Polunadzirana klasifikacija slika

Tehnička dokumentacija, Verzija 2.0

Studentski tim: Bruno Ćorić
Dario Deković
Anja Delić
Antonio Ilinović
Luka Merćep
Marko Minđek
Dario Oreč
Lovro Rabuzin
Jakov Rukavina
Bernard Spiegl

Asistenti: Matej Grcić
Ivan Grubišić

Nastavnik: prof. dr. sc. Siniša Šegvić

Sadržaj

1. Sažetak

U ovom projektu bavili smo se polunadziranim učenjem modela za klasifikaciju slika, to jest metodama poboljšavanja generalizacijske točnosti tog modela. Skup podataka koji smo koristili za provjeravanje uspješnosti našeg modela dobro je poznati skup podataka CIFAR-10. Korištenim metodama proširivali smo osnovni model koji se sastoji od dva konvolucijska sloja i dva potpuno povezana sloja. Umnožavanjem podataka i validiranjem hiperparametara, postigli smo veću klasifikacijsku točnost bez promjene arhitekture modela. Umnožavanje podataka rezultiralo je povećanjem točnosti za otprilike 3 postotna boda u odnosu na osnovni model s najboljom kombinacijom hiperparametara. Uvođenjem normalizacije po grupi i rezidualnih modela, postigli smo dramatično veću točnost u usporedbi s osnovnim modelom (poboljšanje od otprilike 9 postotnih bodova). Kako bismo povećali točnost s manjim brojem označenih podataka, koristili smo pseudooznačavanje i učenje konzistencijom. Pseudooznačavanje je rezultiralo povećanjem točnosti od 3 do 4 postotna boda u odnosu na učenje samo na smanjenom skupu označenih slika. Nasuprot tome učenje konzistencijom rezultiralo je povećanjem točnosti od 0.06 postotnih bodova za učenje na 4000 označenih slika u odnosu na učenje na samo 4000 označenih slika.

2. Uvod

Vjerojatno najbitnije ljudsko osjetilo je vid. Naviknuti smo svaki dan većinu informacija dobivati upravo kroz osjetilo vida. Za ljude, to je trivijalan problem, čak i mala djeca znaju prepoznavati lica i razlučiti lica svojih roditelja u masi drugih ljudi. Nasuprot tome, analiza slike koja ljudima dolazi prirodno vrlo je složen problem za računala.

Računalni je vid područje umjetne inteligencije koje se bavi računalnom ekstrakcijom informacije iz slika na način sličan ljudima. Problemi kojima se bavi računalni vid razni su i uključuju detekciju objekata, semantičku segmentaciju, klasifikaciju predmeta i mnoge druge.

U ovom projektu, bavimo se klasifikacijom slika, to jest, pridjeljivanjem kategorija slici na temelju toga što se nalazi na slici. Budući da je klasifikacija slika već dobro poznat klasičan problem u analizi multimedijskih sadržaja, u ovom projektu primijeniti ćemo niz već istraženih metoda te vidjeti kakve rezultate one daju pri klasifikaciji dosad neviđenih slika.

Kao temelj ovog projekta, za klasifikaciju slika koristimo konvolucijske neuronske mreže koje se pokazuju učinkovitima pri klasifikaciji slika, ali i drugim problemima u računalnom vidu.
Arhitektura mreže koja služi kao baza za ostale postupke kojima se bavimo prikazana je u poglavlju 4.
U poglavljima 6, 7 i 8 opisane su metode koje koristimo za povećavanje točnosti pri učenju samo s označenim podacima.
U poglavljima 9 i 10, opisane su metode polunadziranog učenja, to jest korištenja i označenih i neoznačenih podataka kako bismo povećali točnost našeg modela. Polunadzirano je učenje važno jer može povećati točnost čak i ako nemamo označene podatke. Iako na ovom projektu imamo "školski" skup podataka gdje su sve slike označene, i dalje je vrijedno naučiti raditi s neoznačenim podatcima.

3. Skupovi podataka i tehnologije

3.1 CIFAR-10

CIFAR-10 skup je slika koji se često koristi za treniranje i evaluaciju algoritama računalnog vida. Skup se sastoji od 60000 32x32 slika u boji koje su ravnomjerno podijeljene u 10 različitih klasa. Klase koje se nalaze u skupu slika su: automobil, kamion, brod, zrakoplov, jelen, pas, mačka, žaba, konj i ptica. Skup slika organiziran je u dva podskupa od kojih jedan sadrži 50000 slika i koristi se za treniranje, a drugi sadrži 10000 slika koje se koriste za testiranje. U svakom se podskupu nalazi jednak broj slika iz svake klase.

3.2 Python

Python je programski jezik visoke razine i opće namjene. Zbog svojih brojnih biblioteka koje nadopunjavaju njegove osnovne funkcionalnosti jedan je od najkorištenijih jezika za probleme strojnog učenja i računalnog vida. Za potrebe našeg projekta mi smo koristili sljedeće biblioteke:

3.2.1 PyTorch

PyTorch je biblioteka otvorenog koda za strojno učenje zasnovana na biblioteci Torch za programski jezik Lua. Biblioteka omogućava rad s tenzorima uz značajno ubrzanje zbog integracije s grafičkim procesorom i jednostavno modeliranje dubokih neuronskih mreža. Zbog ovih karakteristika smo izabrali biblioteku Pytorch za korištenje na projektu te se veliki dio našeg programa bazira na funkcionalnostima koje nam ona pruža [1].

3.2.2 NumPy

NumPy je biblioteka za numeričko računanje i operacije s vektorima i matricama u Pythonu. Često se koristi za razne probleme strojnog učenja dok smo mi koristili maleni dio njenih funkcionalnosti, primarno za evaluaciju performansi našeg

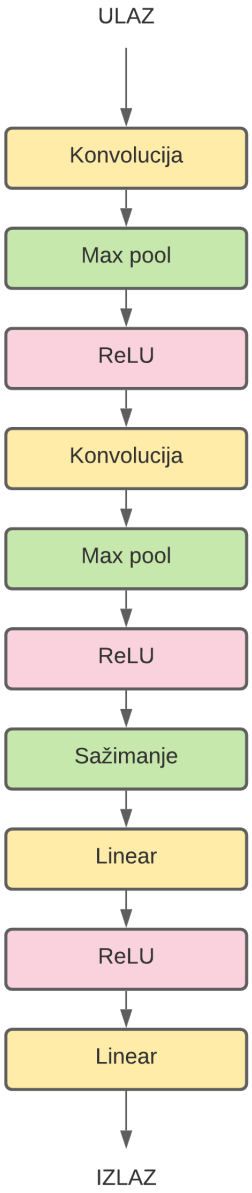
modela.

3.2.3 Matplotlib

Matplotlib je biblioteka za crtanje u Pythonu i NumPyu. Pruža razne funkcionalnosti za iscrtavanje grafova i slika, a mi smo ju koristili za vizualizaciju preciznosti našeg modela, kretanje funkcije gubitka, itd.

4. Arhitektura osnovnog modela

Kao osnovu za primjenjivanje ostalih metoda u projektu koristili smo jednostavnu arhitekturu iz [2], prikazanu na slici 4.1. Pritom smo koristili algoritam za optimizaciju Adam i u svakoj epohi smanjivali smo stopu učenja za fiksni faktor.



Slika 4.1: Arhitektura osnovnog modela

5. Usluga okoline za izvođenje u oblaku - Google Colab

5.1 Uvod

Google Colab proizvod je Google Research-a. Colab omogućava bilo kome da napiše i pokrene programe u Pythonu u svojem web pregledniku te je zbog toga posebno prikladan za strojno učenje, analizu podataka i edukaciju.
Interakciju s Colabom provodimo preko Jupyterovih bilježnica te za korištenje ne zahtjeva nikakvo postavljanje i/ili instalaciju te nudi besplatne računalne resurse uključujući i GPU.

5.2 Korištenje

Interakcija s Colabom započinje otvaranjem nove bilježnice. U bilježnicu vrlo je lagano dodavati kod i tekst. Za uređivanje teksta koristi se markdown.
Bilježnice se spremaju na osobni Google Drive ili mogu biti učitane s Githuba. Colab omogućava lagano dijeljenje bilježnica klikom na ”Share” gumb te odabirom određenih opcija kojima kontroliramo pristup bilježnici ljudima s kojima smo ju podijelili. Virtualni stroj i podatci koje smo koristili za pokretanje svoje bilježnice neće biti podijeljeni ostalim korisnicima.
Colab može pokretati programe u različitim izvršnim okolinama. Ako pokrenemo lokalnu izvršnu okolinu program će se izvoditi lokalno, a ako odaberemo Google-ovu izvršnu okolinu, program će se izvoditi u oblaku. Izvršnu okolinu možemo modificirati klikom na ”Runtime” gumb te tu možemo i promijeniti želimo li program izvoditi na GPU ili CPU klikom na ”Change Runtime type”.
Možemo koristiti podatke spremljene na naš Google Drive klikom na gumb ”Files” i zatim na ”Mount Drive”. Podatke možemo spremati na udaljeni runtime pa ih odmah i koristiti ili ih možemo spremati na Google Drive.

6. Umnožavanje podataka

6.1 Uvod

Kao što i samo ime kaže, umnožavanje podataka tehnika je koju koristimo za povećanje količine podataka za treniranje kako bismo postigli što veću točnost našega modela. To ostvarujemo modifikacijom osnovnih podataka odnosno stvaranjem sintetičkih podataka.

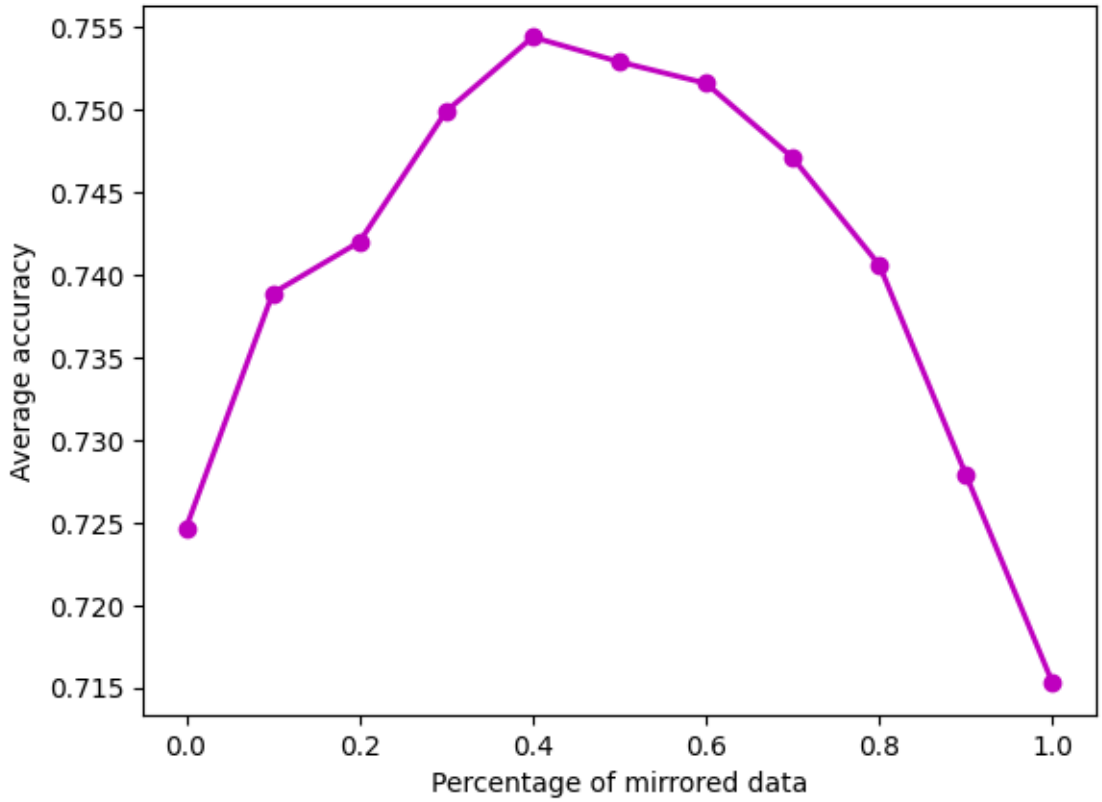
6.2 Metode umnožavanja podataka

Isprobali smo razne metode umnožavanja podataka od kojih je većina već implementirana u Torchvisionu. Samo su nam neke od tih metoda uspjele ostvariti poboljšanja. Neke od isprobanih metoda bile su: zrcaljenje, dodavanje šuma (npr. Gaussov), izrezivanje, brisanje dijelova slike, skaliranje, rotiranje i mnoge druge.
Najuspješnije od njih bile su zrcaljenje i afina transformacija, s time da se afina zapravo sastoji od transformacija poput pomaka, skaliranja, nagiba te rotacije, a izvodi se efikasno množenjem matrica. Obje transformacije ostvarile su poboljšanje točnosti za oko 2 do 3 postotna boda, s time da je njihova kombinacija poboljšala točnost za malo više od 3 postotna boda.

6.3 Korištenje

Transformacije se kombiniraju predajući listu odabranih transformacija konstruktoru klase torchvision.transforms.Compose. Poredak u listi bitan je najviše zato što različite transformacije kao ulaz primaju različite formate slika. Tako na primjer normalizaciju slika obično ostavljamo za kraj budući da se slika prvo mora pretvoriti u tenzor kako bi se transformacija provela.
Zaključili smo kako je umnožavanje podataka bolje provoditi samo na dijelu podataka kako bi se naša mreža učila i na originalnim slikama. Tako smo otkrili da je najbolje zrcaliti 40% do 50% podataka dok afinu transformaciju provodimo na

80% slika.



Slika 6.1: Utjecaj postotka zrcaljenih podataka na točnost modela

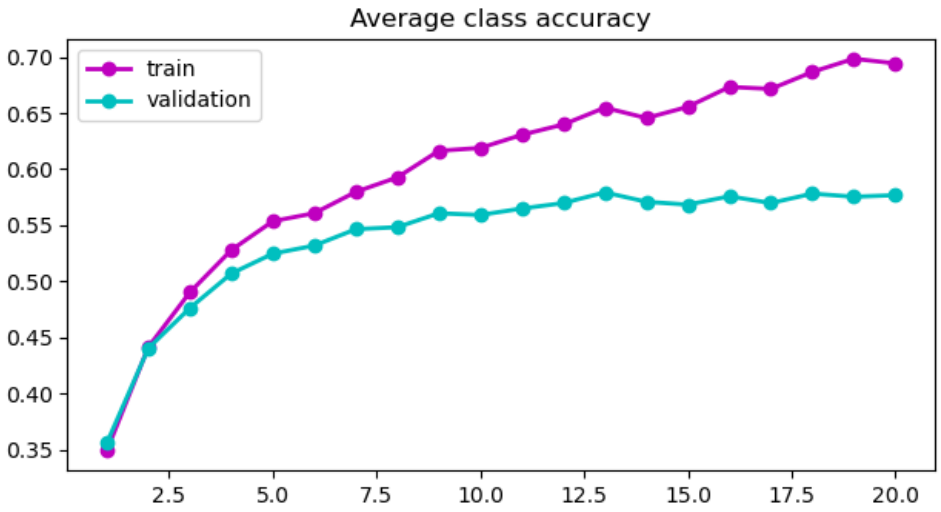
7. Validiranje hiperparametara

7.1 Uvod

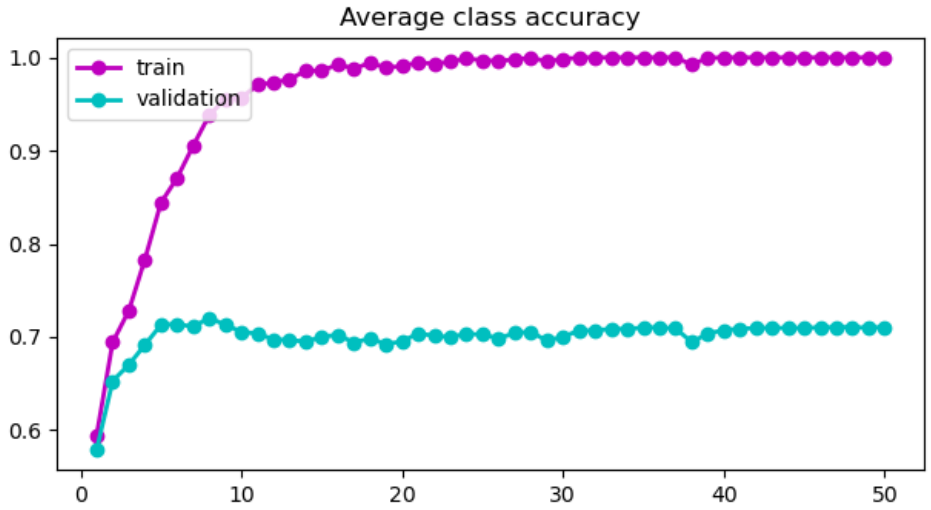
Hiperparametri u strojnom učenju [3] varijable su koje određuju način na koji će mreža biti trenirana te ih je potrebno pravilno uskladiti kako bi se postigle maksimalne performanse mreže. Konkretno, mi smo testirali utjecaj stope učenja, regularizacije, veličine mini-grupe i broja epoha.

7.2 Korak učenja

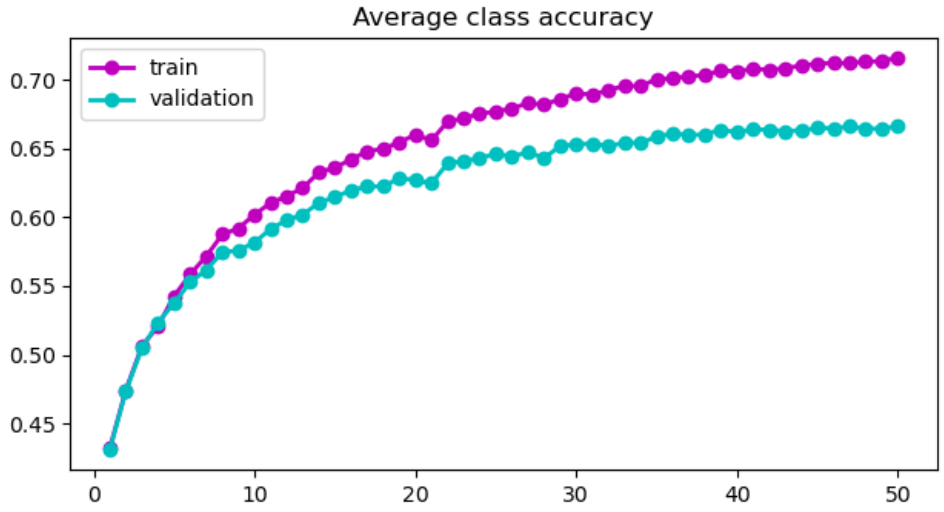
Korak učenja određuje pomak kojim se optimizator kreće u smjeru gradijenata. Samim time korak učenja jedan je od ključnih hiperparametara koje je potrebno odrediti neuronskoj mreži. Preveliki ili premali korak učenja može rezultirati jako lošim performansama mreže. Nadalje, sami korak učenja, odnosno njegova vrijednost koja rezultira zadovoljavajućim performansama mreže, ovisi, između ostalog, o korištenom optimizatoru i arhitekturi mreže. U našem istraživanju odredili smo korak učenja kroz metodu pokušaja i pogreške. Korištenjem te metode došli smo do zaključka kako korak učenja veličine 0.001 rezultira najboljim performansama našeg modela. Birali smo između sljedećih koraka učenja: 0.01, 0.001 i 0.0001. Točnost na skupu za testiranje bila je primarni kriterij izbora koraka učenja kao što je vidljivo iz grafova.



Slika 7.1: Stopa učenja 0.01



Slika 7.2: Stopa učenja 0.001



Slika 7.3: Stopa učenja 0.0001

7.3 Regularizacija

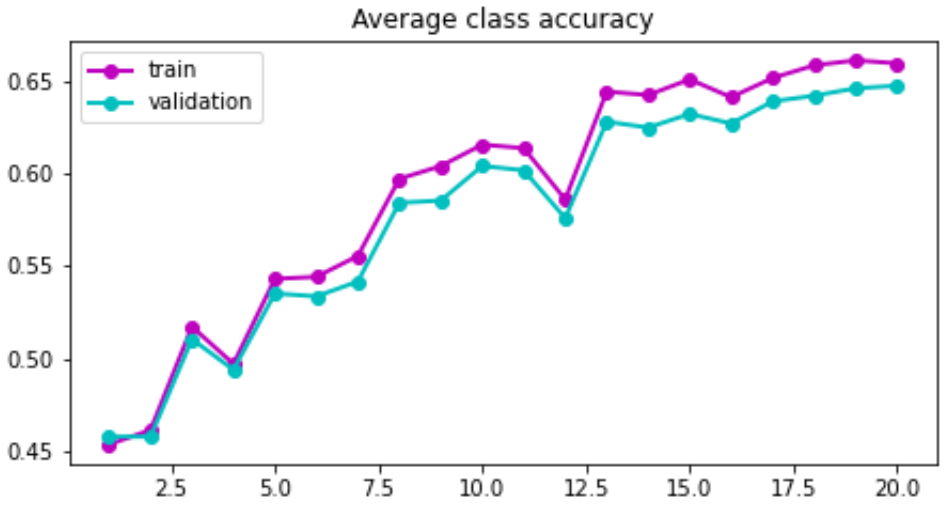
Regularizacija je svaka tehnika koja poboljšava generalizaciju bez da se nužno poboljša i točnost na skupu za učenje. Regularizacija može spriječiti prenaučenost i na taj način poboljšati točnost učenja modela. Kažemo da je model prenaučen kada postiže visoku točnost na skupu podataka za učenje, ali ne i na validacijskom skupu podataka. Koristili smo optimizator Adam [4] čija implementacija u Pytorchu u konstruktoru kao jedan od argumenata prima hiperparametar prigušenje težina (eng. *weight decay*). Iznos toga hiperparametra odredili smo iscrpnim pretraživanjem i zaključili da se najveća točnost na validacijskom skupu podataka, 73.65%, postiže kada je prigušenje težina 0.005 pri 50 epoha, veličini mini-grupe 100 i stopi učenja 0.001. Slika 7.4 prikazuje točnost na skupu za učenje i validacijskom skupu pri iznosu prigušenja težina od 0.005.

Slika 7.4: Točnost na skupu za učenje i validacijskom skupu pri iznosu prigušenja težina od 0.005

7.4 Veličina mini grupe

Tijekom treniranja mreže, u svakoj epohi prolazimo kroz čitavi skup podataka, no parametre modela mijenjamo više puta unutar jednog prolaska po podatcima. Jedna mini grupa (eng. *batch*) podskup je cijelog skupa podataka. Nakon prolaska po svakoj mini grupi, dobiveni rezultati uspoređuju se s oznakama zadanima u skupu podataka, računa se gradijent funkcije gubitka te se sukladno tome mijenjaju parametri mreže.

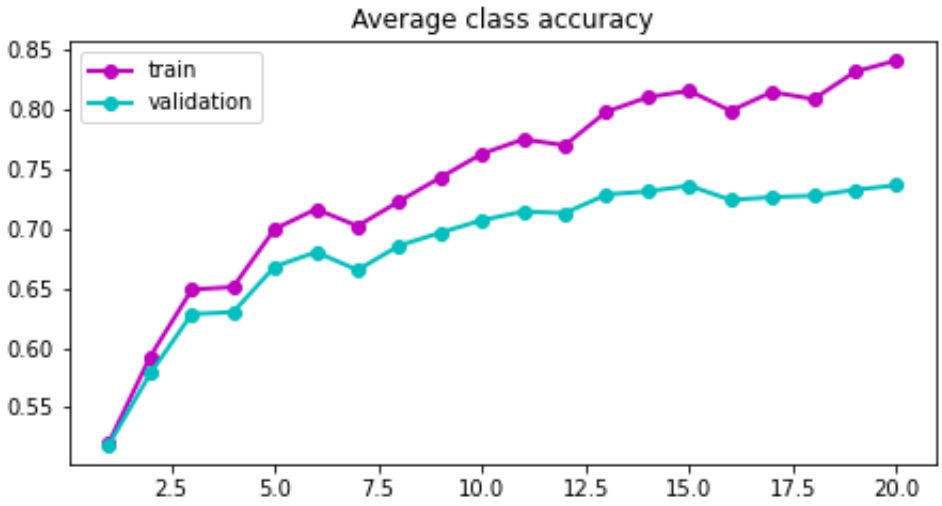
Veličina mini grupe može imati veliki utjecaj na stabilnost treniranja modela i konačnu postignutu točnost pa je važno odabrati odgovarajuću veličinu. Kako je gradijent funkcije gubitka statistička procjena, ako prije računanja gradijenta prođemo po većem broju podataka, procjena gradijenta bit će preciznija te je vjerojatnije da će parametri modela biti promijenjeni tako da povećavaju učinkovitost mreže [5]. Na slici 7.5 vidimo da je treniranje mreže nestabilno jer smo odabrali premalu veličinu mini grupe.



Slika 7.5: Točnost pri treniranju s veličinom mini grupe od 2 slike

Nasuprot tome, ako je veličina mini grupe prevelika, često je potrebno više epoha treniranja da bismo postigli željenu točnost. Također, ograničeni smo sklopovljem koje koristimo jer nam velike mini grupe nekada jednostavno neće stati u radnu memoriju uređaja na kojem treniramo model.

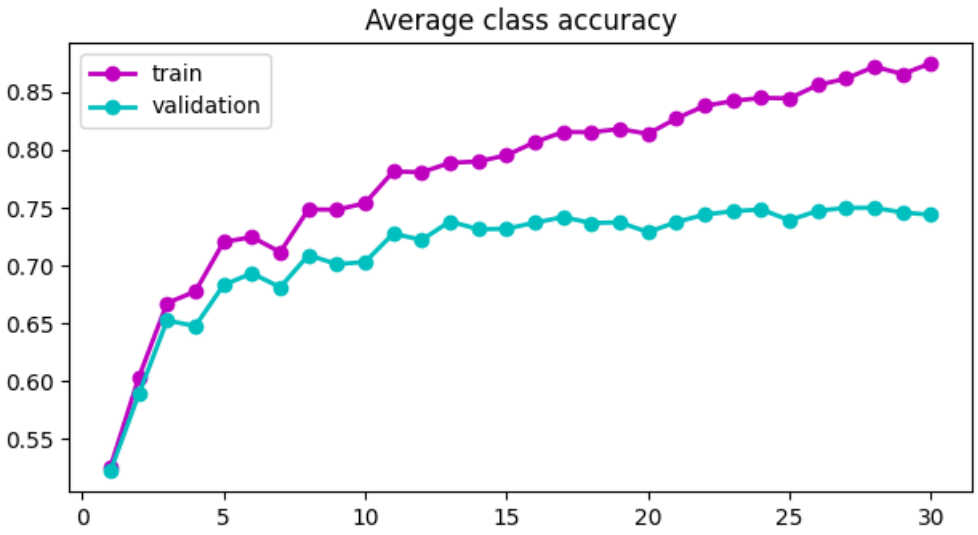
Za naš model pokazalo se da je odgovarajuća veličina mini grupe 100, iako su sve veličine između 50 i 200 davale vrlo slične rezultate. Na slici 7.6 vidimo da je treniranje stablinije za tu veličinu te da na kraju treniranja postižemo veću klasifikacijsku točnost.



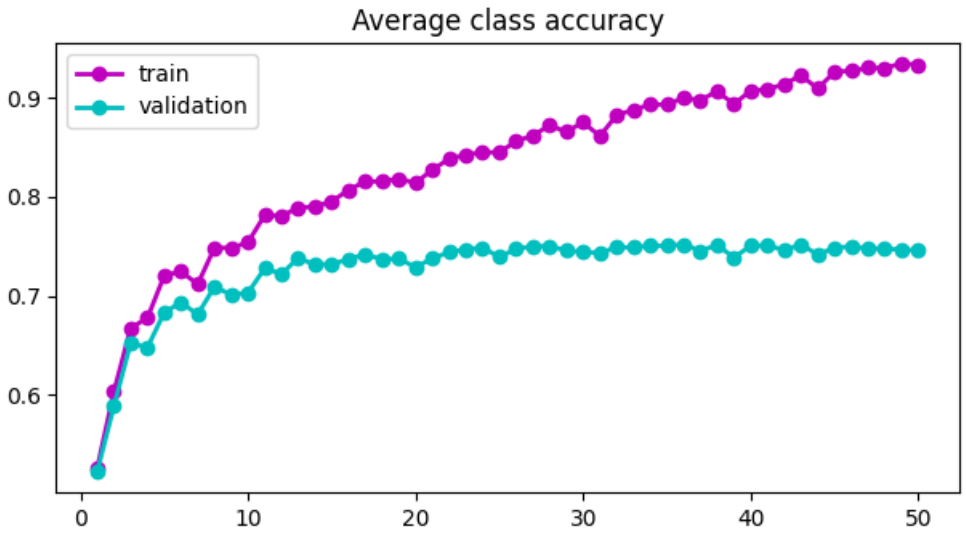
Slika 7.6: Točnost pri treniranju s veličinom mini grupe od 100 slika

7.5 Broj epoha

Zadavanjem broja epoha prilikom treniranja mreže mi zapravo govorimo koliko puta će se proći kroz skup podataka na kojem treniramo mrežu. Važno je odabrati ispravan broj epoha s obzirom da odabirom premalog broja dolazi do podučenosti (eng. *underfitting*) modela, a odabirom prevelikog broja može doći do prenaučenosti (eng. *overfitting*). Podučenost modela događa se jer se u kratkom roku parametri modela ne stižu dovoljno prilagoditi podacima, dok se prenaučenost događa jer se parametri modela mijenjaju sukladno rezultatima koji se postižu na skupu za treniranje, veći broj epoha, tj. veći broj prolazaka kroz taj skup, sve više prilagođava model isključivo tom skupu i smanjuje njegovu sposobnost za generalizaciju.
Za naš inicijalni model od dva konvolucijska i dva potpuno povezana sloja se pokazalo da nema smisla ići preko 30-ak epoha.
To možemo bolje vidjeti promotrimo li grafove koji prikazuju ovisnost preciznosti modela i broja epoha.



Slika 7.7: 30 epoha



Slika 7.8: 50 epoha

Uočljivo je da nakon 30 epoha više nema nikakvog rasta u preciznosti na setu za validaciju dok iz preciznosti koja se postiže na setu za treniranje vidimo da model teži k overfittanju.

7.6 Zaključak

Nakon testiranja svih hiperparametara došli smo do zaključka da za naš model najbolje odgovara sljedeća kombinacija:

- stopa učenja - 0.001
- regularizacija - 0.005
- veličina mini-grupe - 100
- broj epoha - 30

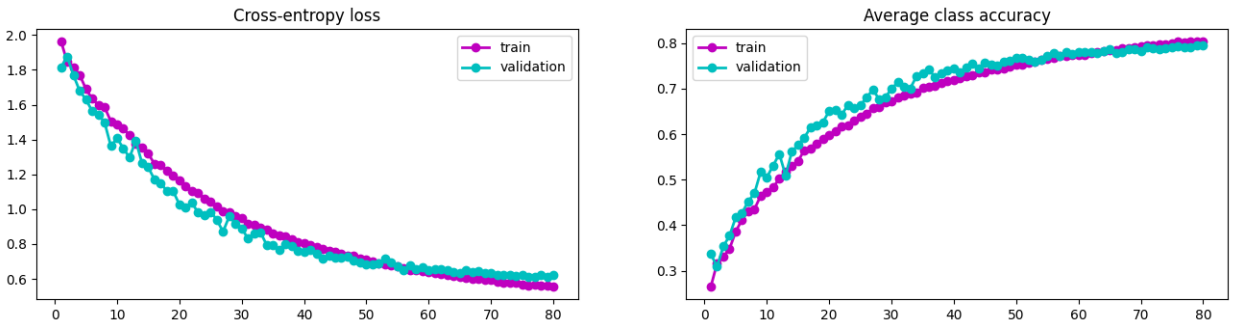
8. Uvođenje normalizacije po grupi i rezidualnih modela

8.1 Normalizacija po grupi

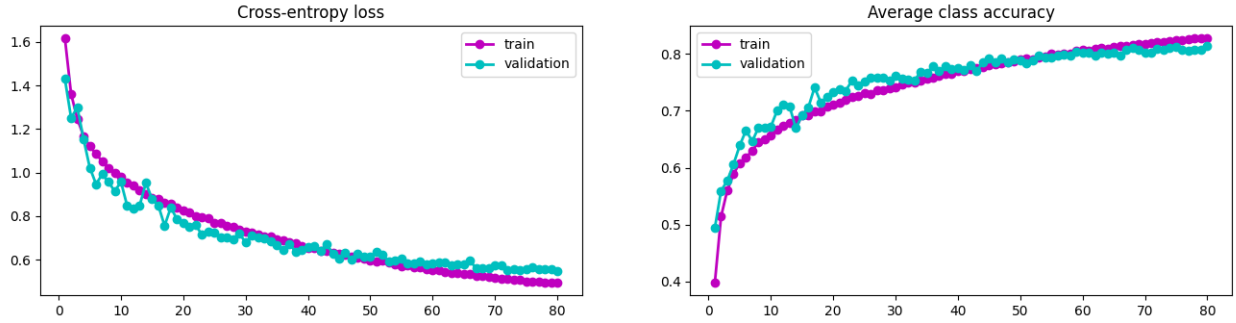
Normalizacija po grupi (eng. *batch normalization*) metoda je u dubokom učenju koja može znatno ubrzati i stabilizirati treniranje modela. U ovom kontekstu, normalizacija znači da se podacima iz skupa oduzima srednja vrijednost i rezultat dijeli sa standardnom devijacijom. Kao rezultat takav skup ima srednju vrijednost 0 i standardnu devijaciju 1. Normalizacija po grupi radi na način da normalizira izlaz iz sloja i time gladi pejzaž funkcije cilja [6]. Batchnorm poboljšava generalizaciju jer model u svakoj iteraciji uči na različito normalizairanim podatcima. Zbog toga teško dolazi do prenaučenosti.

8.2 Efekti uvođenja normalizacije po grupi

Uvođenjem normalizacije po grupi u naš osnovni model koji sadrži samo dva konvolucijska sloja zaključili smo kako ta metoda omogućava treniranje s mnogo većim korakom učenja te kako se sam postupak znatno ubrzava.
Konkretno, sada počinjemo učenje s korakom 0.1 dok smo prije dobivali najbolje rezultate s 10 puta manjom veličinom. Ovo rezultira mnogo strmijom krivuljom porasta točnosti u prvih nekoliko desetaka epoha što znači da puno brže uspijevamo naučiti naš model.



Slika 8.1: Jednostavan konvolucijski model, početni korak učenja 0.1, 80 epoha



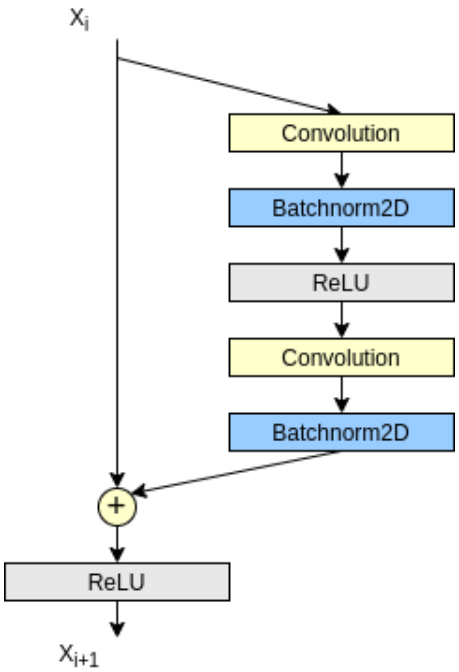
Slika 8.2: Jednostavan konvolucijski model s normalizacijom po grupi, početni korak učenja 0.1, 80 epoha

8.3 Rezidualni modeli

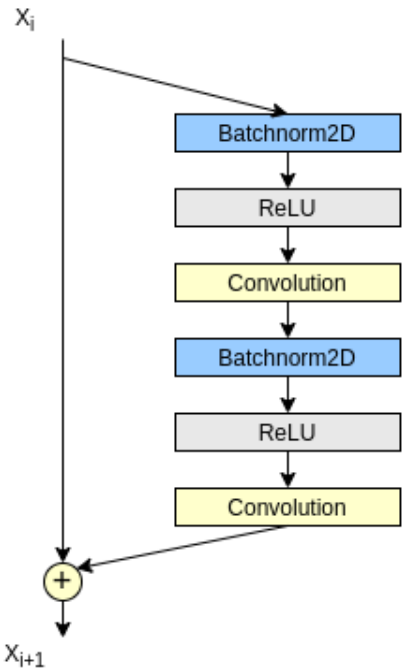
Rezidualne neuronske mreže su tip mreža koje osim direktno povezanih slojeva koji slijede jedan za drugim također sadrže i preskočne veze, tzv. *skip connections*, koje zbrajaju slojeve koji ne slijede izravno jedan iza drugog. Važnost takve arhitekture vidljiva je kod izrazito dubokih mreža. Naime, prilikom unatražnog prolaska (eng. *backprop*) težine svakog sloja se prilagođavaju u ovisnosti o gradijentu te ako mreža ne koristi normalizaciju po grupi, a sadrži previše slojeva gradijent može poprimiti izrazito male vrijednosti koje spriječavaju daljnje treniranje. Taj je problem kod rezidualnih mreža riješen upravo tim vezama koje preskaču pojedine slojeve mreže. Takve veze rade na način da uzimaju izlaz iz jednog sloja te ga dodaju na izlaz sloja koji se nalazi nekoliko slojeva kasnije.

Testirali smo nekoliko rezidualnih modela: ResNet 14, ResNet18, ResNet28, ResNet50, ResNet101, ResNet152. ResNet14, ResNet18 i ResNet28 su se pokazali najboljima, iako je i ResNet101 davao solidne rezultate unatoč njegovoj veličini. ResNet152 se pokazao kao najlošiji izbor, što je bilo i za očekivati uzimajući u obzir

njegovu veličinu koja je znatno prevelika za CIFAR-10. U testovima smo isprobali dvije vrste rezidualnih blokova:



Slika 8.3: Originalni rezidualni blok



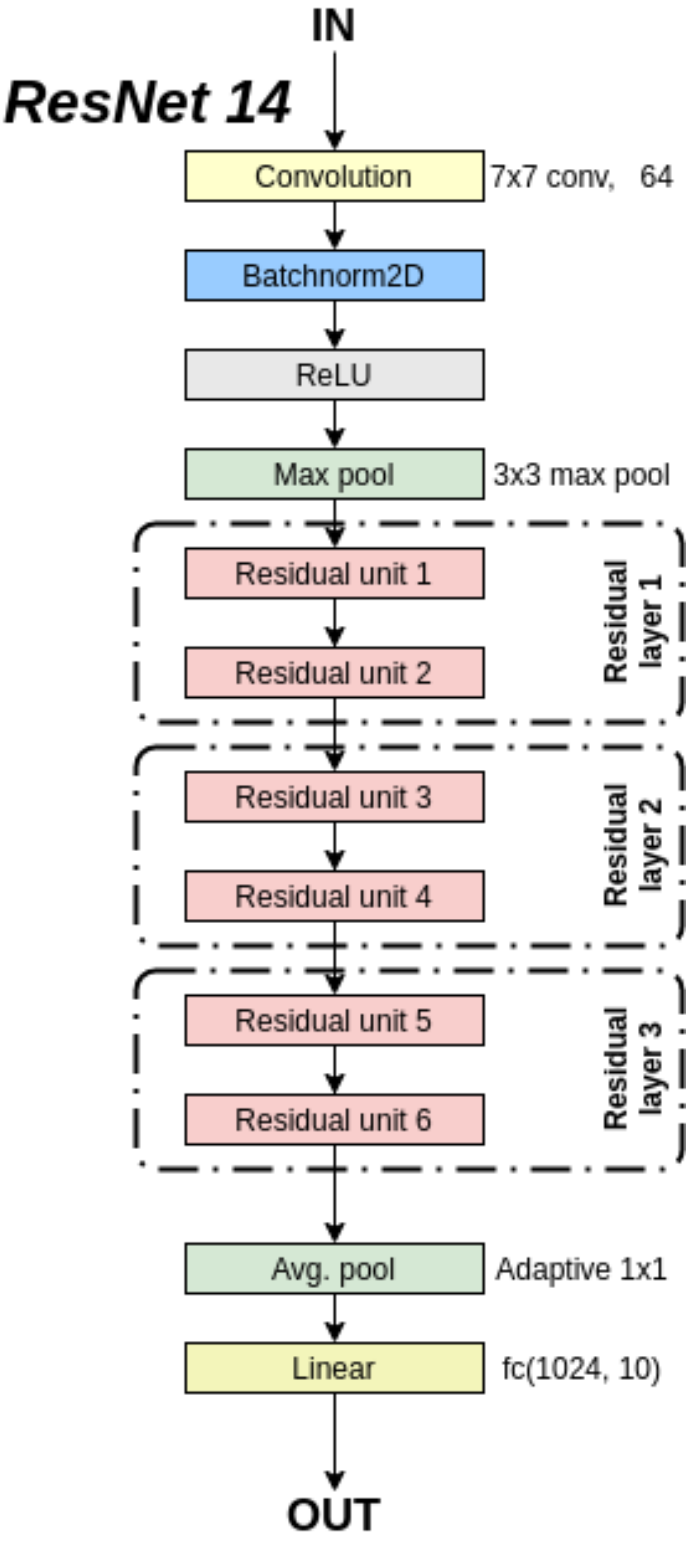
Slika 8.4: Rezidualni blok s predaktivacijskom normalizacijom po grupi

Oba rezidualna bloka sadrže po dva konvolucijska sloja, ali razlika je u redosljedu

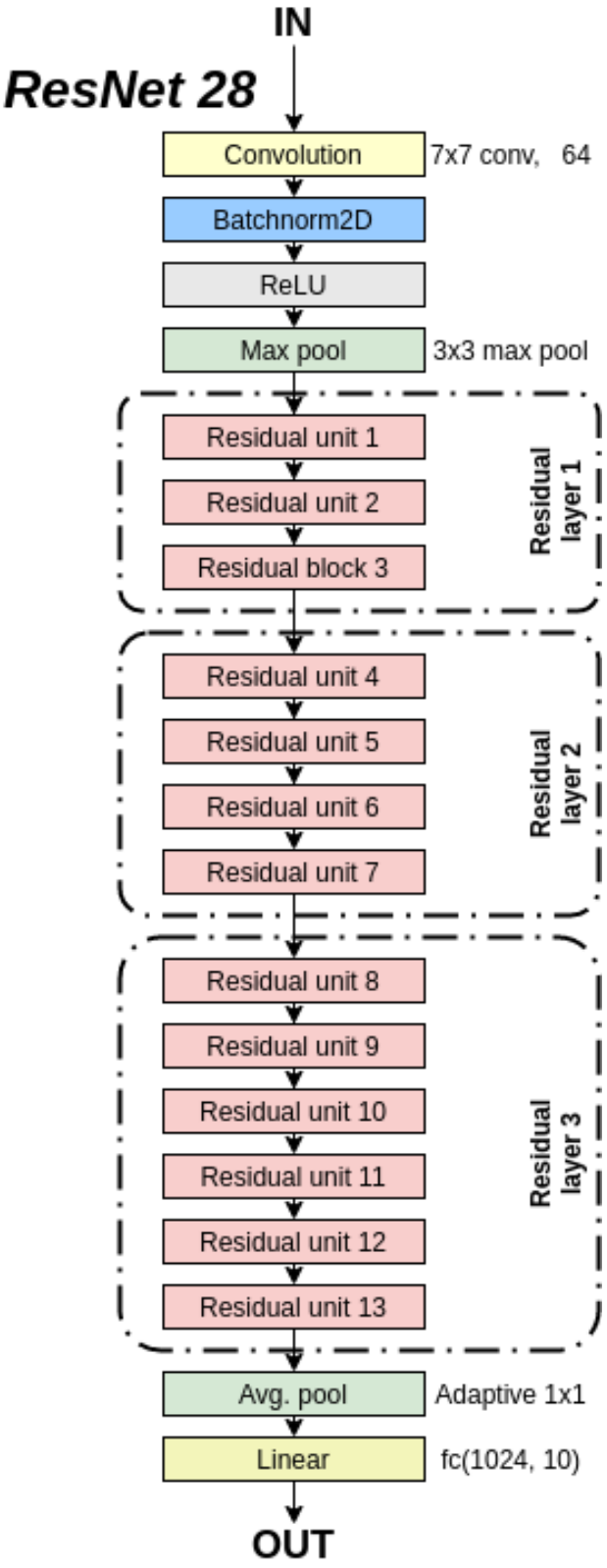
komponenti. Također, razlika je i u tzv. preskočnoj vezi (eng. *skip-connection*). U originalnom rezidualnom bloku, originalni ulaz se zbrajao sa izlaskom iz unutarnjih slojeva i zatim se dovodila na funkciju aktivacije. U rezidualnom bloku sa mapiranjem identiteta, ulaz je dobiven direktno kao zbroj ulaza i izlaza iz unutarnjih slojeva. Rezidualni modeli s predaktivacijskom normalizacijom po grupi pokazali su značajno bolje rezultate (poboljšanje 3-4 postotna boda na setu za testiranje).

8.3.1 Rezultati testiranja

Testirali smo sve gore navedene mreže, sa vrlo različitim skupom hiper-parametara, različitim optimizatorima i postupcima dinamičkog mijenjanja stope učenja. Najbolje smo rezultate dobili uz ResNet14, ResNet18 te ResNet28 koje su koristile blokove sa mapiranjem identiteta, što je otprilike bilo i za očekivati zbog intuicije koja nam govori da će za ovakav skup podataka biti dovoljna nešto manja mreža. Na svakom testiranju podaci su uvećani horizontalnim zrcaljenjem (40% slika) i afinom transformacijom (80% slika). Pokazalo se kako Adam optimizator može polučiti dobre rezultate koji nisu potpuno zavisni od parametara i konstrukcije mreže. S druge strane SGD optimizator može mrežu odvesti u dva potpuno različita pravca, ovisno o hiper-parametrima (najviše je osjetljiv na learning rate). Pažljivo kalibriranje parametara u paru sa SGD optimizatorom u pravilu daje bolje rezultate. Također smo u kombinaciji s *Adam* optimizatorom pokušali izostaviti uporabu normalizacije po grupi iz modela što je za rezultat onemogućilo njihovo treniranje (gubitak se nije smanjivao).
U testovima koji slijede korišten je SGD optimizator. Stopa učenja se svaku epohu množi sa *gamma = 0.99*. Batch size iznosi 100, a regularizacija 0.0005. Modeli su trenirani na 50 epoha.



Slika 8.5: ResNet14



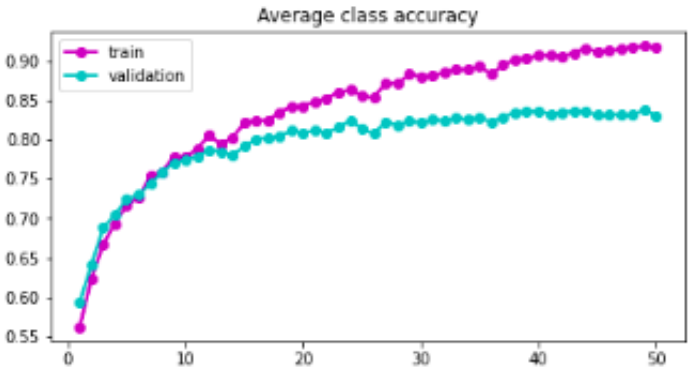
Slika 8.6: ResNet28

8.3.2 Test 1 (learning rate = 0.05)

ResNet 14 :

Točnost na podacima za treniranje = 91.90%

Točnost na podacima za testiranje = 83.75%

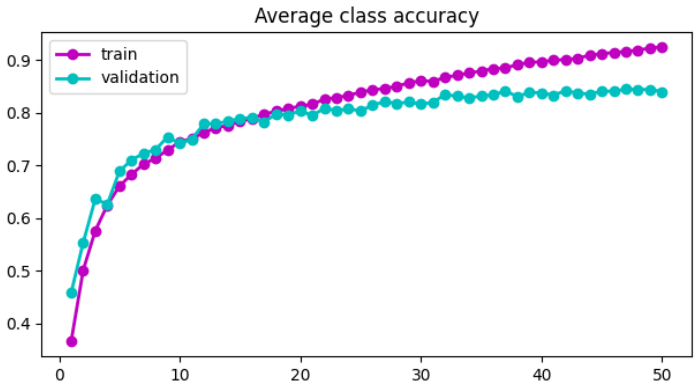


Slika 8.7: ResNet14 Točnost kroz epohe

ResNet 18 :

Točnost na podacima za treniranje = 92.54%

Točnost na podacima za testiranje = 84.51%

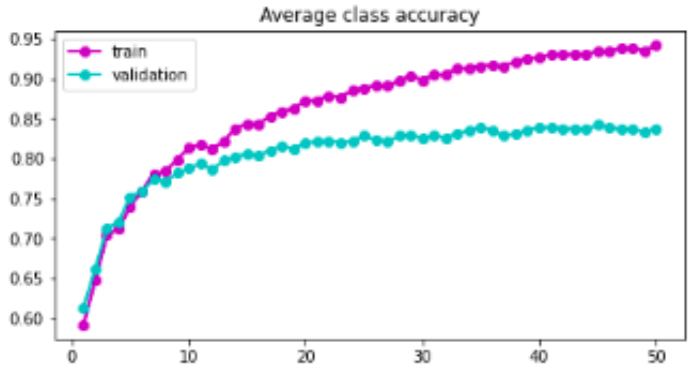


Slika 8.8: ResNet18 Točnost kroz epohe

ResNet 28 :

Točnost na podacima za treniranje = 93.47%

Točnost na podacima za testiranje = 84.23%



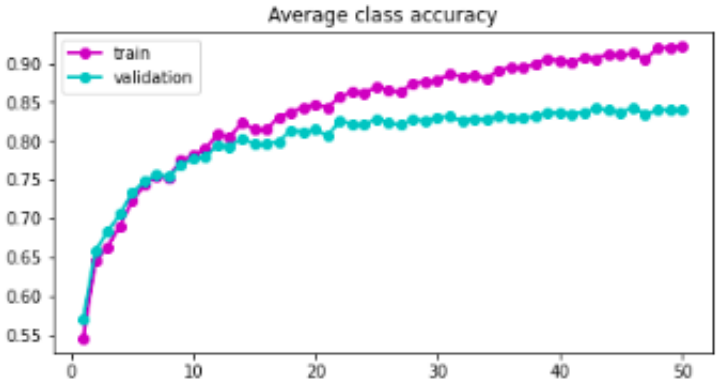
Slika 8.9: ResNet28 Točnost kroz epohe

8.3.3 Test 2 (learning rate = 0.1)

ResNet 14 :

Točnost na podacima za treniranje = 91.29%

Točnost na podacima za testiranje = 84.36%

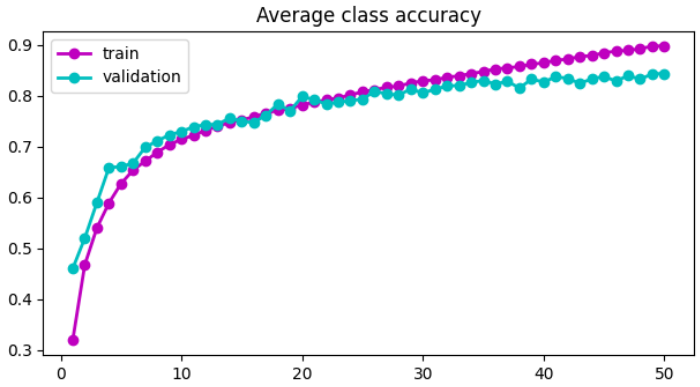


Slika 8.10: ResNet14 Točnost kroz epohe

ResNet 18 :

Točnost na podacima za treniranje = 89.73%

Točnost na podacima za testiranje = 84.32%

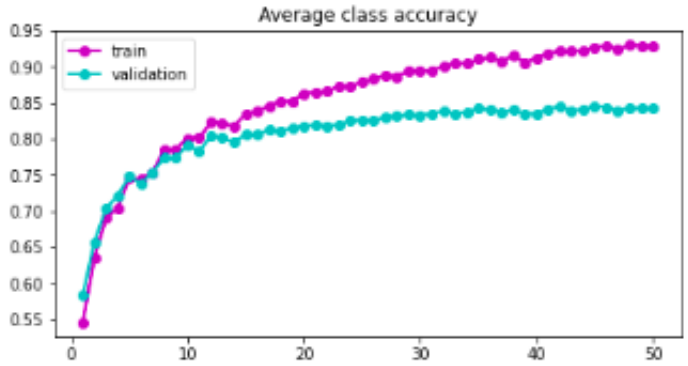


Slika 8.11: ResNet18 Točnost kroz epohe

ResNet 28 :

Točnost na podacima za treniranje = 92.21%

Točnost na podacima za testiranje = **84.60%**



Slika 8.12: ResNet28 Točnost kroz epohe

9. Polunadzirano učenje pseudooznačavanjem

9.1 Uvod

Polunadzirano učenje koristi mali skup označenih podataka i velik skup neoznačenih podataka. To je kombinacija nadziranog učenja koje koristi označene podatke i nenadziranog učenja koje koristi neoznačene podatke. Polunadzirano učenje važno je jer omogućuje korištenje neoznačenih podataka za učenje modela kada su označeni podatci skupi ili nedostupni.
Jedna od jednostavnih i efikasnih metoda polunadziranog učenja je pseudooznačavanje. U našem eksperimentu koristimo tehniku koja je predstavljena u članku [7]. Prvo određeni broj epoha učimo model na označenim podatcima. Zatim učenje nastavljamo na skupu i označenih i neoznačenih podataka. Neoznačenim podatcima na temelju prednaučenog modela pridjeljujemo pseudooznake koje odgovaraju predviđenoj klasi najveće vjerojatnosti. Takve pseudooznake koristimo kao da su prave, a postupak pseudooznačavanja ponavlja se nakon svake iteracije učenja.
Također, u članku [7] predlaže se računati gubitak na temelju gubitka izračunatog na označenim podatcima i gubitka izračunatog na neoznačenim podatcima prema jednadžbi (1):

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L(y'^m_i, f'^m_i) \tag{1}$$

gdje je n broj mini-batcheva u označenom skupu podataka, a n' u neoznačenom skupu podataka, f_i^m je izlaz m-tog uzorka iz označenog skupa, a y_i^m oznaka toga uzorka, f'^m_i je izlaz m-tog uzorka iz neoznačenog skupa, a y'^m_i pseoudooznaka toga uzorka. Funkcija $\alpha(t)$ kontrolira doprinos neoznačenih podataka ukupnom gubitku i računa se kao:

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1 \le t < T_2 \\ \alpha_f & T_2 \le t \end{cases} \tag{2}$$

9.2 Izvedba i rezultati

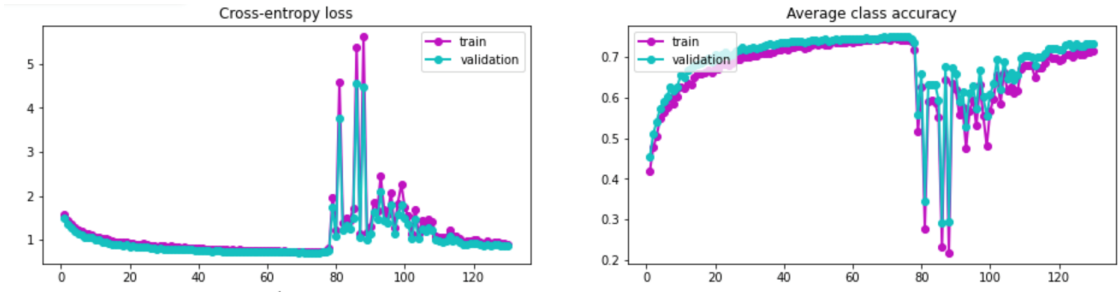
U našem eksperimentu proveli smo polunadzirano učenje pseudooznačavanjem s 250, 1000, 4000 i 50000 označenih podataka. Koristili smo konvolucijski model s dva konvolucijska i dva potpuno povezana sloja i batch normalizacijom. Koristili smo umnožavanje podataka kako je opisano u poglavlju 5 i hiperparametre navedene u poglavlju 6.6. Veličinu batcha označenih podataka smo postavili na 5 za slučaj kada učimo model na 250 označenih podataka, a u ostalim slučajevima na 20. Za svaki određeni broj označenih podataka model smo trenirali na grupama neoznačenih podataka veličine 200 i 500. Model smo trenirali na ukupno 130 epoha, a u funkciji $\alpha(t)$ koristili smo sljedeće hiperparametre: α_f=1 za slučaj kada model učimo na 250 označenih podataka, a inače α_f=3, T_1=75 i T_2=120. Rezultati eksperimenta prikazani su u tablici 9.1 Primjećujemo poboljšanje točnosti u svim slučajevima, osim za 250 označenih podataka i veličinu batcha neoznačenih podataka 200. Takav rezultat pripisujemo maloj veličini batcha neoznačenih podataka i pretpostavljamo da se rezultat mogao poboljšati prilagođavanjem hiperparametara. Također, primjećujemo da veći batch neoznačenih podataka daje veću i nadziranu i nenadziranu točnost.

Tablica 9.1: Rezultati polunadziranog učenja pseudooznačavanjem s 250, 1000, 4000 i 50000 označenih podataka za veličine batcha neoznačenih podataka 200 i 500. Prikazane su nadzirana i nenadzirana točnost, tj. točnost ostvarena u 75. epohi učenja i u 130.

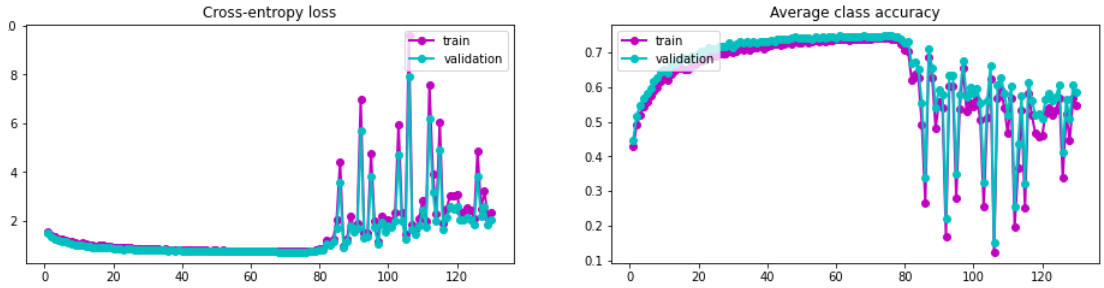
Broj označenih podataka	Batch size - neoznačeni	Nadzirana točnost [%]	Nenadzirana točnost [%]
250	200	38.6	37.98
	500	37.61	41.37
1000	200	50.84	51.39
	500	52.3	55.06
4000	200	65.3	68.26
	500	67.02	69.00
50000	-	79.29	-

Prilikom implementacije pseudooznačavanja susreli smo se s nekim problemima. U svrhu testiranja naše implementacije pseudooznačavanja učili smo model na 25000 označenih podataka. Dok još nismo bili prijmijenili batch noralizaciju, nakon uvođenja neoznačenih podataka, u 75. epohi točnost bi počela brzo padati, gubitak naglo rasti, a u matrici zabune svi primjeri bi se klasificirali u istu klasu.

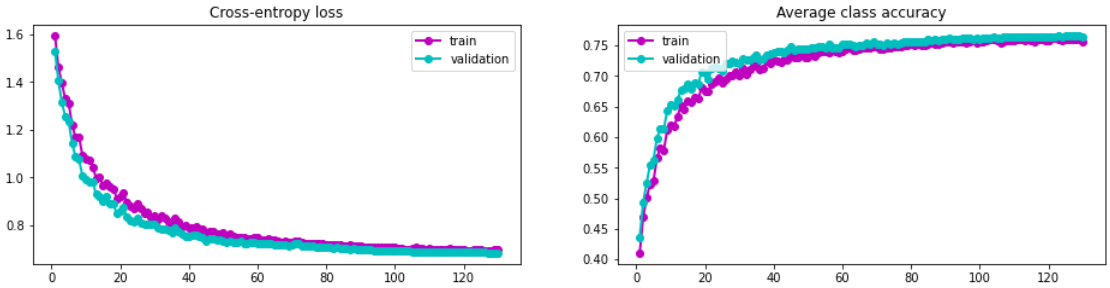
Ta situacija, tj. učenje modela na 25000 označenih je prikazana na slici 8.1. Profesor nam je sugerirao da napravimo sljedeće promjene: uvedemo normalizaciju po grupi, nakon 74. epohe poništimo Adamove momente i jednom prođemo kroz nove podatke bez učenja modela i s normalizacijom po grupi u trening načinu te da u 75. epohi smanjimo learning rate dva puta. Kad smo to primijenili, rezultati su odmah postali puno bolji.



Slika 9.1: Prikaz rezultata učenja na 25000 označenih podataka, bez normalizacije po grupi, smanjenja koraka učenja i resetiranja Adamovih momenata nakon 74.epohe



Slika 9.2: Prikaz rezultata učenja na 25000 označenih podataka nakon smanjenja koraka učenja i resetiranja Adamovih momenata



Slika 9.3: Prikaz rezultata učenja na 25000 označenih podataka nakon uvođenja normalizacije po grupi, smanjenja koraka učenja i resetiranja Adamovih momenata nakon 74. epohe

Slika 9.2 prikazuje rezultat učenja modela bez normalizacije po grupi na 25000 označenih podataka s veličinom batcha neoznačenih podataka 200 nakon smanjenja learning ratea i resetiranja Adamovih momenata nakon 74.epohe, a slika 9.3 prikazuje rezultat nakon što smo uveli normalizaciju po grupi. Vidimo da je uvođenje batch normalizacije riješilo naš problem.

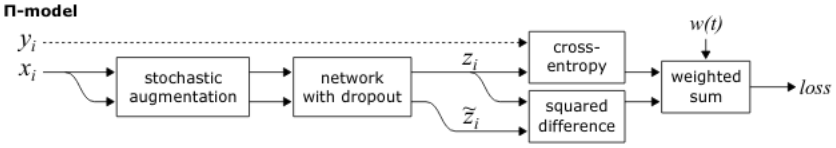
10. Polunadzirano učenje konzistencijom (Π-model)

10.1 Uvod

Označavanje podataka, u našem slučaju slika, jedan je od najskupljih procesa prilikom nadziranog učenja. Ta činjenica dovela je do razvoja polunadziranog učenja. Kao što mu i samo ime govori, polunadzirano učenje je tehnika učenja na malom skupu označenih i velikom skupu neoznačenih primjera. Postoji nekoliko izvedbi polunadziranog učenja, a mi smo se u ovom dijelu fokusirali na polunadzirano učenje konzistencijom. Preciznije, odlučili smo se za implementaciju učenja konzistencijom Π -modelom kao što je to opisano u radu [8].

10.2 Π-model

Osnova pretpostavka Π-modela je stohastičnost mreže i stohastičnost ulaznih podataka. Stohastičnost mreže postižemo dodavanjem dropout sloja u arhitekturu mreže. Dropout sloj nasumično, prilikom učenja, postavlja kanale između dva sloja na nulu. Time se osigurava različiti izlaz za dva ista ulazna podatka što je jedan od preduvjeta za uspješan rad Π-modela.

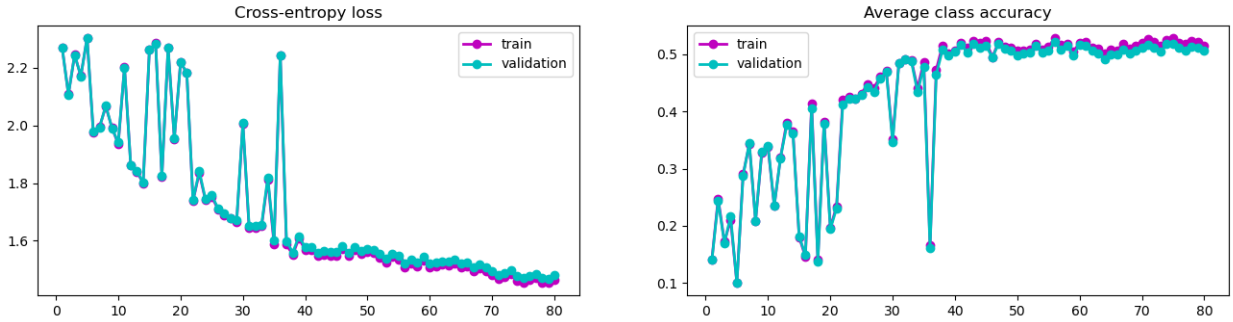


Slika 10.1: Skica Π-modela
[8]

Kao što je opisano u skici modela na slici 10.1 duplicirali smo ulazne mini-grupe te na njih odvojeno primijenili nasumične transformacije. Time smo postigli drugi preduvjet za ispravan rad Π-modela, stohastičnost ulaznih podataka.

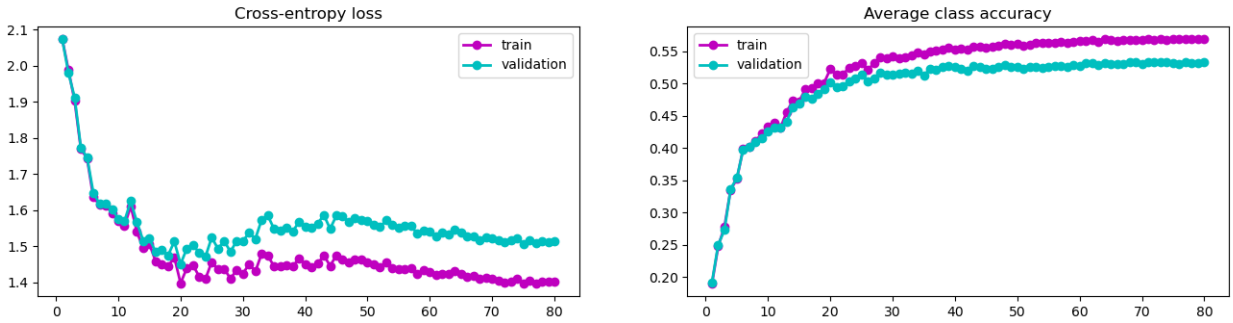
10.3 Priprema podataka

U prvom pokušaju implementacije Π-modela razdvojili smo označene i neoznačene podatke na dvije grupe. Takvim pristupom dobili smo zadovoljavajuće početne rezultate, ali izrazito neravnu površinu grafa gubitka i točnosti modela kao što je vidljivo iz slike 10.2.



Slika 10.2: Prikaz rezultata učenja na 4000 označenih podataka sa odvojenim označenim i neoznačenim podatcima

Ipak, puno uspješnijom metodom pokazalo se miješanje označenih i neoznačenih podataka prije učenja. Tim postupkom smo uspjeli značajno izgladiti površine prije spomenutih grafova.



Slika 10.3: Prikaz rezultata učenja na 4000 označenih podataka sa izmiješanim označenim i neoznačenim podatcima

10.4 Treniranje

Postupak treniranja detaljno je opisan u pseudokodu na slici 10.4. Važno je primijetiti kako funkcija gubitka ima dvije komponente. Nadzirana komponenta je klasični gubitak unakrsnom entropijom koji računamo samo za označene podatke,

a nenadzirana komponenta funkcije gubitka je srednji kvadrat razlike od dva izlaza iz mreže koji računamo za sve podatke. Tom komponentom funkcije gubitka zahtijevamo od mreže konzistentnost u predviđanju između dva izlaza. Naravno takvim zahtjevom ne postižemo puno ako ulazni podaci i sama mreža nisu stohastički. Nadalje, nenadziranu komponentu množimo s eksponencijalnom funkcijom w(t) koja se povećava s brojem epoha. Funkciju w(t) smo implementirali kao što je opisano u radu[8]. Broj epoha kroz koji se povećava funkcija postavili smo na 50, a njenu maksimalnu vrijednost na 1. Dodatno kroz tih 50 epoha eksponencijalno smo povećavali stopu učenja od 0.0001 s faktorom 1.05, a zadnjih 30 epoha eksponencijalno smo ju smanjivali s faktorom 0.95.

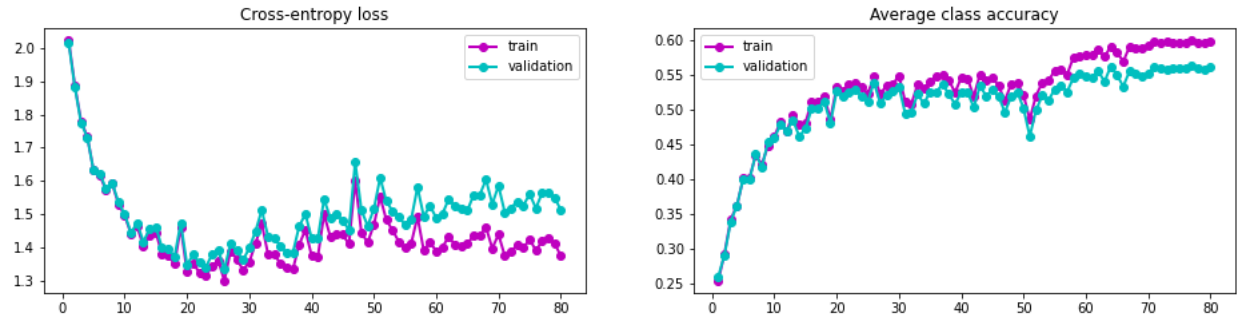
Algorithm 1 Π-model pseudocode.

Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function
for t in $[1, num\_epochs]$ **do**
  for each minibatch B **do**
    $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ evaluate network outputs for augmented inputs
    $\tilde{z}_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ again, with different dropout and augmentation
    $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$ ▷ supervised loss component
      $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
    update θ using, e.g., ADAM ▷ update network parameters
  end for
end for
return θ

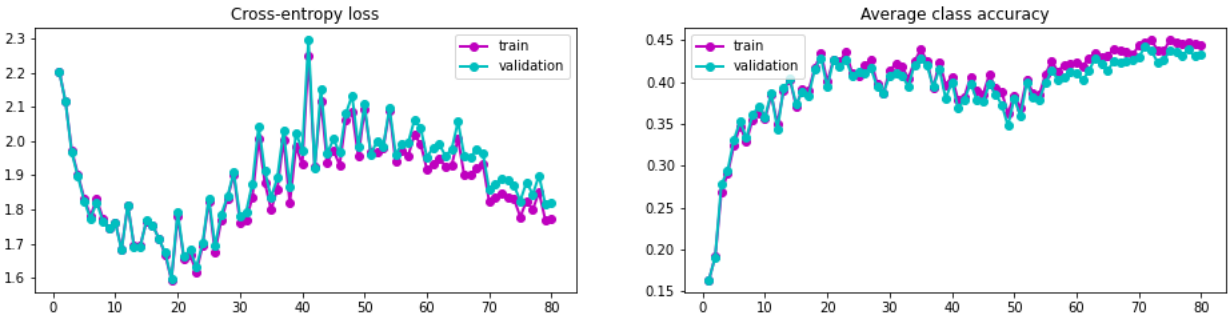
Slika 10.4: Pseudokod treniranja Π-modelom
[8]

10.5 Izvedba i rezultati

Proveli smo treniranje na 4000, 1000 i 250 označenih podataka sa Π-modelom i s nadziranim učenjem. U tablici 10.1 prikazani su rezultati treniranja te vidimo kako smo uspjeli postići bolju točnost samo na treningu s 4000 označenih podataka. Razlozi za to se nalaze u jednostavnoj arhitekturi mreže i manjem broju epoha učenja. Unatoč tome grafovi gubitka i točnosti za 1000 i 4000 označenih podataka prikazuju kako smo ipak uspješno implementirali učenje na neoznačenim podatcima.



Slika 10.5: Grafovi gubitka i točnosti Π-modela na 4000 označenih podataka



Slika 10.6: Grafovi gubitka i točnosti Π-modela na 1000 označenih podataka

Na slikama 10.6 i 10.5 jasno vidimo kako je mreža ostvarila primjetno povećanje točnosti što je indikacija učenja na neoznačenim podatcima što je i bio cilj ovog projekta.

Tablica 10.1: Rezultati polunadziranog učenja kozistencijom s 250, 1000, 4000 označenih podataka za 80 epoha učenja. Prikazani su rezultati učenja samo na označenim podatcima te na označenim i neoznačenim podatcima

Broj označenih podataka	Nadzirana točnost [%]	Nenadzirana točnost[%]
250	34.3	28.85
1000	46.03	43.24
4000	55.99	56.05

11. Zaključak

U ovom radu proučavali smo metode za povećanje generalizacijske točnosti modela za klasifikaciju slika.
Podešavanjem hiperparametara i uvođenjem umnožavanja podataka, uspjeli smo poboljšati točnost bez mijenjanja arhitekture mreže. Uvođenjem normalizacije po grupi i rezidualnih modela dramatično smo povećali točnost, no s puno složenijom mrežom.
Polunadzirano učenje implementirali smo u vidu pseudooznačavanja i Π-modela. Ovdje smo pokušali čim bolje naučiti naš model učenjem na kombinaciji malog podskupa označenih podataka s neoznačenim podacima. Iako tako ne postižemo točnost na razini učenja na svim označenim primjerima, ipak uspijevamo ”istisnuti” malo bolje rezultate od učenja samo na ograničenom podskupu označenih podataka. Takav rezultat može biti osobito važan u realnim projektima gdje nam je pribavljanje označenih podataka skupo i tako možemo poboljšati uspješnost projekta bez pretjeranog troška.

”Opipljivi” rezultat ovog projekta prvenstveno je izvorni kod modela za klasifikaciju slika iz skupa CIFAR-10 s implementiranim metodama za povećanje točnosti. No uz to, bitan je rezultat projekta stečeno znanje svih studenata u studentskom timu koje će poslužiti za daljnji angažman na završnom radu.

Popis slika

Literatura

[1] Stevens, E., Antiga, L., Viehmann, T., *Deep Learning with PyTorch*. Manning Publications Co., 2020.

[2] *Druga laboratorijska vježba iz Dubokog učenja*, `https://dlunizg.github.io/lab2/`

[3] Goodfellow, I., Bengio, Y. and Courville, A., "Machine Learning Basics," in *Deep Learning*, MIT Press, 1964, p. 118.

[4] Kingma, D. P. and Ba, J., "Adam: A Method for Stochastic Optimization", *arXiv e-prints*, 2014.

[5] Goodfellow, I., Bengio, Y. and Courville, A., "Optimization for Training Deep Models," in *Deep Learning*, MIT Press, 1964, p. 276.

[6] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A., "How Does Batch Normalization Help Optimization?", *arXiv e-prints*, 2018.

[7] Dong-Hyun, L., "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks", *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 2013

[8] Laine, S. and Aila, T., "Temporal Ensembling for Semi-Supervised Learning", *arXiv e-prints*, 2016.

[9] He, K., Zhang, X., Ren, S., and Sun, J., "Identity Mappings in Deep Residual Networks", *arXiv e-prints*, 2016.

[10] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition", *arXiv e-prints*, 2015.

[11] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A., "How Does Batch Normalization Help Optimization?", *arXiv e-prints*, 2018.

[12] Tarvainen, A. and Valpola, H., "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results", *arXiv e-prints*, 2017.

[13] Miyato, T., Maeda, S.-. ichi ., Koyama, M., and Ishii, S., "Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning", *arXiv e-prints*, 2017.