatrovani skup. Dok se učenje na čistom skupu provodi uz standardni gradijentni spust, učenje na zatrovanom skupu provodi se uz gradijentni uspon kako bi model odučili od stražnjih vrata. Ovo je moguće zato što je napad najčešće realiziran uz samo jedan ciljni razred tj. uz *all-to-one* način izmjene oznaka. Gubitak u drugom koraku možemo prikazati jednadžbom:

$$\mathcal{L}_2 = \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{\mathcal{D}}_c}\left[\ell(f_\theta(\boldsymbol{x}),y)\right] - \mathbb{E}_{(\boldsymbol{x},y)\sim\hat{\mathcal{D}}_b}\left[\ell(f_\theta(\boldsymbol{x}),y)\right] \tag{2.4}$$

Pritom $\hat{\mathcal{D}}_c$ označava procijenjeni čisti skup, a $\hat{\mathcal{D}}_b$ procijenjeni zatrovani skup.

2.2.2. DBD

Algoritam *Decoupling based defense* (DBD) obranu od napada umetanjem stražnjih vrata razdvaja na tri koraka. Pritom DBD model tretira kao dvije povezane cjeline: ekstraktor značajki (engl. *feature extractor*) te klasifikator (najčešće nekoliko potpuno-povezanih slojeva). Ekstraktor značajki za ulazni primjer na izlazu daje vektor značajki tj. ugrađivanje (engl. *embedding*) u latentnom metričkom prostoru. S druge strane, klasifikator za ugrađivanje na ulazu predviđa jednu od mogućih oznaka.



Slika 2.4. Prikaz glavnih koraka obrane DBD. Preuzeto iz [6].

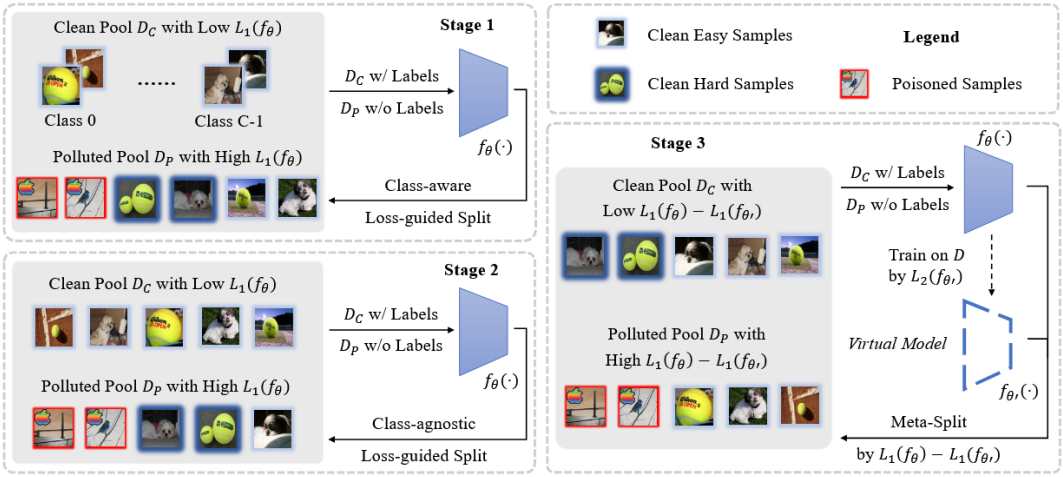
U prvom koraku, DBD uči ekstraktor značajki koristeći proizvoljan algoritam samonadziranog učenja [17] na slikama bez oznaka. Budući da algoritmi samonadziranog učenja uobičajeno uključuju korištenje jakih augmentacija ulaznih primjera, kvaliteta okidača kod zatrovanih primjera bit će narušena. Dodatno, jer se model u ovoj fazi uči bez oznaka, u prvom koraku nije moguće ugraditi stražnja vrata u ekstraktor značajki. Drugim riječima, na kraju prvog koraka imat ćemo naučen čisti ekstraktor značajki.

U drugom koraku, parametri ekstraktora značajki su zamrznuti, a klasifikator učimo koristeći klasično nadzirano učenje na podatcima s oznakama. Pritom kao funkciju gubitka koristimo simetričnu unakrsnu entropiju - pokazano je da korištenje iste rezultira višim gubitkom kod zatrovanih primjera [18]. Ipak, ako bismo koristili samo ova dva koraka, dobiveni model ne bi imao performanse jednake stanju tehnike jer smo ekstraktor značajki učili bez oznaka. Zbog toga, važno je napraviti podjelu skupa podataka na čisti i zatrovani skup te nakon toga ugoditi (engl. *fine-tune*) cijeli model. Kod algoritma DBD, ova podjela se radi na temelju iznosa gubitka: udio α primjera s najnižim iznosom gubitka smatrat ćemo vjerodostojnim tj. čistim skupom, dok ćemo preostale primjere smatrati zatrovanima.

Treći korak koristi podjelu na čisti i zatrovani skup kako bi dodatno ugodio parametre cijelog modela. Konkretno, zatrovanom skupu uklanjamo oznake te potom model učimo koristeći proizvoljni algoritam polunadziranog učenja [19]. Na kraju ovog koraka, imat ćemo naučen cjelokupni model otporan na napade umetanjem stražnjih vrata.

2.2.3. ASD

Algoritam *Adaptively splitting dataset-based defense* (ASD) obranu od napada umetanjem stražnjih vrata razdvaja na tri koraka. Tijekom sva tri koraka, održavaju se dva skupa podataka: čisti i zagađeni skup. Pritom se u čistom skupu nalaze podatci za koje je velika vjerojatnost da su čisti, dok se u zagađenom skupu nalaze zatrovani podatci, kao i preostali čisti podatci. Kroz učenje, čisti skup se povećava dodavanjem primjera iz zagađenog skupa. Konačno, na kraju učenja bi zagađeni skup trebao sadržavati isključivo zatrovane podatke. Parametri modela uvijek se ažuriraju na temelju proizvoljnog polunadziranog gubitka, pri čemu se zagađeni skup koristi za učenje bez oznaka.



Slika 2.5. Prikaz glavnih koraka obrane ASD. D_C predstavlja čisti skup, a D_P zagađeni skup. $\mathcal{L}_1$ odgovara gubitku $\mathcal{L}_{SCE}$, a $\mathcal{L}_2$ gubitku $\mathcal{L}_{CE}$. Preuzeto iz [7].

U prvom koraku, čisti skup se inicijalizira na mali broj provjereno čistih primjera (na primjer, po 10 primjera za svaki razred), dok se zagađeni skup inicijalizira na cijeli skup podataka. Svakih t epoha učenja, u čisti skup se iz zagađenog skupa dodaje n primjera iz svakog pojedinog razreda koji imaju najniži gubitak $\mathcal{L}_{SCE}$. Pritom kao funkciju gubitka $\mathcal{L}_{SCE}$ koristimo simetričnu unakrsnu entropiju kako bi zatrovani primjeri imali što viši gubitak. Ovu podjelu zovemo razredno-svjesna podjela vođena gubitkom (engl. *class-aware loss-guided split*).

Tijekom drugog koraka, čisti skup značajno proširujemo dodavanjem udjela α zagađenog skupa. Pritom se dodaju primjeri iz cijelog skupa (neovisno o razredu) koji imaju najniži gubitak $\mathcal{L}_{SCE}$. Ovu podjelu zovemo razredno-nesvjesna podjela vođena gubitkom (engl. *class-agnostic loss-guided split*).

Nakon prva dva koraka ASD-a, u zagađenom skupu preostaju zatrovani primjeri, ali i neki teški primjeri specifični za model. Zbog toga što model nismo učili na teškim primjerima s oznakama, performanse modela trenutno su niže od stanja tehnike. Kako bismo dodali i teške primjere u čisti skup, svaku epohu trećeg koraka konstruiramo virtualni model. Virtualni model inicijaliziramo parametrima glavnog modela te potom provodimo jednu epohu nadziranog učenja na zagađenom skupu uz korištenje gubitka $\mathcal{L}_{CE}$. Kao funkciju gubitka $\mathcal{L}_{CE}$ koristimo standardnu unakrsnu entropiju. Nakon učenja virtualnog modela, mjerimo smanjenje gubitka definirano jednadžbom:

$$\Delta\mathcal{L}_{SCE} = \mathcal{L}_{SCE}(f_{\theta}) - \mathcal{L}_{SCE}(f_{\theta'}) \tag{2.5}$$

Pritom f_{θ} označava glavni model s parametrima θ, a $f_{\theta'}$ predstavlja virtualni model s parametrima θ' dobivenim nakon jedne epohe nadziranog učenja na zagađenom skupu. Konačno, udio γ primjera s najmanjim smanjenjem gubitka dodajemo u čisti skup. Intuicija iza ove podjele je da su zatrovani podatci lagani za naučiti, tako da već nakon jedne epohe nadziranog učenja isti imaju veoma nizak gubitak, dok teški čisti primjeri i dalje imaju visok gubitak. Važno je napomenuti da se virtualni model koristi isključivo za provođenje podjele na temelju smanjenja gubitka. Ovu podjelu zovemo meta-podjela (engl. *meta-split*) jer je pristup s učenjem virtualnog modela inspiriran područjem meta-učenja (engl. *meta-learning*) [20]. Na kraju trećeg koraka, u čistom skupu će se nalaziti gotovo svi čisti primjeri, a dobiveni model će imati izvrsne performanse uz otpornost na napade umetanjem stražnjih vrata.

3. Zašumljene oznake

Proces stvaranja skupa vizualnih podataka otprilike možemo podijeliti u tri koraka. U prvom koraku, potrebno je definirati skup razreda tj. oznaka koje mogu biti dodijeljene svakom primjeru. Tijekom drugog koraka, cilj nam je prikupiti što veći skup slika, pritom pazeći na to da slike precizno prikazuju situacije s kojima će se naš model susretati. Dodatno, veoma je važno da je prikupljeni skup što raznovrsniji kako bi model mogao naučiti dobro generalizirati. Konačno, u trećem koraku anotatori označavaju slike na temelju definiranog skupa oznaka. Drugi i treći korak ovog procesa mogu se provoditi jedan za drugim, ali i paralelno - nakon što se prikupi određen broj slika, taj skup se šalje na označavanje, a istovremeno je moguće prikupiti još slika.

Iako anotatori podataka prolaze kroz brojne edukacije kako bi se što bolje upoznali s pravilima na temelju kojih će označavati podatke, određeni podatci su prirodom teži od drugih pa može doći do pogrešnog označavanja. Prisutnost zašumljenih oznaka slična je napadima umetanjem stražnjih vrata po tome što se kod oba problema model mora nositi s pogrešnim oznakama. Ipak, kod problema zašumljenih oznaka podrazumijevamo da su ulazne slike netaknute tj. da ne postoji nikakva izmjena ulaza. Dodatno, kod problema zašumljenih oznaka ne postoji konkretan cilj - podatci sa zašumljenim oznakama nastaju slučajno, bez ikakvih loših namjera.

Dok je kod napada umetanjem stražnjih vrata problem prelako učenje poveznice između prisutnosti okidača i određenog ciljnog razreda, kod podataka sa zašumljenim oznakama je problem činjenica da model uz dovoljan kapacitet može naučiti oznake čak i ako su one nasumično generirane [1]. Drugim riječima, ako nismo oprezni, lako je doći do prenaučenosti (engl. *overfitting*) modela. Naravno, ovakav model će loše generalizirati na ispravno označenim podatcima.

Kako bismo mogli usporediti performanse različitih algoritama za učenje na zašumljenim oznakama, najčešće određenom udjelu čistog skupa podataka dodajemo sintetičke zašumljene oznake. Hiperparametar koji opisuje udio podataka sa zašumljenim oznakama zvat ćemo stopa šuma (engl. *noise rate*). Nakon učenja na zašumljenim podatcima, performanse modela evaluiramo na čistom, ali i na u potpunosti zašumljenom skupu. Cilj algoritama za učenje na zašumljenim oznakama tada je zadržati performanse modela učenog na zašumljenim podatcima što bližima performansama modela učenog na čistim podatcima.

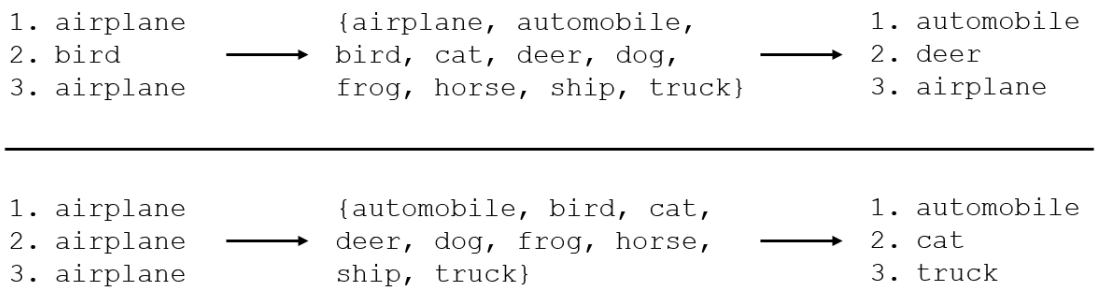
3.1. Metode zašumljivanja

U ovome radu, fokusiramo se na dvije metode zašumljivanja oznaka: simetrično (engl. *symmetric*) i asimetrično (engl. *asymmetric*) zašumljivanje [21].

3.1.1. Simetrično zašumljivanje

Kod simetričnog zašumljivanja, nasumično biramo zadani udio primjera iz čistog skupa podataka te istima dodjeljujemo nasumično generirane nove oznake. Ovu metodu zašumljivanja također zovemo i uniformno zašumljivanje jer svaki razred ima jednaku vjerojatnost postati nova oznaka za neki zadani primjer.

Pritom razlikujemo dvije vrste simetričnog zašumljivanja: inkluzivnu (engl. *symm-inc*) i ekskluzivnu (engl. *symm-exc*) vrstu [21]. Ako govorimo o inkluzivnom simetričnom zašumljivanju, stvarna oznaka za neki zadani primjer može biti odabrana i kao zašumljena oznaka. S druge strane, kod ekskluzivnog simetričnog zašumljivanja ignoriramo stvarnu oznaku tj. zašumljena oznaka se uvijek razlikuje od stvarne oznake primjera. U našem radu, fokusiramo se na inkluzivno simetrično zašumljivanje.

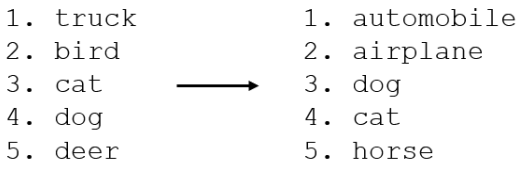


Slika 3.1. Prikaz inkluzivnog (gore) odnosno ekskluzivnog (dolje) simetričnog zašumljivanja za tri odabrana primjera iz skupa CIFAR-10 [22]. Čiste oznake (lijevo) preslikavaju se na jednu od mogućih oznaka (sredina) i nastaju zašumljene oznake (desno). Vidimo da primjeri s istim čistim oznakama nemaju nužno i iste zašumljene oznake.

3.1.2. Asimetrično zašumljivanje

Kod asimetričnog zašumljivanja, prvo nasumično biramo zadani udio primjera zasebno za svaki razred. Drugim riječima, ako je u pitanju skup CIFAR-10 [22], za svaki od 10 razreda nasumično ćemo odabrati zadani udio primjera kojima ćemo potencijalno dodijeliti zašumljene oznake.

Nakon početnog odabira, svakom primjeru dodjeljujemo novu oznaku na temelju predodređenog preslikavanja. Pritom je moguće da određeni razredi nemaju definirano preslikavanje u neki drugi razred, tako da odabranim primjerima iz tih razreda neće biti dodijeljene zašumljene oznake. Preslikavanje na temelju kojeg se pojedinim primjerima dodjeljuje nova oznaka unaprijed je definirano za pojedine skupove podataka poput skupova MNIST [23], CIFAR-10 i CIFAR-100 [22]. Kod skupova s hijerarhijskim odnosom razreda poput skupa CIFAR-100, preslikavanja su definirana nasumično unutar pojedinih grupa razreda.



Slika 3.2. Prikaz preslikavanja razreda za skup CIFAR-10. Odabranim primjerima se na temelju čistih oznaka (lijevo) dodjeljuju zašumljene oznake (desno). Pritom odabrani primjeri s istim čistim oznakama uvijek imaju i iste zašumljene oznake.

3.2. Učenje na zašumljenim oznakama

U ovome radu, rezultate prilagođenog okvira VIBE uspoređujemo s rezultatima dvaju algoritama za učenje na podatcima sa zašumljenim oznakama: *Sparse over-parameterization* (SOP) [8] te *Imprecise label learning* (ILL) [9].

3.2.1. SOP

Osnovna ideja algoritma *Sparse over-parameterization* (SOP) bazira se na činjenici da su primjeri sa zašumljenim oznakama u većini slučajeva rijetki. Jedan od mogućih načina za učenje modela na zašumljenim oznakama tada je modeliranje šuma za pojedine primjere koristeći rijetke vektore.

Konkretno, za svaki primjer $(\boldsymbol{x}_i, y_i)$ iz skupa podataka $\mathcal{D}$ definiramo rijetki vektor $\boldsymbol{s}_i$ koji služi kao odstupanje između uočene (potencijalno zašumljene) oznake $\boldsymbol{y}_i^{OH}$ i čiste (skrivene) oznake $\boldsymbol{l}_i^{OH}$. Pritom $\boldsymbol{y}_i^{OH}$ označava jednojedinično kodiranu (engl. *one-hot encoding*) oznaku y. Uobičajeno, učenje modela se svodi na potragu za parametrima θ koji minimiziraju gubitak definiran jednadžbom:

$$\mathcal{L} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\ell(f_\theta(\boldsymbol{x}), y)\right] \tag{3.1}$$

Pritom f_θ predstavlja model s parametrima θ, a $\ell(f_\theta(\boldsymbol{x}), y)$ funkciju gubitka između izlaza modela $f_\theta(\boldsymbol{x})$ i dane oznake y. Uz uvođenje rijetkog vektora $\boldsymbol{s}_i$ za svaki primjer, učenje se svodi na potragu za parametrima $(\theta, \{\boldsymbol{s}_i\}_{i=1}^N)$ koji minimiziraju gubitak:

$$\mathcal{L}_{SOP} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\ell(f_\theta(\boldsymbol{x}) + \boldsymbol{s}, y)\right] \tag{3.2}$$

Dodatno, kako bi se osigurala rijetkost vektora $\boldsymbol{s}_i$, isti definiramo pomoću vektora $\boldsymbol{u}_i$ i vektora $\boldsymbol{v}_i$ kao:

$$\boldsymbol{s}_i = \boldsymbol{u}_i \odot \boldsymbol{u}_i - \boldsymbol{v}_i \odot \boldsymbol{v}_i \tag{3.3}$$

Drugim riječima, za svaki podatak $(\boldsymbol{x}_i, y_i)$ definiramo dva vektora parametara: $\boldsymbol{u}_i$ i $\boldsymbol{v}_i$.

Ove vektore učimo zajedno s parametrima modela θ koristeći gradijentni spust. Pritom je stopa učenja za parametre $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}_{i=1}^N$ skalirana hiperparametrom α kako bi se izbjeglo trivijalno rješenje optimizacijskog problema kod kojeg vrijedi $\boldsymbol{u}_i \equiv \boldsymbol{v}_i \equiv \mathbf{0}$, $\forall i \in N$.

Uočimo da vektor $\boldsymbol{s}_i$ za primjer $(\boldsymbol{x}_i, y_i)$ dodatno mora poštovati nekoliko uvjeta: u vektoru $\boldsymbol{s}_i$ se pozitivna vrijednost može nalaziti isključivo na mjestu koje odgovara nenegativnoj vrijednosti u $\boldsymbol{y}_i^{OH}$, dok se negativne vrijednosti mogu nalaziti isključivo na mjestima koje odgovaraju vrijednosti 0 u $\boldsymbol{y}_i^{OH}$. Dodatno, svaki element vektora $\boldsymbol{s}_i$ mora biti u rasponu $[-1, 1]$. Kako bismo ovo osigurali, uvodimo konačnu definiciju vektora $\boldsymbol{s}_i$:

$$\boldsymbol{s}_i = \boldsymbol{u}_i \odot \boldsymbol{u}_i \odot \boldsymbol{y}_i^{OH} - \boldsymbol{v}_i \odot \boldsymbol{v}_i \odot (\mathbf{1} - \boldsymbol{y}_i^{OH}) \tag{3.4}$$

Pritom su vektori parametara $\boldsymbol{u}_i$ i $\boldsymbol{v}_i$ definirani kao:

$$\boldsymbol{u}_i \in [-1, 1]^K \ , \ \boldsymbol{v}_i \in [-1, 1]^K \tag{3.5}$$

U algoritmu SOP, za učenje parametara $(\theta, \{\boldsymbol{u}_i\}_{i=1}^N)$ se kao funkcija gubitka koristi standardna unakrsna entropija. Nažalost, unakrsna entropija se ne može koristiti za ispravno učenje parametara $\{\boldsymbol{v}_i\}_{i=1}^N$. Umjesto toga, za učenje tih parametara kao funkciju gubitka koristimo srednju kvadratnu pogrešku.

Kako bi poboljšali performanse osnovnog algoritma SOP, autori predlažu dodavanje dvije nove komponente gubitka: konzistencijski gubitak (engl. *consistency loss*) [24] te gubitak ravnoteže razreda (engl. *class-balance loss*) [25]. Inačicu algoritma koja dodatno koristi ove dvije komponente gubitka zovemo SOP+, a ista općenito postiže bolje performanse od osnovne inačice algoritma.

3.2.2. ILL

Algoritam *Imprecise label learning* (ILL) obuhvaća i pokušava riješiti nekoliko problema vezanih uz nesavršenosti oznaka: učenje na djelomičnim oznakama (engl. *partial label learning*) [26], polunadzirano učenje te učenje na zašumljenim oznakama. Konkretno, ILL sve ove probleme smatra vrstama problema nepreciznih oznaka (engl. *imprecise labels*). Skup podataka tada sadrži ulazne slike X i neprecizne oznake I, dok su čiste oznake Y skrivene (latentne). Pritom su neprecizne oznake apstraktne - u stvarnosti to može biti skup oznaka (u problemu učenja na djelomičnim oznakama), ali i zašumljena oznaka (u problemu učenja na zašumljenim oznakama). Cilj učenja modela tada je pronaći parametre θ koji maksimiziraju zajedničku vjerojatnost definiranu kao:

$$p(X, I|\theta) = \sum_{Y} p(X, I, Y|\theta) \tag{3.6}$$

Kako bismo maksimizirali logaritam očekivanja uz prisutnost latentne varijable, koristimo algoritam maksimizacije očekivanja (algoritam EM) [27]. Pritom u E koraku algoritma maksimizacije očekivanja računamo očekivanje zajedničke vjerojatnosti $p(X, I, Y|\theta)$ uz zadanu uvjetnu vjerojatnost $p(Y|X, I, \theta^t)$ u vremenskom koraku t. S druge strane, u M koraku tražimo parametre θ koji maksimiziraju donju varijacijsku granicu (engl. *evidence lower bound* - ELBO) vjerojatnosti $p(X, I|\theta)$. Budući da se različite vrste problema razlikuju primarno po prirodi nepreciznosti oznaka, algoritmi za pojedine probleme najviše će se razlikovati u načinu izračuna vjerojatnosti $p(Y|X, I, \theta^t)$. Ovime algoritam ILL pruža unificirani okvir koji podržava različite vrste nepreciznosti oznaka, kao i kombinacije istih.

Dakle, glavni cilj algoritma ILL je pronaći parametre θ koji maksimiziraju zajedničku vjerojatnost $p(X, I|\theta)$ odnosno logaritam iste. Prema algoritmu EM, ovaj problem možemo riješiti maksimizacijom očekivanja:

$$\mathbb{E}_{Y|X,I,\theta^t}\left[\log p(X, Y, I|\theta)\right] = \mathbb{E}_{Y|X,I,\theta^t}\left[\log p(Y|X, \theta) + \log p(I|X, Y, \theta)\right] \tag{3.7}$$

Pritom ignoriramo član $p(X)$ jer isti ne ovisi o parametrima θ. Iz dobivenog izraza vidimo da izračunom očekivanja razmatramo sve moguće oznake Y na temelju danih nepreciznih oznaka I, umjesto da se fokusiramo na samo jednu prepravljenu oznaku. Na temelju ove formulacije dalje možemo izvesti cilj za pojedini problem. U našem radu, fokusirat ćemo se na izvod za problem učenja na zašumljenim oznakama.

Kod problema učenja na zašumljenim oznakama, neprecizne oznake I se manifestiraju kao zašumljene oznake $\hat{Y}$. Vjerojatnost $p(\hat{Y}|Y, X, \theta)$ tada predstavlja model šuma ovisan o pojedinom uzorku X. Ovaj problem pojednostavit ćemo razmatranjem modela šuma neovisnog o uzorku $\mathcal{T}(\hat{Y}|Y, \omega)$ s parametrima ω. Cilj koji želimo maksimizirati tada dobivamo kao:

$$\begin{aligned}
\mathbb{E}_{Y|X,I,\theta^t}\left[\log p(X, Y, I|\theta)\right] &= \mathbb{E}_{Y|X,I,\theta^t}\left[\log p(Y, I|X, \theta)\right] \\
&= \mathbb{E}_{Y|X,\hat{Y},\theta^t}\left[\log p(Y, \hat{Y}|X, \theta)\right] \\
&= \mathbb{E}_{Y|X,\hat{Y},\theta^t}\left[\log p(Y|\hat{Y}, X, \theta) + \log p(\hat{Y}|X, \theta)\right] \\
&= \sum_{Y} p(Y|\hat{Y}, X, \theta^t) \log p(Y|\hat{Y}, X, \theta) + \log p(\hat{Y}|X, \theta)
\end{aligned} \tag{3.8}$$

Funkcija gubitka koju želimo minimizirati tada postaje:

$$\mathcal{L}_{ILL} = \mathcal{L}_{CE}(p(y|\boldsymbol{x}, \hat{y}, \theta, \omega^t), p(y|\boldsymbol{x}, \hat{y}, \theta^t, \omega^t)) + \mathcal{L}_{CE}(p(\hat{y}|\boldsymbol{x}, \theta, \omega), \hat{y}) \tag{3.9}$$

Pritom $\mathcal{L}_{CE}$ označava unakrsnu entropiju. Prva komponenta dobivenog gubitka odgovara konzistencijskom gubitku čistih oznaka uvjetovanih zašumljenim oznakama, dok druga komponenta odgovara nadziranom gubitku na zašumljenim oznakama. Pritom se za izračun obje komponente koristi model šuma uz zadanu zašumljenu oznaku $\hat{y}$:

$$\begin{aligned}
p(y|\boldsymbol{x}, \hat{y}, \theta, \omega^t) &\propto p(y|\boldsymbol{x}, \theta)\, \mathcal{T}(\boldsymbol{y}|y, \omega^t) \\
p(\hat{y}|\boldsymbol{x}, \theta, \omega) &= \sum_{y} p(y|\boldsymbol{x}, \theta)\, \mathcal{T}(\boldsymbol{y}|y, \omega)
\end{aligned} \tag{3.10}$$

4. Samonadzirano učenje

Samonadzirano učenje [17] paradigma je strojnog učenja kod koje model uči izlučivati korisne reprezentacije tj. značajke ulaznih podataka na temelju posebno osmišljenih zadataka bez oznaka (engl. *pretext tasks*). Ovim pristupom pokušava se adresirati problem cijene i vremena potrebnog za označavanje velikih skupova podataka. Model učen proizvoljnim algoritmom samonadziranog učenja dalje se može koristiti kao okosnica (engl. *backbone*) za model koji rješava neki nizvodni (engl. *downstream*) zadatak poput klasifikacije ili detekcije objekata. Pritom se okosnici dodaje klasifikacijska glava (najčešće nekoliko potpuno-povezanih slojeva), a cijeli model se dodatno ugađa nadziranim učenjem na manjem skupu podataka prilagođenom konkretnom problemu.

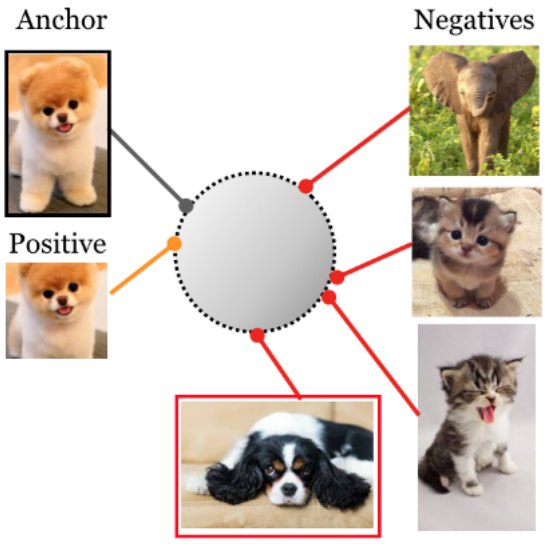
Ključno pitanje kod samonadziranog učenja je formiranje zadatka učenja bez oznaka tj. odlučivanje o tome na temelju čega će model dobivati signal za učenje. Tipični zadatci uključuju rekonstrukciju ulaznih podataka na temelju izlučenih reprezentacija, grupiranje reprezentacija tj. ugrađivanja (engl. *embedding*) semantički sličnih podataka u prostoru ugrađivanja ili predviđanje maskiranih piksela ulaznih slika (engl. *masked image modeling* - MIM) [28]. Rješavanjem jednog od zadataka učenja, model posredno uči izlučivati korisne reprezentacije ulaznih podataka ili uočavati korisne odnose između podataka.

Područje samonadziranog učenja možemo podijeliti na temelju korištenog tipa zadatka učenja, a neka od najpoznatijih područja su autoasocijativno samonadzirano učenje [29] i kontrastno samonadzirano učenje [17]. Kod autoasocijativnog samonadziranog učenja, osnovni zadatak je rekonstruirati ulazni podatak na temelju izlučene reprezentacije. Pritom je model koji učimo odgovoran za izlučivanje reprezentacije, dok za rekonstrukciju uobičajeno koristimo drugi model koji nakon učenja odbacujemo. U našem radu, fokusirat ćemo se na kontrastno samonadzirano učenje.

4.1. Kontrastno samonadzirano učenje

Kontrastno učenje jedno je od područja samonadziranog učenja. Ono podrazumijeva učenje izlučivanja korisnih reprezentacija tj. ugrađivanja ulaznih podataka na temelju parova podataka. Ako su dobivena ugrađivanja normirana, a sličnost dvaju ugrađivanja možemo izračunati koristeći neku od standardnih metrika (na primjer, kosinusnu sličnost), tada govorimo o metričkim ugrađivanjima tj. ugrađivanjima u metrički prostor [30]. Možemo reći da naučeni model preslikava ulazne slike na $(d-1)$-dimenzionalnu hipersferu S^{d-1}, pri čemu sličnost ugrađivanja odgovara semantičkoj sličnosti ulaza.

Kod kontrastnog učenja razlikujemo sidro, pozitivne i negativne primjere. Trenutno promatrani podatak iz minigrupe nazivamo sidro, podatak sličan sidru nazivamo pozitivan primjer, a podatak različit od sidra nazivamo negativan primjer. Budući da primjeri nisu označeni, pozitivne primjere najčešće dobivamo perturbacijom sidra, dok negativnim primjerima smatramo sve ostale podatke iz minigrupe.



Slika 4.1. Prikaz osnovne ideje kontrastnog samonadziranog učenja. Cilj učenja je približiti sidro (gore lijevo) i pozitivan primjer (dolje lijevo), ali i međusobno udaljiti sidro i negativne primjere (desno). Preuzeto iz [31].

Glavni cilj kontrastnog učenja je približiti ugrađivanja pozitivnih parova (sidra i pozitivnog primjera), ali i istovremeno udaljiti ugrađivanja negativnih parova (sidra i negativnih primjera). Kako bismo ovo postigli, veoma je važno prikladno definirati funkciju gubitka, a neke od mogućih su trojni gubitak [32] te gubitak N parova. Gubitak N parova također je poznat i kao infoNCE gubitak [33], a možemo ga definirati jednadžbom:

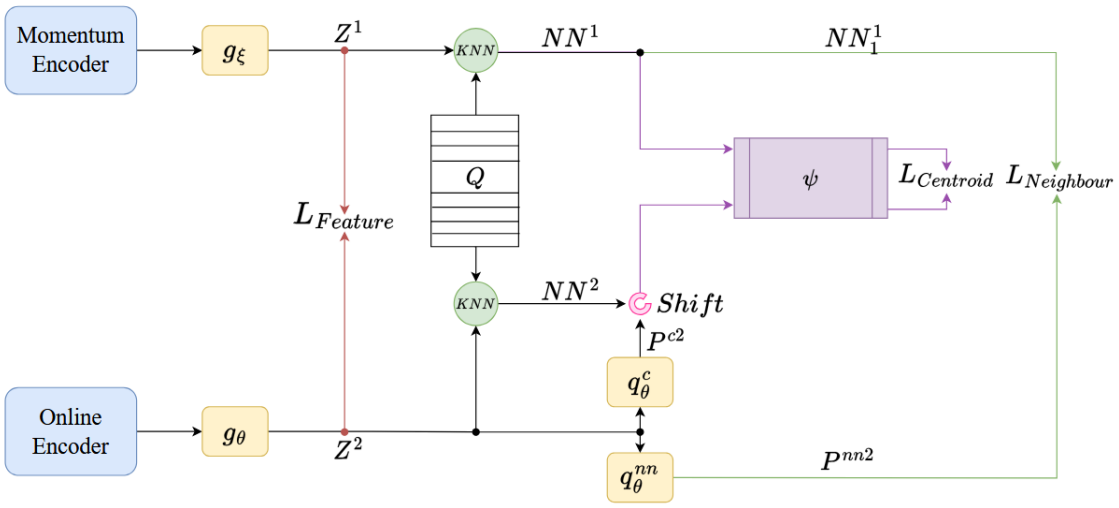
$$\mathcal{L}_{infoNCE} = -\log \frac{\exp(\langle \boldsymbol{z}_a, \boldsymbol{z}_p \rangle / \tau)}{\sum_{i=1}^{N} \exp(\langle \boldsymbol{z}_a, \boldsymbol{z}_{ni} \rangle / \tau)} \tag{4.1}$$

Pritom $\boldsymbol{z}_a$ označava ugrađivanje sidra $\boldsymbol{x}_a$, $\boldsymbol{z}_p$ ugrađivanje pozitivnog primjera $\boldsymbol{x}_p$, $\boldsymbol{z}_{ni}$ ugrađivanje i-tog negativnog primjera $\boldsymbol{x}_{ni}$ iz minigrupe, a τ hiperparametar temperature. Oznaka $\langle ... \rangle$ označava skalarni produkt vektora unutar zagrada.

U okviru za obranu od napada umetanjem stražnjih vrata VIBE, samonadzirana inicijalizacija parametara modela veoma je važan korak koji osigurava uspješnost obrane. Pritom za inicijalizaciju koristimo okvir All4One [34] koji se bazira na kontrastnom samonadziranom učenju.

4.2. All4One

Okvir All4One kombinira tri zasebna pristupa kontrastiranju kako bi naučeni modeli postizali što bolje rezultate. Konkretno, optimizacijski cilj okvira All4One sadrži komponentu kontrastiranja najbližih susjeda (engl. *nearest neighbour contrast*) [35], komponentu kontrastiranja centroida (engl. *centroid contrast*) te komponentu kontrastiranja značajki (engl. *feature contrast*) [36].



Slika 4.2. Prikaz arhitekture okvira All4One. Komponenta kontrastiranja najbližih susjeda označena je zelenom bojom, komponenta kontrastiranja centroida ljubičastom bojom, a komponenta kontrastiranja značajki crvenom bojom. Preuzeto iz [34].

U okviru All4One, model koji učimo služi kao koder značajki f_θ. Osim glavnog kodera koji učimo gradijentnim spustom, tijekom učenja se dodatno održava i koder s momentom f_ξ čiji parametri odgovaraju eksponencijalnom pomičnom prosjeku (engl. *exponential moving average* - EMA) parametara glavnog kodera. Dodatno, osim para kodera, učimo i par projektora g_θ i g_ξ. Za svaku sliku $\boldsymbol{x}$ na ulazu, generiraju se dvije augmentirane slike $\boldsymbol{x}^1$ te $\boldsymbol{x}^2$. Jedna od dobivenih augmentiranih slika tada prolazi kroz koder s momentom i pripadni projektor te dobivamo ugrađivanje $\boldsymbol{z}^1 = g_\xi(f_\xi(\boldsymbol{x}^1))$, dok druga augmentirana slika prolazi kroz glavni koder i pripadni projektor te dobivamo ugrađivanje $\boldsymbol{z}^2 = g_\theta(f_\theta(\boldsymbol{x}^2))$. Dobivena ugrađivanja dalje se mogu koristiti za izračun pojedinih komponenti optimizacijskog cilja. Osim para kodera i para projektora, dodatno se održavaju i dva prediktora q_θ^{nn} te q_θ^c koji služe za prilagodbu ugrađivanja za zadatak kontrastiranja najbližih susjeda odnosno zadatak kontrastiranja centroida.

4.2.1. Kontrastiranje najbližih susjeda

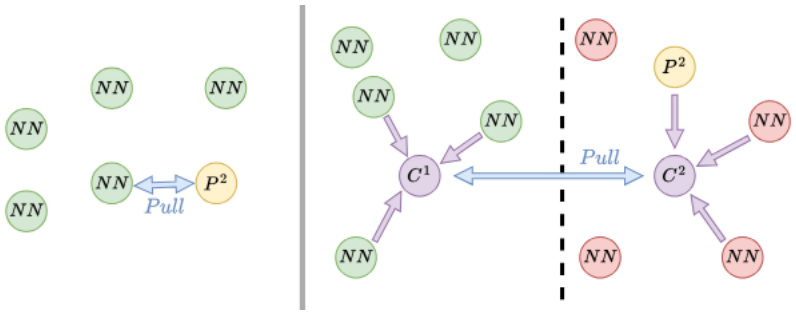
Kontrastiranje najbližih susjeda često je korišten pristup kontrastnog samonadziranog učenja kod kojega se ugrađivanje jedne od perturbiranih slika zamijeni sa svojim najbližim susjedom. Konkretno, tijekom učenja održavamo skup prethodnih ugrađivanja (engl. *support set*) Q koji možemo koristiti za pronalazak najbližih susjeda za neko zadano ugrađivanje. Ovu operaciju označit ćemo operatorom KNN. Gubitak za i-ti primjer $\boldsymbol{x}_i$ u minigrupi tada možemo definirati jednadžbom:

$$\mathcal{L}_{NNCLR} = -\log \frac{\exp(\langle \boldsymbol{nn}_i^1, \boldsymbol{p}_i^{nn,2} \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle \boldsymbol{nn}_i^1, \boldsymbol{p}_k^{nn,2} \rangle / \tau)} \tag{4.2}$$

Pritom $\boldsymbol{nn}_i^1$ označava najbližeg susjeda ugrađivanja prve perturbacije primjera $\boldsymbol{x}_i$, $\boldsymbol{p}_i^{nn,2}$ izlaz prediktora q_θ^{nn} za ugrađivanje druge perturbacije primjera $\boldsymbol{x}_i$, a τ hiperparametar temperature. Naspram osnovne inačice kontrastnog učenja, kontrastiranje najbližih susjeda povećava raznolikost primjera koje model vidi tijekom učenja, a time i generalizacijsku moć naučenog modela.

4.2.2. Kontrastiranje centroida

Ipak, pokazano je da modeli učeni kontrastiranjem najbližih susjeda mogu postići još bolje performanse ako se pritom koristi više od jednog najbližeg susjeda [37]. Nažalost, učenje s više od jednog najbližeg susjeda značajno povećava trajanje učenja jer se funkcija cilja treba evaluirati za svakog susjeda pojedinačno. Autori okvira All4One ovome problemu doskaču uvođenjem kontrastiranja centroida.



Slika 4.3. Prikaz kontrastiranja najbližih susjeda (lijevo) i kontrastiranja centroida (desno). Kod kontrastiranja najbližih susjeda, kontrastiraju se najbliži susjed jedne perturbacije i druga perturbacija. S druge strane, kod kontrastiranja centroida, kontrastiraju se centroidi pojedinih perturbacija. Preuzeto iz [34].

Konkretno, nakon što smo za zadani primjer $\boldsymbol{x}$ generirali ugrađivanja $\boldsymbol{z}^1$ te $\boldsymbol{z}^2$, za ista prvo pronalazimo skupove najbližih susjeda NN^1 odnosno NN^2. Ugrađivanje $\boldsymbol{z}^2$ tada prolazi kroz prediktor q_θ^c te dobivamo ugrađivanje $\boldsymbol{p}^{c,2}$. Ovim ugrađivanjem zamijenit ćemo posljednjeg najbližeg susjeda u skupu NN^2 te potom provesti operaciju kružnog posmaka (engl. *shift*) kojim će ugrađivanje $\boldsymbol{p}^{c,2}$ doći na prvo mjesto u skupu.

Kada imamo pripremljene skupove najbližih susjeda NN^1 te NN^2, iz istih želimo dobiti po jedno ugrađivanje koje sadrži informacije o svim najbližim susjedima iz pripadnog skupa. Kako bismo ovo postigli, koristimo mehanizam samopažnje (engl. *self-attention*) [38]. Konkretno, koristimo koder transformera ψ kojemu na ulaz dovodimo niz ugrađivanja Seq iz skupa NN obogaćen sinusoidalnim pozicijskim kodiranjem [38]. Na izlazu kodera ψ tada dobivamo niz ugrađivanja Seq^c obogaćenih kontekstualnim informacijama o preostalim susjedima iz skupa. Kako bismo za svaki skup imali samo jedno ugrađivanje, kao centroid $\boldsymbol{c}$ proglašavamo prvo ugrađivanje s izlaza kodera transformera Seq_1^c.

Ovaj postupak provodimo zasebno za skup NN^1 odnosno skup NN^2 kako bismo dobili centroide $\boldsymbol{c}^1$ odnosno $\boldsymbol{c}^2$. Konačno, gubitak za i-ti primjer $\boldsymbol{x}_i$ u minigrupi možemo definirati jednadžbom:

$$\mathcal{L}_{centroid} = -\log \frac{\exp(\langle \boldsymbol{c}_i^1, \boldsymbol{c}_i^2 \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle \boldsymbol{c}_i^1, \boldsymbol{c}_k^2 \rangle / \tau)} \tag{4.3}$$

4.2.3. Kontrastiranje značajki

Kontrastiranje značajki veoma se razlikuje od uobičajenih pristupa kontrastnom učenju. Kod ovog pristupa, ideja je direktno kontrastirati značajke ugrađivanja. Kako bismo ovo postigli, potrebno je izračunati korelacijsku matricu značajki za svaku minigrupu. Tada je cilj naučiti parametre modela koji korelacijsku matricu značajki što više približavaju prema jediničnoj matrici.

Konkretno, prvo za svaku minigrupu konstruiramo matrice $\boldsymbol{Z}^1$ odnosno $\boldsymbol{Z}^2$ u kojima reci označavaju ugrađivanja prve odnosno druge perturbacije pripadnih primjera iz minigrupe. Nakon provođenja L_2 normalizacije matrice po stupcima, korelacijsku matricu $\boldsymbol{CC}^1$ dobivamo računanjem kosinusne sličnosti između transponirane matrice $\boldsymbol{Z}^1$ te matrice $\boldsymbol{Z}^2$. Dodatno, korelacijsku matricu $\boldsymbol{CC}^2$ dobivamo uz zamjenu grana tj. zamjenu glavnog kodera i kodera s momentom te ponovan izračun sličnosti. Kada smo izračunali obje korelacijske matrice, gubitak za zadanu minigrupu možemo definirati jednadžbom:

$$\begin{aligned} \mathcal{L}_{feature} = & \frac{1}{2} \sqrt{\frac{1}{2D} \sum_{i=1}^{D} ((1 - \boldsymbol{CC}_{i,i}^1)^2 + (1 - \boldsymbol{CC}_{i,i}^2)^2)} \\ & + \frac{1}{2} \sqrt{\frac{1}{2D(D-1)} \sum_{i=1}^{D} \sum_{j \neq i}^{D} ((\boldsymbol{CC}_{i,j}^1)^2 + (\boldsymbol{CC}_{i,j}^2)^2)} \end{aligned} \tag{4.4}$$

Pritom D označava broj značajki tj. dimenzionalnost vektora ugrađivanja. Vidimo da će gubitak biti to manji što su korelacijske matrice bliže jediničnoj matrici. Prvi član gubitka pokušava povećati invarijantnost pojedinih značajki na augmentacije ulaznih slika, dok drugi član pokušava smanjiti međusobnu redundantnost značajki.

4.2.4. Kombinirano kontrastiranje

Dok prethodni algoritmi samonadziranog učenja većinom koriste samo jednu vrstu kontrastiranja (na primjer, samo kontrastiranje najbližih susjeda), okvir All4One kombinira sva tri navedena pristupa. Konkretno, gubitak koji želimo minimizirati možemo definirati jednadžbom:

$$\mathcal{L}_{All4One} = \sigma\mathcal{L}_{NNCLR} + \kappa\mathcal{L}_{centroid} + \eta\mathcal{L}_{feature} \tag{4.5}$$

Pritom su σ, κ i η hiperparametri koji određuju jačinu utjecaja pojedine vrste kontrastiranja. Kombiniranjem više pristupa kontrastnog učenja, okvir All4One poboljšava učenje korisnih reprezentacija ulaznih podataka.

5. Algoritam maksimizacije očekivanja

Zamislimo da imamo skup opaženih varijabli X te želimo pronaći parametre θ pretpostavljenog modela za koje je log-izglednost $\log p(X|\theta)$ maksimalna. Drugim riječima, zanima nas MLE (engl. *maximum likelihood estimation*) procjena [39] parametara θ. Osim što ovaj zadatak ponekad ima rješenje u zatvorenoj formi (engl. *closed-form solution*), općenito ga možemo riješiti gradijentnim postupcima poput stohastičkog gradijentnog spusta [40]. Ako uz opažene varijable X postoje i skrivene (latentne) varijable Z, problem pronalaska MLE procjene parametara θ postaje kompliciraniji. Konkretno, nepotpunu log-izglednost parametara tada možemo definirati kao:

$$\log p(X|\theta) = \log \int p(X, Z|\theta) dZ = \log \int p(X|Z, \theta) p(Z|\theta) dZ \tag{5.1}$$

Gledajući da optimizacijski cilj sada sadrži varijable Z čije vrijednosti ne znamo, više ga nije moguće direktno maksimizirati. Jedan od mogućih algoritama koji se nosi s postojanjem skrivenih varijabli je algoritam maksimizacije očekivanja (engl. *expectation maximization algorithm* - algoritam EM) [27]. Kao i gradijentni spust, i algoritam EM je iterativne prirode, ali se pritom svaka iteracija sastoji od dva zasebna koraka: E koraka i M koraka.

Pokazuje se da algoritam EM osigurava monotoni rast nepotpune log-izglednosti parametara $\log p(X|\theta)$, ali pritom ne postoji garancija da će algoritam konvergirati u globalni maksimum [41]. Drugim riječima, moguće je da dobiveni parametri θ odgovaraju lokalnom maksimumu nepotpune log-izglednosti, tako da se preporučuje koristiti heurističke pristupe poput ponovnog pokretanja algoritma s nasumično odabranim početnim vrijednostima parametara θ.

5.1. Osnovna ideja

Osnovna ideja algoritma EM je jednostavna. Tijekom učenja, u svakoj iteraciji prvo provodimo E korak, a nakon njega provodimo M korak. Gledajući da ne znamo vrijednosti skrivenih varijabli Z, kao ni vrijednosti optimalnih parametara θ, učenje možemo započeti uz nasumično odabrane vrijednosti parametara $\theta^{(0)}$.

Na početku E koraka fiksiramo trenutne vrijednosti parametara $\theta^{(t)}$. Tada računamo očekivanje potpune log-izglednosti $\log p(X, Z|\theta)$ uz uvjetnu distribuciju skrivenih varijabli $p(Z|X, \theta^{(t)})$. Izračunato očekivanje možemo definirati jednadžbom:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{Z\sim p(\cdot|X,\theta^{(t)})}\left[\log p(X, Z|\theta)\right] \tag{5.2}$$

U M koraku, cilj nam je pronaći parametre θ koji maksimiziraju izračunato očekivanje $Q(\theta|\theta^{(t)})$. Nakon što smo izračunali nove parametre $\theta^{(t+1)}$, možemo ih iskoristiti za provođenje E koraka sljedeće iteracije. Općenito, jednu iteraciju algoritma EM sažeto možemo prikazati kao:

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{Z\sim p(\cdot|X,\theta^{(t)})}\left[\log p(X, Z|\theta)\right] \tag{5.3}$$

Pokažimo sada detaljniji izvod algoritma maksimizacije očekivanja.

5.2. Izvod algoritma

Glavni cilj algoritma EM je pronalazak parametara θ koji maksimiziraju log-izglednost $\log p(X|\theta)$ uz postojanje skrivenih varijabli Z. Koristeći Bayesovu formulu, vjerojatnost $p(X|\theta)$ možemo prikazati kao:

$$p(X|\theta) = \frac{p(X|Z, \theta)p(Z|\theta)}{p(Z|X, \theta)} \tag{5.4}$$

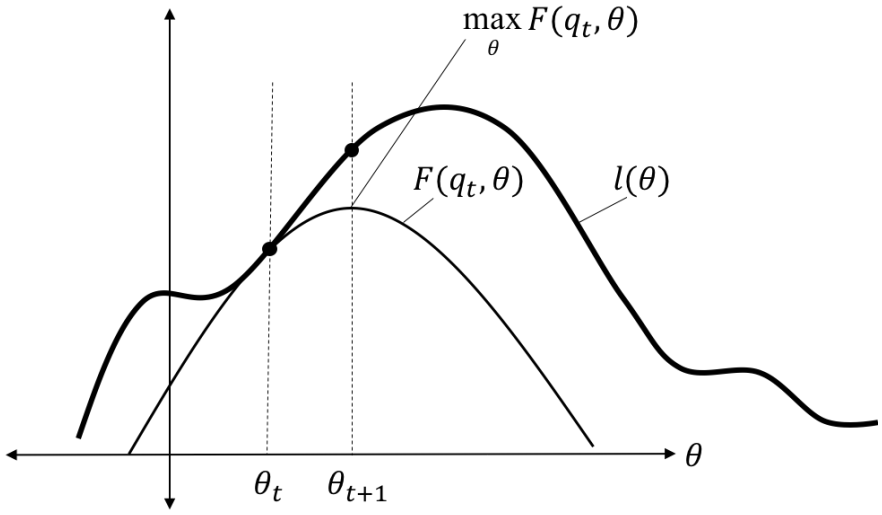
Definirajmo sada zamjensku distribuciju $q(Z)$ kao proizvoljnu distribuciju skrivenih varijabli Z. Ako izraz za vjerojatnost $p(X|\theta)$ pomnožimo i podijelimo s $q(Z)$ te potom primijenimo operaciju logaritmiranja, dobivamo:

$$\log p(X|\theta) = \log \frac{p(X|Z,\theta)p(Z|\theta)}{q(Z)} + \log \frac{q(z)}{p(Z|X,\theta)} \tag{5.5}$$

Izračunajmo sada očekivanje dobivenog izraza s obzirom na distribuciju $q(Z)$. Budući da se na lijevoj strani jednadžbe ne pojavljuju skrivene varijable Z, nepotpunu log-izglednost $\log p(X|\theta)$ konačno možemo definirati jednadžbom:

$$\begin{aligned} \log p(X|\theta) &= \int q(Z) \log \frac{p(X|Z,\theta)p(Z|\theta)}{q(Z)} dZ + \int q(Z) \log \frac{q(Z)}{p(Z|X,\theta)} dZ \\ &= F(q(Z),\theta) + KL(q(Z)||p(Z|X,\theta)) \end{aligned} \tag{5.6}$$

Pritom $F(q(Z),\theta)$ predstavlja donju varijacijsku granicu (engl. *evidence lower bound* - ELBO) log-izglednosti $\log p(X|\theta)$, a $KL(q(Z)||p(Z|X,\theta))$ predstavlja Kullback-Leibler (KL) divergenciju [42] između distribucija $q(Z)$ te $p(Z|X,\theta)$. Algoritam maksimizacije očekivanja sada možemo interpretirati kao koordinatni uspon (engl. *coordinate ascent*) [43] s ciljem maksimizacije donje varijacijske granice $F(q(Z),\theta)$.



Slika 5.1. Prikaz rada algoritma EM. $l(\theta)$ odgovara nepotpunoj log-izglednosti $\log p(X|\theta)$. E korak algoritma maksimizira donju varijacijsku granicu $F(q(Z),\theta^{(t)})$ s obzirom na distribuciju $q(Z)$, a M korak algoritma maksimizira istu s obzirom na parametre θ. Preuzeto iz [44].

E korak algoritma maksimizacije očekivanja sada maksimizira donju varijacijsku granicu $F(q(Z), \theta^{(t)})$ s obzirom na distribuciju $q(Z)$ uz fiksirane vrijednosti parametara $\theta^{(t)}$. Budući da lijeva strana jednadžbe 5.6 ne ovisi o distribuciji $q(Z)$, vidimo da je problem maksimizacije funkcije $F(q(Z), \theta^{(t)})$ s obzirom na distribuciju $q(Z)$ ekvivalentan problemu minimizacije izraza $KL(q(Z)||p(Z|X, \theta^{(t)}))$. Iznos KL divergencije jednak je 0 jedino kada vrijedi:

$$q(Z) = p(Z|X, \theta^{(t)}) \tag{5.7}$$

Ovime dobivamo rješenje E koraka, a uz dobivenu distribuciju $q^{(t+1)}(Z)$ dodatno vrijedi da je donja varijacijska granica $F(q^{(t+1)}(Z), \theta^{(t)})$ jednaka nepotpunoj log-izglednosti $\log p(X|\theta^{(t)})$. Uz uvrštavanje izraza za distribuciju $q^{(t+1)}(Z)$, donju varijacijsku granicu možemo zapisati kao:

$$\begin{aligned} F(p(Z|X, \theta^{(t)}), \theta) &= \int p(Z|X, \theta^{(t)}) \log \frac{p(X, Z|\theta)}{p(Z|X, \theta^{(t)})} dZ \\ &= \int \left[p(Z|X, \theta^{(t)}) \log p(X, Z|\theta) - p(Z|X, \theta^{(t)}) \log p(Z|X, \theta^{(t)}) \right] dZ \\ &= \mathbb{E}_{Z \sim p(\cdot|X, \theta^{(t)})} \left[\log p(X, Z|\theta) \right] - \mathbb{E}_{Z \sim p(\cdot|X, \theta^{(t)})} \left[\log p(Z|X, \theta^{(t)}) \right] \\ &= Q(\theta|\theta^{(t)}) + \mathrm{H}(p(Z|X, \theta^{(t)})) \end{aligned} \tag{5.8}$$

Pri čemu $\mathrm{H}(p(Z|X, \theta^{(t)}))$ označava entropiju [45] distribucije $p(Z|X, \theta^{(t)})$. Uočimo da je dobiveni izraz za donju varijacijsku granicu jednak izrazu $Q(\theta|\theta^{(t)})$, ali uvećan za iznos entropije $\mathrm{H}(p(Z|X, \theta^{(t)}))$. Kada smo odredili distribuciju $q^{(t+1)}(Z)$ koja maksimizira donju varijacijsku granicu, možemo prijeći na M korak.

M korak algoritma maksimizacije očekivanja sada maksimizira donju varijacijsku granicu $F(q^{(t+1)}(Z), \theta)$ s obzirom na parametre θ uz fiksiranu distribuciju $q^{(t+1)}(Z)$ dobivenu u E koraku. Dakle, cilj je pronaći parametre $\theta^{(t+1)}$ koji maksimiziraju izraz:

$$\begin{aligned} F(p(Z|X,\theta^{(t)}),\theta) &= Q(\theta|\theta^{(t)}) + \mathrm{H}(p(Z|X,\theta^{(t)})) \\ &= Q(\theta|\theta^{(t)}) + const. \end{aligned} \tag{5.9}$$

Pritom je član $\mathrm{H}(p(Z|X,\theta^{(t)}))$ konstantan s obzirom na trenutne parametre θ, tako da on ne utječe na tijek optimizacije. Vidimo da je izvedeni M korak identičan M koraku osnovne ideje - jedina razlika je prisutnost konstantnog člana u izrazu koji maksimiziramo. Dobivene parametre $\theta^{(t+1)}$ dalje možemo iskoristiti za provođenje E koraka sljedeće iteracije.

Pokažimo još da provođenjem algoritma maksimizacije očekivanja postižemo monotoni rast nepotpune log-izglednosti $\log p(X|\theta)$. Na kraju E koraka raspolažemo distribucijom $q^{(t+1)}(Z)$ uz koju vrijedi:

$$\log p(X|\theta^{(t)}) = F(q^{(t+1)}(Z),\theta^{(t)}) \tag{5.10}$$

Provođenjem M koraka dobivamo parametre $\theta^{(t+1)}$ uz koje vrijedi:

$$F(q^{(t+1)}(Z),\theta^{(t)}) \leq F(q^{(t+1)}(Z),\theta^{(t+1)}) \tag{5.11}$$

Gledajući da je funkcija $F(q(Z),\theta)$ donja granica log-izglednosti $\log p(X|\theta)$, vrijedi i:

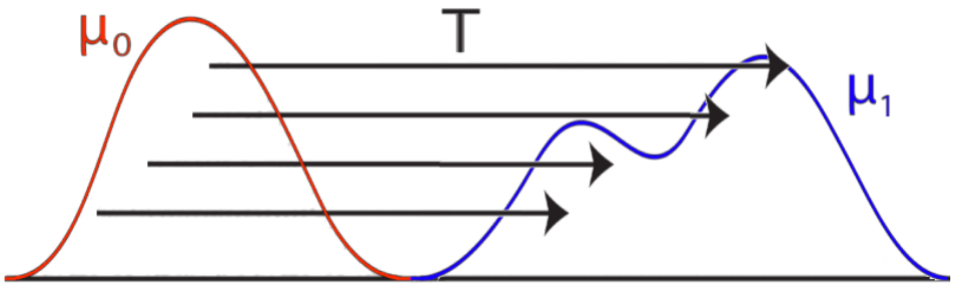
$$F(q^{(t+1)}(Z),\theta^{(t+1)}) \leq \log p(X|\theta^{t+1}) \tag{5.12}$$

Konačno, dobivamo početnu tvrdnju:

$$\log p(X|\theta^{(t)}) \leq \log p(X|\theta^{t+1}) \tag{5.13}$$

6. Problem optimalnog transporta

Problem optimalnog transporta (engl. *optimal transport problem*) [46] bavi se pitanjem premještanja jedne raspodjele (distribucije) mase u drugu, pritom nastojeći minimizirati ukupnu cijenu transporta. Drugim riječima, cilj je pronaći optimalni plan transporta za koji je ukupna cijena premještanja jedne distribucije u drugu minimalna.



Slika 6.1. Prikaz problema optimalnog transporta. Plan transporta T opisuje kako preoblikovati distribuciju μ_0 u distribuciju μ_1. Preuzeto iz [47].

Općenito govoreći, dvije najpoznatije formulacije problema su Mongeova formulacija te Kantorovičeva formulacija, a po istima se problem optimalnog transporta često naziva i Monge-Kantorovičev problem [48]. Kod Mongeove formulacije problema, optimalni plan transporta je bijekcija - jedna vrijednost iz prve distribucije može se preslikati u samo jednu vrijednost iz druge distribucije. Ova formulacija ekvivalentna je problemu uparivanja bipartitnog grafa (engl. *bipartite graph matching*) [49].

S druge strane, kod Kantorovičeve formulacije problema, optimalni plan transporta možemo prikazati matricom zajedničke vjerojatnosti uz nekoliko dodatnih ograničenja. U okviru našeg rada, fokusirat ćemo se na Kantorovičevu formulaciju problema optimalnog transporta, kao i jedno moguće rješenje istoga.

6.1. Općenita formulacija

Neka su $\boldsymbol{\mu}$ i $\boldsymbol{\nu}$ vektori distribucija vjerojatnosti nad prostorima X odnosno Y. Vektor $\boldsymbol{\mu}$ neka je duljine M, a vektor $\boldsymbol{\nu}$ neka je duljine N. Cilj problema je pronaći optimalni plan transporta koji distribuciju $\boldsymbol{\mu}$ preslikava u distribuciju $\boldsymbol{\nu}$ uz što nižu ukupnu cijenu. Cijenu transporta između točke $x \in X$ i točke $y \in Y$ označit ćemo s $\boldsymbol{C}_{x,y}$, pri čemu je $\boldsymbol{C}$ matrica cijena transporta. Neki zadani plan transporta definirat ćemo matricom $\boldsymbol{Q}$. Važno je napomenuti da su matrice $\boldsymbol{C}$ i $\boldsymbol{Q}$ definirane nad prostorom $X \times Y$ odnosno dimenzija su $M \times N$. Dodatno, svaki plan transporta mora zadovoljavati sljedeće uvjete:

$$\begin{aligned} \boldsymbol{Q}\mathbf{1}_N &= \boldsymbol{\mu} \\ \boldsymbol{Q}^T\mathbf{1}_M &= \boldsymbol{\nu} \end{aligned} \tag{6.1}$$

Drugim riječima, plan transporta $\boldsymbol{Q}$ mora se marginalizirati po stupcima u vektor $\boldsymbol{\mu}$ odnosno po recima u vektor $\boldsymbol{\nu}$. Gledajući da je u pitanju matrica vjerojatnosti, svaki element q_{ij} mora biti nenegativan. Ovim skupom linearnih ograničenja definiran je politop [50]:

$$\boldsymbol{Q}\left[\boldsymbol{\mu}, \boldsymbol{\nu}\right] = \{\boldsymbol{Q} \in \mathbb{R}_+^{M \times N} | \boldsymbol{Q}\mathbf{1}_N = \boldsymbol{\mu}\,, \boldsymbol{Q}^T\mathbf{1}_M = \boldsymbol{\nu}\} \tag{6.2}$$

Svako rješenje problema mora se nalaziti na zadanom politopu $\boldsymbol{Q}\left[\boldsymbol{\mu}, \boldsymbol{\nu}\right]$. Problem optimalnog transporta tada možemo definirati kao potragu za matricom $\boldsymbol{Q}$ koja zadovoljava:

$$\min_{\boldsymbol{Q}} \operatorname{tr}(\boldsymbol{C}^T\boldsymbol{Q}) \tag{6.3}$$

Pritom tr označava operaciju traga matrice, a $\boldsymbol{C}^T$ označava transponiranu matricu $\boldsymbol{C}$. Ovako formuliran problem možemo riješiti proizvoljnim algoritmom linearnog programiranja [51]. Matrica $\boldsymbol{Q}$ dobivena kao rješenje zadanog problema optimalnog transporta često će biti rijetka (engl. *sparse*). Ako želimo osigurati da rješenje $\boldsymbol{Q}$ što manje odgovara rijetkoj matrici, u problem možemo dodati komponentu entropijske regularizacije. Problem optimalnog transporta s entropijskom regularizacijom [52] možemo definirati jednadžbom:

$$\min_{\boldsymbol{Q}} \left[\operatorname{tr}(\boldsymbol{C}^T \boldsymbol{Q}) - \frac{1}{\lambda} \mathrm{H}(\boldsymbol{Q}) \right] \tag{6.4}$$

Pri čemu je λ regularizacijski hiperparametar, a $\mathrm{H}(\boldsymbol{Q})$ entropija matrice vjerojatnosti dana kao:

$$\mathrm{H}(\boldsymbol{Q}) = -\sum_{i=1}^{M} \sum_{j=1}^{N} \boldsymbol{Q}_{i,j} \log \boldsymbol{Q}_{i,j} \tag{6.5}$$

Ovisno o iznosu hiperparametra λ, član $-\frac{1}{\lambda} \mathrm{H}(\boldsymbol{Q})$ može potaknuti gustoću matrice plana transporta $\boldsymbol{Q}$ te time spriječiti degeneraciju rješenja. Jedan od algoritama kojim možemo riješiti problem optimalnog transporta s entropijskom regularizacijom je algoritam Sinkhorn-Knopp (algoritam SK) [53].

6.2. Algoritam Sinkhorn-Knopp

Sinkhornov teorem [54] govori da za svaku matricu s pozitivnim vrijednostima $\boldsymbol{A}$ dimenzija $N \times N$ postoje dijagonalne matrice $\boldsymbol{U}$ i $\boldsymbol{V}$ s pozitivnim vrijednostima takve da je matrica $\boldsymbol{UAV}$ dvostruko stohastička tj. da se svaki redak i stupac dobivene matrice sumiraju u 1. Pritom matrice možemo definirati kao $\boldsymbol{U} = \operatorname{diag}(\boldsymbol{u})$ odnosno $\boldsymbol{V} = \operatorname{diag}(\boldsymbol{v})$, pri čemu je $\operatorname{diag}(\boldsymbol{u})$ dijagonalna matrica s vrijednostima vektora $\boldsymbol{u}$. Originalna namjena algoritma Sinkhorn-Knopp bila je proizvesti dvostruko stohastičku matricu $\boldsymbol{UAV}$ na temelju dane matrice $\boldsymbol{A}$.

Uočimo da matrice $\boldsymbol{U}$ i $\boldsymbol{V}$ zapravo skaliraju retke odnosno stupce matrice $\boldsymbol{A}$. Zadatak algoritma tada postaje pronaći pripadne vektore $\boldsymbol{u}$ i $\boldsymbol{v}$ koji ispravno skaliraju zadanu matricu. Kako bismo ovo postigli, kod algoritma SK alterniramo između ažuriranja vektora $\boldsymbol{u}$ na temelju suma pojedinih redaka odnosno vektora $\boldsymbol{v}$ na temelju suma pojedinih stupaca. Konkretno, nakon inicijalizacije vektora na proizvoljne vrijednosti (na primjer, na vektore jedinica $\mathbf{1}_N$), u svakoj iteraciji ih ažuriramo koristeći sljedeće jednadžbe:

$$\begin{aligned} \boldsymbol{u}^{(t+1)} &= \frac{\mathbf{1}_N}{\boldsymbol{A}\boldsymbol{v}^{(t)}} \\ \boldsymbol{v}^{(t+1)} &= \frac{\mathbf{1}_N}{\boldsymbol{A}^T \boldsymbol{u}^{(t)}} \end{aligned} \tag{6.6}$$

Nakon dovoljnog broja iteracija, vektori $\boldsymbol{u}$ i $\boldsymbol{v}$ će konvergirati u očekivane vrijednosti, a konačnu dvostruko stohastičku matricu dobivamo kao $\boldsymbol{UAV}$. Osim za postizanje dvostruke stohastičnosti dane matrice, pokazano je da se algoritam Sinkhorn-Knopp može koristiti i za rješavanje problema optimalnog transporta s entropijskom regularizacijom [55].

Prisjetimo se danog problema. Uz vektore distribucija vjerojatnosti $\boldsymbol{\mu}$ i $\boldsymbol{\nu}$ definiranih nad prostorima X odnosno Y te matricu cijena transporta $\boldsymbol{C}$ dimenzija $M \times N$, cilj je pronaći matricu plana transporta $\boldsymbol{Q}$ koja se nalazi na politopu $\boldsymbol{Q}[\boldsymbol{\mu}, \boldsymbol{\nu}]$ te za koju vrijedi:

$$\min_{Q} \left[\operatorname{tr}(\boldsymbol{C}^T \boldsymbol{Q}) - \frac{1}{\lambda} \operatorname{H}(\boldsymbol{Q}) \right] \tag{6.7}$$

Pritom je λ regularizacijski hiperparametar, a $\operatorname{H}(\boldsymbol{Q})$ entropija matrice vjerojatnosti. Kako bismo izveli pravila za ažuriranje temeljena na algoritmu SK, definirajmo prvo Lagrangeovu funkciju [56]:

$$\mathcal{L}(\boldsymbol{Q}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \operatorname{tr}(\boldsymbol{C}^T \boldsymbol{Q}) - \frac{1}{\lambda} \operatorname{H}(\boldsymbol{Q}) + \boldsymbol{\alpha}^T (\boldsymbol{Q}\mathbf{1}_N - \boldsymbol{\mu}) + \boldsymbol{\beta}^T (\boldsymbol{Q}^T \mathbf{1}_M - \boldsymbol{\nu}) \tag{6.8}$$

Vektori $\boldsymbol{\alpha}$ i $\boldsymbol{\beta}$ odgovaraju Lagrangeovim multiplikatorima uz linearne uvjete jednakosti. Pogledajmo sada kako izgleda derivacija dobivene Lagrangeove funkcije $\mathcal{L}(\boldsymbol{Q}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ po proizvoljnom elementu matrice plana transporta $\boldsymbol{Q}_{i,j}$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{Q}_{i,j}} = \boldsymbol{C}_{i,j} + \frac{1}{\lambda}(1 + \log \boldsymbol{Q}_{i,j}) + \alpha_i + \beta_j \tag{6.9}$$

Izjednačavanjem dobivene parcijalne derivacije s 0, dobivamo izraz za element matrice plana transporta $\boldsymbol{Q}_{i,j}$:

$$\boldsymbol{Q}_{i,j} = \exp(-1) \exp(-\lambda \boldsymbol{C}_{i,j}) \exp(-\lambda \alpha_i) \exp(-\lambda \beta_j) \tag{6.10}$$

Pritom član $\exp(-1)$ možemo ignorirati jer je on konstantan $\forall i, j$. Uvođenjem:

$$\begin{aligned} \boldsymbol{A}_{i,j} &= \exp(-\lambda \boldsymbol{C}_{i,j}) \\ u_i &= \exp(-\lambda \alpha_i) \\ v_j &= \exp(-\lambda \beta_j) \end{aligned} \tag{6.11}$$

Dolazimo do formulacije iz Sinkhornovog teorema:

$$\boldsymbol{Q} = \operatorname{diag}(\boldsymbol{u})\, \boldsymbol{A}\, \operatorname{diag}(\boldsymbol{v}) \tag{6.12}$$

Gledajući da se svako rješenje $\boldsymbol{Q}$ mora nalaziti na politopu $\boldsymbol{Q}\,[\boldsymbol{\mu}, \boldsymbol{\nu}]$, mora vrijediti:

$$\begin{aligned} \operatorname{diag}(\boldsymbol{u})\, \boldsymbol{A}\, \operatorname{diag}(\boldsymbol{v}) \mathbf{1}_N &= \boldsymbol{\mu} \\ \operatorname{diag}(\boldsymbol{v})\, \boldsymbol{A}^T\, \operatorname{diag}(\boldsymbol{u}) \mathbf{1}_M &= \boldsymbol{\nu} \end{aligned} \tag{6.13}$$

Ove izraze možemo dodatno pojednostaviti:

$$\begin{aligned} \boldsymbol{u} \odot (\boldsymbol{A}\boldsymbol{v}) &= \boldsymbol{\mu} \\ \boldsymbol{v} \odot (\boldsymbol{A}^T \boldsymbol{u}) &= \boldsymbol{\nu} \end{aligned} \tag{6.14}$$

Pronalaskom vektora $\boldsymbol{u}$ i $\boldsymbol{v}$ koji zadovoljavaju dana ograničenja, pronašli smo i rješenje $\boldsymbol{Q}$. Konačno, slijedeći algoritam Sinkhorn-Knopp, na temelju dobivenih ograničenja izvodimo izraze za iterativno ažuriranje vektora $\boldsymbol{u}$ i $\boldsymbol{v}$:

$$\begin{aligned} \boldsymbol{u}^{(t+1)} &= \frac{\boldsymbol{\mu}}{\boldsymbol{A}\boldsymbol{v}^{(t)}} \\ \boldsymbol{v}^{(t+1)} &= \frac{\boldsymbol{\nu}}{\boldsymbol{A}^T \boldsymbol{u}^{(t)}} \end{aligned} \tag{6.15}$$

7. VIBE

Okvir VIBE (engl. *Variational inference for backdoor elimination*) [10] zasniva se na ideji da su podatci i zatrovane oznake realizacije uočenih slučajnih varijabli, dok su čiste oznake skrivene varijable. Cilj okvira tada je naučiti model koji na izlazu daje vjerojatnost čistih oznaka, bez obzira na mogućnost da je skup podataka zatrovan. Kako bismo ovo postigli, model učimo koristeći algoritam EM. Pritom u E koraku procjenjujemo čiste pseudooznake rješavajući problem optimalnog transporta s entropijskom regularizacijom, a u M koraku ažuriramo parametre modela koristeći gradijentni spust.

Neka je $\mathcal{D}_{clean} = \{(\tilde{\boldsymbol{x}}^i, l^i)\}_{i=1}^N$ čisti skup podataka, pri čemu je $\tilde{\boldsymbol{x}}^i \in X$ ulazni primjer, a $l^i \in Y$ čista oznaka. Napadač modificira udio γ čistog skupa podataka te time nastaje zatrovani skup $\mathcal{D} = \{(\boldsymbol{x}^i, y^i)\}_{i=1}^N$. Pritom ulazni primjer $\boldsymbol{x}^i \in X$ može sadržavati okidač, a oznaka $y^i \in Y$ može biti zatrovana. Cilj okvira VIBE je naučiti model h koji potencijalno zatrovanom ulaznom primjeru $\boldsymbol{x}^i$ dodjeljuje čistu oznaku l^i. Naravno, čiste oznake $\boldsymbol{l}$ su skrivene, a dostupne su nam samo zatrovane oznake $\boldsymbol{y}$.

Uobičajeni napadi umetanjem stražnjih vrata modificiraju mali udio γ čistog skupa jer žele izbjeći detekciju. Drugim riječima, za većinu podataka će zatrovane i čiste oznake biti identične. Zbog ovoga, prirodan optimizacijski cilj za okvir VIBE je maksimizacija log-izglednosti skupa $\mathcal{D}$ uz pretpostavku o nezavisno i identično distribuiranim primjerima (IID pretpostavku):

$$\ell_{VD}(\cdot|\mathcal{D}) = \log \prod_{i=1}^{N} p(y^i|\boldsymbol{x}^i) = \sum_{i=1}^{N} \log \sum_{l=1}^{K} p(y^i|l)p(l|\boldsymbol{x}^i) \tag{7.1}$$

Pritom smo problem pojednostavili aproksimiranjem distribucije $p(y|l, \boldsymbol{x})$ distribucijom $p(y|l)$. Drugim riječima, uveli smo model šuma koji ne ovisi o konkretnom primjeru. Kako bismo mogli optimirati zadani cilj, potrebno je definirati distribuciju $p(l|\boldsymbol{x})$ odnosno distribuciju $p(y|l)$.

7.1. Parametrizacija distribucija

Model h podijelit ćemo na okosnicu tj. ekstraktor značajki g_θ te klasifikacijsku glavu. Pritom ekstraktor značajki na ulazu dobiva primjer $\boldsymbol{x}^i$, a na izlazu daje ugrađivanje $\boldsymbol{v}^i = g_\theta(\boldsymbol{x}^i)$ definirano na $(d-1)$-dimenzionalnoj jediničnoj hipersferi S^{d-1}. Uvjetnu vjerojatnost ugrađivanja $\boldsymbol{v}^i$ s obzirom na čistu oznaku l^i definirat ćemo kao von Mises-Fisherovu distribuciju (distribuciju vMF) [57]:

$$p_{\boldsymbol{\phi}}(\boldsymbol{v}^i|l^i) = C_d(\kappa)\exp(\kappa\boldsymbol{\mu}_{l^i}^T\boldsymbol{v}^i) \tag{7.2}$$

Pri čemu vektor $\boldsymbol{\mu}_{l^i} \in S^{d-1}$ određuje smjer središta distribucije na hipersferi, hiperparametar κ određuje raspršenost distribucije, a $C_d(\kappa)$ je normalizacijska konstanta. Primjer $\boldsymbol{x}^i$ jednoznačno određuje ugrađivanje $\boldsymbol{v}^i$, tako da vrijedi:

$$p_{\theta,\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i) = p_{\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{v}^i) \tag{7.3}$$

Koristeći Bayesovu formulu, vjerojatnost $p_{\theta,\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x})$ sada možemo definirati kao mješavinu distribucija vMF:

$$p_{\theta,\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i) = \frac{p_{\boldsymbol{\phi}}(\boldsymbol{v}^i|l^i)p_{\pi}(l^i)}{\sum_{l'} p_{\boldsymbol{\phi}}(\boldsymbol{v}^i|l')p_{\pi}(l')} = \frac{p_{\boldsymbol{\phi}}(\boldsymbol{v}^i|l^i)\boldsymbol{\pi}_{l^i}}{\sum_{l'} p_{\boldsymbol{\phi}}(\boldsymbol{v}^i|l')\boldsymbol{\pi}_{l'}} = \frac{\exp(\kappa\boldsymbol{\mu}_{l^i}^T\boldsymbol{v}^i + \log\boldsymbol{\pi}_{l^i})}{\sum_{l'}\exp(\kappa\boldsymbol{\mu}_{l'}^T\boldsymbol{v}^i + \log\boldsymbol{\pi}_{l'})} \tag{7.4}$$

Pritom koeficijenti miješanja $\boldsymbol{\pi}$ odgovaraju apriornoj distribuciji čistih oznaka $\boldsymbol{l}$. Kako bismo osigurali da distribuciju $\boldsymbol{\pi}$ možemo učiti, definiramo ju kao $\boldsymbol{\pi} = \sigma(c \cdot \hat{\boldsymbol{\pi}})$, pri čemu je σ funkcija softmax, c hiperparametar temperature, a $\hat{\boldsymbol{\pi}}$ vektor parametara koje učimo. Ovom formulacijom osiguravamo da se vektor $\boldsymbol{\pi}$ sumira u 1.

Uočimo da dobivena uvjetna vjerojatnost $p_{\theta,\boldsymbol{\phi},\pi}(l|\boldsymbol{x})$ odgovara dubokom modelu s funkcijom softmax, L_2 normalizacijom pred-logita te skupom razrednih prototipova $\boldsymbol{\phi}$. Parametri definiranog modela tada su: skup razrednih tj. čistih prototipova $\boldsymbol{\phi} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$, koeficijenti miješanja $\boldsymbol{\pi}$ te parametri θ ekstraktora značajki g_θ. Uz ovu parametrizaciju, model h koji na ulazu prima primjer $\boldsymbol{x}^i$ te na izlazu daje čistu oznaku l^i možemo prikazati kao kompoziciju nekoliko funkcija:

$$h = \arg\max \circ \text{cos-sim}_{\boldsymbol{\phi},\pi} \circ g_\theta \tag{7.5}$$

Pritom funkcija $\text{cos-sim}_{\boldsymbol{\phi},\pi}$ računa kosinusnu sličnost između prototipova iz skupa $\boldsymbol{\phi}$ i ugrađivanja $\boldsymbol{v}^i$ otežanu parametrima $\boldsymbol{\pi}$.

Sada trebamo definirati distribuciju $p(y|l)$. Kako bismo to postigli, uvedimo još i skup zatrovanih prototipova $\boldsymbol{\psi} = \{\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_K\}$ na istoj hipersferi S^{d-1}. Distribuciju $p(y|l)$ definirat ćemo kao normaliziranu kosinusnu sličnost između čistih i zatrovanih prototipova:

$$p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i) = \frac{\exp(\nu \boldsymbol{\mu}_{l^i}^T \boldsymbol{\eta}_{y^i})}{\sum_{y'} \exp(\nu \boldsymbol{\mu}_{l^i}^T \boldsymbol{\eta}_{y'})} \tag{7.6}$$

Pri čemu je ν hiperparametar temperature. Uočimo da koristeći naučenu distribuciju $p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y|l)$ možemo rekonstruirati pravila trovanja korištena za stvaranje zatrovanog skupa podataka $\mathcal{D}$. Skup svih parametara modela označit ćemo s $\boldsymbol{\Omega} = \theta \cup \boldsymbol{\pi} \cup \boldsymbol{\phi} \cup \boldsymbol{\psi}$. Nakon što smo parametrizirali obje potrebne distribucije, vrijeme je za definiranje algoritma učenja.

7.2. Optimizacija varijacijskog cilja

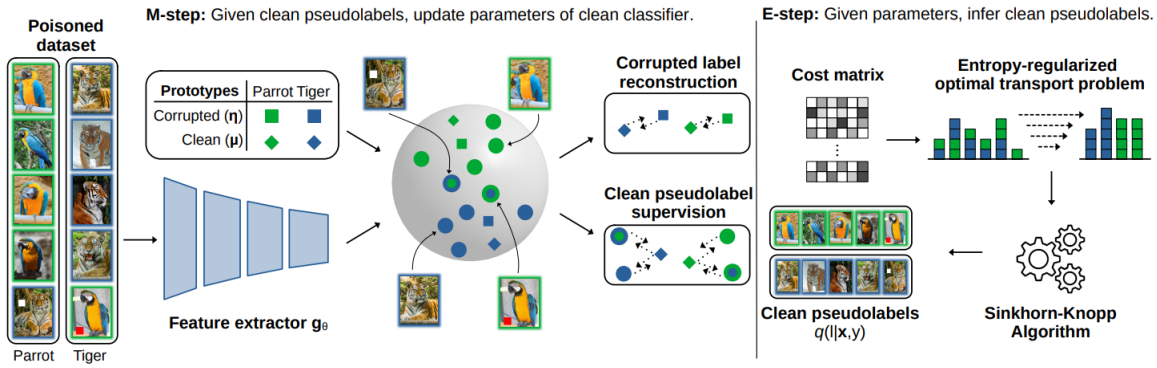
Nažalost, pokazuje se da direktna optimizacija cilja ℓ_{VD} ne osigurava ispravno učenje distribucije $p_{\theta,\boldsymbol{\phi},\pi}(l|\boldsymbol{x})$ [58]. Kako bismo doskočili ovom problemu, uvodimo zamjensku distribuciju $q(l|\boldsymbol{x}, y)$. Cilj ℓ_{VD} sada možemo zapisati kao:

$$\begin{aligned}\ell_{VD}(\mathbf{\Omega}|\mathcal{D}) &= \sum_{i=1}^{N} \log \sum_{l=1}^{K} p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l)\, p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i) \frac{q(l|\boldsymbol{x}^i, y^i)}{q(l|\boldsymbol{x}^i, y^i)} \\ &= \sum_{i=1}^{N} \log \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\frac{p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i)\, p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i)}{q(l^i|\boldsymbol{x}^i, y^i)} \right]\end{aligned} \tag{7.7}$$

Prisjetimo se da je logaritam konkavna funkcija. Primjenom Jensenove nejednakosti [59] dalje dobivamo:

$$\begin{aligned}\ell_{VD}(\mathbf{\Omega}|\mathcal{D}) &= \sum_{i=1}^{N} \log \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\frac{p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i)\, p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i)}{q(l^i|\boldsymbol{x}^i, y^i)} \right] \\ &\geq \sum_{i=1}^{N} \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\log \frac{p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i)\, p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i)}{q(l^i|\boldsymbol{x}^i, y^i)} \right] \\ &= \sum_{i=1}^{N} \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i) + \log p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i) - \log q(l^i|\boldsymbol{x}^i, y^i) \right] \\ &= \mathcal{L}_{VD}(\mathbf{\Omega}, q|\mathcal{D})\end{aligned} \tag{7.8}$$

Dobivena funkcija $\mathcal{L}_{VD}(\mathbf{\Omega}, q|\mathcal{D})$ predstavlja donju varijacijsku granicu (ELBO) log-izglednosti $\ell_{VD}(\mathbf{\Omega}|\mathcal{D})$, a istu možemo optimirati algoritmom maksimizacije očekivanja. Konkretno, u E koraku ćemo uz fiksirane vrijednosti parametara $\mathbf{\Omega}$ procijeniti distribuciju $q(l|\boldsymbol{x}, y)$ za skup podataka $\mathcal{D}$ rješavajući problem optimalnog transporta s entropijskom regularizacijom. Nakon toga, u M koraku ćemo uz fiksiranu distribuciju $q(l|\boldsymbol{x}, y)$ ažurirati parametre modela $\mathbf{\Omega}$ koristeći gradijentni spust.



Slika 7.1. Prikaz učenja okvirom VIBE. U E koraku (desno) procjenjujemo pseudooznake tj. distribuciju $q(l|\boldsymbol{x}, y)$, a u M koraku (lijevo) ažuriramo parametre modela $\mathbf{\Omega}$. Preuzeto iz [10].

Kako bismo osigurali da učenje algoritmom maksimizacije očekivanja konvergira k parametrima $\boldsymbol{\Omega}$ koji imaju dobru generalizacijsku moć, važno je prikladno inicijalizirati model. Nasumična inicijalizacija parametara ne nudi nikakvu garanciju uspješnosti algoritma EM, tako da umjesto nje koristimo samonadziranu inicijalizaciju okvirom All4One. Pokazuje se da korištenje samonadzirane inicijalizacije parametara ubrzava konvergenciju učenja, a i rezultira modelom koji bolje generalizira [10].

7.2.1. E korak

Cilj E koraka algoritma maksimizacije očekivanja je pronaći distribuciju $q(l|\boldsymbol{x}, y)$ koja maksimizira donju varijacijsku granicu $\mathcal{L}_{VD}(\boldsymbol{\Omega}, q|\mathcal{D})$ uz fiksirane vrijednosti parametara $\boldsymbol{\Omega}$. Donju varijacijsku granicu uprosječenu preko N primjera možemo zapisati kao:

$$\begin{aligned}
\frac{1}{N}\mathcal{L}_{VD}(\boldsymbol{\Omega}, q|\mathcal{D}) &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{l^i\sim q(\cdot|\boldsymbol{x}^i,y^i)}\left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i) + \log p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i) - \log q(l^i|\boldsymbol{x}^i, y^i)\right] \\
&= \sum_{i=1}^{N}\sum_{l=1}^{K}\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l) + \log p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i) - \log q(l|\boldsymbol{x}^i, y^i)\right] \\
&= \sum_{i=1}^{N}\sum_{l=1}^{K}\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\log\left[p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l)p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i)\right] \\
&- \sum_{i=1}^{N}\sum_{l=1}^{K}\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\log\left[N\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\right]
\end{aligned} \tag{7.9}$$

Uvedimo sada matrice $\boldsymbol{P}$ i $\boldsymbol{Q}$ za koje vrijedi:

$$\begin{aligned}
\boldsymbol{P}_{i,l} &= p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l)p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i) \\
\boldsymbol{Q}_{i,l} &= \frac{1}{N}q(l|\boldsymbol{x}^i, y^i)
\end{aligned} \tag{7.10}$$

Pritom član $\frac{1}{N}$ osigurava da je matrica $\boldsymbol{Q}$ ispravna matrica zajedničke vjerojatnosti. Cilj $\frac{1}{N}\mathcal{L}_{VD}(\boldsymbol{\Omega}, q|\mathcal{D})$ sada možemo prikazati matričnim operacijama:

$$
\begin{aligned}
\frac{1}{N}\mathcal{L}_{VD}(\boldsymbol{\Omega}, q|\mathcal{D}) &= \sum_{i=1}^{N}\sum_{l=1}^{K}\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\log\left[p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l)p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i)\right] \\
&\quad - \sum_{i=1}^{N}\sum_{l=1}^{K}\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\log\left[N\frac{1}{N}q(l|\boldsymbol{x}^i, y^i)\right] \\
&= \sum_{i=1}^{N}\sum_{l=1}^{K}\boldsymbol{Q}_{i,l}\log\boldsymbol{P}_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{K}\boldsymbol{Q}_{i,l}\log\left[N\boldsymbol{Q}_{i,l}\right] \\
&= \sum_{i=1}^{N}\sum_{l=1}^{K}\boldsymbol{Q}_{i,l}\log\boldsymbol{P}_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{K}\boldsymbol{Q}_{i,l}\log\boldsymbol{Q}_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{K}\boldsymbol{Q}_{i,l}\log N \\
&= \operatorname{tr}(\boldsymbol{Q}^T\log\boldsymbol{P}) + \mathrm{H}(\boldsymbol{Q}) - \log N
\end{aligned}
\tag{7.11}
$$

Pri čemu tr označava operaciju traga matrice, a $\mathrm{H}(\boldsymbol{Q})$ entropiju matrice $\boldsymbol{Q}$ danu jednadžbom 6.4 . Član $-\log N$ pritom možemo ignorirati jer je konstanta. Uz uvođenje regularizacijskog hiperparametra λ za koji vrijedi $\lambda > 1$, dobivamo:

$$
\begin{aligned}
\frac{1}{N}\mathcal{L}_{VD}(\boldsymbol{\Omega}, q|\mathcal{D}) &= \operatorname{tr}(\boldsymbol{Q}^T\log\boldsymbol{P}) + \mathrm{H}(\boldsymbol{Q}) - \log N \\
&\geq \operatorname{tr}(\boldsymbol{Q}^T\log\boldsymbol{P}) + \frac{1}{\lambda}\mathrm{H}(\boldsymbol{Q})
\end{aligned}
\tag{7.12}
$$

Maksimizacija dobivenog cilja ekvivalentna je pronalasku matrice $\boldsymbol{Q}$ za koju vrijedi:

$$
\min_{\boldsymbol{Q}}\left[-\operatorname{tr}(\boldsymbol{Q}^T\log\boldsymbol{P}) - \frac{1}{\lambda}\mathrm{H}(\boldsymbol{Q})\right] \tag{7.13}
$$

Uočimo da ovo odgovara problemu optimalnog transporta s entropijskom regularizacijom uz matricu cijena $-\log\boldsymbol{P}$. Pritom matrica $\boldsymbol{Q}$ mora zadovoljavati ograničenja:

$$
\begin{aligned}
\boldsymbol{Q}\mathbf{1}_K &= \frac{1}{N}\mathbf{1}_N \\
\boldsymbol{Q}^T\mathbf{1}_N &= \boldsymbol{\pi}
\end{aligned}
\tag{7.14}
$$

Rješenje ovog problema možemo pronaći algoritmom Sinkhorn-Knopp uz iterativno ažuriranje vektora $\boldsymbol{u}$ i $\boldsymbol{v}$ koristeći pravila:

$$\begin{aligned} \boldsymbol{u}^{(t+1)} &= \frac{\mathbf{1}_N}{N \cdot \boldsymbol{P}^{-\lambda}\boldsymbol{v}^{(t)}} \\ \boldsymbol{v}^{(t+1)} &= \frac{\boldsymbol{\pi}}{(\boldsymbol{P}^{-\lambda})^T\boldsymbol{u}^{(t)}} \end{aligned} \tag{7.15}$$

Konačnu matricu vjerojatnosti $\boldsymbol{Q}$ tada dobivamo kao:

$$\boldsymbol{Q} = \text{diag}(\boldsymbol{u})\, \boldsymbol{P}^{-\lambda}\, \text{diag}(\boldsymbol{v}) \tag{7.16}$$

Važno je napomenuti da algoritmom Sinkhorn-Knopp dobivamo matricu vjerojatnosti $\boldsymbol{Q}$ koja odgovara skupu podataka $\mathcal{D}$, a ne rješenje u zatvorenoj formi za distribuciju $q(l|\boldsymbol{x}, y)$. Ipak, ovo nam ne predstavlja problem jer je matrica $\boldsymbol{Q}$ dovoljna za provođenje M koraka.

7.2.2. M korak

Cilj M koraka algoritma maksimizacije očekivanja je pronaći parametre $\boldsymbol{\Omega}$ koji maksimiziraju donju varijacijsku granicu $\mathcal{L}_{VD}(\boldsymbol{\Omega}, q|\mathcal{D})$ uz fiksiranu distribuciju $q(l|\boldsymbol{x}, y)$ odnosno matricu vjerojatnosti $\boldsymbol{Q}$. Maksimizacija donje varijacijske granice ekvivalentna je minimizaciji cilja:

$$\begin{aligned} -\mathcal{L}_{VD}(\boldsymbol{\Omega}|\mathcal{D}) &= -\sum_{i=1}^{N} \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i) + \log p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l^i|\boldsymbol{x}^i) - \log q(l^i|\boldsymbol{x}^i, y^i)\right] \\ &= \sum_{i=1}^{N}\sum_{l=1}^{K} q(l|\boldsymbol{x}^i, y^i)\left[-\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l) - \log p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i) + \log q(l|\boldsymbol{x}^i, y^i)\right] \\ &= \sum_{i=1}^{N}\left[\sum_{l=1}^{K} -q(l|\boldsymbol{x}^i, y^i)\log p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i) - \sum_{l=1}^{K} q(l|\boldsymbol{x}^i, y^i)\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l) - \text{H}(q)\right] \\ &= \sum_{i=1}^{N}\left[\mathcal{L}_{CE}(p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i), q(l|\boldsymbol{x}^i, y^i)) - \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)}\left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i)\right] - \text{H}(q)\right] \end{aligned} \tag{7.17}$$

Pritom $\mathcal{L}_{CE}(p_{\theta,\phi,\pi}(l|\boldsymbol{x}^i), q(l|\boldsymbol{x}^i, y^i))$ označava funkciju gubitka unakrsne entropije između izlaza modela $p_{\theta,\phi,\pi}(l|\boldsymbol{x}^i)$ i cilja $q(l|\boldsymbol{x}^i, y^i)$. Član H($q$) ne ovisi o parametrima $\boldsymbol{\Omega}$, tako da ga možemo izostaviti te dobivamo konačan cilj koji želimo minimizirati:

$$\mathcal{L}_{VIBE}(\boldsymbol{\Omega}|\mathcal{D}) = \sum_{i=1}^{N} \left[\mathcal{L}_{CE}(p_{\theta,\phi,\pi}(l|\boldsymbol{x}^i), q(l|\boldsymbol{x}^i, y^i)) - \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\log p_{\phi,\psi}(y^i|l^i) \right] \right] \tag{7.18}$$

Izraženi gubitak možemo minimizirati koristeći gradijentni spust, a sastoji se od dvije komponente. Prva komponenta pritom odgovara standardnom nadziranom gubitku između izlaza modela $p_{\theta,\phi,\pi}(l|\boldsymbol{x}^i)$ i pseudooznaka dobivenih na temelju distribucije $q(l|\boldsymbol{x}^i, y^i)$ procijenjene u E koraku. S druge strane, druga komponenta gubitka odgovara rekonstrukciji zatrovanih oznaka $\boldsymbol{y}$ na temelju procijenjenih pseudooznaka $\boldsymbol{l}$.

7.3. Konzistencijski gubitak

Osnovna ideja konzistencijskog gubitka (engl. *consistency loss*) [24] je osigurati da model za slične ulaze daje slične izlaze. Dodavanjem ove komponente, modelu možemo povećati invarijantnost na perturbacije ulaza, kao i potencijalno spriječiti prenaučenost. Drugim riječima, konzistencijski gubitak može služiti kao regularizacijska tehnika.

Kako bismo definirali sami gubitak, prvo je potrebno uvesti pojam slabih i jakih augmentacija. Kod uobičajenog učenja dubokih modela, većinom imamo definiran jedan skup augmentacija za učenje koji primjenjujemo na ulaze prije nego što ih damo modelu. S druge strane, kod implementacije konzistencijskog gubitka definiramo dva pogleda (engl. *view*) na ulazne podatke koristeći zasebne skupove augmentacija. Pritom skup slabih augmentacija u manjoj mjeri perturbira ulaz, a skup jakih augmentacija u većoj mjeri mijenja isti taj ulaz.

Skup jakih augmentacija lako možemo dobiti dodavanjem augmentacije AutoAugment [60] u postojeći skup slabih augmentacija. Augmentacija AutoAugment zapravo je strategija primjena niza pojedinih augmentacija specijalizirana za neki skup podataka (na primjer, za skup CIFAR-10), a određena podržanim učenjem (engl. *reinforcement learning*) [61].

Nakon što smo odredili oba pogleda ulaznih podataka, iste dajemo kao ulaz modelu. Konzistencijski gubitak tada možemo definirati kao udaljenost između izlaza modela za slabo odnosno jako augmentiran ulaz. Pritom uobičajeno koristimo mjeru poput unakrsne entropije ili KL divergencije:

$$\mathcal{L}_{con} = KL(p(y|\mathcal{A}_W(\boldsymbol{x}))||p(y|\mathcal{A}_S(\boldsymbol{x}))) \tag{7.19}$$

Pri čemu $p(y|\mathcal{A}_W(\boldsymbol{x}))$ označava izlaz modela za slabu augmentaciju ulaza $\mathcal{A}_W(\boldsymbol{x})$, a $p(y|\mathcal{A}_S(\boldsymbol{x}))$ označava izlaz modela za jaku augmentaciju $\mathcal{A}_S(\boldsymbol{x})$. Važno je napomenuti da kod konzistencijskog gubitka uobičajeno kao cilj tj. učitelja (engl. *teacher*) koristimo slabu augmentaciju, a kao učenika (engl. *student*) koristimo jaku augmentaciju. Sveukupni gubitak za zadani primjer $\boldsymbol{x}$ tada definiramo kao:

$$\mathcal{L}_{total}(\boldsymbol{x}) = \mathcal{L}_{supervised}(\mathcal{A}_S(\boldsymbol{x})) + \sigma\mathcal{L}_{con}(\boldsymbol{x}) \tag{7.20}$$

Pritom hiperparametar σ zovemo stopom konzistencije (engl. *consistency rate*). Dodatno, uočimo da se nadzirani gubitak sada računa koristeći jaku augmentaciju ulaza $\mathcal{A}_S(\boldsymbol{x})$. Dok algoritmi SOP+ te ILL među ostalom koriste i konzistencijski gubitak, osnovna inačica okvira VIBE ga ne koristi. U našem radu istražujemo učinak dodavanja konzistencijskog gubitka u formulaciju gubitka okvira VIBE.

Konkretno, glavna izmjena okvira VIBE je u M koraku. Dok E korak za zadani skup podataka $\mathcal{D}$ aproksimira distribuciju $q(l|\boldsymbol{x}, y)$, M korak na temelju procijenjene distribucije optimira parametre $\boldsymbol{\Omega}$ koristeći gradijentni spust. Pritom smo gubitak u M koraku definirali kao:

$$\mathcal{L}_{VIBE}(\boldsymbol{\Omega}|\mathcal{D}) = \sum_{i=1}^{N} \left[\mathcal{L}_{CE}(p_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\pi}}(l|\boldsymbol{x}^i), q(l|\boldsymbol{x}^i, y^i)) - \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)}\left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i)\right]\right] \tag{7.21}$$

Uz korištenje jake augmentacije ulaza za izračun standardnog gubitka te dodavanje komponente konzistencijskog gubitka, konačni gubitak koji minimiziramo u M koraku proširene inačice okvira VIBE možemo definirati kao:

$$\mathcal{L}_{VIBE+CL}(\boldsymbol{\Omega}|\mathcal{D}) = \sum_{i=1}^{N} \left[\mathcal{L}_{CE}(p_{\theta,\boldsymbol{\phi},\pi}(l|\mathcal{A}_S(\boldsymbol{x}^i)), q(l|\boldsymbol{x}^i, y^i)) - \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)} \left[\log p_{\boldsymbol{\phi},\boldsymbol{\psi}}(y^i|l^i) \right] \right] + \sigma \sum_{i=1}^{N} KL(p_{\theta,\boldsymbol{\phi},\pi}(l|\mathcal{A}_W(\boldsymbol{x}^i)) || p_{\theta,\boldsymbol{\phi},\pi}(l|\mathcal{A}_S(\boldsymbol{x}^i))) \tag{7.22}$$

7.4. Pretprocesiranje

Okvir VIBE algoritmom maksimizacije očekivanja implicitno prepravlja oznake zatrovanih primjera te ih potom koristi za učenje. Pritom učenje uspijeva jer većina napada nastoji što manje izmijeniti ulazne primjere. Ipak, napadi s čistim oznakama (engl. *clean-label attacks*) uobičajeno moraju više izmijeniti ulazne primjere kako bi uspjeli ugraditi stražnja vrata u model. Ova značajna perturbacija ulaza vidljiva je i u prostoru samonadziranih značajki: primjeri zatrovani napadom s čistim oznakama prilično su udaljeni od mnogostrukosti (engl. *manifold*) na kojoj leže čisti podatci. Kako bi iskoristili ovu činjenicu, okvir VIBE uključuje korak pretprocesiranja čiji je cilj ukloniti primjere zatrovane napadom s čistim oznakama.

Konkretno, korak pretprocesiranja promatra mnogostrukost podataka u prostoru samonadziranih značajki te zatrovane primjere identificira kao najudaljeniju zajednicu. Kako bi ovo postigli, prvo konstruiramo graf k najbližih susjeda predstavljen matricom susjedstva $\boldsymbol{A}_k$. Potom koristimo Leidenov algoritam [62] kako bismo podijelili graf najbližih susjeda $\boldsymbol{A}_k$ na $K+1$ zajednicu. Za svaku detektiranu zajednicu zatim računamo prosječnu udaljenost do preostalih K zajednica te izdvajamo zajednicu s najvećom prosječnom udaljenosti. Ako je prosječna udaljenost te zajednice veća od zadanog praga δ, iz skupa podataka $\mathcal{D}$ uklanjamo primjere koji joj pripadaju.

U slučaju da je skup podataka $\mathcal{D}$ zatrovan napadom s čistim oznakama, najudaljenija zajednica odgovarat će zatrovanim primjerima. S druge strane, ako skup $\mathcal{D}$ nije zatrovan, najudaljenija zajednica će obuhvaćati nekolicinu čistih primjera koji odstupaju od mnogostrukosti podataka. Ipak, prosječna udaljenost najudaljenije zajednice u ovom slučaju bit će manja od praga δ pa stoga ti primjeri neće biti odbačeni. Štoviše, čak i ako je prag δ loše odabran pa zbog toga odbacimo nekolicinu čistih primjera, naučeni model će podjednako dobro generalizirati [10].

8. Duboki konvolucijski modeli

Arhitektura LeNet-5 [63] predstavlja jedan od prvih dubokih konvolucijskih modela [64], a dizajnirana je s ciljem klasifikacije ručno pisanih znamenki iz skupa podataka MNIST [23]. Ipak, duboki konvolucijski modeli postali su popularni tek razvojem duboke arhitekture AlexNet [65] koja je 2012. godine pobijedila na natjecanju ImageNet ILSVRC [66] te time pokazala potencijal konvolucijskih mreža.

Pod duboki konvolucijski model općenito mislimo na model koji sadrži barem jedan konvolucijski sloj. Osim konvolucijskih slojeva (engl. *convolutional layer*), duboki modeli uobičajeno sadrže i slojeve sažimanja (engl. *pooling layer*) [67], aktivacijske funkcije poput funkcije zglobnice (engl. *rectified linear unit* - ReLU), slojeve normalizacije nad grupom (engl. *batch normalization layer*) [68] te potpuno-povezane slojeve (engl. *fully-connected layer*).

8.1. Konvolucijski sloj

Konvolucijski sloj dubokih modela zasniva se na operaciji konvolucije. Konvoluciju možemo definirati kao integral umnoška dviju funkcija, pri čemu je jedna reflektirana i posmaknuta:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{8.1}$$

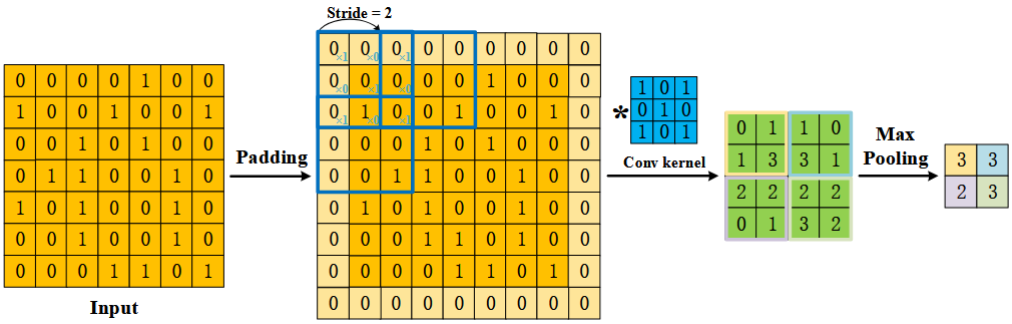
U dubokom učenju, pod konvolucijom najčešće mislimo na operaciju unakrsne korelacije (engl. *cross-correlation*) kod koje izostavljamo reflektiranje jedne od funkcija:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t + \tau) d\tau \tag{8.2}$$

Operacija konvolucije izvrsna je za primjenu na podatcima rešetkaste strukture. Kako bismo ju mogli primijeniti na slike, izlaz operacije 2-dimenzionalne konvolucije definiramo kao:

$$\boldsymbol{Y}_{i,j} = (\boldsymbol{X} * \boldsymbol{W})_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \boldsymbol{X}_{i+m,j+n} \boldsymbol{W}_{m,n} \tag{8.3}$$

Pritom $\boldsymbol{X}$ predstavlja ulaznu sliku, a $\boldsymbol{Y}$ rezultat primjene 2-dimenzionalne konvolucije s jezgrom $\boldsymbol{W}$ dimenzija $M \times N$. Uočimo da ovdje zapravo primjenjujemo operaciju unakrsne korelacije. Jezgra $\boldsymbol{W}$ predstavlja parametre koje možemo učiti gradijentnim spustom, a uobičajeno je kvadratnog oblika (na primjer, dimenzija 3×3).



Slika 8.1. Prikaz primjene konvolucije uz jezgru veličine 3×3, korak iznosa 2 te nadopunjavanje iznosa 1. Nakon operacije konvolucije, na izlaz se dodatno primjenjuje operacija sažimanja maksimumom uz jezgru veličine 2×2. Preuzeto iz [64].

Osim veličine jezgre K, neki od hiperparametara konvolucijskih slojeva su korak (engl. *stride*) s te nadopunjavanje (engl. *padding*) p. Pritom korak određuje za koliko se elemenata horizontalno odnosno vertikalno jezgra $\boldsymbol{W}$ pomiče dok klizi po ulazu $\boldsymbol{X}$. S druge strane, nadopunjavanjem možemo izbjeći smanjenje dimenzija izlaza naspram ulaza. Konkretno, primjenom jezgre veličine $K \times K$ uz korak iznosa s na sliku dimenzija $H_{in} \times W_{in}$ s nadopunjavanjem iznosa p dobivamo izlaz dimenzija:

$$
\begin{aligned}
H_{out} &= \lfloor \frac{H_{in} + 2p - K}{s} \rfloor + 1 \\
W_{out} &= \lfloor \frac{W_{in} + 2p - K}{s} \rfloor + 1
\end{aligned}
\tag{8.4}
$$

Ulaz i izlaz konvolucijskog sloja obično su dimenzija $C_{in} \times H_{in} \times W_{in}$ odnosno $C_{out} \times H_{out} \times W_{out}$. Pritom C_{in} i C_{out} označavaju broj kanala ulaza odnosno izlaza. Parametri zadanog sloja tada su dimenzija $C_{out} \times C_{in} \times K \times K$. Ovo možemo zamisliti kao C_{out} zasebnih jezgara (po jedna za svaki kanal izlaza) pri čemu je svaka jezgra dimenzija $C_{in} \times K \times K$. Jedan kanal izlaza tada dobivamo kao:

$$
\boldsymbol{Y}^{(d)} = \sum_{c=1}^{C_{in}} \boldsymbol{X}^{(c)} * \boldsymbol{W}^{(d,c)} \tag{8.5}
$$

Pritom $\boldsymbol{X}^{(c)}$ označava c-ti kanal ulaza $\boldsymbol{X}$, a $\boldsymbol{W}^{(d,c)}$ c-ti kanal d-te jezgre koja odgovara d-tom kanalu izlaza $\boldsymbol{Y}^{(d)}$.

Sloj sažimanja funkcionira slično konvolucijskom sloju. Kao i kod konvolucije, i ovdje klizimo po ulazu te dobivamo okna dimenzija $K \times K$. Pritom je korak najčešće jednak dimenzijama okna. Za razliku od konvolucijskog sloja, kod sloja sažimanja općenito ne postoji parametrizirana jezgra $\boldsymbol{W}$, već na svako okno primjenjujemo zadanu operaciju (na primjer, operaciju traženja maksimalne vrijednosti). Dodatno, sloj sažimanja se na svaki kanal ulaza primjenjuje zasebno tj. nema interakcije između različitih kanala.

8.2. Sloj normalizacije nad grupom

Osnovna ideja normalizacije nad grupom je da normalizacijom svake značajke zasebno možemo osigurati konzistentnu distribuciju ulaza u daljnji sloj modela. Idealno, normalizacija bi se provodila koristeći cijeli skup podataka $\mathcal{D}$, no ovo zbog memorijskih zahtjeva najčešće nije moguće. Umjesto toga, ideja je normalizaciju provoditi na razini minigrupe $\mathcal{B}$. Dodavanje slojeva normalizacije nad grupom u proizvoljni model može ubrzati konvergenciju, stabilizirati učenje te dopustiti učenje s višom stopom učenja (engl. *learning rate*) [68].

Neka je $\boldsymbol{x}_i$ i-ti vektor značajki iz minigrupe $\mathcal{B}$ koja sadrži N vektora. Kako bismo mogli normalizirati značajke, prvo moramo izračunati procjene vektora srednjih vrijednosti $\boldsymbol{\mu}_{\mathcal{B}}$ i varijanci $\boldsymbol{\sigma}^2_{\mathcal{B}}$ na temelju trenutne minigrupe:

$$\begin{aligned} \boldsymbol{\mu}_{\mathcal{B}} &= \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \\ \boldsymbol{\sigma}^2_{\mathcal{B}} &= \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu}_{\mathcal{B}})^2 \end{aligned} \tag{8.6}$$

Normaliziranu k-tu značajku $\hat{x}_i^{(k)}$ i-tog vektora značajki $\boldsymbol{x}_i$ definiramo jednadžbom:

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_{\mathcal{B}}^{(k)}}{\sqrt{(\sigma_{\mathcal{B}}^{(k)})^2 + \epsilon}} \tag{8.7}$$

Pritom je ϵ konstanta malog iznosa koju dodajemo zbog numeričke stabilnosti. Kako bismo povećali ekspresivnost, normalizirane značajke dodatno transformiramo:

$$y_i^{(k)} = \gamma^{(k)} \hat{x}_i^{(k)} + \beta^{(k)} \tag{8.8}$$

Pri čemu je $y_i^{(k)}$ k-ta značajka i-tog izlaza $\boldsymbol{y}_i$ iz sloja normalizacije nad grupom, a $\boldsymbol{\gamma}$ i $\boldsymbol{\beta}$ su vektori parametara linearne transformacije koje učimo. Kako bi sloj normalizacije nad grupom radio ispravno, potrebno je koristiti dovoljno velike minigrupe.

Opisani postupak koristi se tijekom učenja modela. Nakon što smo naučili i upogonili model, primjeri mu najčešće dolaze jedan po jedan (engl. *online*). Kako bi se nosili s ovom situacijom, sloj normalizacije nad grupom definira još jedan režim rada. Konkretno, kada model koristimo za evaluaciju, više ne računamo statistike $\boldsymbol{\mu}_{\mathcal{B}}$ i $\boldsymbol{\sigma}^2_{\mathcal{B}}$. Umjesto toga, za normalizaciju koristimo pomične prosjeke izračunatih statistika $\hat{\boldsymbol{\mu}}$ i $\hat{\boldsymbol{\sigma}}^2$. Alternativno, možemo koristiti i statistike izračunate za cijeli skup za učenje $\mathcal{D}$.

8.3. Rezidualni modeli

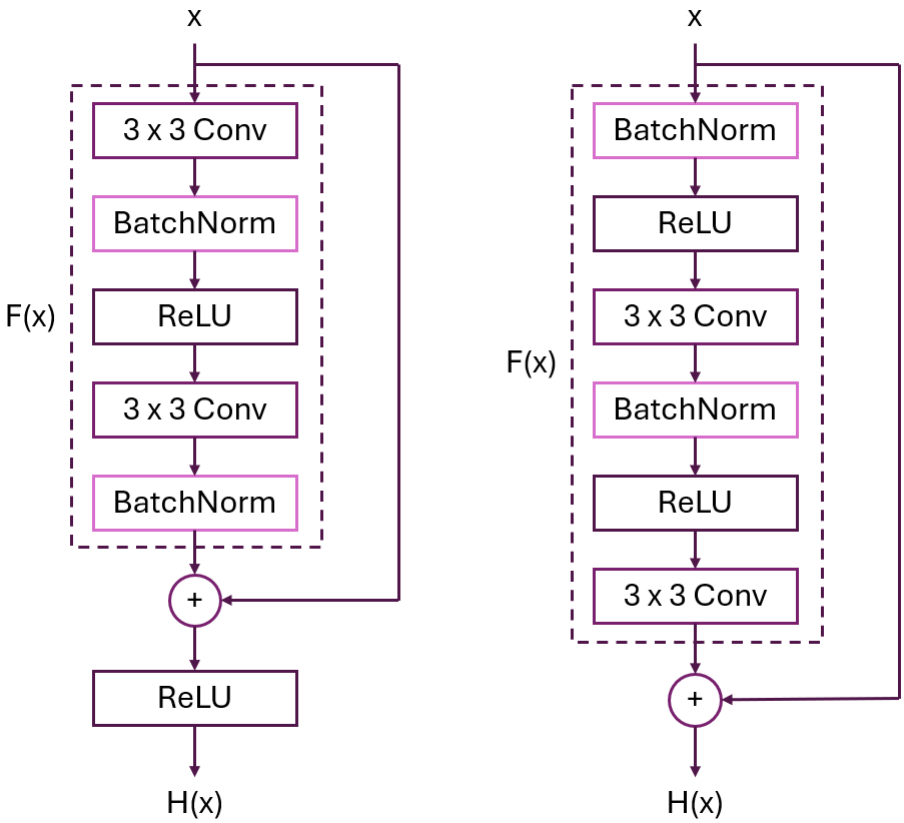
Prije uvođenja rezidualnih blokova [69], duboki konvolucijski modeli bili su ograničeni dubinom. Konkretno, korištenje više od otprilike 20 slojeva vodilo je do pogoršanja performansi modela [69]. Jedan od razloga ovome je činjenica da kod modela s velikim brojem slojeva gradijent teško propagira do početnih slojeva. Pritom može doći do problema nestajućih ili eksplodirajućih gradijenata (engl.*vanishing/exploding gradients*).

Neka je $\boldsymbol{x}$ ulaz u sloj l. Kod modela bez rezidualnih blokova, sloj l treba modelirati preslikavanje $H(\boldsymbol{x})$. Osnovna ideja uvođenja rezidualnih blokova je reformulacija danog problema. Umjesto direktnog modeliranja funkcije $H(\boldsymbol{x})$, cilj postaje modelirati razliku između očekivanog izlaza $H(\boldsymbol{x})$ i danog ulaza $\boldsymbol{x}$. Ovu razliku zovemo rezidual i označavamo s $F(\boldsymbol{x})$. Izlaz rezidualnog bloka tada postaje:

$$H(\boldsymbol{x}) = F(\boldsymbol{x}) + P(\boldsymbol{x}) \tag{8.9}$$

Pritom $P(\boldsymbol{x})$ označava projekciju ulaza $\boldsymbol{x}$ kako bi se isti mogao zbrojiti s rezidualom $F(\boldsymbol{x})$, a $H(\boldsymbol{x})$ označava izlaz rezidualnog bloka. Dakle, rezidualni blok sastoji se od reziduala $F(\boldsymbol{x})$ te preskočne veze $P(\boldsymbol{x})$ (engl. *skip connection*). Rezidual $F(\boldsymbol{x})$ uobičajeno je modeliran nizom od nekoliko konvolucijskih slojeva u kombinaciji s nelinearnim aktivacijama i slojevima normalizacije nad grupom.

Kada govorimo o rezidualnim blokovima, važno je istaknuti razliku između osnovne inačice i predaktivacijske (engl. *pre-act*) inačice [70]. Konkretno, dok osnovna inačica koristi slijed slojeva `Conv -> BatchNorm -> ReLU`, predaktivacijska inačica koristi slijed slojeva `BatchNorm -> ReLU -> Conv`. Osim ovoga, osnovna inačica na zbroj reziduala $F(\boldsymbol{x})$ i projekcije ulaza $P(\boldsymbol{x})$ dodatno primjenjuje nelinearnu aktivaciju. Predaktivacijska inačica uklanja konačnu primjenu nelinearne aktivacije kako bi se dodatno pospješio tok gradijenata [70].



Slika 8.2. Prikaz osnovne inačice (lijevo) i predaktivacijske inačice (desno) rezidualnog bloka. Rezidual $F(\boldsymbol{x})$ označen je iscrtkanom linijom.

Kod jednostavnih problema, već prvih nekoliko slojeva dubokog modela ima dovoljan kapacitet za modelirati potrebno rješenje. Ako naša arhitektura ne koristi rezidualne blokove, svi daljnji slojevi bi trebali modelirati funkciju identiteta. S druge strane, ako koristimo rezidualne blokove, težine svih daljnjih slojeva trebale bi konvergirati u vrijednost 0. Pokazuje se da je učenje "ugašenih" blokova značajno lakši problem od učenja funkcije identiteta [69]. Zbog ovoga, korištenje rezidualnih blokova nam omogućava dizajn arhitektura s velikim brojem slojeva.

Obitelj arhitektura ResNet [69] jedna je od najpoznatijih obitelji arhitektura baziranih na korištenju rezidualnih blokova. Pritom su najčešće korištene arhitekture s 18, 34, 50, 101 ili 152 sloja, ali postoje i još dublje inačice poput arhitekture ResNet-200. U okviru našeg rada, fokusiramo se na arhitekturu ResNet-18 koja se sastoji od 17 konvolucijskih slojeva i 1 potpuno-povezanog sloja.

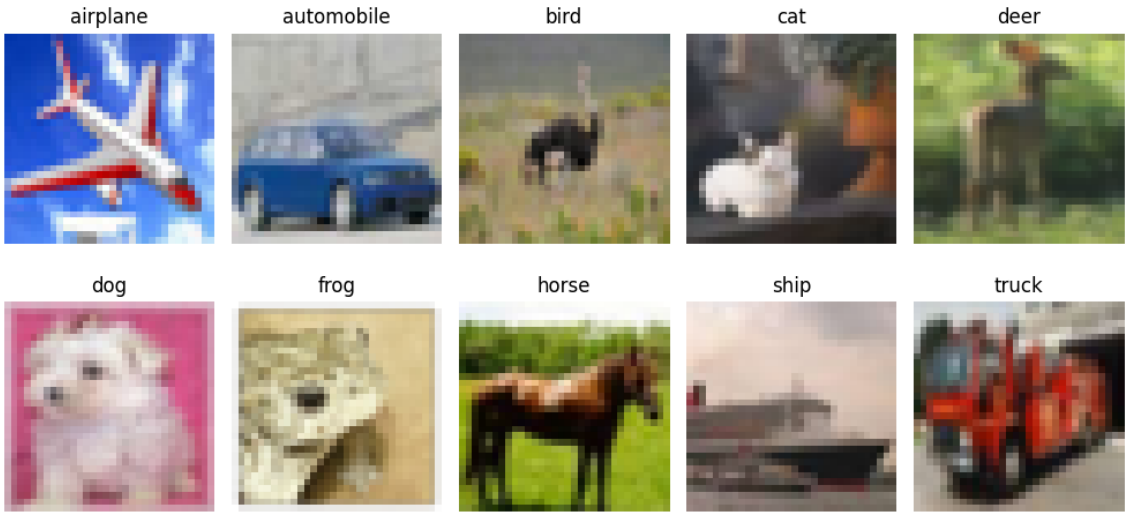
9. Skupovi podataka

U okviru našeg rada, koristili smo nekoliko zatrovanih odnosno zašumljenih inačica skupa podataka CIFAR-10 [22].

9.1. CIFAR-10

Skup podataka CIFAR-10 jedan je od najčešće korištenih skupova za učenje i evaluaciju različitih algoritama i modela dubokog učenja, a nastao je označavanjem podskupa skupa slika Tiny Images [22]. Podijeljen je na 50000 primjera u skupu za učenje i 10000 primjera u skupu za ispitivanje.

Svaka slika je u boji, a dimenzije svih slika su 32×32. Pojedine slike mogu pripadati jednom od 10 razreda. Pritom su razredi isključivi tj. ne postoje preklapanja. Skup za učenje sadrži po 5000 slika iz svakog razreda, a skup za ispitivanje sadrži po 1000 slika iz svakog razreda. Drugim riječima, oba podskupa su balansirana.



Slika 9.1. Prikaz po jedne slike za svaki razred iz skupa podataka CIFAR-10.

10. Eksperimenti

Eksperimente koje smo provodili u okviru našeg rada možemo podijeliti na dvije cjeline: eksperimenti sa zašumljenim oznakama te eksperimenti s napadima umetanjem stražnjih vrata.

10.1. Zašumljene oznake

Eksperimente sa zašumljenim oznakama provodili smo na zašumljenim inačicama skupa CIFAR-10. Pritom smo koristili inkluzivno simetrično zašumljivanje uz stopu šuma iznosa 20%, 50% te 80%, kao i asimetrično zašumljivanje uz stopu šuma iznosa 40%. Za svaku kombinaciju metode zašumljivanja i stope šuma, stvorili smo jednu inačicu zašumljenog skupa CIFAR-10. Dodatno, različite pristupe smo evaluirali i pri učenju na čistom skupu CIFAR-10. U svim eksperimentima, učili smo model arhitekture ResNet-18. Ključna mjera dobrote za eksperimente sa zašumljenim oznakama nam je točnost na čistom skupu za ispitivanje.

Općenito uspoređujemo pet pristupa: nadzirano učenje (engl. *cross-entropy* - CE), nadzirano učenje s konzistencijskim gubitkom (engl. *cross-entropy with consistency loss* - CE + CL), nadogradnju algoritma *Sparse over-parameterization* (SOP+), algoritam *Imprecise label learning* (ILL) te okvir *Variational inference for backdoor elimination* s dodanim konzistencijskim gubitkom (VIBE).

Kod nadziranog učenja odnosno nadziranog učenja s konzistencijskim gubitkom, nasumično inicijalizirani model smo učili 300 epoha uz optimizator SGD s početnom stopom učenja iznosa 0.05. Pritom smo koristili moment (engl. *momentum*) [71] iznosa 0.9, kao i propadanje težina (engl. *weight decay*) [72] iznosa $5 \cdot 10^{-4}$. Kako bi se stopa učenja kroz vrijeme smanjivala, koristili smo višekoračnu strategiju (engl. *multistep scheduler*)

uz hiperparametar γ iznosa 0.1 te smanjivanje stope učenja pri dostizanju 150. odnosno 180. epohe. Skup podataka bio je podijeljen na mini-grupe veličine 128. U slučaju nadziranog učenja s konzistencijskim gubitkom, koristili smo konstantnu stopu konzistencije iznosa 1. Jake augmentacije uključuju slabe augmentacije, augmentaciju AutoAugment te augmentaciju nasumičnog brisanja (engl. *random erasing*) [73]. Dodatno, kod nadziranog učenja s konzistencijskim gubitkom model učimo 300 epoha na čistom skupu odnosno 200 epoha na zašumljenim skupovima. Ovo je ujedno i jedini hiperparametar čiji iznos ne fiksiramo na jednu vrijednost za sve eksperimente.

U slučaju algoritma SOP+, hiperparametri odgovaraju onima iz originalnog rada [8], ali uz korištenje arhitekture ResNet-18 umjesto arhitekture predaktivacijskog ResNet-18. Nasumično inicijalizirani model učili smo 300 epoha uz optimizator SGD s početnom stopom učenja iznosa 0.02. Koristili smo moment iznosa 0.9 te propadanje težina iznosa $5 \cdot 10^{-4}$. Umjesto višekoračne strategije, ovdje smo koristili strategiju kosinusnog kaljenja (engl. *cosine annealing scheduler*) bez restarta te uz minimalnu stopu učenja iznosa $2 \cdot 10^{-4}$. Skup podataka ponovno je podijeljen na mini-grupe veličine 128. Vektore parametara $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}_{i=1}^N$ inicijalizirali smo nasumično iz normalne distribucije sa srednjom vrijednošću iznosa 0 te standardnom devijacijom iznosa $1 \cdot 10^{-8}$, a učili smo ih uz optimizator SGD sa stopom učenja iznosa 1 te faktorom skaliranja stope učenja iznosa 10. Pritom za optimizator vektora $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}_{i=1}^N$ nismo koristili moment ni propadanje težina. Koristili smo konstantnu stopu konzistencije iznosa 0.9, kao i konstantnu stopu ravnoteže razreda iznosa 0.1. Jake augmentacije kod algoritma SOP+ uključuju slabe augmentacije, augmentaciju AutoAugment te augmentaciju izrezivanja (engl. *cutout*) [74].

Kod algoritma ILL, hiperparametri ponovno odgovaraju onima iz originalnog rada [9], ali uz korištenje arhitekture ResNet-18 umjesto arhitekture predaktivacijskog ResNet-18. Kao i kod algoritma SOP+, nasumično inicijalizirani model smo učili 300 epoha uz optimizator SGD s početnom stopom učenja iznosa 0.02. Pritom smo koristili moment iznosa 0.9, ali i propadanje težina iznosa $1 \cdot 10^{-3}$ te propadanje slojeva (engl. *layer decay*) [75] iznosa 1. Za smanjivanje stope učenja, koristili smo strategiju kosinusnog kaljenja bez restarta te uz minimalnu stopu učenja iznosa $2 \cdot 10^{-4}$. Skup podataka ponovno je bio podijeljen na mini-grupe veličine 128. Koristili smo konstantnu stopu konzistencije iznosa 1. Jake augmentacije kod algoritma ILL uključuju slabe augmentacije, augmenta-

ciju AutoAugment te augmentaciju nasumičnog brisanja.

Konačno, kod okvira VIBE s dodanim konzistencijskim gubitkom, model inicijaliziran okvirom All4One smo učili 30000 iteracija (približno 77 epoha) uz optimizator SGD s početnom stopom učenja iznosa 0.01. Koristili smo moment iznosa 0.9, kao i propadanje težina iznosa $5 \cdot 10^{-4}$. Kao i kod algoritama SOP+ te ILL, koristili smo strategiju kosinusnog kaljenja bez restarta, ali uz minimalnu stopu učenja iznosa 0. Skup podataka bio je podijeljen na mini-grupe veličine 256. Hiperparametar raspršenosti von Mises-Fisherove distribucije κ postavili smo na iznos 10, jednako kao i hiperparametar temperature ν korišten pri izračunu distribucije $p(y|l)$. Pojedine vektore iz skupa čistih prototipova $\boldsymbol{\phi}$ odnosno skupa zašumljenih prototipova $\boldsymbol{\psi}$ inicijalizirali smo kao centroide pripadnih razreda. Parametre $\hat{\boldsymbol{\pi}}$ inicijalizirali smo nasumično iz jedinične normalne distribucije. Hiperparametar temperature c korišten za izračun koeficijenata miješanja $\boldsymbol{\pi}$ postavili smo na iznos 0.02. E korak učenja provodili smo svakih 1000 iteracija, pritom koristeći Sinkhorn-Knopp algoritam uz regularizacijski hiperparametar λ iznosa 25 te maksimalan broj iteracija iznosa 10000. Koristili smo konstantnu stopu konzistencije iznosa 5. Kao i kod nadziranog učenja s konzistencijskim gubitkom te algoritma ILL, jake augmentacije uključuju slabe augmentacije, augmentaciju AutoAugment te augmentaciju nasumičnog brisanja.

Ako za pojedini eksperiment nisu eksplicitno navedene vrijednosti određenih hiperparametara, u pitanju su upravo opisane vrijednosti. Na primjer, ako za određeni eksperiment učenja okvirom VIBE ne istaknemo korištenu veličinu mini-grupe, podrazumijevamo već navedenu veličinu 256.

10.1.1. Usporedba sa stanjem tehnike

Prije svega, pogledajmo kako se različiti opisani algoritmi ponašaju pri učenju na čistim podatcima. U tablici 10.1., stupac *Algoritam* predstavlja algoritam korišten za učenje modela na čistom skupu CIFAR-10. Stupac *Točnost [%]* predstavlja točnost naučenog modela na čistom skupu za ispitivanje skupa CIFAR-10. Najbolji rezultat je podebljan.

Tablica 10.1. Točnost modela učenih na čistom skupu CIFAR-10.

Algoritam	Točnost [%]
CE	95.30
CE + CL	96.30
SOP+	96.70
ILL	**96.75**
VIBE	96.10

Kao što možemo vidjeti, algoritam ILL postiže najveću točnost pri učenju na čistom skupu CIFAR-10. Algoritam SOP+ postiže podjednake performanse kao i algoritam ILL, dok okvir VIBE postiže za 0.65% manju točnost. Usporedbom redaka CE i CE + CL, vidimo da dodavanje konzistencijskog gubitka povećava točnost nadziranog učenja za 1%. Zanimljivo je da algoritmi ILL te SOP+ postižu bolje performanse čak i od nadziranog učenja s konzistencijskim gubitkom. S druge strane, okvir VIBE postiže neznatno nižu točnost u usporedbi s nadziranim učenjem s konzistencijskim gubitkom.

Pogledajmo sada točnost različitih modela pri učenju na podatcima sa simetrično zašumljenim oznakama. U tablici 10.2., stupac *Algoritam* predstavlja algoritam korišten za učenje modela na skupu CIFAR-10 sa simetrično zašumljenim oznakama. Znak $*$ prije imena algoritma označava rezultat preuzet iz originalnog rada pripadnog algoritma, dok manjak znaka označava reprodukciju rezultata na temelju originalnog repozitorija algoritma. Stupci *20%*, *50%* i *80%* predstavljaju različite stope šuma. Drugim riječima, svaki stupac odgovara jednoj zašumljenoj inačici skupa CIFAR-10 uz zadanu stopu šuma. Pojedine vrijednosti u tablici predstavljaju točnost odgovarajućeg modela na čistom skupu za ispitivanje skupa CIFAR-10. Najbolji rezultat za svaku zašumljenu inačicu skupa je podebljan. Vrijednosti preuzete iz originalnih radova nisu uzete u obzir pri označavanju najboljih rezultata.

Tablica 10.2. Točnost modela učenih na simetrično zašumljenim inačicama skupa CIFAR-10.

Algoritam	20%	50%	80%
CE	83.80	-	-
CE + CL	95.40	-	-
SOP+	**96.26**	95.58	93.11
*SOP+	96.30	95.50	94.00
ILL	96.22	**95.67**	93.75
*ILL	96.78	96.60	94.31
VIBE	95.70	94.80	**94.40**

Vidimo da za različite stope šuma različiti algoritmi postižu najveću točnost. Konkretno, uz stopu šuma iznosa 20%, algoritam SOP+ postiže najveću točnost. Algoritam ILL podjednake je točnosti, dok okvir VIBE postiže za 0.56% manju točnost. Iako se nadzirano učenje pokazuje kao najlošije u ovom postavu, zanimljivo je da dodavanjem konzistencijskog gubitka možemo postići točnost za samo 0.86% manju od točnosti stanja tehnike. Drugim riječima, iako standardno nadzirano učenje ne postiže visoku točnost pri učenju na skupu sa zašumljenim oznakama, jednostavnim dodavanjem konzistencijskog gubitka možemo drastično povećati točnost naučenog modela.

Uz stopu šuma iznosa 40%, algoritam ILL postiže najveću točnost. Algoritam SOP+ u ovom slučaju je podjednake točnosti, dok okvir VIBE postiže za 0.87% nižu točnost. Ovaj postav predstavlja najgori slučaj za okvir VIBE u usporedbi sa stanjem tehnike. Kada govorimo o postavu uz stopu šuma iznosa 80%, okvir VIBE postiže najveću točnost. Pritom isti postiže točnost za 1.29% veću od točnosti algoritma SOP+, kao i za 0.65% veću od točnosti algoritma ILL. Zanimljivo je da rezultati stanja tehnike za ovaj postav odudaraju od rezultata iz originalnih radova za 0.89% u slučaju algoritma SOP+, odnosno za 0.56% u slučaju algoritma ILL.

Uočimo da povećanjem stope šuma s 20% na 80% kod okvira VIBE gubimo samo 1.3% točnosti, dok kod algoritama SOP+ te ILL gubimo 3.15% odnosno 2.47% točnosti. Robusnost okvira VIBE na povećanje stope šuma, kao i postizanje najboljih performansi u usporedbi sa stanjem tehnike pri visokoj stopi šuma, potencijalno su posljedica samonadzirane inicijalizacije parametara modela korištenjem okvira All4One. Konkretno, samonadzirana inicijalizacija parametara mogla bi povećati važnost semantike slika u usporedbi s važnošću pridijeljenih oznaka te time doprinijeti učenju robusnog modela.

Pogledajmo još i točnost različitih modela pri učenju na podatcima s asimetrično zašumljenim oznakama. U tablici 10.3., stupac *Algoritam* predstavlja algoritam korišten za učenje modela na skupu CIFAR-10 s asimetrično zašumljenim oznakama uz stopu šuma iznosa 40%. Stupac *Točnost [%]* predstavlja točnost naučenog modela na čistom skupu za ispitivanje skupa CIFAR-10. Najbolji rezultat je podebljan.

Tablica 10.3. Točnost modela učenih na asimetrično zašumljenom skupu CIFAR-10.

Algoritam	Točnost [%]
CE	76.80
CE + CL	92.20
SOP+	93.74
*SOP+	93.80
ILL	85.75
*ILL	94.75
VIBE	**95.10**

Vidimo da u ovom slučaju okvir VIBE postiže najveću točnost. Konkretno, točnost okvira VIBE je za 1.36% veća od točnosti algoritma SOP+, kao i za 9.35% veća od točnosti algoritma ILL. Točnost algoritma ILL ovdje drastično odstupa od točnosti navedene u originalnom radu. Pokazuje se da je algoritam ILL nestabilan u teškim postavima - različita pokretanja učenja mogu rezultirati modelima s veoma različitim performansama. Dodatno, vidimo da i u ovom slučaju dodavanje konzistencijskog gubitka nadziranom učenju značajno povećava točnost. Štoviše, dodavanjem konzistencijskog gubitka ovdje postižemo povećanje točnosti iznosa čak 15.4%. Dobiveni rezultati sugeriraju da bismo problem zašumljenih oznaka potencijalno mogli riješiti fino podešenim nadziranim učenjem s konzistencijskim gubitkom.

Dok algoritmi SOP+ i ILL postižu najbolje performanse u postavima sa simetričnim zašumljivanjem uz nisku stopu šuma, okvir VIBE postiže najbolje performanse u teškim postavima tj. pri simetričnom zašumljivanju uz stopu šuma iznosa 80%, kao i pri asimetričnom zašumljivanju uz stopu šuma iznosa 40%. Dodatno, dodavanjem konzistencijskog gubitka nadziranom učenju možemo postići rezultate veoma bliske rezultatima algoritama za učenje na podatcima sa zašumljenim oznakama.

10.2. Napadi umetanjem stražnjih vrata

10.2.1. Usporedba sa stanjem tehnike

11. Zaključak

Literatura

[1] C. Zhang, S. Bengio, M. Hardt, B. Recht, i O. Vinyals, "Understanding deep learning requires rethinking generalization", *arXiv preprint arXiv:1611.03530*, 2016.

[2] A. Voulodimos, N. Doulamis, A. Doulamis, i E. Protopapadakis, "Deep learning for computer vision: A brief review", *Computational intelligence and neuroscience*, sv. 2018, br. 1, str. 7068349, 2018.

[3] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, i H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review", *arXiv preprint arXiv:2007.10760*, 2020.

[4] S. Gupta i A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review", *Procedia Computer Science*, sv. 161, str. 466–474, 2019.

[5] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, i X. Ma, "Anti-backdoor learning: Training clean models on poisoned data", *Advances in Neural Information Processing Systems*, sv. 34, str. 14 900–14 912, 2021.

[6] K. Huang, Y. Li, B. Wu, Z. Qin, i K. Ren, "Backdoor defense via decoupling the training process", *arXiv preprint arXiv:2202.03423*, 2022.

[7] K. Gao, Y. Bai, J. Gu, Y. Yang, i S.-T. Xia, "Backdoor defense via adaptively splitting poisoned dataset", u *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023., str. 4005–4014.

[8] S. Liu, Z. Zhu, Q. Qu, i C. You, "Robust training under label noise by over-parameterization", u *International Conference on Machine Learning*. PMLR, 2022., str. 14 153–14 172.

[9] H. Chen, A. Shah, J. Wang, R. Tao, Y. Wang, X. Li, X. Xie, M. Sugiyama, R. Singh, i B. Raj, "Imprecise label learning: A unified framework for learning with various imprecise label configurations", *Advances in Neural Information Processing Systems*, sv. 37, str. 59 621–59 654, 2024.

[10] I. Sabolić, M. Grcić, i S. Šegvić, "Seal your backdoor with variational defense", *arXiv preprint arXiv:2503.08829*, 2025.

[11] T. Gu, K. Liu, B. Dolan-Gavitt, i S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks", *IEEE Access*, sv. 7, str. 47 230–47 244, 2019.

[12] Y. Chen, X. Gong, Q. Wang, X. Di, i H. Huang, "Backdoor attacks and defenses for deep neural networks in outsourced cloud environments", *IEEE Network*, sv. 34, br. 5, str. 141–147, 2020.

[13] A. Nguyen i A. Tran, "Wanet–imperceptible warping-based backdoor attack", *arXiv preprint arXiv:2102.10369*, 2021.

[14] Y. Li, Y. Li, B. Wu, L. Li, R. He, i S. Lyu, "Invisible backdoor attack with sample-specific triggers", u *Proceedings of the IEEE/CVF international conference on computer vision*, 2021., str. 16 463–16 472.

[15] K. D. Doan, Y. Lao, i P. Li, "Marksman backdoor: Backdoor attacks with arbitrary target class", *Advances in Neural Information Processing Systems*, sv. 35, str. 38 260–38 273, 2022.

[16] M. Barni, K. Kallas, i B. Tondi, "A new backdoor attack in cnns by training set corruption without label poisoning", u *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019., str. 101–105.

[17] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, i F. Makedon, "A survey on contrastive self-supervised learning", *Technologies*, sv. 9, br. 1, str. 2, 2020.

[18] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, i J. Bailey, "Symmetric cross entropy for robust learning with noisy labels", u *Proceedings of the IEEE/CVF international conference on computer vision*, 2019., str. 322–330.

[19] J. E. Van Engelen i H. H. Hoos, "A survey on semi-supervised learning", *Machine learning*, sv. 109, br. 2, str. 373–440, 2020.

[20] R. Vilalta i Y. Drissi, "A perspective view and survey of meta-learning", *Artificial intelligence review*, sv. 18, str. 77–95, 2002.

[21] F. R. Cordeiro i G. Carneiro, "A survey on deep learning with noisy labels: How to train your model when you cannot trust on the annotations?" u *2020 33rd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. IEEE, 2020., str. 9–16.

[22] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images", 2009.

[23] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]", *IEEE signal processing magazine*, sv. 29, br. 6, str. 141–142, 2012.

[24] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, i C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning", *Advances in neural information processing systems*, sv. 32, 2019.

[25] D. Tanaka, D. Ikami, T. Yamasaki, i K. Aizawa, "Joint optimization framework for learning with noisy labels", u *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018., str. 5552–5560.

[26] Y. Tian, X. Yu, i S. Fu, "Partial label learning: Taxonomy, analysis and outlook", *Neural Networks*, sv. 161, str. 708–734, 2023.

[27] T. K. Moon, "The expectation-maximization algorithm", *IEEE Signal processing magazine*, sv. 13, br. 6, str. 47–60, 1996.

[28] V. Hondru, F. A. Croitoru, S. Minaee, R. T. Ionescu, i N. Sebe, "Masked image modeling: A survey", *arXiv preprint arXiv:2408.06687*, 2024.

[29] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks", *AIChE journal*, sv. 37, br. 2, str. 233–243, 1991.

[30] E. Chávez, G. Navarro, R. Baeza-Yates, i J. L. Marroquín, "Searching in metric spaces", *ACM computing surveys (CSUR)*, sv. 33, br. 3, str. 273–321, 2001.

[31] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, i D. Krishnan, "Supervised contrastive learning", *Advances in neural information processing systems*, sv. 33, str. 18 661–18 673, 2020.

[32] F. Schroff, D. Kalenichenko, i J. Philbin, "Facenet: A unified embedding for face recognition and clustering", u *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015., str. 815–823.

[33] A. v. d. Oord, Y. Li, i O. Vinyals, "Representation learning with contrastive predictive coding", *arXiv preprint arXiv:1807.03748*, 2018.

[34] I. G. Estepa, I. Sarasúa, B. Nagarajan, i P. Radeva, "All4one: Symbiotic neighbour contrastive learning via self-attention and redundancy reduction", u *Proceedings of the IEEE/CVF international conference on computer vision*, 2023., str. 16 243–16 253.

[35] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, i A. Zisserman, "With a little help from my friends: Nearest-neighbor contrastive learning of visual representations", u *Proceedings of the IEEE/CVF international conference on computer vision*, 2021., str. 9588–9597.

[36] J. Zbontar, L. Jing, I. Misra, Y. LeCun, i S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction", u *International conference on machine learning*. PMLR, 2021., str. 12 310–12 320.

[37] S. A. Koohpayegani, A. Tejankar, i H. Pirsiavash, "Mean shift for self-supervised learning", u *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021., str. 10 326–10 335.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, i I. Polosukhin, "Attention is all you need", *Advances in neural information processing systems*, sv. 30, 2017.

[39] I. J. Myung, "Tutorial on maximum likelihood estimation", *Journal of mathematical Psychology*, sv. 47, br. 1, str. 90–100, 2003.

[40] S.-i. Amari, "Backpropagation and stochastic gradient descent method", *Neurocomputing*, sv. 5, br. 4-5, str. 185–196, 1993.

[41] C. J. Wu, "On the convergence properties of the em algorithm", *The Annals of statistics*, str. 95–103, 1983.

[42] F. Pérez-Cruz, "Kullback-leibler divergence estimation of continuous distributions", u *2008 IEEE international symposium on information theory*. IEEE, 2008., str. 1666–1670.

[43] R. M. Neal i G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants", u *Learning in graphical models*. Springer, 1998., str. 355–368.

[44] M. N. Bernstein, "Expectation-maximization (em) and coordinate ascent", https://mbernste.github.io/posts/em/, 2021., accessed: 2025-06-02.

[45] A. Rényi, "On measures of entropy and information", u *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, sv. 4. University of California Press, 1961., str. 547–562.

[46] G. Peyré, M. Cuturi *et al.*, "Computational optimal transport: With applications to data science", *Foundations and Trends® in Machine Learning*, sv. 11, br. 5-6, str. 355–607, 2019.

[47] N. Papadakis, "Optimal transport for image processing", doktorska disertacija, Université de Bordeaux; Habilitation thesis, 2015.

[48] V. I. Bogachev i A. V. Kolesnikov, "The monge-kantorovich problem: achievements, connections, and perspectives", *Russian Mathematical Surveys*, sv. 67, br. 5, str. 785, 2012.

[49] R. M. Karp, U. V. Vazirani, i V. V. Vazirani, "An optimal algorithm for on-line bipartite matching", u *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, 1990., str. 352–358.

[50] G. M. Ziegler, "Lectures on polytopes." 1993.

[51] G. B. Dantzig, “Linear programming”, *Operations research*, sv. 50, br. 1, str. 42–47, 2002.

[52] A. Genevay, “Entropy-regularized optimal transport for machine learning”, doktorska disertacija, Université Paris sciences et lettres, 2019.

[53] P. A. Knight, “The sinkhorn–knopp algorithm: convergence and applications”, *SIAM Journal on Matrix Analysis and Applications*, sv. 30, br. 1, str. 261–275, 2008.

[54] R. Sinkhorn, “A relationship between arbitrary positive matrices and doubly stochastic matrices”, *The annals of mathematical statistics*, sv. 35, br. 2, str. 876–879, 1964.

[55] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport”, *Advances in neural information processing systems*, sv. 26, 2013.

[56] M. Slater, “Lagrange multipliers revisited”, u *Traces and emergence of nonlinear programming*. Springer, 2013., str. 293–306.

[57] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, i G. Ridgeway, “Clustering on the unit hypersphere using von mises-fisher distributions.” *Journal of Machine Learning Research*, sv. 6, br. 9, 2005.

[58] G. J. McLachlan i T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 2008.

[59] E. J. McShane, “Jensen’s inequality”, 1937.

[60] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, i Q. V. Le, “Autoaugment: Learning augmentation strategies from data”, u *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019., str. 113–123.

[61] L. P. Kaelbling, M. L. Littman, i A. W. Moore, “Reinforcement learning: A survey”, *Journal of artificial intelligence research*, sv. 4, str. 237–285, 1996.

[62] V. A. Traag, L. Waltman, i N. J. Van Eck, “From louvain to leiden: guaranteeing well-connected communities”, *Scientific reports*, sv. 9, br. 1, str. 1–12, 2019.

[63] Y. LeCun *et al.*, "Generalization and network design strategies", *Connectionism in perspective*, sv. 19, br. 143-155, str. 18, 1989.

[64] Z. Li, F. Liu, W. Yang, S. Peng, i J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects", *IEEE transactions on neural networks and learning systems*, sv. 33, br. 12, str. 6999–7019, 2021.

[65] A. Krizhevsky, I. Sutskever, i G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, sv. 25, 2012.

[66] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge", *International journal of computer vision*, sv. 115, str. 211–252, 2015.

[67] Y.-L. Boureau, J. Ponce, i Y. LeCun, "A theoretical analysis of feature pooling in visual recognition", u *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010., str. 111–118.

[68] S. Ioffe i C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", u *International conference on machine learning*. pmlr, 2015., str. 448–456.

[69] K. He, X. Zhang, S. Ren, i J. Sun, "Deep residual learning for image recognition", u *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016., str. 770–778.

[70] ——, "Identity mappings in deep residual networks", u *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016., str. 630–645.

[71] I. Sutskever, J. Martens, G. Dahl, i G. Hinton, "On the importance of initialization and momentum in deep learning", u *International conference on machine learning*. PMLR, 2013., str. 1139–1147.

[72] A. Krogh i J. Hertz, "A simple weight decay can improve generalization", *Advances in neural information processing systems*, sv. 4, 1991.

[73] Z. Zhong, L. Zheng, G. Kang, S. Li, i Y. Yang, "Random erasing data augmentation", u *Proceedings of the AAAI conference on artificial intelligence*, sv. 34, br. 07, 2020., str. 13 001–13 008.

[74] T. DeVries i G. W. Taylor, "Improved regularization of convolutional neural networks with cutout", *arXiv preprint arXiv:1708.04552*, 2017.

[75] M. Ishii i A. Sato, "Layer-wise weight decay for deep neural networks", u *Pacific-Rim Symposium on Image and Video Technology*. Springer, 2017., str. 276–289.

Sažetak

Varijacijsko učenje na zašumljenim oznakama

Dominik Jambrović

Sažetak...

Ključne riječi: prva ključna riječ; druga ključna riječ; treća ključna riječ