Zahvaljujem se svom mentoru prof. dr. sc. Siniši Šegviću na strpljenju i pomoći iskazanoj tijekom studiranja, pogotovo tijekom izrade diplomskog rada.

SADRŽAJ

1. Uvod

Konvolucijske neuronske mreže nastaju kao koncept u Fukushima (1980), ali nisu značajno korištene sve do 2010. godine. Te godine nastaje ImageNet koji je i danas jedan od najvećih i najpoznatijih skupova podataka za računalni vid. Od tada konvolucijske mreže dominiraju na natjecanjima detekcije objekata, klasifikacije slika, semantičke segmentacije i raznim drugim. Svoj uspjeh temelje na pronalasku dobrih generičnih značajki na slikama. Kako bi to postigli, potreban je velik broj anotiranih slika za svaku od promatranih klasa.

U ovom radu objasnit ćemo kako pristupiti rješavanju problema učenja s malim brojem anotiranih primjera u zadacima računalnog vida. Podaci, pri učenju s malim brojem primjera, su podijeljeni u epizode. Epizoda se sastoji od potpornog skupa i skupa upita. Potporni skup sadrži jako mali broj anotiranih primjera za svaku klasu, dok skup upita sadrži primjere nad kojim radimo predikciju. U sklopu rada implementirali smo konvolucijsku neuronsku mrežu za semantičku segmentaciju koja se temelji na PANet arhitekturi iz Wang et al. (2019). PANet arhitektura sastoji se od dva dijela: izvlačenje prototipa klasa iz potpornog skupa i određivanje pripadnosti korespondencijskom metrikom. Izvlačenje prototipa klasa zaduženo je za generiranje vektora značajki koji bi trebali što bolje opisivati klasu. Korespondencijskom metrikom svaki piksel slike upita uspoređuje se s prototipovima dobivenim iz prethodnog dijela mreže. Prethodni radovi na temu semantičke segmentacije s malim brojem primjera nisu iskoristili svo znanje iz potpornog skupa (Wang et al., 2019) pa autori rada uvode regularizaciju usklađivanjem prototipa. Cilj navedene regularizacije je postići što veću sličnost između prototipova izračunatih sa slika potpornog skupa i prototipova koje računamo iz slike upita i predviđene segmentacijske maske.

U odjeljku 2 objašnjavamo osnovne koncepte potrebne za shvaćanje konvolucijskih neuronskih mreža. Također, predstavljamo pojam učenja prijenosom znanja i kako se on može iskoristiti u treniranju neuronskih mreža. Na kraju je opisana ResNet arhitektura iz He et al. (2015) koju ćemo koristiti u našoj implementaciji. U odjeljku 3 predstavljamo koncept učenja s malim brojem primjera. Navodimo nekoliko pristupa

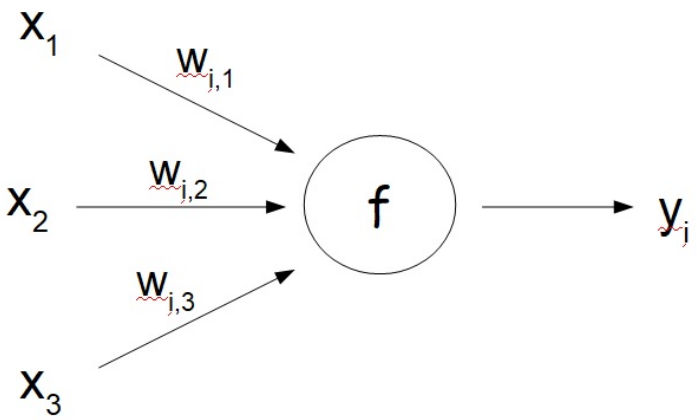
za zadatak klasifikacije s malim brojem primjera i objašnjavamo arhitekturu sijamske neuronske mreže. Slijedi upoznavanje sa semantičkom segmentacijom i detaljno objašnjenje PANet arhitekture. U odjeljku 4 objašnjavamo skup podataka PASCAL VOC na kojem smo provodili eksperimente i PyTorch kao alat korišten za implementaciju mreže. Konačno, predstavljamo rezultate eksperimenata u kojima smo ispitali uspješnost naše mreže u usporedbi sa (Wang et al., 2019) i drugim radovima na ovu temu.

2. Konvolucijske neuronske mreže

Konvolucijska neuronska mreža vrsta je neuronske mreže najčešće korištena za zadatke kao što su prepoznavanje slike, klasifikacije slike i semantička segmentacija videa/slike. S ciljem lakšeg razumijevanja konvolucijskih neuronskih mreža, pojasnit ćemo potpuno povezane neuronske mreže. Osnovna gradivna jedinica potpuno povezane neuronske mreže je neuron. Na slici 2.1 prikazan je neuron sa tri ulaza. U potpuno povezanoj mreži svaki neuron u nekom sloju povezan je sa svim neuronima prethodnog sloja. Izlazna vrijednost neurona dana je idućom jednadžbom:

$$y_i = \sum_j f(w_{ij}x_j)$$

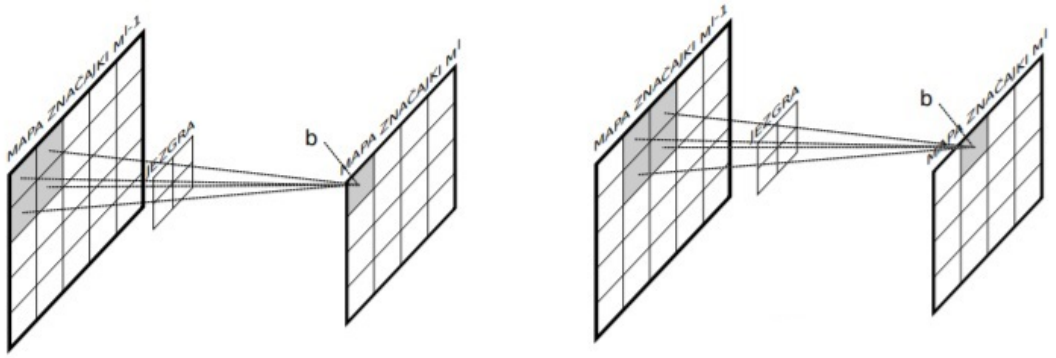
U jednadžbi w_{ij} označava težinu između dva neurona, j predstavlja neuron u prethodnom sloju, i neuron u promatranom sloju, a f aktivacijsku funkciju. Najčešće korištene aktivacijske funkcije su ReLU(*Rectified Linear Unit*), sigmoida i tangens hiperbolički.



Slika 2.1: Neuron sa 3 ulaza i aktivacijskom funkcijom f

Za razliku od potpuno povezane neuronske mreže, u konvolucijskoj neuronskoj mreži neuroni nisu predstavljeni kao skalari nego kao mape značajki, a odnos između dviju mapa značajki iz susjednih konvolucijskih slojeva određen je konvolucijskom

jezgrom. Vrijednosti konvolucijskih slojeva određuju se pomicanjem jezgre po tenzoru značajki iz prethodnog sloja. Pri svakom pomaku množe se težine iz jezgre s vrijednostima iz tenzora značajki koje u tom pomaku jezgra "pokriva". Pomicanje jezgre preko slike prikazano je na slici 2.2.



Slika 2.2: Jezgra veličine 2x2 prelazi preko mape značajki. Na lijevoj slici nalazi se prvi korak, a na desnoj drugi korak u slučaju kada je pomak jezgre jednak jedan. Preuzeto iz Vukotić (2014)

Dimenzija konvolucijskog sloja definira je pomoću tri broja koji označavaju broj mapa značajki u tom sloju te širinu i visinu mapa značajki. Dok je broj mapa značajki slobodno odabran hiperparametar, njena širina i visina ovise o veličini mape značajki prethodnog sloja, veličini i pomaku jezgara prethodnog sloja te sloju sažimanja između ta dva konvolucijska sloja. Na ulazu konvolucijske mreže nalazi se ili monokromatska slika odnosno jedan kanal ili višekanalna slika u boji (najčešće 3 kanala). Na izlazu mreže, odnosno nakon konvolucijskih slojeva, postavi se nekoliko potpuno povezanih slojeva, a širina zadnjeg od tih slojeva najčešće odgovara željenom broju klasa u koje se slika klasificira.

Obilježje je većine konvolucijskih modela za računalni vid rast broja mapa značajki i smanjenje prostornih dimenzija u konvolucijskim slojevima povećanjem dubine. Smanjenjem mape značajki, receptivno polje elemenata unutar mape značajki raste. Na taj način povećava se i značenje svake značajke pa postoje značajke visoke, srednje i niske razine. Značajke niske razine imaju malo receptivno polje pa često predstavljaju rubove, bridove ili neke jednostavne oblike koji se mogu primjeniiti na par piksela, a značajke srednje i visoke razine predstavljaju kompleksnije oblike.

U tablici 2.1 nalazi se lista najpoznatijih arhitektura konvolucijskih neuronskih mreža, broj slojeva tih arhitektura i broj parametara za svaku arhitekturu. Za treniranje mreža ovih veličina potrebni su veliki skupovi anotiranih podataka. Najčešće

Tablica 2.1: Najpoznatije arhitekture konvolucijskih neuronskih mreža

Model	Godina	Broj parametara	Dubina
AlexNet	2012.	60M	8
VGG	2014.	138M	19
ResNet-50	2016.	25.56M	50
ResNet-152	2016.	60.19M	152
Xception	2017.	22.8M	126

korišteni skupovi podataka su:

- **ImageNet** - Sadrži preko 14 milijuna anotiranih slika u više od 20000 kategorija. Trenutno je najveći i najpoznatiji skup podataka za klasifikaciju slika koji je pokrenuo rast u korištenju konvolucijskih neuronskih mreža. Skup podataka sadrži klase poput velikog broja vrsta životinja (škorpion, tarantula...), svakodnevnih predmeta (boca za pivu, otirač, čekić...), građevina (samostan, knjižnica...) i mnogih drugih.
- **MNIST** - Skup podataka namijenjen za klasifikaciju znamenki koji sadrži 60000 slika ručno napisanih znamenki za treniranje i 10000 za testiranje.
- **MS COCO** - Skup podataka korišten pri treniranju modela za detekciju objekata i semantičku segmentaciju slike. Sadrži preko 330000 slika i 91 kategoriju.
- **CIFAR** - Ovi skupovi podataka sadrže po 60000 slika od kojih se 50000 koristi za treniranje, a 10000 za testiranje. Postoje dvije inačice ovoga skupa. CIFAR-10 sadrži 10 klasa od kojih svaka ima 6000 slika, a CIFAR-100 ima 100 klasa sa po 600 slika. Skupovi podataka CIFAR se koriste za klasifikaciju slika.
- **PASCAL VOC** - Skup podataka korišten za natjecanja u detekciji objekata, semantičkoj segmentaciji slika, detekciji akcije na slici i klasifikaciji slika. Sadrži preko 17000 slika od kojih svaki zadatak ima svoj skup slika i svoje klase.

Prethodno nabrojani skupovi podataka koriste se za treniranje modela za klasifikaciju, semantičku segmentaciju slika, detekciju objekata i razne druge zadatke. Svaki od ovih skupova sadrži dovoljan broj anotiranih primjera u svrhu treniranja prethodno navedenih arhitektura konvolucijskih mreža. Međutim, što napraviti u slučaju kad ne postoji dovoljno velik skup primjera za klasu koju želimo klasificirati?

2.1. Učenje prijenosom

Ljudski mozak sposoban je naučiti novi vizualni koncept iz samo jednog primjera. Na primjer, čovjeku je dovoljno vidjeti samo nekoliko Segwaya da bi ga razlikovao od ostalih prijevoznih sredstava kao što su romobil ili skuter. Iako nije sasvim sigurno kako čovjeku to uspijeva, neki od znanstvenika predlažu da se radi o takozvanom učenju prijenosom (engl. *tranfer learning*)(Oquab et al., 2014). Učenje prijenosom ideja je pomoću koje se prethodna znanja odnosno iskustva koriste za brže učenje različitih novih koncepata.

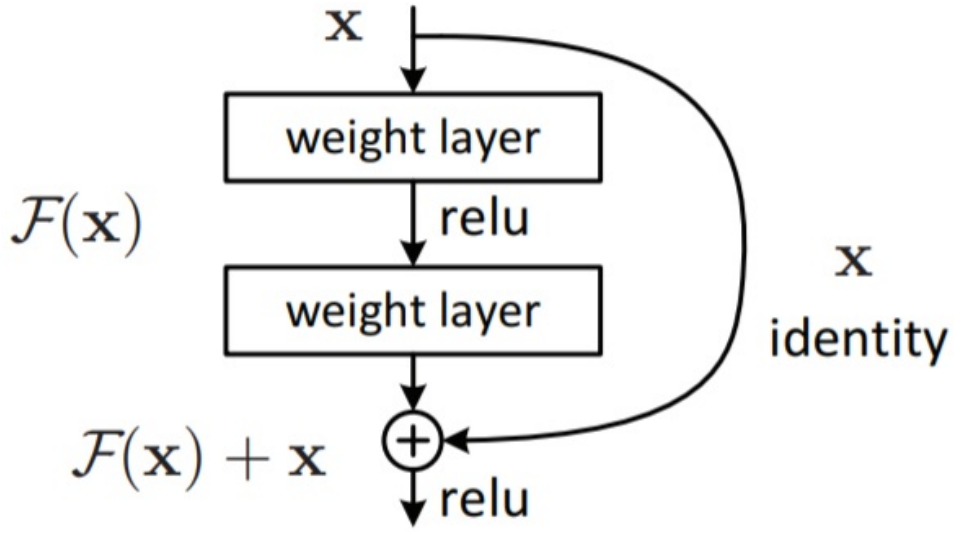
Slična ideja može se primjeniti kod konvolucijskih neuronskih mreža. Mreža se trenira na klasama za koje postoji dovoljan broj primjera tako da mreža nauči izvlačiti smislene generične značajke. Nakon što je mreža istrenirana za izvlačenje generičkih značajki, njen prednji dio se koristi kao ekstraktor značajki (engl. *feature extractor*). Na posljednji sloj ekstraktora značajki primjenjujemo gubitak, na primjer dodavanjem nekoliko potpuno povezanih slojeva. Težine između nadodanih slojeva treniraju se za prilagodbu prepoznavanju novih koncepata, odnosno novih klasa koje mreža do tad nije vidjela. Najčešće se kao ekstraktori značajki koriste prethodno navedene arhitekture trenirane na velikim skupovima podataka poput ImageNeta.

2.2. Arhitektura ResNet

Povećanjem popularnosti konvolucijskih neuronskih mreža uočena je važnost dubine mreža. Konkretno, povećanjem broja slojeva mreže dobiveni su sve bolji rezultati na ImageNetu i drugim natjecanjima. Slijedeći logiku moglo se zaključiti da dublji modeli mogu davati samo bolje ili barem jednako dobre rezultate kao plitki modeli budući da svaki dodani sloj na postojeći plitki model može biti samo preslika vrijednosti iz prošlog sloja čime bi duboki model trebao davati jednake rezutate kao plitki. Autori rada He et al. (2015) eksperimentima su došli do saznanja da postojeći algoritmi nisu sposobni doći do takvih rješenja i da dodavanje slojeva na već jako duboke modele rezultira smanjenjem točnosti modela na skupu za učenje. Kao rješenje problema predlažu ResNet arhitekturu.

Autori smatraju da je uzrok tog problema kompliciranost postizanja funkcije mapiranja identitetitom koristeći više nelinearnih slojeva. Uzmimo da je $\mathcal{H}(x)$ preslikavanje ulaza x nakon nekoliko slojeva mreže. Umjesto da mreža pokušava naučiti preslikavanje $\mathcal{H}(x)$, predlaže se da mreža uči rezidulano preslikavanje odnosno:

$$\mathcal{F}(x) := \mathcal{H}(x) - x$$



Slika 2.3: Osnovna konvolucijska jedinica od 2 sloja u ResNet arhitekturi. Preuzeto iz He et al. (2015)

čime se originalno mapiranje onda pretvara u $\mathcal{F}(x) + x$. Ovim pristupom ako mreža treba postići mapiranje identitetom potrebno je samo postaviti sve težine konvolucijske jedinice na nulu, a ne pokušavati naučiti mapiranje identitetom nizom nelinearnih slojeva.

Rezidualno preslikavanje postiže se koristeći preskočne veze (engl. *skip connections*). Preskočne veze u ResNet arhitekturi "preskaču" po dva ili tri sloja iako je moguće i više. Osnovna konvolucijska jedinica od dva sloja prikazana je na slici 2.3. Funkcija $\mathcal{F}$ za prikazanu konvolucijsku jedinicu odgovara izrazu:

$$\mathcal{F}(x) = W_2\sigma(W_1x)$$

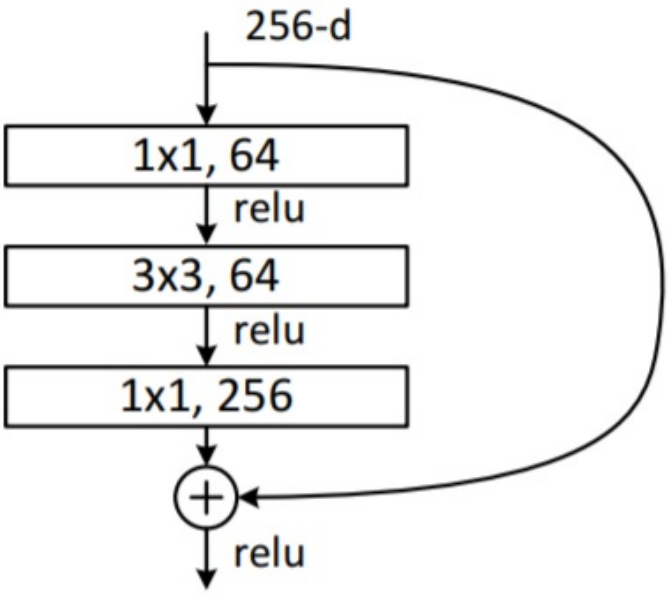
gdje σ označava ReLU. Da bi se $\mathcal{F}(x)$ i x mogli zbrojiti potrebno je osigurati slaganje njihovih dimenzija. Kad nisu istih dimenzija, potebno je napraviti ili nadopunjavanje nulama ili linearnu projekciju W_s tako da izlaz osnovne konvolucijske jedinice glasi:

$$y = \mathcal{F}(x) + W_sx$$

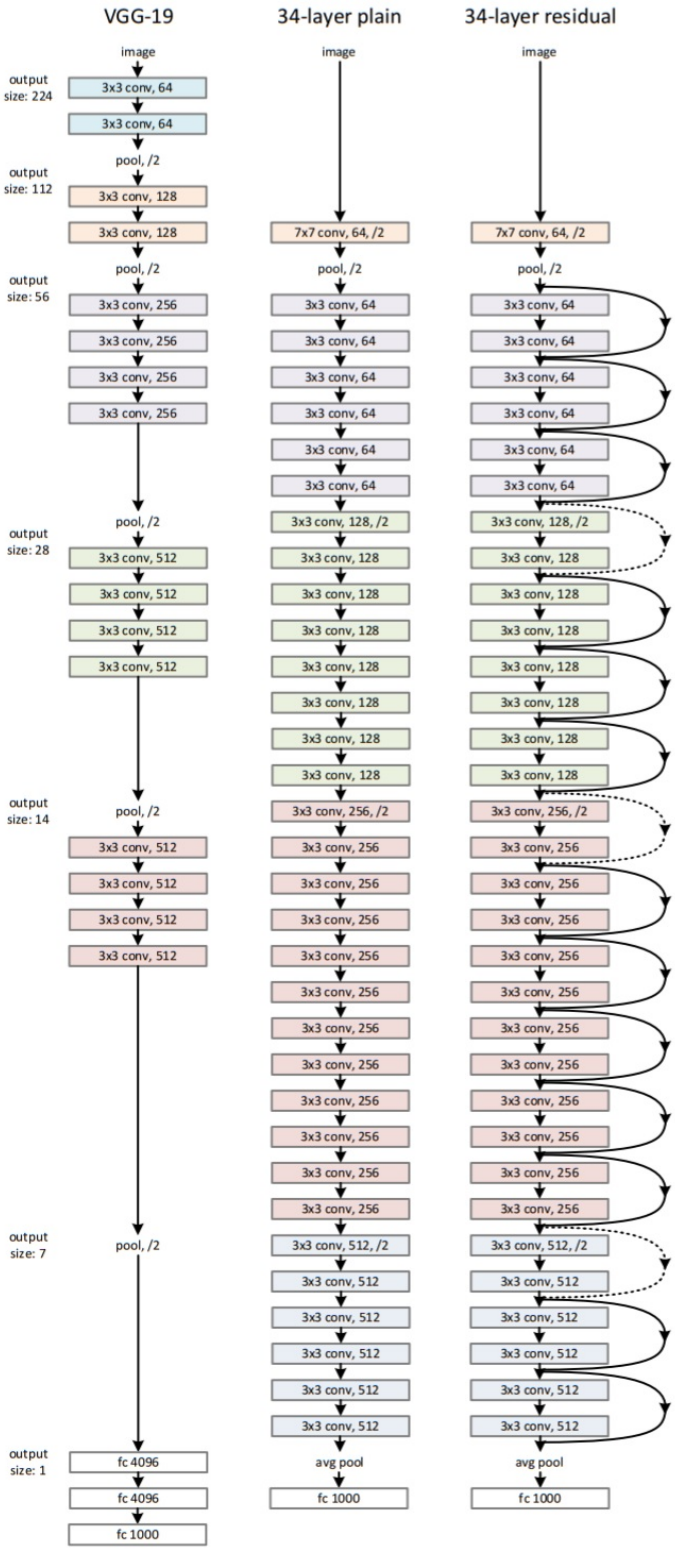
Budući da dodavanje linearnih projekcija povećava složenost, preporuča se koristiti ih samo za prilagođavanje dimenzija.

Dok se u plićim ResNet arhitekturama od 18 ili 34 sloja koriste konvolucijske jedinice prikazane na slici 2.3, u dubljim ResNet arhitekturama od 50 ili više slojeva preporuča se koristiti konvolucijsku jedinicu uskog grla (engl. *bottleneck*) prikazanu na slici 2.4. Konvolucijska jedinica s uskim grlom sadrži 3 sloja. Prvi i zadnji imaju

1x1 konvolucije u svrhu smanjivanja i povećanja dimenzionalnosti što omogućuje srednjem sloju s konvolucijom 3x3 rad s manjim dimenzijama ulaza i izlaza. Korištenje rezidualnih jedinica znatno poboljšava performanse modela.



Slika 2.4: Konvolucijska jedinica s uskim grlom u ResNet arhitekturi. Preuzeto iz He et al. (2015)



Slika 2.5: Prikazana je arhitektura VGG-19 mreže, "obične" mreže s 34 konvolucijska sloja i ResNet-34 mreže. Dok VGG-19 mreža zahtjeva 19.6 milijardi FLOP-ova, prikazana "obična" mreža s 34 sloja zahtjeva samo 3.6 milijardi FLOP-ova, a razlika u zahtjevnosti "obične" i ResNet mreže je u preskočnim vezama odnosno u zbrajanju $\mathcal{F}(x)$ i x koje je zanemarivo. Preuzeto iz He et al. (2015)

3. Učenje s malim brojem primjera

U prethodnom poglavlju opisano je učenje prijenosom kao način iskorištavanja prethodnog iskustva za shvaćanje novih koncepata i za primjenu s neuronskim mrežama. Spomenili smo da kada imamo dovoljno slika klasa koje želimo klasificirati možemo uzeti ekstraktor značajki treniran na nekim drugim klasama i dotrenirati model da prepoznaje nove klase sa slika. Ekstremni primjer toga je učenje s malim brojem primjera (engl. *few-shot learning*) ili čak učenje s jednim primjerom (engl. *one-shot learning*). Kako u *few-shot* i *one-shot* učenju ne postoji dovoljno slika da se model retrenira, potrebno je pronaći drugi način kako odrediti pripadnost klasi. Najčešći je pristup korištenje raznih metrika udaljenosti između ispitne slike i slika koje su anotirane.

3.1. Klasfikacija s malim brojem primjera

Iako *few-shot* učenje nije popularna tema među znanstvenicima strojnog učenja, pogotovo od pojavljivanja velikih skupova podataka, još uvijek postoje zanimljivi pristupi na tu temu te ćemo u ovom poglavlju proći kroz par njih. Podijeliti ćemo ove algoritme u tri vrste: algoritmi temeljeni na inicijalizaciji, algoritmi koji uče metrike udaljenosti i algoritmi temeljeni na "halucinacijama".

- **Algoritmi temeljni na inicijalizaciji** - (engl. *Initialization based methods*) Algoritmi koji pokušavaju postići što bolju inicijalizaciju parametara modela tako da je dovoljan mali broj primjera da se mreža nauči. Sličan pristup je učenje optimizatora. Jedan od primjera radova na tu temu je Ravi i Larochelle (2017) koji predlaže LSTM (engl. *Long Short-Term Memory*) *meta-learner* model koji predviđa točni algoritam optimizacije i prikladne promjene parametara potrebne drugom modelu koji se trenira u *few-shot* režimu da dođe do što brže konvergencije. Algoritmi temeljeni na dobroj inicijalizaciji brzo uče i sposobni su brzo naučiti nove klase, ali postižu znatno lošije rezultate kada nove klase koje model treba naučiti i one na kojima je model istreniran dolaze iz različitih domena(Chen et al., 2019).

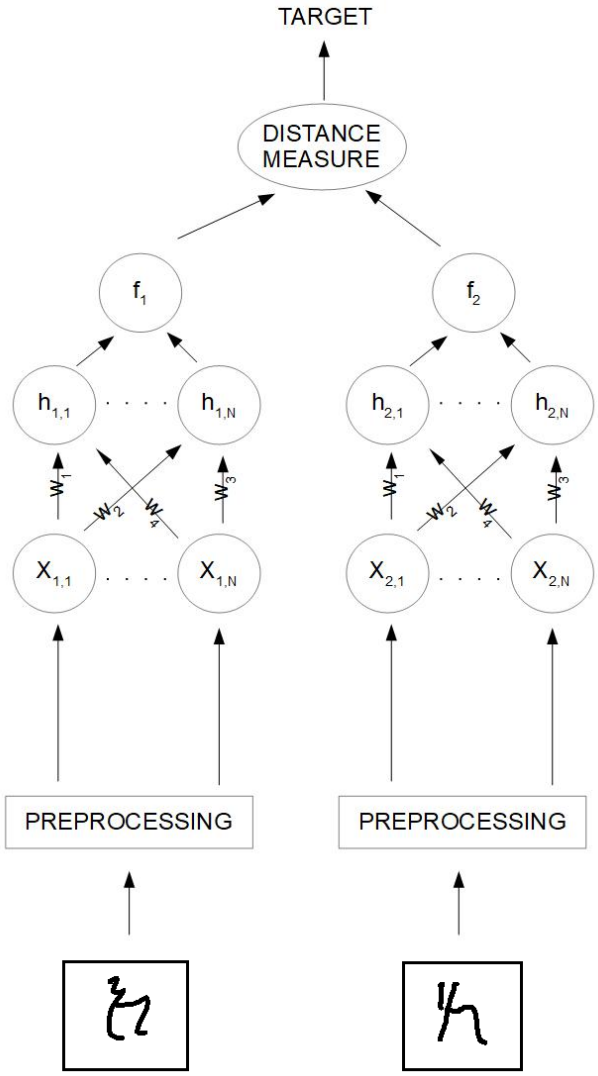
– **Algoritmi temeljeni na "halucinacijama"** - (engl. *Hallucination based models*) Cilj je ovih algoritama rješiti se problema nedostatka podataka odnosno generirati slike klasa koje imaju malo primjera na kojima će trenirati modele. Kako bi to uspjelo, na klasama za koje imaju dovoljno slika treniraju se generativni modeli koji se primjenjuju na klasama s malo primjera. Generativni modeli su često ili modeli koji na početnim klasama uče varijancu pa tu varijancu primjene na novim klasama (Hariharan i Girshick, 2017) ili GAN-ovi čiji je cilj prenijeti stil (Antoniou et al., 2018). Bitno je napomenuti da ovi algoritmi temeljeni na "halucinacijama" ne isključuju korištenje drugih *few-shot* algoritama odnosno često se kombinaraju sa algoritmima temeljenim na učenju metrika.

– **Algoritmi koji uče metriku udaljenosti** - (engl. *Distance metric learning based models*) Algoritmi koji uče metriku udaljenosti slijede čovjekov način razmišljanja jer im je cilj naučiti kako najbolje odrediti razliku između dvije slike. Ideja pristupa je da ako model može naučiti sličnost između dviju slika onda je sposoban iz anotiranih slika odnosno klasa prepoznati kojoj klasi promatrana slika najviše sliči. Neke od metrika s kojima se sličnost između slika može mjeriti su: kosinusna udaljenost, euklidska udaljenost i hrbatna regresija (engl. *ridge regression*).

3.1.1. Sijamske neuronske mreže

Sijamske neuronske mreže predstavljene su 1993. godine kao rješenje za verifikaciju potpisa (Bromley et al., 1993). Sijamska neuronska mreža algoritam je koji se sastoji od dviju identičnih neuronskih podmreža koje izvlače vektore značajki iz dviju slika. Dobiveni se vektori kasnije uspoređuju izračunom vektora udaljenosti odabranom metrikom. Budući da su mreže identične, za slične slike dobit će se slični vektori značajki, a za slike koje ne pripadaju istoj klasi očekuju se znatno različiti vektori značajki. Za potpise se može reći da pripadaju istoj klasi/čovjeku ako je vektor udaljenosti ispod neke vrijednosti. Unatoč tome što sijamske neuronske mreže uobičajeno imaju identične težine među slojevima, događa se da dvije podmreže ne trebaju dijeliti težine. Na primjer kad se sijamske neuronske mreže koriste na različitim domenama slika. Konkretnije, primjer je takvog zadatka provjera podudaranja skice odnosno crteža s pravom slikom. Neki od zadataka za koje se najčešće koriste sijamske neuronske mreže su:

– provjera podudaranja - Primjer toga je provjera potpisa. Kako za provjeru slaganja potpisa najčešće nemamo dostupan dovoljno velik skup podataka slika,



Slika 3.1: Sijamska neuronska mreža se sastoji od dvije identične podmreže koje se spajaju na vrhu sa mjerom sličnosti

u tom su slučaju prikladan algoritam sijamske mreže i računanje udaljenosti odnosno sličnosti između slika.

– dohvaćanje sličnih objekata iz velikih baza podataka - Osim što su sijamske neuronske mreže odlične za računanje sličnosti između slika, još jedna prednost je jednostavnost *inference*-a u ovom primjeru. Ta jednostavnost se postiže tako da prilikom dodavanja objekata u bazu podataka odmah izračunamo i spremimo njihove vektor značajki. Jednom kad imamo spremljene vektore značajki za cijelu bazu, za računanje sličnosti s novim objektom samo je potrebno izračunati vektor značajki novom objektu te ga usporediti s vektorima iz baze.

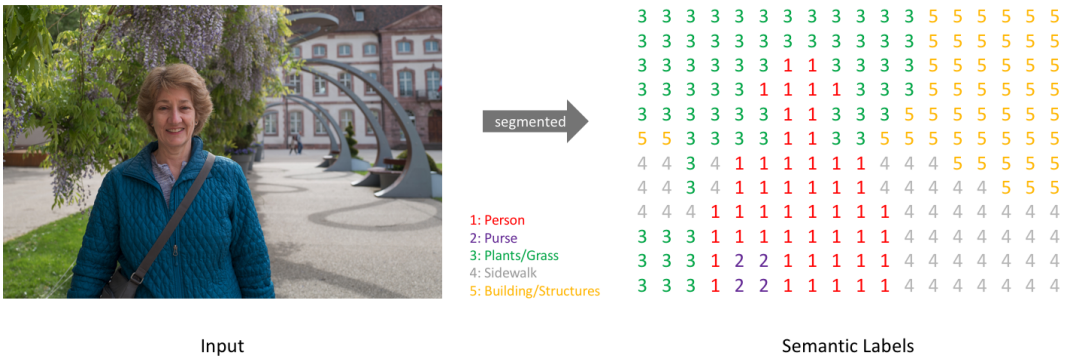
– *few-shot* klasfikacija slika - U prethodno poglavlju spomenili smo algoritme te-

meljene na udaljenosti i zašto su dobri za klasifikaciju s malim brojem primjera. Primjer jednog od tih algoritama su sijamske mreže. U Koch et al. (2015) autori pokušavaju rješiti problem *one-shot* klasifikacije znakova iz Omniglot skupova podataka koristeći konvolucijske sijamske neuronske mreže. Omniglot skup podataka sadrži 50 različitih abeceda koje autori podijele na skup za učenje i skup za evaluaciju. Iz skupa za učenje izvlače parove slika istih i različitih znakova pa na njima treniraju sijamsku neuronsku mrežu koju kasnije testiraju na parovima slika iz skupa za evaluaciju. Autori rada su dobili zavidne rezultate i pokazali prednost korištenja sijamskih neuronskih mreža u *one-shot* režimu..

3.2. Semantička segmentacija s malim brojem primjera

3.2.1. Semantička segmentacija

Semantička segmentacija je zadatak klasifikacije svakog piksela slike u neku od predodređenih klasa. Većina današnjih rješenja semantičke segmentacije temelji se na konvolucijskim neuronskim mrežama. Ulaz mreže je slika koja najčešće sadrži 3 kanala, a izlaz je segmentacijska mapa u kojoj je svakom pikselu dodijeljena neka klasa. Primjer segmentacijske mape prikazan je na slici 3.2.

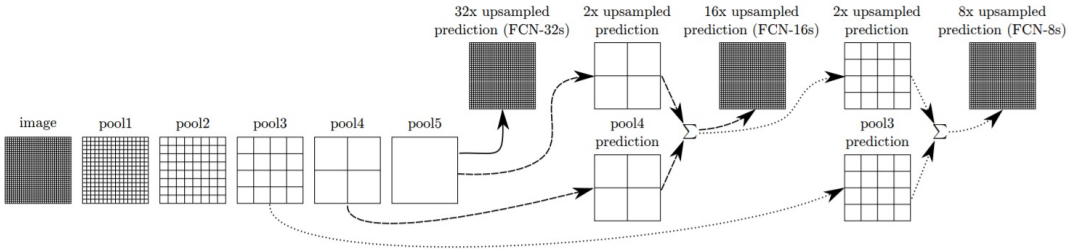


Slika 3.2: Segmentacijska mapa. Preuzeto iz Jordan (2018)

U drugom poglavlju objasnili smo razliku između značajki niske, srednje i visoke razine, no nismo objasnili gubitak informacija koji nastaje odlaskom u dubinu. Kretajući se sve dublje u mrežu možemo reći da imamo sve veći gubitak informacija. Ulazna slika sadrži sve informacije, a iz nje proizlaze značajke niske razine koje već sadrže manje informacija jer su njene značajke agregirane iz ulazne slike. Nakon značajki niske razine, slijede značajke srednje razine koje više nemaju pristup svim informacijama ulazne slike nego agregiranim informacijama iz niske razine. Isto vrijedi i za

značajke visoke razine koje imaju pristup samo informacijama iz srednjih slojeva. Dok je u klasifikaciji dovoljno imati pristup značajkama visoke razine da bi donijeli odluku jer se slika klasificira kao cjelina, u semantičkoj segmentaciji klasificiramo svaki piksel pa gubitak informacija, kada se oslanjamo samo na informacije iz dubokih slojeva, nije prihvatljiv. Kako bismo mogli klasificirati svaki piksel, potrebno je imati pristup značajkama i visoke i srednje i niske razine. Jedan način kako bismo to ostvarili su FCN (*Fully Convolutional Network*) arhitekture.

Fully Convolutional Network arhitektura je za semantičku segementaciju koja primjenjuje naduzorkovanje dekonvolucijom na dubljim slojevima sažimanja te ih zbraja sa slojevima sažimanja ranije u mreži. Postupak je prikazan na slici 3.3. Dekonvolucija se može naučiti ili fiksirati koristeći algoritme poput bilinearne interpolacije.



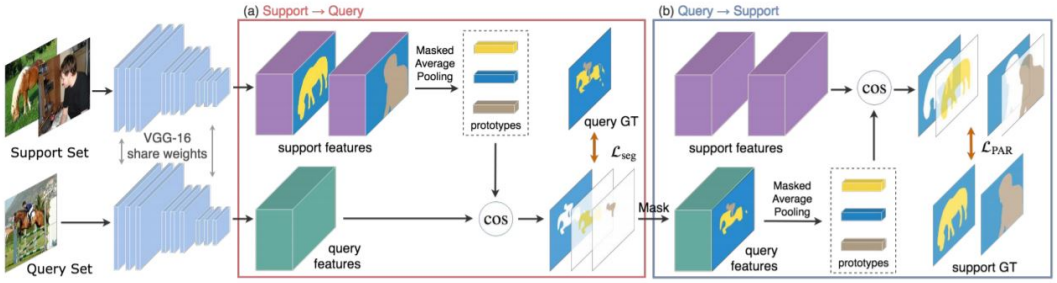
Slika 3.3: Prikaz postupka naduzorkovanja dekonvolucijom i zbrajanja sa mapama značajki većih dimenzija kod FCN-a. Preuzeto iz Long et al. (2015)

.

3.2.2. PANet - *Prototype Alignment Network*

PANet ili *Prototype Alignment Network* pristup je predložen u Wang et al. (2019). Dotadašnji radovi na temu *few-shot* semantičke segmentacije su se temeljili na izvlačenju znanja iz potpornog skupa i ubacivanje tog znanja u parametarski modul koji provodi segementaciju. Autori rada predlažu odvajanje procesa u dva dijela: izvlačenje prototipova i određivanje pripadnosti korespondencijskom metrikom. Cilj prvog dijela je naučiti model da uspješno izvlači robusne i fleksibilne prototipove za svaku od semantičkih klasa. U drugom dijelu se provodi semantička segmentacija svakog piksela tako da ih se uspoređuje sa dobivenim prototipovima iz prvog dijela.

Kako bismo ispravno objasnili ideju PANet-a, potrebno je detaljno objasniti zadatak. Skup podataka D podijeljen je u dvije disjunktne skupine klasa: viđene klase C_{seen} i neviđene klase C_{unseen}. Viđene klase čine skup za treniranje D_{train}, a neviđene klase čine skup za testiranje D_{test}. Iz oba skupa izvlače se epizode koje se sastoje



Slika 3.4: Algoritam PANet: Na lijevoj strani slike iz potpornih slika i slike upita pomoću ekstraktora značajki računaju se mape značajke. a) Računaju se prototipovi iz potpornih slika te se pomoću kosinusne udaljenosti uspoređuju s pikselima iz slike upita. b) Iz predviđene maske iz dijela a) na slici upita računaju se prototipovi koji se uspoređuju s pikselima sa slika iz potpornog skupa. Preuzeto iz Wang et al. (2019)

.

od potpornih slika S (engl. *support images*) i slika upita Q (engl. *query images*) pa vrijedi:

$$D_{train/test} = \{(S_i, Q_i)\}_{i=1}^{N_{train/test}}$$

gdje N_{train} i N_{test} označuju broj epizoda za treniranje odnosno testiranje. Tijekom treniranja potporne slike i slike upita su anotirane dok u fazi testiranja samo potporne slike imaju anotacije.

Razlikujemo dvije vrste učenja: učenje značajki i učenje prototipa. Učenje značajki odnosi se na fazu treniranja gdje težine u mreži mijenjamo tako da izvlače što bolje značajke. Težine mreže ažuriraju se nakon svake epizode. Učenje prototipa događa se tijekom svake epizode faze treniranja i faze testiranja. Cilj učenja prototipa je izvlačenje znanja o klasama iz malog broja anotiranih primjera odnosno potpornog skupa. Učenje značajki prikazano je na slici 3.4, a učenje prototipa nalazi se na *a)* dijelu iste slike.

Eksperimente razlikujemo po broju neviđenih klasa i veličini potpornog skupa svake klase. Potporni skup se sastoji od malog broja slika (manje od deset), a slika upita je najčešće jedna po epizodi. C*-way* K*-shot* eksperiment znači da imamo C neviđenih klasa i K slika u potpornom skupu svake klase. Ukupni potporni skup sadrži $C * K$ parova slika i maski (K parova za svaku od C klasa). U slučaju kada se na nekoj od slika nalazi više promatranih klasa, potrebno je tu sliku računati kao dio potpornog skupa za sve klase koje se nalaze na slici. Tada bi veličina ukupnog potpornog skupa bila manja od $C * K$. I potporne slike i slike upita se izvlače iz istog skupa neviđenih klasa koji se mijenja svaku epizodu.

Izvlačenje prototipa

Prototip je vektor značajki koji dobro generalizira piksele klase koju opisuje, ali ih u isto vrijeme razlikuje od ostalih klasa. To postižemo tako da izračunamo prosječnu vrijednost značajki nad anotiranim područjem promatrane klase na toj slici. To je moguće napraviti ranim ili kasnim stapanjem. U ranom stapanju primjenjujemo masku nad ulaznom slikom prije prolaska kroz *feature extractor* dok u kasnom stapanju masku primjenjujemo direktno na mape značajki. U našem radu koristimo kasnu fuziju što nam daje sljedeću formulu za računanje prototipa klase c:

$$\mathbf{p}_c = \frac{1}{K} \sum_k \frac{\sum_{x,y} \mathbf{F}_{c,k}^{(x,y)} \mathbf{1}\left[M_{c,k}^{(x,y)} = c\right]}{\sum_{x,y} \mathbf{1}\left[M_{c,k}^{(x,y)} = c\right]}$$

gdje je $F_{c,k}$ mapa značajki za sliku $I_{c,k}$, K broj potpornih slika, x i y lokacija piksel na mapi značajki, a $\mathbf{1}\left[\cdot\right]$ funkcija koja poprima vrijednost 1 ako je izraz unutar zagrada istinit, a 0 ako nije. Osim računanja prototipa klasa iz skupa C, potrebno je izračunati i prototip pozadine odnosno područja na slici koji ne pripadaju nijednoj poznatoj klasi. Formula za prototip pozadine je:

$$\mathbf{p}_{\text{bg}} = \frac{1}{CK} \sum_{c,k} \frac{\sum_{x,y} \mathbf{F}_{c,k}^{(x,y)} \mathbf{1}\left[M_{c,k}^{(x,y)} \notin \mathcal{C}_i\right]}{\sum_{x,y} \mathbf{1}\left[M_{c,k}^{(x,y)} \notin \mathcal{C}_i\right]}$$

Određivanje pripadnosti korespondencijskom metrikom

Kako bismo klasificirali piksele potrebno je svaki piksel iz mape značajki usporediti s prototipovima klasa i pozadine. Sličnost između piksela i klasa računamo korespondencijskom metrikom. U našem modelu to je kosinusna udaljenost. Na izračunati vektor udaljenosti primjenimo *softmax* da dobijemo mapu vjerojatnosti.

$$\tilde{M}_{q:j}^{(x,y)} = \frac{\exp\left(-\alpha d\left(\mathbf{F}_q^{(x,y)}, \mathbf{p}_j\right)\right)}{\sum_{p_j \in \mathcal{P}} \exp\left(-\alpha d\left(\mathbf{F}_q^{(x,y)}, \mathbf{p}_j\right)\right)}$$

U navedenoj jednadžbi težinski faktor α postavljen je na 20 po uzoru na Wang et al. (2019). Težinski faktor α recipročna je vrijednost temperature *softmaxa*. Njime određujemo nagib funkcije tako da što je veći α to će *softmax* biti strmiji, a model sigurniji u svoje predikcije. Pripadnost klasi odredimo tako da odaberemo klasu s najvećom predikcijom u mapi vjerojatnosti:

$$\hat{M}_q^{(x,y)} = \arg\max_j \tilde{M}_{q;j}^{(x,y)}$$

Funkcija gubitka semantičke segmentacije slika upita se računa sljedećom formulom:

$$\mathcal{L}_{\text{seg}} = -\frac{1}{N} \sum_{x,y} \sum_{p_j \in \mathcal{P}} \mathbf{1}\left[M_q^{(x,y)} = j\right] \log \tilde{M}_{q;j}^{(x,y)}$$

gdje N označuje broj piksela na slici, M_q pravu segmentacijsku masku slike upita, a $\tilde{M}_q$ mapu vjerojatnosti predikcije. Optimizacijom ovog gubitka pokušavamo naučiti mrežu da generira što preciznije prototipe za svaku klasu.

Regularizacija usklađivanjem prototipa

Autori rada Wang et al. (2019) smatraju da dotadašnji radovi na temu *few-shot* semantičke segmentacije nisu iskoristili svo znanje koje se može izvući iz potpornog skupa. Oni predlažu regularizaciju usklađivanjem prototipa odnosno PAR (*Prototype Alignment Regularization*). Cilj regularizacije usklađivanja prototipa je provjeriti ispravnost izvučenih prototipova odnosno uskladiti ih sa slikom upita.

Ako naš model iz potpornog seta dobro predviđa segmentacijsku masku na slici upita onda je za očekivati da bi se iz slike upita i predviđene maske mogli izvući prototipovi dovoljno dobri da segmentiramo slike iz potpornog skupa. Drugim riječima, regularizacija usklađivanjem prototipa provodi *few-shot* semantičku segmentaciju u obrnutom smjeru tako da da potporni skup postaje skup koji se skup slika upita, a slika upita i predviđena maska postaju potporni skup. Formule za izvlačanje prototipova i računanje maski su iste kao prije. Mape vjerojatnosti dobijemo za C*K slika iz potpornog skupa:

$$\tilde{M}_{c,k;j}^{(x,y)} = \frac{\exp\left(-\alpha d\left(\mathbf{F}_{c,k}^{(x,y)}, \bar{\mathbf{p}}_j\right)\right)}{\sum_{\bar{p}_j \in \left\{\bar{p}_c, \bar{p}_{bg}\right\}} \exp\left(-\alpha d\left(\mathbf{F}_{c,k}^{(x,y)}, \bar{\mathbf{p}}_j\right)\right)}$$

Funkcija gubitka za regularizaciju glasi:

$$\mathcal{L}_{\text{PAR}} = -\frac{1}{CKN} \sum_{c,k,x,y} \sum_{p_j \in \mathcal{P}} \mathbf{1}\left[M_q^{(x,y)} = j\right] \log \tilde{M}_{q;j}^{(x,y)}$$

Dobiveni gubitak $\mathcal{L}_{\text{PAR}}$ pomnožimo s faktorom regularizacije λ i zbrojimo ga sa gubitkom $\mathcal{L}_{\text{seg}}$ da dobijemo ukupni gubitak.

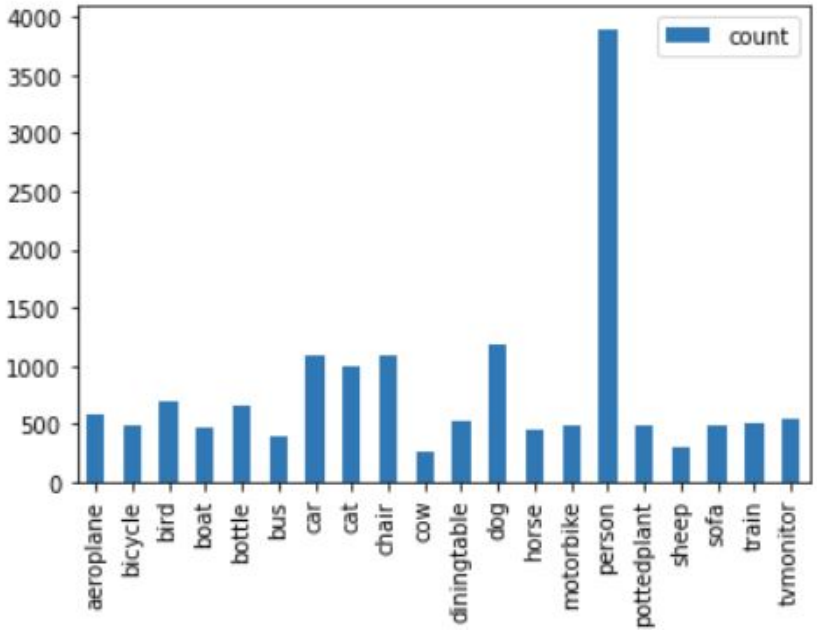
$$\mathcal{L} = \mathcal{L}_{\text{seg}} + \lambda \mathcal{L}_{\text{PAR}}$$

Faktor regularizacije λ je fiksiran na 1 jer mijenjanje njegove vrijednosti rezultira zanemarivim poboljšanjem rezultata (Wang et al., 2019). Regularizacija usklađivanjem prototipa pomaže modelu da izvlači što robusnije prototipove koje dobro generaliziraju nakon samo par slika.

4. Podatci i alati

U ovom poglavlju pojasnit ćemo korištene alate i podatke na kojima smo provodili eksperimente. Programski kod rada implementiran je u programskom jeziku Python. Korištene su razne biblioteke od kojih je najbitniji PyTorch. Za skup podataka odabran je skup PASCAL VOC zbog njegove česte primjene za zadatak sematičke segmentacije.

4.1. Skup podataka PASCAL VOC



Slika 4.1: Histogram broja slika po klasi

.

Eksperimente provodimo na skupu VOC 2012 koji proširujemo skupom SBD (*Semantic Boundaries Dataset*). SBD proširenje predloženo je u Hariharan et al. (2011). Konačni skup podataka sastoji se od 11355 slika. VOC (*Visual Object Classes*) natjecanja počinju 2005. godine kada skup podataka ima samo četiri klase sa 1578 slika.

Za natjecanje 2007. godine skup podataka se proširio na 20 klasa i te klase su do sada ostale iste:

- **Osoba**: osoba
- **Životinja** : ptica, mačka, krava, pas, konj, ovca
- **Vozilo**: zrakoplov, bicikl, brod, autobus, automobil, motocikl, vlak
- **Namještaj**: boca, stolica, stol za blagovanje, lončasta biljka, kauč, televizija/monitor

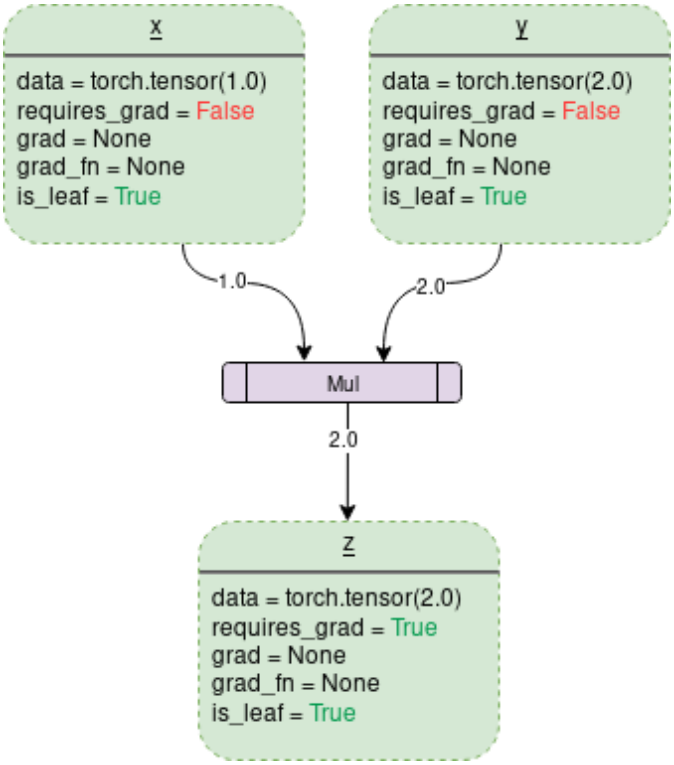
Pascal skup podataka se prestao proširivati 2012. godine kada je znatno povećan skup slika za semantičku segmentaciju.

Ovih 20 klasa smo ravnomjerno podjelili u četiri skupine tako da svaka skupina sadrži pet klasa. Model se trenira na tri skupine, a testira na četvrtoj. Podjela klasa u kategorije može se pronaći u Shaban et al. (2017).

4.2. PyTorch

PyTorch je *open source* bibilioteka za duboko učenje koju je razvio laboratorij FAIR (*Facebook Artificial Intelligence Research*). Iako PyTorch nudi podršku za programske jezike C++ i Java , najčešće se koristi u kombinaciji sa Pythonom. Poput Tensorflowa, Kerasa i sličnih biblioteka za razvoj neuronskih mreža, PyTorch nudi izvršavanje operacija s tenzorima pomoću GPU (*Graphic Processing Unit*) jedinica. Dok se CPU sastoji od malog broja jako kompleksnih jezgri, GPU sadrži veliki broj jednostavnih jezgri koji nam omogućuju paralelizaciju računanja operacija nad tenzorima.

Osim korištenja GPU-a, PyTorch nudi i *autograd* paket. *Autograd* omogućava jednostavno računanje gradijenata odnosno derivacija funkcija. Treniranje neuronskih mreža se temelji na algoritmu *backpropagation* koji računa gradijente funkcije gubitka s obzirom na parametre modela. Pomoću PyTorch biblioteke ne trebamo se brinuti oko računanja gradijenata naše mreže jer *autograd* paket obavlja to u pozadini. *Autograd* dinamički generira graf naše mreže i brine se o toku podataka kroz mrežu za vrijeme izvršavanja koda. Vrhovi u grafu predstavljaju tenzore, a bridovi između grafa operacije nad tenzorima. Pozivanjem *backward()* funkcije *autograd* sam računa gradijente prolaskom unazad kroz graf odnosno mrežu.



Slika 4.2: Prikaz grafa za množenje dva tenzora

5. Eksperimenti

Eksperimenti su izvedeni na skupu podataka PASCAL VOC, a za evaluaciju rezultata korištene su dvije mjere. Prva mjera je prosječni omjer presjeka i unije (engl. mean-IoU/*mean Intersection-over-Union*) gdje se računa IoU za pozadinu i svaku od klasa pa se kao rezultat uzme prosjek pozadine i svih klasa. IoU se računa tako da se površina presjeka segmenata predikcije i istine za promatranu klasu podijeli sa unijom. Druga mjera je binarni omjer presjeka i unije (engl. binary-IoU/*binary Intersection-over-Union*) koji se računa tako da se sve klase, osim pozadine, skupe u jednu klasu pa se uprosječi IoU te jedne klase i pozadine. Mean-IoU se smatra ispravnijom mjerom jer se u njoj razlikuju segmenti različitih klasa. Za evaluaciju pokrećemo pet puta testiranje od 1000 epizoda, a konačnim rezultatom smatramo prosjek tih pet testova.

Za ekstraktor značajki korištena je ResNet-18 arhitektura i napravljena je usporedba s ResNet-34 arhitekturom. Model je inicijaliziran s težinama pretreniranim na skupu podataka ImageNet. Skup slika proširen je nasumičnim vodoravnim zrcaljenjem, a slike su sklarine na veličinu (417, 417). Optimizacija modela se vrši stohastičkim gradijentnim spustom s momentom 0.9 tijekom svih 30000 iteracija. Stopa učenja je postavljena na 0.001 i smanjuje se za 0.1 svakih 10000 iteracija, propadanje težina je 0.0005, a veličina grupe je 1.

Treniranje modela se izvršavalo na kartici NVIDIA GTX GeForce 1070.

5.1. *Few-shot* rezultati

Tablice 5.1 i 5.2 prikazuju mean-IoU rezultate naše PANet mreže s ResNet-18 *feature extractor*-om i bez regularizacije usklađivanja prototipa. U tablici 5.1 mogu se primjetiti znatno lošiji rezultati od Wang et al. (2019) i Zhang et al. (2018) za *1-way 1-shot* eksperiment. Međutim, rezultati našeg model za *1-way 5-shot* eksperiment, iako još uvijek lošiji od Wang et al. (2019), znatno su bolji od Zhang et al. (2018). Iz navedenog se može zaključiti da povećanje potpornog skupa znatno više doprinosi PANet arhitekturi nego SG-One arhitekturi. Isto se može zaključiti iz razlike mean-IoU rezul-

tata *1-way 5-shot* i *1-way 1-shot* segmentacije. Mean-IoU PANet ResNet-18 modela poveća se za 9.6, PANet modela iz Wang et al. (2019) za 7.6, a ostalih modela za 3.1 ili manje. Ovo upućuje na uspješnost PANet arhitekture u izvlačenju znanja iz potpornog skupa. Mogući razlog velikog poboljšanja u razultatu prilikom povećanja potpornog skupa je izvlačenje boljih prototipova. Povećanjem potpornog skupa očito je da se povećao i broj instanci promatranih klasa. Kako PANet model uprosječuje prototip klase po slikama, tako povećanjem broja slika i instanci klase, model uspijeva izvući smislenije značajke koje bolje generaliziraju.

Tablica 5.1: Mean-IoU rezultati ResNet-18 PANet modela bez usklađivanja prototipa i modela iz drugih radova za zadatak *1-way 1-shot* semantičke segmentacije na skupu podataka PASCAL VOC.

Metoda	1-way 1-shot					Parametri
	set-1	set-2	set-3	set-4	Prosjek	
OSLM (Shaban et al., 2017)	33.6	55.3	40.9	33.5	40.8	272.6M
co-FCN (Rakelly et al., 2018)	36.7	50.6	44.9	32.4	41.1	34.2M
SG-One (Zhang et al., 2018)	40.2	58.4	48.4	38.4	46.3	19.0M
PANet (Wang et al., 2019)	42.3	58.0	51.1	41.2	48.1	14.7M
PANet-ResNet18 bez PAR	37.5	50.3	44.1	38.0	42.5	**11.2M**

Tablica 5.2: Mean-IoU rezultati ResNet-18 PANet modela bez usklađivanja prototipa i modela iz drugih radova za zadatak *1-way 5-shot* semantičke segmentacije na skupu podataka PASCAL VOC.

Metoda	1-way 5-shot					Parametri
	set-1	set-2	set-3	set-4	Prosjek	
OSLM (Shaban et al., 2017)	35.9	58.1	42.7	39.1	43.9	272.6M
co-FCN (Rakelly et al., 2018)	37.5	50.0	44.1	33.9	41.4	34.2M
SG-One (Zhang et al., 2018)	41.9	58.6	48.6	39.4	47.1	19.0M
PANet (Wang et al., 2019)	51.8	64.6	59.8	46.5	55.7	14.7M
PANet-ResNet18 bez PAR	47.8	61.6	52.5	46.5	52.1	**11.2M**

5.2. Eksperimenti s regularizacijom usklađivanja prototipa

Jedna od zanimljivih ideja koju su autori PANet arhitekture predložili je regularizacija usklađivanja prototipa zamjenom potpornog skupa i skupa upita. U ovom dijelu provesti ćemo analizu korisnosti navedene regularizacije.

U tablici 5.4 nalaze se rezultati PANet modela s dva različita ekstraktora značajki. Za oba ekstraktora značajki proveli smo *1-way 1-shot* segmentaciju sa i bez regularizacije usklađivanja prototipa. Usklađivanje prototipa prosječno poboljšava mean-IoU rezultat za 0.8 kod ResNet18 modela, a za 0.5 kod ResNet34 modela. Ako promatramo rezultate u ovisnosti o klasama nad kojim se provodila evaluacija, usklađivanje prototipa doprinosi točnosti modela od približno 0 do 1.6. U tablici 5.3 nalaze se vremena treniranja navedenih modela. Utjecaj regularizacije na vrijeme treniranja je zanemarivo. Također, graf gubitka na slici 5.1 pokazuje da usklađivanje prototipa, osim što poboljšava točnost modela, doprinosi bržoj konvergenciji.

Tablica 5.3: Vrijeme treniranja PANet modela

Model	Vrijeme treniranja
ResNet18 bez PAR	4 sata i 36 min
ResNet18 sa PAR	4 sata i 48 min
ResNet34 bez PAR	4 sata i 51 min
ResNet34 sa PAR	4 sata i 45 min

Konačno, može se zaključiti da regularizacija usklađivanjem prototipa podiže točnost modela i doprinosi bržoj konvergenciji uz nikakvo ili zanemarivo povećanje vremena treniranja.

Tablica 5.4: Mean-IoU rezultati PANet modela nad značajkama ResNet18 i ResNet34 sa i bez regularizacije usklađivanja prototipa.

Model	Koristi PAR?	1-way 1-shot				
		set-1	set-2	set-3	set-4	Prosjek
ResNet18	Ne	37.5	50.3	44.1	38.0	42.5
	Da	38.1	51.4	45.7	38.1	43.3
ResNet34	Ne	40.2	53.4	46.4	40.7	45.2
	Da	40.6	54.7	46.7	40.7	45.7

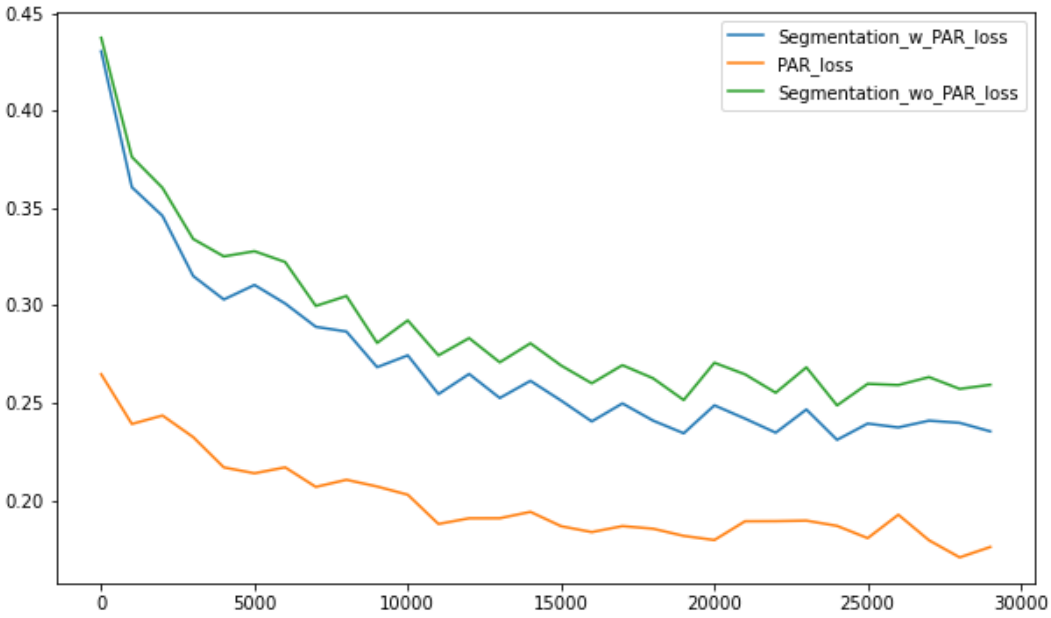
Tablica 5.5: Binary-IoU rezultati PANet modela nad značajkama ResNet18 i ResNet34 sa i bez regularizacije usklađivanja prototipa.

Model	Koristi PAR?	1-way 1-shot				
		set 1	set 2	set 3	set 4	Prosjek
Resnet18	Ne	63.7	68.0	61.7	59.1	63.1
	Da	63.8	68.8	62.8	58.9	63.6
Resnet34	Ne	65.0	70.0	62.7	61.3	64.8
	Da	65.2	70.6	62.9	60.8	64.9

5.3. Predikcije

Na stranicama koje slijede nalaze se slike predikcija *1-way 5-shot* modela s ResNet18 *feature extractorom* i bez regularizacije usklađivanja prototipa. Iz predikcija se može zaključiti da naš model poprilično dobro pogađa generalno područje objekta. Model najviše griješi na rubnim djelovima objekta koje je teže prepoznati iz različitih kutova kao što su krila aviona na slici 5.2.

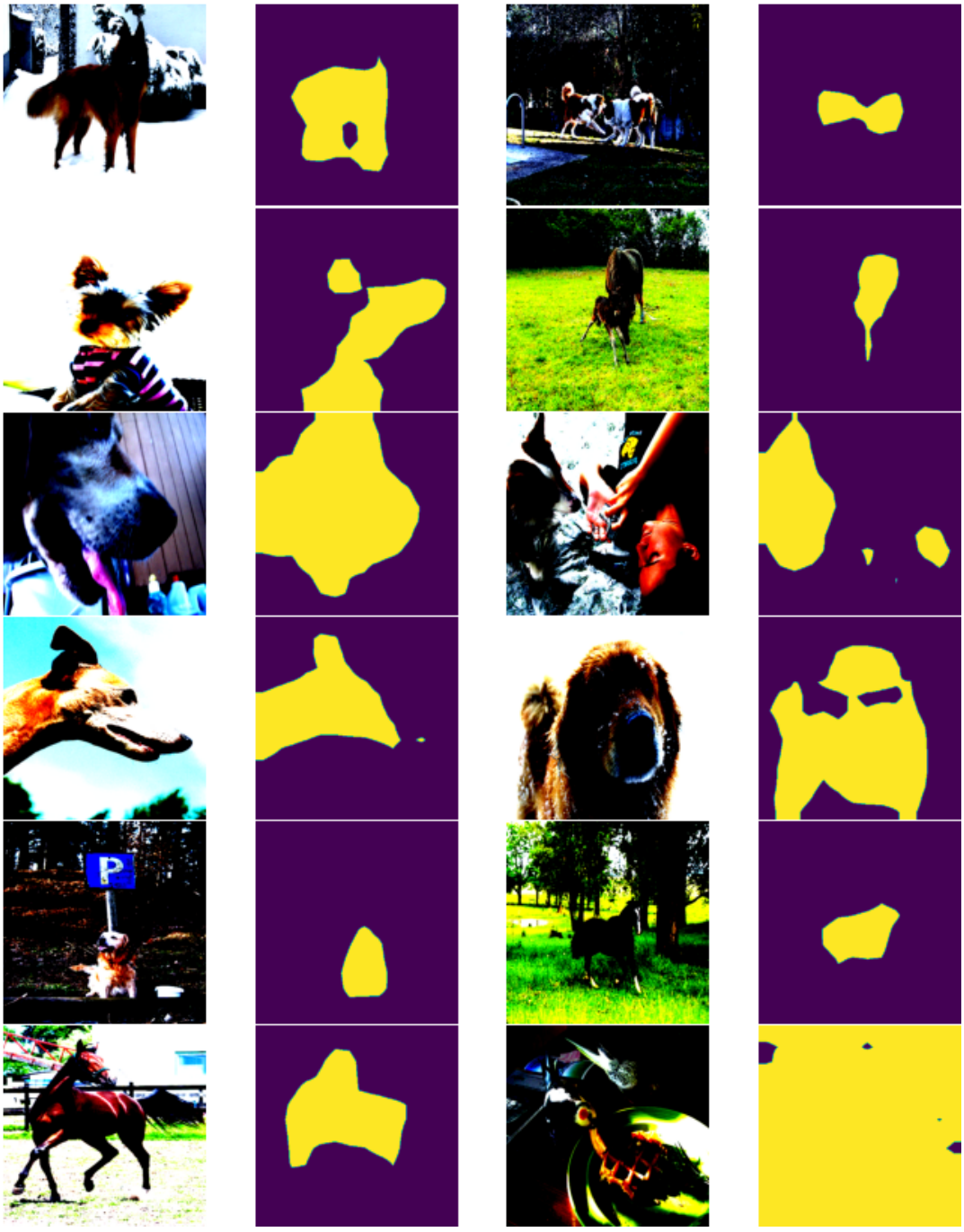
Promatrajući sliku 5.2 zaključujemo da model prepoznaje trup aviona, ali nemože prepoznati rubne dijelove kao što su propeleri i krila. Na primjerima mačaka i autobusa na slici 5.3 primjećujemo da model predviđa neprirodne rupe u objektima. Smatramo da bi se ovaj problem mogao rješiti nekom vrstom postprocesiranja predikcija. Model ponavlja slične greške na primjerima sa slika 5.4 i 5.5. Neprirodne rupe možemo primjetiti na slikama pasa, a neprepoznavanje rubnih dijelova, u ovom primjeru nogu, na slikama konja.



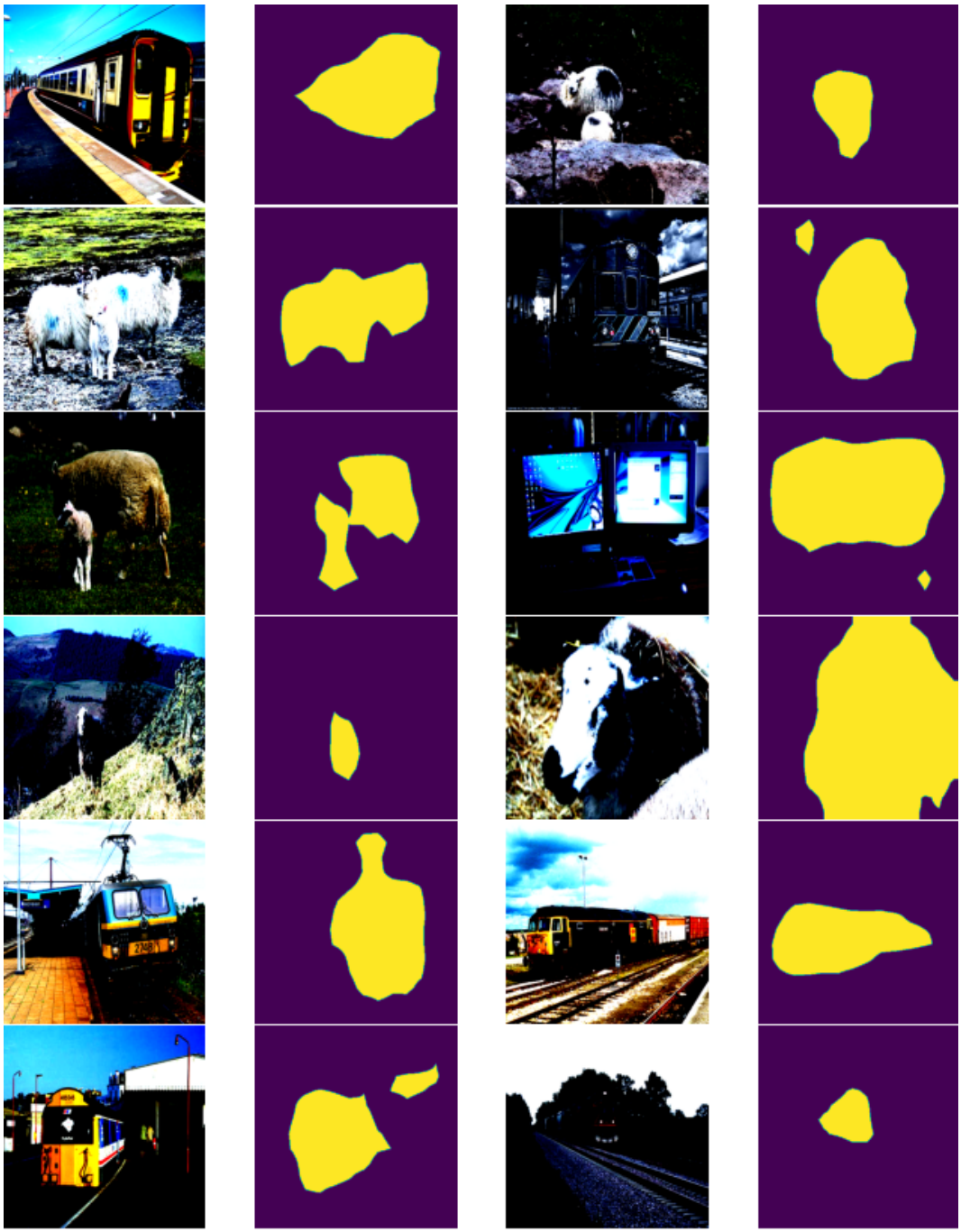
Slika 5.1: Graf gubitka za model sa i bez regularizacije usklađivanja prototipa. Model koji primjenjuje usklađivanje prototipa brže konvergira i postiže bolje rezultate.

Slika 5.2: Predikcije *1-way 5-shot* ResNet 18 modela bez regularizacije usklađivanja prototipa na prvom setu klasa

Slika 5.3: Predikcije *1-way 5-shot* ResNet 18 modela bez regularizacije usklađivanja prototipa na drugom setu klasa



Slika 5.4: Predikcije *1-way 5-shot* ResNet 18 modela bez regularizacije usklađivanja prototipa na trećem setu klasa



Slika 5.5: Predikcije *1-way 5-shot* ResNet 18 modela bez regularizacije usklađivanja prototipa na četvrtom setu klasa

6. Zaključak

Konvolucijske neuronske mreže zahtjevaju veliki broj slika za treniranje da bi postigle izvrsne rezultate. Ovaj rad razmatra metode koje tim modelima omogućavaju učenje na vrlo malenom broju primjera. Metode smo podijelili u tri skupine: algoritmi temeljeni na inicijalizaciji čiji je cilj što točnije postaviti parametre modela prije treniranja, algoritmi temeljeni na "halucinacijama" čiji je cilj povećati skup za treniranje i algoritmi koji uče metriku udaljenosti čiji je cilj naučiti uspoređivati slike. Fokus rada je na algoritmima koji uče metriku udaljenosti. Napravljen je pregled sijamskih neuronskih mreža kao jednog od takvih algoritama.

U okviru rada implementirali smo PANet arhitekturu za semantičku segmentaciju (Wang et al., 2019). PANet arhitektura može se podijeliti u dva dijela: izvlačenje prototipa i određivanje pripadnosti korespondencijskom metrikom. Cilj mreže je izvući robusne prototipove koji se uspoređuju s pikselima originalne slike. Naš model treniran u *1-way 1-shot* režimu postiže usporedive rezultate s prijašnjim radovima iako sadrži znatno manji broj parametara za treniranje. Međutim, isti model treniran u *1-way 5-shot* režimu nadmašuje rezultate prijašnjih radova što dokazuje veću sposobnost izvlačenja znanja iz potpornog skupa. U radu je objašnjena i regularizacija usklađivanjem prototipa čiji je cilj omogućiti tok informacija u oba smjera odnosno od potpornih slika do slika upita i obrnuto. Navedeno se postiže tako da prototipove izvlačimo iz slika upita, a segmentaciju radimo na potpornim slikama. Usklađivanje prototipa ubrzava konvergenciju i povećava točnost modela bez znatnog vremena treniranja. Daljni bi rad na ovu temu uključivao eksperimentiranje s drugim načinima naduzorkovanja, dodavanje modula za postprocesiranje predikcija ili evaluacija rezultata modela kad su klase korištene prilikom treniranja i klase nad kojima se radi predikcija iz različitih domena.

LITERATURA

Antreas Antoniou, Amos Storkey, i Harrison Edwards. Data augmentation generative adversarial networks. *Proceedings of the International Conference on Learning Representations Workshops (ICLR Workshops)*, 2018.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Sickinger, i Roopak Shah. Signature verification using a "siamese" time delay neural network. *Advances in Neural Information Processing Systems 6 (1993)*, 1993.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, i Jia-Bin Huang. A closer look at few-shot classification. *arXiv*, 1904.04232, 2019. URL `https://arxiv.org/pdf/1904.04232.pdf`.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980. URL `https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf`.

Bharath Hariharan i Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, i Jitendra Malik. Semantic contours from inverse detectors. *International Conference on Computer Vision*, 2011. URL `http://home.bharathh.info/pubs/pdfs/BharathICCV2011.pdf`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. *CoRR*, 1512.03385, 2015. URL `https://arxiv.org/abs/1512.03385`.

Jeremy Jordan. An overview of semantic image segmentation. 2018. URL `https://www.jeremyjordan.me/semantic-segmentation`.

Gregory Koch, Richard Zemel, i Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. *Proceedings of the International Conference on Machine Learning Workshops (ICML Workshops)*, 2015.

Jonathan Long, Evan Shelhamer, i Trevor Darrell. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015. URL https://arxiv.org/pdf/1411.4038.pdf.

Maxime Oquab, Leon Bottou, Ivan Laptev, i Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alyosha Efros, i Sergey Levine. Conditional networks for few-shot semantic segmentation. *International Conference on Learning Representations(ICLR)*, 2018.

Sachin Ravi i Hugo Larochelle. Optimization as a model for few-shot learning. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/pdf?id=rJY0-Kcll.

Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, i Byron Boots. One-shot learning for semantic segmentation. *arXiv*, 1709.03410, 2017. URL https://arxiv.org/pdf/1709.03410.pdf.

Vedran Vukotić. Raspoznavanje objekata dubokim neuronskim mrežama. Magistarski rad, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2014.

Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, i Jiashi Feng. Panet: Few-shot image semantic segmentation with prototype alignment. *arXiv*, 1411.4038, 2019. URL https://arxiv.org/pdf/1908.06391.pdf.

Xiaolin Zhang, Yunchao Wei, Yi Yang, i Thomas Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *arXiv*, 1810.09091, 2018. URL https://arxiv.org/pdf/1810.09091.pdf.

Sažetak

Konvolucijske neuronske mreže postižu izvrsne rezultate u zadacima klasifikacije i semantičke segmentacije slika, ali za to im je potreban veliki skup anotiranih slika za učenje. U ovom radu razmotreni su različiti pristupi učenja konvolucijskih mreža s malim brojem anotiranih primjera. Implementirana je PANet arhitektura za semantičku segmentaciju. PANet model iz malog broja slika za učenje izvlači prototipove klasa koje uspoređuje s pikselima slike upita pomoću neparametarskog računanja udaljenosti. Model postiže usporedive rezultate s prijašnjim radovima kada ima samo jednu potpornu sliku, a nadmašuje neke od njih kad se poveća potporni skup. Regularizacija usklađivanjem prototipa ubrzava konvergenciju i poboljšava točnost modela bez znatnog povećanja vremena treniranja.

Ključne riječi: računalni vid, konvolucijske neuronske mreže, učenje s malim brojem primjera, semantička segmentacija

Few-shot learning for discriminative convolutional models

Abstract

Convolutional neural networks achieve state-of-the-art results in image classification and semantic segmentation tasks, but they require an enormous amount of training images. This thesis examines various approaches in training convolutional neural networks when only a few annotated examples are available. We implemented PANet architecture used for semantic segmentation. PANet model extracts a prototype for each class from the support set and performs segmentation using non-parametric distance calculation. Model achieves comparable results with previous works when training in one-shot regime, but outperforms those works when support set is increased. Prototype alignment regularization speeds up model's convergence and increases accuracy without a significant increase in model's training time.

Keywords: computer vision, convolutional neural networks, few-shot learning, semantic segmentation