1. Kontrastivno ucenje i SimCLR

1.1 Kontrastivno učenje

Kontrastivno učenje je tehnika mašinskog učenja gde se ulazni podaci transformišu u tačke n dimenzionalnog prostora, a model se trenira da tačke sličnih podatka u tom prostoru približi a tačke različitih podataka udalji jedne od drugih. Odatle je i nastao naziv jer se uči kontrast između sličnih (pozitivnih) primera i različitih (negativnih) primera.

Prvi radovi koji su sadržali ideje koje se danas koriste u kontrastivnom učenju pojavili su se još tokom 1990-ih godina, a neki od njih su:

- **1993.** godine, rad Bromleja "Signature Verification using a 'Siamese' Time Delay Neural Network" je prvi poznati rad koji je koristio siamsku mrežu: dve mreže sa istim težinama koje porede dva ulaza i mere njihovu sličnost. Koristio se za prepoznavanje potpisa.
- **1998.** godine, rad LeCuna "Gradient-Based Learning Applied to Document Recognition" predstavlja primer rada koji koristi sličan princip poređenja uzoraka bez eksplicitnog nadzora. Takođe koristi siamsku arhitekturu, a služi za klasifikaciju rukopisa i dokumenata.

Međutim, ono što se smatra prvim pravim kontrastivnim radom u modernom smislu je rad iz **2006.** godine, Hadsell et al.: "Dimensionality Reduction by Learning an Invariant Mapping". Ovaj rad eksplicitno uvodi **contrastive loss** i koristi ga za mapiranje slika u zajednički prostor gde su slični uzorci blizu jedni drugima, a različiti udaljeni.

Iako je ovaj pravac postojao više od decenije, tek nakon **2018. godine** dolazi do značajnog razvoja zahvaljujući rastu računarske moći i dostupnosti velikih skupova podataka. U tom periodu pojavljuju se različite metode kontrastivnog učenja koje postižu impresivne rezultate u zadacima vizuelnog razumevanja, među kojima je i **SimCLR** (**Simple Framework for Contrastive Learning of Visual Representations)**.

1.2 SimCLR

SimCLR je posebno značajan zbog svoje jednostavnosti i velike efikasnosti. Razvijen je od strane Google Brain tima, a autori koji stoje iza ove metode su **Ting Chen**, **Simon Kornblith**, **Mohammad Norouzi** i **Geoffrey Hinton**. Njihov rad, objavljen **2020. godine** pod nazivom ***"A Simple Framework for Contrastive Learning of Visual Representations"***, predstavlja jedno od najuticajnijih istraživanja u oblasti učenja bez nadzora, sa otvorenim kodom koji je poslužio kao osnova za mnoga kasnija istraživanja.

Spada u metode bez nadzora **(self-supervised)**, jer tokom treniranja ne koristi oznake podataka. Umesto toga, model se uči pomoću jednostavnih zadataka koje samostalno stvara iz podataka. Najčešći takav zadatak je da model nauči da prepozna da su dve različite verzije slike (nastale augmentacijom) zapravo prikaz istog originala, bez potrebe za ručnim označavanjem slika. Glavna prednost ovakvog pristupa je sposobnost skaliranja. Možemo ga trenirati na ogromnim neoznačenim skupovima i zatim lako prilagoditi na konkretne zadatke sa malim brojem oznaka (tzv. fine-tuning). Takođe, pošto ne zavisi od anotacija, značajno se smanjuju troškovi pripreme podataka.

Iako je SimCLR originalno razvijen za obradu slika (testiran je prvenstveno na ImageNet i CIFAR-10 skupovima), njegova osnova nije vezana isključivo za slike. Kasnija istraživanja su uspešno adaptirala SimCLR principe i na druge domene, poput teksta, govora i bioinformatike, gde se takođe može definisati ideja pozitivnih i negativnih parova **(rad iz 2020. koji nudi detaljnu analizu razvoja metoda kontrastivnog i self supervised učenja)**.

U ovom projektu implementirana je pojednostavljena verzija SimCLR metode korišćenjem PyTorch biblioteke, sa ciljem da se analiziraju uticaji različitih parametara i kombinacija augmentacija na kvalitet naučenih reprezentacija. Kvalitet reprezentacija se ocenjuje putem linearne evaluacije na zadatku klasifikacije CIFAR-10 skupa.

2. Opis rešenja

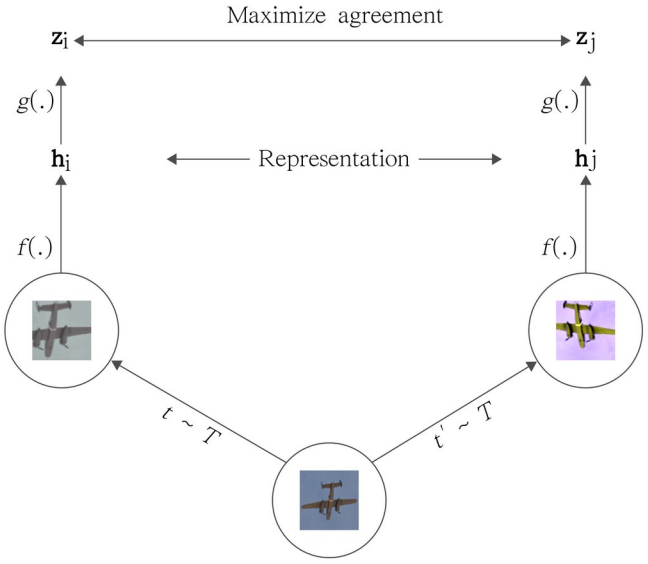
2.1 Algoritam

Pre nego što pređemo na analizu koda, važno je razumeti osnovnu ideju algoritma SimCLR.
Na početku se za svaku sliku iz skupa podataka primenjuju dve različite kombinacije augmentacija (npr. sečenje, izmena boja, rotiranje), čime dobijamo dve slične verzije iste slike. Te dve slike čine tzv. **pozitivni par**, jer potiču od iste slike.

Međutim, slike u sirovom obliku nisu pogodne za kompleksniju numeričku obradu i poređenje. Da bismo ih mogli upoređivati na intuitivniji način, predstavljamo ih u uprošćenom vektorskom obliku koji sadrži njihove najvažnije karakteristike. U tome nam pomaže **enkoder** f(), neuronska mreža koja svakoj augmentaciji dodeljuje njen uprošćeni prikaz, tj. vektor (označen kao h).

Ti vektori se zatim prosleđuju kroz manju mrežu nazvanu **projekciona glava** g(), koja ih mapira u novi prostor (označen kao *z*) u kojem se vrši učenje sličnosti.
Cilj algoritma je da nauči da vektori pozitivnih parova budu što bliži, dok negativni parovi (parovi koji potiču od različitih slika) budu što udaljeniji. Tačan stepen sličnosti meri se pomoću **kontrastivne funkcije gubitka**, koja uz pomoć dodatnog hiperparametra **temperature** (*T*) kontroliše koliko strogo se kažnjavaju odstupanja od željenog rezultata.

Kada se treniranje završi, projekciona glava se odbacuje, a enkoder ostaje kao koristan alat za dalje zadatke, poput klasifikacije slika.

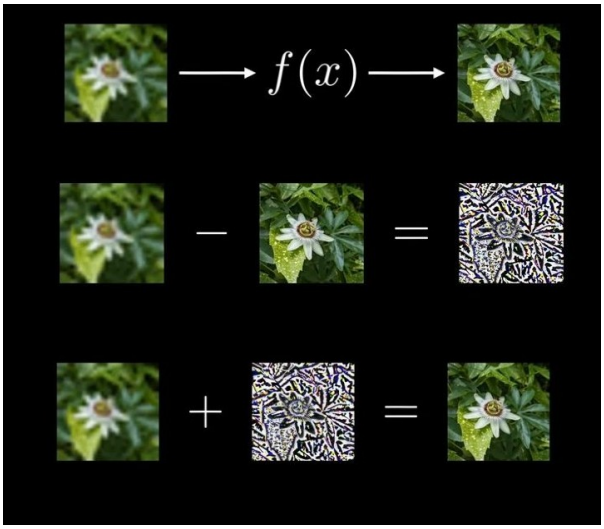


Slika 1. Na slici je prikazan tok obrade jedne slike u SimCLR metodi. Najpre se nad originalnom slikom primenjuju dve različite augmentacije, čime dobijamo dve varijacije iste slike. Obe varijacije zatim prolaze kroz enkoder f(), koji generiše njihove vektorske reprezentacije $\mathbf{h}_i$ i $\mathbf{h}_j$. Ti vektori se dalje prosleđuju kroz projekcionu glavu g(), gde se transformišu u nove reprezentacije $\mathbf{z}_i$ i $\mathbf{z}_j$. Ovaj par (z_i, z_j) čini **pozitivni par**, jer obe varijacije potiču od iste slike. Upravo na ovim parovima se trenira model da nauči sličnosti među povezanim uzorcima.

2.2 Implementacija

2.2.1 Arhitektura modela

Kao osnova enkodera obično se koristi **ResNet** mreža (slika 2). Ona koristi rezidualne blokove, koji omogućavaju da informacije iz ulaza lakše prolaze kroz mrežu. Umesto da mreža svaki put pokušava potpuno da izmeni podatke, ona uči samo šta treba dodati ili promeniti u odnosu na ulaz, i to zatim sabira sa originalnim signalom. Ovakav pristup omogućava treniranje dubljih mreža bez gubitka važnih informacija. U kontekstu kontrastivnog učenja to pomaže modelu da sažme slike u reprezentacije koje zadržavaju ključne karakteristike potrebne za razlikovanje među primerima.



Slika 2 predstavlja vizuelnu ilustraciju osnovne ideje ResNeta. Zadat je problem transformacije slike niže rezolucije u sliku više rezolucije. U prvom redu prikazan je klasičan pristup: model direktno uči da transformiše sliku niže rezolucije u sliku više rezolucije pomoću neke funkcije *f*. Drugi red prikazuje drugačiji pogled na problem: rezultat se može posmatrati kao početna slika uvećana za razliku, tj. masku koju je potrebno dodati. Ta maska se dobija kao razlika početne i ciljne slike. Time dolazimo do ideje ResNeta gde umesto da model uči celu transformaciju, uči samo tu razliku (**rezidual**). Model predviđa masku, koja se zatim sabira sa originalnom slikom, kako bi se dobio konačan rezultat (slika više rezolucije).

U većini istraživanja se koristi ResNet-50, koji ima više slojeva i daje bolje rezultate na velikim skupovima podataka kao što je ImageNet (sa preko milion slika). Međutim, za potrebe ovog projekta, zbog ograničenih hardverskih resursa, **ResNet-18** predstavlja praktičniji izbor. Iako je pravljen za rad sa ImageNet setom možemo ga lako prilagoditi radu i sa manjim setovima poput **CIFAR-10**. ResNet-18 arhitektura je jednostavnija, brža i zahteva manje memorije.

Iz ResNet arhitekture **izostavljen završni fully connected (fc) sloj**, jer nije potreban za kontrastivno učenje. Umesto njega koristi se projekciona glava, koja pretvara izlaz enkodera u prostor u kojem se uči sličnost između parova slika.

Za **projekcioni sloj** koristi se MLP koji uzima vektore dimenzije 512 (izlaz iz enkodera) i pretvara ih u vektore dimenzije **64**. Ovaj sloj se sastoji od **dva linearna sloja**, između kojih se nalazi **ReLU** funkcija aktivacije.

ReLU (Rectified Linear Unit) zamenjuje negativne vrednosti nulom, čime uvodi **nelinearnost** u model. Iako to može značiti da se određeni deo informacija gubi, ta nelinearnost omogućava mreži da uči složenije obrasce i efikasnije razdvaja slične od različitih primera, što je ključno za uspešno kontrastivno učenje.

2.2.2 Loss funkcija

U srži kontrastivnog učenja nalazi se **funkcija gubitka** (loss), čiji je cilj da približi reprezentacije sličnih primera, a udalji one koje nisu povezane. Konkretno, SimCLR koristi **NT-Xent** gubitak (*Normalized Temperature-scaled Cross Entropy Loss*).

Za svaku sliku u batch-u, model generiše njen reprezentacijski vektor *z*. Pošto se za svaku sliku prave dve različite augmentacije, dobijamo po dva vektora označena kao $\boldsymbol{z_i}$ i $\boldsymbol{z_j}$ koji čine jedan pozitivni par. Prvi korak je **normalizacija svih vektora**, kako bi im dužina bila jednaka 1. Ovo omogućava da se koristi **kosinusna sličnost**, koja meri ugaonu sličnost između dva vektora. Formula za kosinusnu sličnost između vektora $\boldsymbol{z_i}$ i $\boldsymbol{z_j}$ je:

$$similarity(z_i, z_j) = \frac{z_i \cdot z_j}{||z_i|| \cdot ||z_j||}$$

Nakon normalizacije ($\|z\| = 1$), formula se pojednostavljuje u običan **skalarni proizvod**:

$$similarity(z_i, z_j) = z_i \cdot z_j$$

Ovo nam omogućava da efikasno izračunamo matricu sličnosti S (u kodu sim_matrix), u kojoj je svaki element $S_{i,j}$ kosinusna sličnost između vektora z_i *i* z_j **.** Vrednosti ove matrice su u rasponu od -1 do 1, gde:

- 1 označava potpunu sličnost
- 0 znači da nema sličnosti (vektori su ortogonalni)
- −1 označava da su vektori potpuno suprotni

Pošto je svaka slika najsličnija samoj sebi, na dijagonali ove matrice pojavljuju se jedinice (sličnosti između svakog vektora i njega samog). Međutim, te vrednosti nisu korisne za treniranje i mogu narušiti izračunavanje gubitka, pa se zamenjuju vrednostima koje neće uticati na optimizaciju, najčešće velikim negativnim brojem. Na taj način, softmax funkcija neće favorizovati te pozicije prilikom računanja verovatnoća.

Takođe se koristi tzv. temperaturni parametar **T**, kojim se reguliše oštrina distribucije verovatnoća. On se primenjuje tako što se vrednosti sličnosti u matrici **S** dele sa **T**:

$$similarity(z_i, z_j) = \frac{z_i \cdot z_j}{T}$$

Kada je temperatura manja, na primer **T = 0.1**, razlike u kosinusnoj sličnosti između parova se dodatno pojačavaju. To znači da i male razlike u sličnosti dovode do velikih razlika u izlaznim vrednostima softmax funkcije. Model tada postaje sigurniji u svojim odlukama, jer snažnije favorizuje pozitivne parove u odnosu na negativne. Međutim, ovakav model može teže da generalizuje jer postaje osetljiviji na sitne varijacije u podacima.

Suprotno tome, kada je temperatura veća, na primer **T = 0.5**, eksponencijalna funkcija ublažava razlike među vrednostima. Model tada postaje tolerantniji, što može poboljšati njegovu sposobnost generalizacije, ali to istovremeno može smanjiti kontrast između pozitivnih i negativnih parova, čineći učenje težim.

U praksi, manja temperatura može dovesti do nižeg gubitka (*loss-a*), pod uslovom da model pravilno prepoznaje pozitivne parove, jer će njihova veća sličnost eksponencijalno dominirati nad ostalima. Ipak, ukoliko model greškom dodeli veću sličnost negativnim parovima, loss može ostati nizak, što stvara iluziju uspešnog učenja, dok zapravo vodi do pogrešnih reprezentacija. Zbog toga je važno pažljivo izabrati temperaturu kao jedan od ključnih hiperparametara.

Kao što smo ranije pomenuli, svaka originalna slika prolazi kroz dve različite augmentacije, koje zajedno čine pozitivan par. Ako imamo **N originalnih slika**, nakon augmentacija dobijamo ukupno **2N uzoraka**. Tokom treniranja, ove slike se organizuju tako da se augmentacije istih slika pojavljuju u susednim batch-ovima, a ne u istom. Na primer, ako se prva augmentacija slike A nalazi na poziciji *i* u jednom batch-u, onda će se druga augmentacija te iste slike nalaziti na istoj poziciji *i*, ali u narednom batch-u. Na ovaj način se obezbeđuje da pozitivni parovi budu jasno identifikovani, dok se sve ostale kombinacije tretiraju kao negativni parovi.

Batch 1

Slika (original)	Augmentacija 1	Indeks
Slika A	A1	0
Slika B	B1	1
Slika C	C1	2

Batch 2

Slika (original)	Augmentacija 2	Indeks
Slika A	A2	0
Slika B	B2	1
Slika C	C2	2

Prilikom spajanja vektora iz dva susedna batch-a unutar loss funkcije, moguće je precizno izračunati poziciju pozitivnog para za svaku augmentaciju. Naime, ako augmentacija slike A zauzima poziciju *i*, njen pozitivni par će se nalaziti na poziciji $N + i$ (gde je N broj originalnih slika). Na osnovu toga, za svaku poziciju *i* možemo tačno odrediti indeks *j* njenog pozitivnog para:

- Ako je $i < N$, tada je pozitivni par na poziciji $j = i + N$.
- Ako je $i \geq N$, tada je par na poziciji $j = i - N$.

Ova informacija se koristi za kreiranje vektora pozitivnih indeksa, koji loss funkciji precizno određuje koji indeks predstavlja pozitivni par za svaku poziciju.

Spojeni susedni batch-ovi		
Augmentacija	**Slika (original)**	**Indeks**
A1	A	0
B1	B	1
C1	C	2
A2	A	3
B2	B	4
C2	C	5

Tokom izračunavanja NT-Xent loss-a, računa se kosinusna sličnost između svakog vektora z_i i svih ostalih vektora u batch-u. Međutim, samo pozicija odgovarajućeg pozitivnog para koristi se kao ciljna vrednost za treniranje. Na osnovu toga, gubitak za svaku sliku računa se prema sledećoj

formuli:

$$L_i = -\log \frac{\exp(similarity(z_i, z_j)/T)}{\sum_{k=1}^{2*N} k \neq i \exp(similarity(z_i, z_k)/T)}$$

gde je:

- z_i jedan prikaz slike
- z_j njegov pozitivni par
- z_k svi ostali uzorci u batchu
- T temperatura.

Konačni gubitak se računa kao prosek svih L_i vrednosti za sve parove u batch-u.

2.2.3 Augmentacije

Za kreiranje pozitivnih parova koristi se niz transformacija nad istom slikom. Izbor konkretnih transformacija i njihovih kombinacija može značajno uticati na treniranje modela. Prilikom njihovog izbora treba uzeti u obzir dimenzije slika, njihovu rezoluciju, a možemo se poslužiti i znanjem iz ranijih istraživanja i uočiti šta se pokazalo kao dobra praksa. Na osnovu preporuka iz originalnog rada tima Google Brain , odlučili smo se za sledeće augmentacije: **random crop and resize**, **random horizontal flip** (sa verovatnoćom 0.5) i **color jitter** (sa verovatnoćom 0.8) u kombinaciji sa **grayscale** (sa verovatnoćom 0.2).

2.2.4 Trening

Model se trenira tokom 100 epoha nad **CIFAR-10** skupom (bez korišćenja etiketa). Za trening od dostupnih 50.000 slika CIFAR-10 seta koristimo podskup od **10.000** a veličina **batch-a 64**. Kao optimizator koristimo **Adam**.

2.2.5 Linearna evaluacija

Nakon treniranja SimCLR modela, **zamrzavaju se težine enkodera**, a **projekciona glava se odbacuje**. Na taj način ostaje samo enkoder, koji za svaku sliku daje fiksiran vektor reprezentacije. Na izlazne vektore iz enkodera trenira se jednostavan **linearni klasifikator** (jedan linearni sloj) koji pokušava da predvidi kojoj klasi slika pripada. U ovom delu koristi se ceo označeni trening skup **CIFAR-10,** a trening klasifikatora traje **50 epoha**. Ovaj proces je znatno brži od prethodnog jer se trenira samo jedan sloj, dok su težine enkodera fiksirane. Koristimo standardnu funkciju gubitka za probleme klasifikacije, **CrossEntropyLoss**, a kao optimizator opet **Adam**. Dobijena tačnost klasifikatora koristi se kao mera kvaliteta naučenih reprezentacija. Ako linearni klasifikator uspešno klasifikuje slike, to implicira da enkoder uspešno razdvaja slike različitih klasa u prostoru karakteristika.

3. Eksperimentalni rezultati

U ovom poglavlju predstavljeni su eksperimentalni rezultati dobijeni korišćenjem implementiranog SimCLR sistema za kontrastivno učenje. Eksperimenti su sprovedeni nad CIFAR-10 skupom podataka, sa ciljem da se proceni uticaj različitih parametara na kvalitet naučenih reprezentacija, kao i da se rezultati uporede sa sličnim pristupima iz literature.

3.1 Okruženje za eksperimente

- **Procesor**: AMD Ryzen 5 5600
- **Grafička kartica**: NVIDIA RTX 3060 (12GB VRAM)
- **RAM memorija**: 32 GB DDR4 3200MHz
- **Operativni sistem**: Windows 11
- **Programski jezik**: Python 3.12.2
- **Biblioteke**: PyTorch 2.7.1+cu128
- **Hardversko ubrzanje**: CUDA

3.2 Uporedna analiza sa literaturom

U dostupnoj literaturi nismo uspeli da pronađemo čist SimCLR model treniran isključivo na CIFAR-10 skupu, čak ni u konfiguracijama znatno jačim od naših. Uglavnom su u radovima za CIFAR-10 korišćeni modeli sa većim kapacitetima (npr. ResNet-50, veći broj epoha, veći batch size), a često su prisutni i elementi fine-tuninga, prethodnog pretreniranja modela na drugim skupovima podataka, kao i različite hibridne verzije SimCLR algoritma. U takvim slučajevima, tačnost na CIFAR-10 može dostići i do **95.3%** (npr. tim Google Brain-a, uz prethodno pretreniranje na ImageNet skupu). Zbog svega navedenog naš rezultat nije direktno uporediv sa rezultatima iz naučnih radova. Postoje neoficijalni GitHub projekti koji primenjuju SimCLR direktno na CIFAR-10 skupu, uz nešto bolje performanse od našeg modela. Primera radi, ostvareni su rezultati poput **92.06%** tačnosti (ResNet-18, batch size 512, 1000 epoha) i **69.82%** (ResNet-18, batch size 512, 100 epoha). Ipak, kako ovi radovi nisu zvanično recenzirani, glavno poređenje rezultata baziraćemo na sopstvenim implementacijama i eksperimentima.

3.2 Parametri testiranja

Osnovni model je baziran na ResNet-18 arhitekturi, bez prethodno naučenih težina. Nakon učenja kontrastivnih reprezentacija pomoću SimCLR okvira, vršena je linearna evaluacija koristeći jednostavni linearni klasifikator. Za sve eksperimente meren je NT-Xent gubitak tokom kontrastivnog učenja, kao i tačnost tokom linearne evaluacije.

Testirani su sledeći parametri:

- **Temperature (T)**: 0.1, **0.2**, 0.3
- **Learning rate (encoder)**: 1e-4, **3e-4**, 7e-4
- **Augmentacije**: testirano odvojeno samo crop + horizontal flip, samo color jitter + grayscale, **kombinacija svih**

- **Veličina dataseta: 10000**, 50000
- **Broj epoha (SimCLR)**: **100**, 300

Kao difoltnu postavku ovih parametara kroz eksperimente koristili smo boldovane vrednosti. A vreme izvršavanja bi bilo do dva i po sata uključujući i vreme linearne evaluacije.

3.4 Rezultati testova

Uticaj temperature:

Temperature (T)	**Loss**	**Accuracy**
0.1	**0.6149**	79.15%
0.2	1.5721	**82.16%**
0.3	2.4434	81.97%

Možemo uočiti da je gubitak manji za manje vrednosti temperature ali i to da manji gubitak ne znači nužno i bolje rezultate tačnosti. Niža temperatura dovodi do oštrijih razlika među pozitivnim i negativnim parovima ali i teže generalizuje, dok model sa većom temperaturom teže uči ali bolje generalizuje. Zato treba pažljivo odabrati vrednost ovog parametra tako da dobijemo što bolje rezultate za naš ciljni problem. Detaljnije o temperaturi smo objasnili u poglavlju *2.2.2 Loss funkcija.*

Uticaj learning reta optimizatora Adam vezanog za model SimCLR:

Learning rate	**Loss**	**Accuracy**
7e-4	**1.5608**	81,38%
3e-4	1.5721	**82.16%**
1e-4	1.6301	80.88%

Opet kao i u slucaju sa temperaturom primećujemo da manju gubitak ne znači i bolju tačnost. U ovom slučaju learning rate nam određuje veličinu koraka tokom traženja minimuma. Tako da ako uzmemo veći korak možemo ići brže ka nekom minimumu uz mogućnost da se i neki minimum preskoči. Kada uzmemo manji learning rate manja je mogućnost da preskočimo neki minimum ali se generalno sporije krećemo ka minimumu i zbog toga gubitak sporije opada.

Uticaj augmentacija:

Augmentations	**Accuracy**
Crop + Horizontal flip	56.15%
Color Jitter + Grayscale	50.54%
All together	**82.16%**

Uočavamo da kombinacija randomCropAndResize i randomHorizontalFlip daju bolje rezultate od korišćenja samo ColorJitter i Grayscale ali daleko bolje rezultate postižemo korišćenjem njihove kombinacije. Ono što se dalo primetiti iz rada Google brain tima je da ako nemamo neku vrste distorzije boje program može koristiti “prečice” u učenju i prepoznavati pozitivne parove po intezitetu piksela na slici umesto po karakteristikama izvučenih iz celokupne slike.

Uticaj velicine dataseta u treningu:

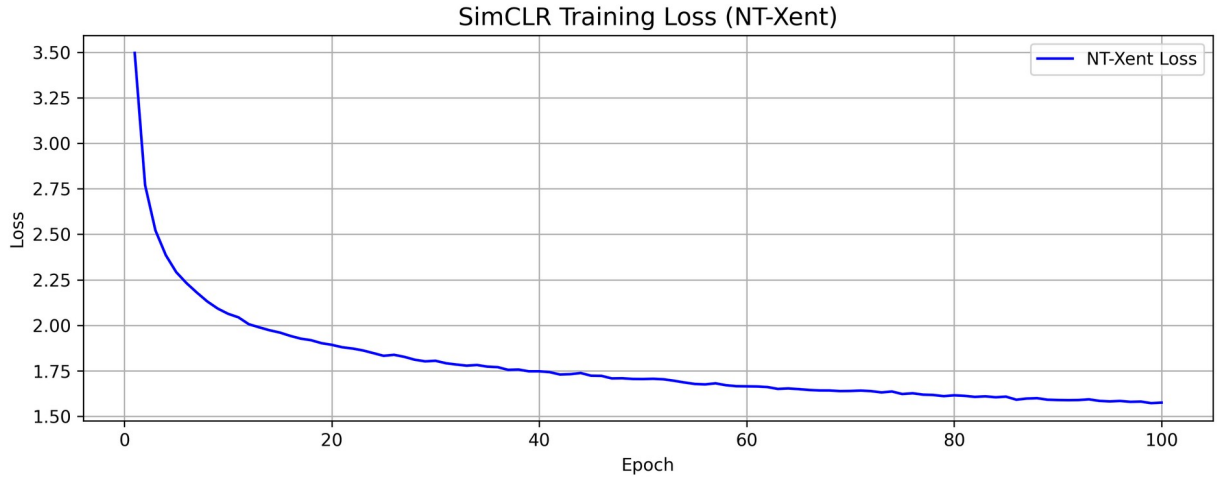
Datasize	**Accuracy**
10.000	**82.16%**
50.000	81.73%

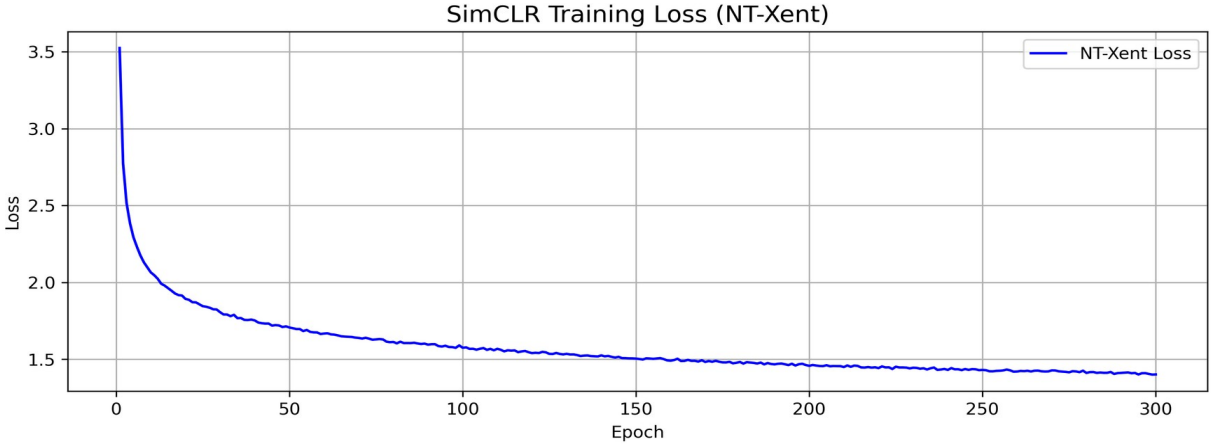
U eksperimentima je primećeno da linearna evaluacija postiže veću tačnost kada je model treniran na 10.000 slika u odnosu na 50.000 slika. Ovaj rezultat ne ukazuje na to da manji skup podataka nužno dovodi do boljih reprezentacija, već prikazuje ograničenja performansi. Zbog slabijeg hardvera korišćene su i skromnije konstrukcije koje nisu uspele da u datom opsegu vremena i memorije izvuku maksimum iz datih podataka. Sa druge strane, model sa manjim brojem slika je uspeo da stabilizuje svoje rezultate i bolje prilagodi linearnoj evaluaciji i time dobio nešto bolje rezultate. Uz jači hardver i duže treniranje mogli bismo da očekujemo da bi skup s većim brojem slika dao bolje rezultate.

Uticaj broja epoha:

Epoch	**Loss**	**Accuracy**
100	1.5721	82.16%
300	1.3998	**85.62%**

Očekivano, povećanje broja epoha dovelo je do veće tačnosti modela, ali uz značajan dodatni trošak u vremenu izvršavanja. Konkretno, sa 300 epoha postignuti su bolji rezultati u poređenju sa treniranjem od 100 epoha, iako je povećanje iznosilo svega nekoliko procenata. To je u skladu sa očekivanjima, jer rast performansi nije linearan. Kako se model približava globalnom optimumu, napredak postaje sve sporiji, pa dodatne epohe donose sve manje povećanje tačnosti.



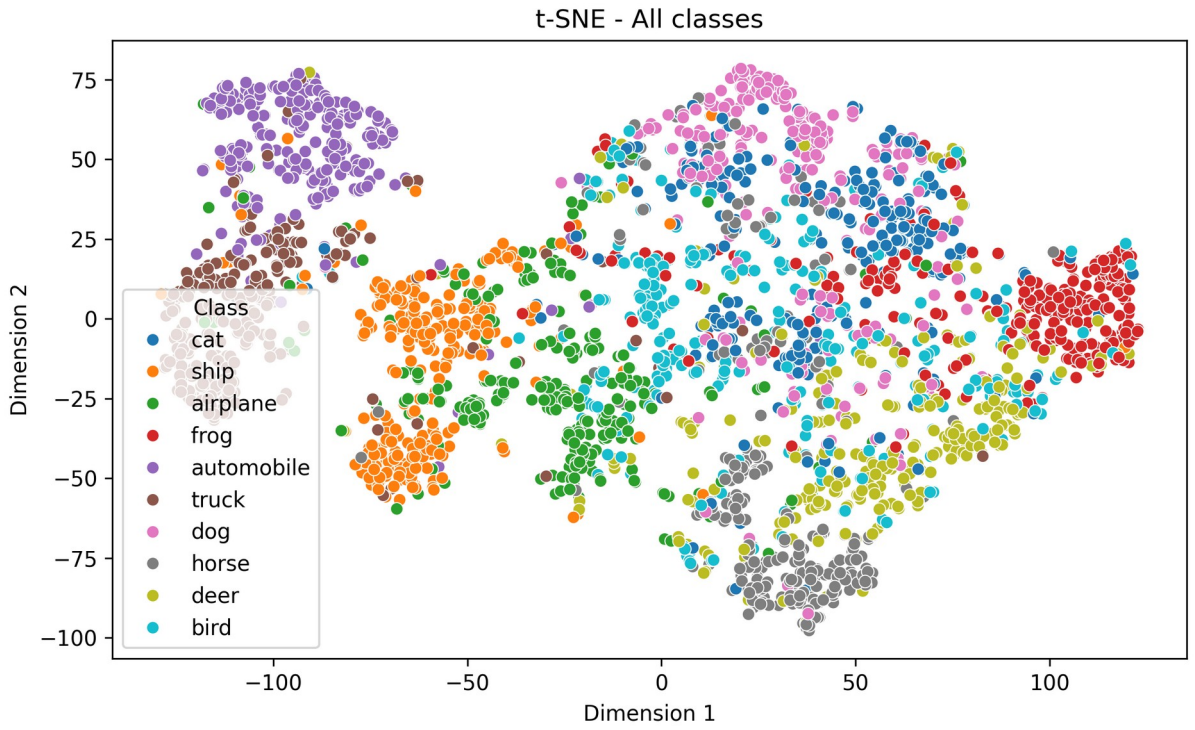


Slika 3 prikazuje opadanje loss-a tokom treninga na 100 i 300 epoha. Na graficiku se vidi da loss i kod 300 epoha i dalje nastavlja da opada, iako sporije nego na početku, dok ni u jednom trenutku ne postaje potpuno konstantan. Opadanje je kontinuirano, pa makar i po malim koracima.

Iako je model treniran sa 300 epoha ostvario nešto veću tačnost, zbog značajno dužeg vremena izvršavanja ne smatramo ga praktičnim za dalju analizu. Kao naše najbolje rešenje uzimamo eksperiment sa podrazumevanom postavkom iz prethodnog poglavlja, koji je dostigao tačnost od **82,16%**.

5.5 Vizualizacija

Da bismo dodatno prikazali kvalitet naučenih reprezentacija primenjena je metoda **t-SNE** (t-distributed Stochastic Neighbor Embedding) radi njihove vizuelizacije u dvodimenzionalnom prostoru. Ova tehnika omogućava da se složene višedimenzionalne strukture predstave na način koji je intuitivno razumljiv pri čemu se slični primeri grupišu blizu jedni drugih. Na taj način moguće je vizuelno uočiti koliko su reprezentacije različitih klasa dobro ili loše razdvojene, što predstavlja dopunski pokazatelj uspešnosti modela. U nastavku je prikazan najbolji rezultat postignut u eksperimentima.



Na **slici 4** prikazana je vizuelizacija reprezentacija naučenih našim najboljim modelom. Jasno se primećuju grupacije klasa u prostoru, kod nekih je to više izraženo (klasa automobila, kamiona, brodova, žaba) a kod nekih manje (klase ptica, jelena, mačaka). Primetno je da model generalno bolje razdvaja vozila od živih bića, što je verovatno posledica toga što vozila sadrže prepoznatljive i stabilne linije koje se ne menjaju značajno između slika, dok se linije i oblici životinja razlikuju u zavisnosti od poza i perspektive. Takođe, klasa ptica je najslabije grupisana, što je i razumljivo jer u CIFAR-10 u okviru te klase spadaju prilično različite vrste ptica, kao što su vrabac, patka, paun koje se znatno razlikuju po obliku, šarama i proporcijama tela, pa model mora da ih mapira u jednu zajedničku klasu.

4. Zaključak

U ovom radu implementiran je SimCLR pristup kontrastivnom učenju reprezentacija na CIFAR-10 skupu koristeći 10 000 slika za trening, arhitekturu ResNet-18, batch size 64, 100 epoha i 64-dimenzionalnu projekcionu glavu. U eksperimentima su testirani različiti hiperparametrari i augmentacije kako bi se demonstrirao njihov uticaj na kvalitet naučenih reprezentacija. U najboljem slučaju smo uspeli da ostvarimo tačnost linearne evaluacije od 82.16%. Iako nije blizu rezultata iz naučnih radova ovaj rezultat je prilično solidan za skromnije performanse koje smo koristili.

Moguća buduća poboljšanja po cenu više vremena izvršavanja i korišćenja jačih perfromansi su:

- korišćenje veće arhitekture (ResNet-34 i veće)
- veći batch-size (256 i veći)
- veća projekciona glava (128 dimenzija)
- više epoha

5. Korišćena literatura

1. Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, Roopak Shah: Signature Verification using a "Siamese" Time Delay Neural Network
2. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner: Gradient-based Learning Applied to Document Recognition
3. Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, Fillia Makedon: A Survey on Contrastive Self-supervised Learning
4. **Ting Chen**, **Simon Kornblith**, **Mohammad Norouzi** i **Geoffrey Hinton: A Simple Framework for Contrastive Learning of Visual Representations**
5. Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee: A survey on contrastive self-supervised learning