SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 231

RASPOZNAVANJE SLIKA PRIMJENOM SLOJEVA PAŽNJE

Bruno Ćorić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 231

RASPOZNAVANJE SLIKA PRIMJENOM SLOJEVA PAŽNJE

Bruno Ćorić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 231

Pristupnik:	**Bruno Ćorić (0036517561)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Raspoznavanje slika primjenom slojeva pažnje**

Opis zadatka:

Raspoznavanje slika važan je zadatak računalnog vida s mnogim zanimljivim primjenama. Trenutno stanje tehnike koristi duboke modele koji se uče s kraja na kraj. U posljednjih nekoliko godina posebno su zanimljivi modeli koji umjesto konvolucije koriste slojeve pažnje. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće pristupe za klasifikaciju slike. Odabrati slobodno dostupni skup slika te oblikovati podskupove za učenje, validaciju i testiranje. Predložiti prikladnu arhitekturu dubokog klasifikacijskog modela. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele te prikazati i ocijeniti ostvarene rezultate. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

SADRŽAJ

1. Uvod

Računalni vid grana je umjetne inteligencije čiji je cilj dobiti korisne informacije iz slike, a najčešće želimo dobiti informaciju što se to nalazi na slici. Tako je i jedna od najčešćih primjena računalnog vida klasifikacija slika.

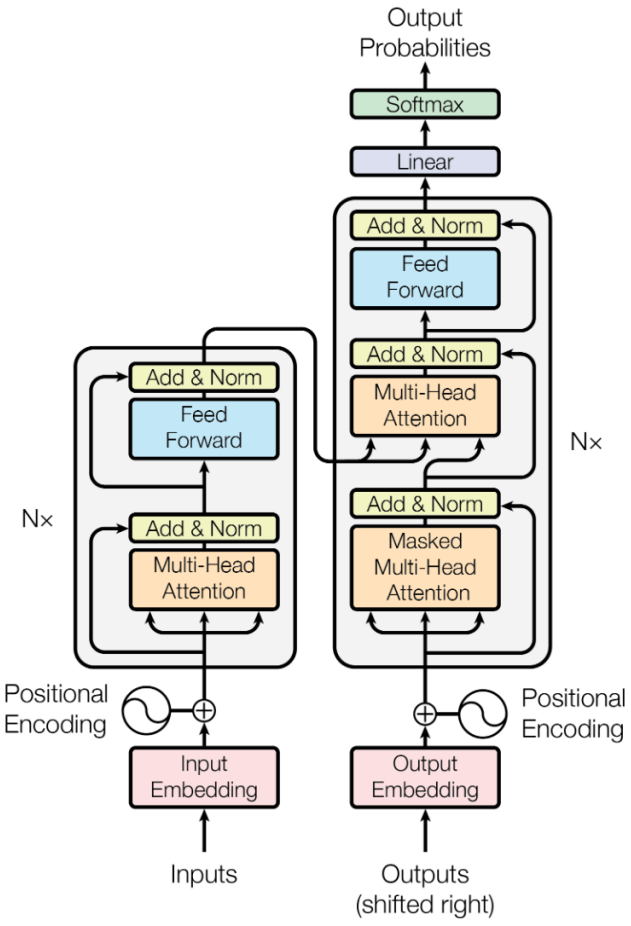
Klasifikacija slika je proces u kojemu se slici dodjeljuje neka oznaka. Oznake su najčešće razredi kao što su avion, auto i tako dalje. Klasifikacija slika je temeljni problem računalnog vida te je kao takav i jako dobro istražen.

Dugo godina su raspoznavanjem slika dominirali duboki konvolucijski modeli, no vremena se mijenjaju. 2017. godine izašla je nova arhitektura *transformer* [26] koja je u potpunosti promijenila područje obrade prirodnog jezika, ali ne i računalnog vida. Glavno obilježje *transformera* je primjena sloja pažnje.

U ovom završnom radu opisat će se koncept *transformera* te razni problemi i prednosti u korištenju te arhitekture u računalnom vidu. Predstavit će se arhitektura DeiT te će se njezini rezultati usporediti s dobro istraženim i popularnim dubokim konvolucijskim modelima.

2. Arhitektura *Transformer*

Transformer je arhitektura dubokih modela predstavljena 2017. godine. U daljnjem radu pri spominjanju *transformera* mislit ćemo na identičnu arhitekturu opisanu u radu *Attention is all you need* [26]. Glavna odlika ove arhitekture je sloj pažnje. *Transformer* se sastoji od 2 dijela, kodera i dekodera (Slika 2.1). U ovom radu fokusirat ćemo se na klasifikaciju slike te ćemo se zbog toga fokusirati na koder dio *transformera*.



Slika 2.1: *Transformer* arhitektura [26].

2.1. Ulaz

Posvetit ćemo poseban odjeljak samo ulazu u *transformer* koji se razlikuje od ulaza kod konvolucijskih mreža.

Ulaz kod konvolucijskih mreža je tenzor popunjen vrijednostima piksela te je ulaz često slika s normaliziranim vrijednostima piksela. Često je taj tenzor dimenzije CxHxW gdje C predstavlja broj kanala dok H i W predstavljaju visinu i širinu slike.

S druge strane, *transformeri* djeluju na neuređene kolekcije podataka. Prije nego što podaci uđu u *transformer* provodi se ugrađivanje ulaza (eng. *input embedding*). Najlakše je taj proces objasniti na primjeru obrade prirodnog jezika. U kontekstu *transformera* rečenica bi bila slijed koja se sastoji od više tokena (riječi). Svaki token (riječ) iz slijeda (rečenice) prolazi kroz ugrađivanje ulaza gdje se njegova vrijednost (u obradi prirodnog jezika broj u vokabularu) pretvara u vektor (na primjer *word2vec* algoritmom [15]) koji je sada reprezentacija te riječi. U računalnom vidu za primjer bi mogli uzeti vrijednosti svakog piksela u slici te s nekim postupkom dobivati vektorske reprezentacije tih piksela.

Osim što se provodi ugradnja ulaza, provodi se i pozicijsko kodiranje (eng. *positional encoding*). Pošto, za razliku od konvolucijske mreže, *transformer* nema informacije o pozicijama tokena u slijedu mora se pronaći način kako *transformeru* dati tu informaciju. Zato se poslije ugradnje ulaza slijeda, s kojom nastaje slijed vektora, svakom vektoru u slijedu dodaje i vektor koji predstavlja njegovu poziciju u slijedu. Postoje različite metode pozicijskog kodiranja, no u ovom radu je to implementirano kao tenzor dimenzija NxE gdje N predstavlja broj tokena u slijedu dok E predstavlja veličinu tokena. Vrijednosti tenzora pozicijskog kodiranja se onda zbrajaju s tenzorom koji predstavlja slijed dobiven iz ulaznih podataka.

2.2. Koder

Koder je osnovna gradbena jedinica *transformera* te se koristi kako bi predstavila određenu informaciju u vektorskom obliku. Glavna odlika kodera je da su veličina njegovog ulaza i izlaza jednake. Što znači da svaki token u slijedu ima svoj izlaz.

Koder se u originalom radu [26] koristio zajedno sa dekoderom, no to ne mora biti uvijek slučaj. Koder možemo koristiti na isti način kao što i koristimo konvolucijske mreže u raspoznavanju slika. S koderom možemo dobiti vektorski prikaz ulaza te pomoću višeslojnog perceptrona možemo izlazni vektor klasificirati.

Koder se sastoji od sloja pažnje nakon kojeg slijedi rezidualni sloj i sloj normali-

zacije. Nakon njega slijedi višeslojni perceptron te opet rezidualni sloj i sloj normalizacije (Slika 2.1).

2.3. Pažnja

Pažnja je ključni mehanizam *transformera* te će se ovaj rad fokusirati baš na nju. Postoje tri ključna pojma kada se priča o pažnji, a to su upit (eng. *query*), ključ (eng. *key*) i vrijednost (*value*). To su vektori koji sudjeluju u samom procesu pažnje i iz kojih dobivamo izlaz koji je također vektor. Ti se vektori dobiju množenjem svakog vektora ulaza (ulaz je slijed vektora gdje je svaki vektor reprezentacija neke vrijednosti) s 3 tenzora čiji se parametri uče prilikom procesa treniranja. Takvim postupkom dobivamo 3 nova vektora za svaki vektor u slijedu, vektor upita (Q), ključa (K) i vrijednosti (V).

Postoji više funkcija pažnje, no ovaj rad promatra onu predstavljenu originalnom radu [26], a to je *Scaled dot-product attention* (Jednadžba 2.1).

$$Paznja(Q, K, V) = softmax(\frac{QK^\intercal)}{\sqrt{d_k}})V \quad [26] \tag{2.1}$$

U ovoj formuli Q, K i V označavaju tenzore upita ključa i vrijednosti te je d_k dužina vektora upita. Dimenzije tenzora Q, K i V su NxD gdje je N veličina slijeda, a D je odabrana veličina. Uočavamo da su Q, K i V tenzori, a ne vektori kao što je opisano u prijašnjem dijelu. Pošto se ulaz sastoji od više vektora, zbog jednostavnosti i efikasnosti postupka dobiveni vektori upit, ključ i vrijednost se stavljaju u tenzore. Tako svaki od tih tenzora u svakom redu sadržava vektor za svaki token ulaznog slijeda. Primjećujemo da vektor upita svakog tokena množi sve vektore ključeva dobivene iz ulaza. a tako se množi i sa svojim vektorom ključa te se onda množi sa svim vektorima vrijednosti dobivene iz ulaza.

Također možemo zamijetiti iz formule (Jednadžba 2.1) da u pažnji pozicija tokena ne igra nikakvu uloga u izračunavanju vrijednosti. Zbog toga se i uvodi pozicijsko kodiranje kako bi postupak pažnje imao i informaciju u položaju tokena u slijedu.

2.4. Pažnja s više glava

Umjesto provođenja procesa pažnje jednom, pokazalo da je korisno provesti taj proces više puta. U praksi se ovaj postupak provodi tako da se dobiju tenzori Q, K i V kao u običnoj pažnji, ali se ti tenzori podijele na nekoliko jednakih dijelova (glavama). Tada

provodimo pažnju nad svakim od tih dijelova, spojimo izlaze pažnje te pomnožimo s tenzorom koji se također uči prilikom treniranja (Jednadžba 2.2).

$$ViseGlava(Q, K, V) = Spoji(glava_1, glava_2, ..., glava_h)W^o \quad [26] \tag{2.2}$$

gdje je $glava_i$ određena jednadžbom (Jednadžba 2.3).

$$glava_i = Paznja(QW_i^Q, KW_i^K, VW_i^V) \quad [26] \tag{2.3}$$

3. Primjena pažnje u računalnom vidu

Glavna zadaća ovog rada je primijeniti pažnju u području računalnog vida, no to se pokazalo puno problematičnije nego u nekim drugim područjima kao što je obrada prirodnog jezika. Tako su *transformeri* postali standard u području obrade prirodnog jezika, ali ne i u računalnom vidu.

Jedan od glavnih razloga zato je kvadratna vremenska i prostorna složenost pažnje. Pažnja za svaki token iz ulaznog slijeda radi vektor upita te taj vektor množi sa svim ostalim vektorima ključeva te onda i vektorima vrijednosti. Taj proces je spor i zahtjeva puno resursa. Ta činjenica omogućava laganu primjenu *transformera* u području obrade prirodnog jezika jer se često kao ulazi koriste rečenice ili paragrafi koji se sastoje od riječi te takvi slijedovi često ne budu predugački, no to nije i slučaj kod računalnog vida. Slike često sadrže i preko 100,000 piksela. To je znatno otežalo korištenje pažnje u području računalnog vida te su se morali razviti posebni procesi kako bi se omogućilo korištenje pažnje s razumnim resursima.

Postoji također još jedan veliki problem, a to je da *transformerima* treba više podataka nego dubokim konvolucijskim mrežama kako bi postigli iste rezultate. To se pokazalo posebno problematično u području računalnog vida. Do slika nije toliko lagano doći kao do teksta te je i puno zahtjevnije skladištiti ogromnu količinu podataka.

Pitanje je zašto uopće koristiti *transformere* ako imaju toliko velike prepreke. Pokazalo se da *transformeri* imaju određene prednosti kao što su: manja induktivna sklonost, odlična skalabilnost za jako duboke mreže i ogromnu količinu podataka, jako dobri rezultati kada se predtreniraju na ne označenim velikim količinama podataka te se onda treniraju na manjim skupom označenih podataka [3], težine su dinamički kalkulirane dok su kod konvolucijskih mreža one fiksne [11]. To su samo neki od razloga zašto nas primjena pažnje, specifično arhitektura *transformera*, zanima u području računalnog vida.

U području obrade prirodnog jezika *transformer* je praktički zamijenio najkorišteniju do tada arhitekturu LSTM [9]. Zbog lakše paralelizacije i sposobnosti učenja dugačkih ovisnosti *transformer* je postao široko primjenjivan u raznim zadacima obrade

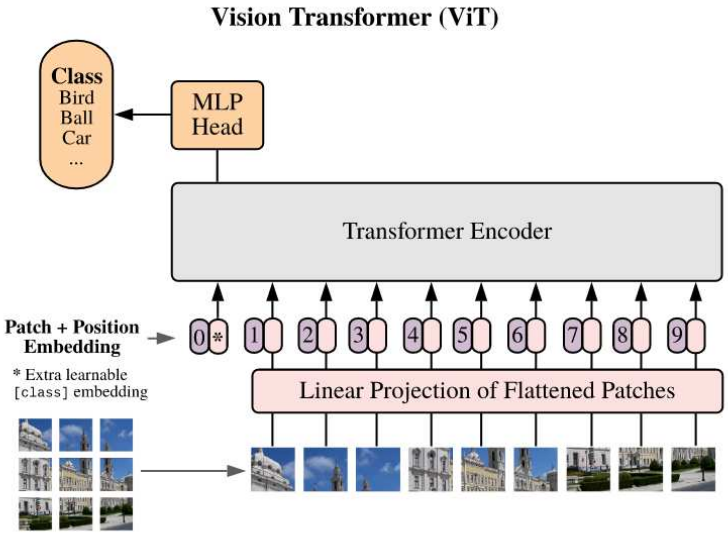
prirodnog jezika. Usprkos tome još uvijek se LSTM često koristi kod zadataka gdje nije dostupna tolika količina podataka.

U ovom poglavlju navedene su neke prednosti korištenja pažnje u računalnom vidu, ali i razlozi protiv. Hoće li *transformer* uvelike zamijeniti konvolucijske mreže kao što je to napravio LSTMovima ili će se hibridna arhitektura *transformera* i konvolucijske mreže pokazati kao najboljom, teško je reći. U ovom radu ćemo se fokusirati na arhitekturu koja je napravila znatne korake u primjeni *transformera*, a tim putem i samog sloja pažnje, u računalnom vidu.

4. Vision Transformer (ViT)

Arhitektura ViT (*Vision transformer*) [4] postala je standard za raspoznavanje slika modelima bez konvolucijskih slojeva. Ona pokušava prebroditi neke od problema navedenih u prijašnjem poglavlju te postići kompetitivne rezultate u usporedbi s tadašnjim najboljim mrežama, koje su većinom konvolucijske mreže, koristeći samo *transformer* arhitekturu bez ikakvih konvolucijskih slojeva.

Arhitektura je zapravo vrlo jednostavna te se sastoji od naslaganih kodera identičnih onima opisanima u originalnom *transformeru* [26]. Sama arhitektura *vision transformera* (Slika 4.1) je zapravo identična onoj predloženoj u originalnom radu, no razlog zbog kojeg je *vision transformer* postao vrlo popularan je način na koji se radi slijed od ulazne slike te zbog načina na koji se raspoznaje slika. U daljnjem radu koristit ćemo baš ViT arhitekturu za raspoznavanje slika pomoću slojeva pažnje.



Slika 4.1: ViT arhitektura [4]. *Transformer Encoder* dio ViTa je identičan originalnom *transformeru* (Slika 2.1), jedino što izlaz iz kodera ne ide u dekoder nego se uzima dio izlaza koji onda ide u višeslojni perceptron. *Transformer Encoder* se sastoji od sloja pažnje s više glava nakon kojeg slijedi rezidualni sloj i sloj normalizacije. Nakon njih slijede višeslojni perceptron te opet rezidualni sloj i sloj normalizacije kako je ilustrirano u originalnom radu [26] (Slika 2.1).

4.1. Ulaz

Jedan od najvećih problema pri primjeni *transformera* u računalnom vidu je složenost pažnje. Najjednostavniji i na prvu najsmisleniji način bi bio pretvoriti svaki piksel u vektorsku reprezentaciju te napraviti slijed koja se sastoji od svih piksela u slici. Taj pristup za većinu zadataka u računalnom vidu nije moguć jer su resursi potrebni za obavljanje pažnje preveliki.

Kako bi prebrodili taj problem u radu se predstavlja drugačiji pristup tokeniziranja. Slika se prvo dijeli u kvadratna okna. Tada se dobiva vektorska reprezentacija tih okna te se dodaje informacija o poziciji pomoću pozicijskog kodiranja. Tako naprimjer sa slike veličine $224 * 224$ umjesto slijeda dužine 50176, uzimanjem okna veličine $16x16$ dobivamo slijed dužine 196.

Takav pristup je omogućio korištenje *transformera* u računalnom vidu te je postao i standard kojeg su prihvatili kasniji pristupi [22] [28].

4.2. Klasifikacija

Kada se napravi slijed od slike, procesom opisanim u prethodnom ulomku, tom slijedu dodaje se još jedan token zvan razredni token (Slika 4.1).

Taj token je sličan onome koji se koristi u vrlo popularnoj arhitekturi BERT [3]. To je token koji se uči tijekom treniranja i koristi pri klasifikaciji. On ide zajedno s ostatkom slijeda kroz model te se "ponaša" isto kao i svi ostali tokeni. Tijekom klasifikacije slike u razred koristi se samo izlaz za sam razredni token, a ne i za ostale tokene u slici koji predstavljaju okna.

To je zanimljiv način klasifikacije slika koji se ne primjenjuje u mrežama kao što su konvolucijske mreže, a daje dobre rezultate.

4.3. Rezultati rada

Pokazalo se da ova arhitektura omogućava dobivanje konkurentnih rezultata sa samo *transformer* arhitekturom nad klasifikacijskim zadacima. ViT postiže odlične rezultate nad ImageNet [20] podatkovnom skupom koji je standard za klasifikaciju slika. ViT prestiže konkurentne modele kada je predtreniran na Googleovom JFT-300M podatkovnom skupu. JFT-300M je privatni podatkovni skup koji se sastoji od 300 milijuna slika. Iako postiže odlične rezultate uočen je i jedan veliki problem, a to je da ovakva arhitektura zahtjeva treniranje nad ogromnim količinama podataka. Tako da na 300 milijuna slika arhitektura postiže odlične rezultate, ali smanjivanjem broja slika i treniranjem na nekim manjim podatkovnim skupovima, kao što je ImageNet, rezultati ViTa postaju lošiji od konkurencije.

Tako je ViT uspio riješiti problem velike složenosti primjenjivanja pažnje nad slikama, ali ne i problem potrebe za ogromnim količinama podataka pri treniranju modela.

5. Data efficient image transformer (DeiT)

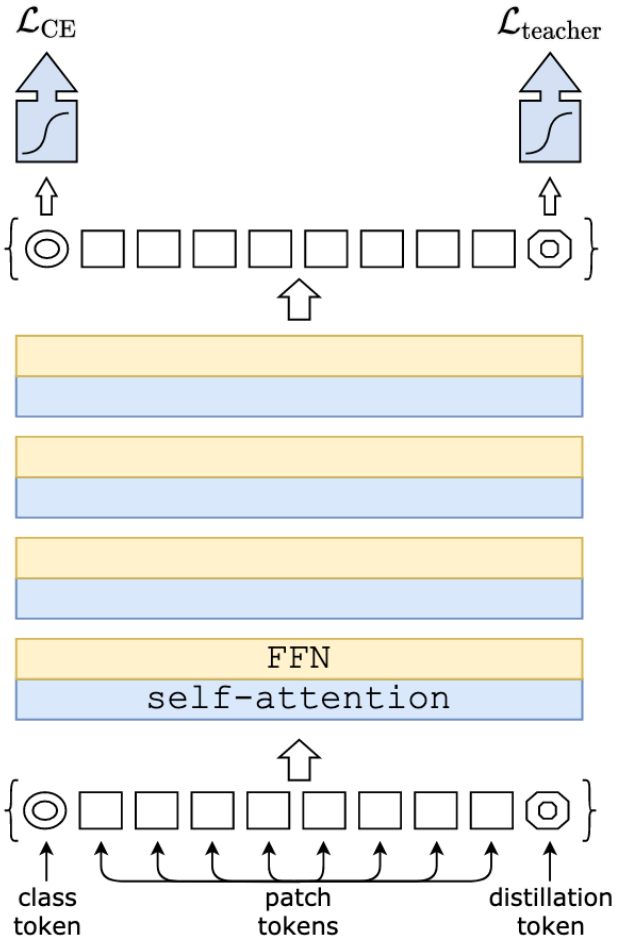
DeiT [23] je nadogradnja na ViT arhitekturu s kojom se pokušava riješiti problem potrebe za velikom količinom podatka kako bi se dobili kompetitivni rezultati s arhitekturom koja sadržava samo *transformer*. DeiT uvodi neke zanimljive metode kojim se ublažava ovaj problem te zadržavaju dobri rezultati dobiveni ViTom.

5.1. Arhitektura

Arhitektura DeiTa (Slika 5.1) je skoro pa identična ViT arhitekturi uz jednu promjenu, a to je da se osim razrednog tokena (token koji se koristi kod klasifikacije u ViTu), u slijed stavlja i destilacijski token (Slika 5.1).

Destilacijski token je identičan kao i razredni token jedino što se koristi u drugačiju svrhu. Razredni token koristi se za klasifikaciju slike dok se destilacijski token koristi za destilaciju znanja.

Postoji više arhitektura DeiTa u kojoj su se probavale inačice samo s destilacijskim tokenim i s obje vrste tokena te su se najbolji rezultati pokazali pri korištenju i razrednog tokena i destilacijskog tokena. U daljnjem radu opisivat će se model koji u sebi sadrži oba tipa tokena.



Slika 5.1: Arhitektura DeiT [23] je vrlo slična ViTu (Slika 4.1), ono po čemu se razlikuje su ulaz i izlaz. Ulaz je isti kao kod ViTa, samo se osim razrednog tokena stavlja i destilacijski token. Kod ViTa kao izlaz se uzima samo izlaz razrednog tokena dok se kod DeiTa uzima i izlaz destilacijskog tokena. Na slici su označeni pravokutnici obojeni plavom bojom (*self-attention*) i žutom bojom (*FFN*) te svaki od tih pravokutnika predstavlja koder identičan onom u ViTu (Slika 4.1).

5.2. Destilacijski token

Destilacijski token (Slika 5.1) ključan je koncept u ovom radu te i jedna od ključnih stvari koja je pridonijela uspjehu ovoga modela.

Destilacijski token funkcionira identično kao i razredni token, no njegova uloga je drugačija. Iz imena možemo zaključiti da ovaj token ima veze s destilacijom znanja za razliku od razrednog tokena koji služi samo pri klasifikaciji slike. Destilacijski token

omogućava modelu da uči od modela učitelja putem destilacije. Tako se DeiT arhitektura sastoji od modela učitelja i modela studenta. Model student je ViT dok je učitelj model neki drugi model naučen na podatkovnom skupu kojeg želimo klasificirati.

Destilacijski token omogućava ViTu da uči uz pomoć modela učitelja, a ne samo uz pomoć razrednog tokena kao što je to bio slučaj u originalnom radu [4]. Tako da ova arhitektura ima dva izlaza, jedan koji proizlazi iz razrednog tokena te jedan koji proizlazi iz destilacijskog tokena. Kod učenja oba se izlaza gledaju zasebno dok se kod zaključivanja uzima srednja vrijednost oba izlaza.

5.3. Destilacija znanja

Destilacija znanja [8] je metoda u kojem se prenosi znanje sa, uglavnom, većeg modela (model učitelj) na manji model (model student). Manji modeli su uglavnom efikasniji i brži od većih modela, no dobivaju lošije rezultate. Ideja je istrenirati veći model na skupu podataka, a manji bi se model prilikom učenja nad tim istim skupom podataka koristio i izlazom većeg modela.

5.3.1. Destilacija znanja u DeiTu

U DeiTu destilacija znanja provodi se pomoću destilacijskog tokena. Kao student model uzima se arhitektura ViT s nadodanim destilacijskim tokenom, a kao model učitelj uzima se konvolucijska mreža.

Eksperimentacijama se pokazalo da je konvolucijska mreža bolji učitelj od mreža sastavljenih od samo *transformera*. Zašto je to tako nije formalno dokazano, no u radu se pretpostavlja da je to zbog toga što *transformeri* nasljeđuju induktivnu sklonost od konvolucijskih učitelja putem destilacije znanja.

5.4. Funkcija gubitka

Predstavljene (Slika 5.1) su dvije moguće funkcije gubitka. To su *soft distillation* i *hard-label distillation* [22].

5.4.1. Soft distillation

Prvi tip funckije gubitka je *soft distillation*. Ta funkcija gubitka minimizira Kullback-Leiblerovu divergenciju između funkcije softmax model učitelja i funkcije softmax

modela studenta [23].

$$\mathcal{L}_{global} = (1-\lambda)\mathcal{L}_{CE}(softmax(Z_s)y) + \lambda\tau^2 KL(softmax(Z_s/\tau), softmax(Z_t/\tau)) \tag{5.1}$$

Z_s izlaz iz student modela, a Z_t izlaz iz modela učitelja. τ je temperatura destilacije, a λ je koeficijent koji balansira Kullback-Leiblerovu divergenciju.

5.4.2. Hard-label distillation

Drugi način destilacije je *hard-label distillation*. Ta funkcija gubitka se može razdvojiti na dva djela, gdje se od svakog djela uzima polovica dobivene vrijednosti.

Prvi dio je klasični gubitak unakrsne entropije: $\mathcal{L}_1 = \mathcal{L}_{CE}(softmax(Z_s), y)$
U kojem je $\mathcal{L}_{CE}$ funkcija gubitka unakrsne entropije, Z_s je izlaz iz student modela te je y zadana točna oznaka za tu sliku.

Drugi dio sadrži identičnu formulu kao i prvi dio, no umjesto y koji predstavlja točnu oznaku, stavlja se y_t koji predstavlja predikciju klasifikacije modela učitelja. Tako da formula onda glasi (Jednadžba 5.2):

$$\mathcal{L}_2 = \mathcal{L}_{CE}(softmax(Z_s), y_t) \tag{5.2}$$

gdje je: $y_t = argmax_c Z_t(c)$. Tako krajnja formula glasi (Jednadžba 5.3):

$$\mathcal{L}_{global}^{hardDistill} = 0.5 * \mathcal{L}_1 + 0.5 * \mathcal{L}_2 \tag{5.3}$$

5.5. Uvećanje podataka

Jedna od ključnih stvari koje je DeiT uveo kao poboljšanje nad ViTom je uvećanje podataka. Uvećanje podataka je jedna od metoda koja je omogućila DeiTu da postiže dobre rezultate nad puno manjim skupovima podataka nego ViT.

Primjenjivanjem slučajnih uvećavanja podataka umjetno se dobiva veći podatkovni skup. To je osobito korisno pri učenju modela nad manjim podatkovnim skupom (npr. CIFAR-10) te se pokazalo da primjenom posebno odabranih metoda uvećavanja podataka te destilacije znanja možemo dobiti konkurentne rezultate trenirajući model na manjem skupu podataka, kao što je to CIFAR-10 što kod ViTa to nije bio slučaj.

5.6. Rezultati rada

Za razliku od ViTa, DeiT postiže odlične rezultate i kada je treniran samo na ImageNet podskupom podataka, ali i CIFAR-10 skupom podataka.

Pokazalo se da je pomoću DeiT moguće dobiti jako kompetitivne rezultae i u točnosti, ali i u performansama. Ono najbitnije s DeiTom su se uspjeli dobiti puno bolji rezultati od ViTa kada se trenirana na manjem skupu podataka kao što je ImageNet (Tablica 5.1). DeiT je tako uspio donekle popraviti taj veliki problem kod ViTa vezan uz podatke.

Ime modela	Broj parametara	Broj slika po sekundi	ImageNet
ViT-B/16 [4]	86M	85.9	77.9
ViT-L/16 [4]	307M	27.3	76.5
DeiT-Ti [22]	5M	2536.5	72.2
DeiT-S [22]	22M	940.4	79.8
DeiT-B [22]	86M	292.3	81.8

Tablica 5.1: Tablica preuzeta iz rada [22] koja uspoređuje rezultate ViTa i DeiTa treniranim nad ImageNet skupom podataka.

Gledajući ove rezultate (Tablica 5.1) vidi se veliki potencijal u korištenju transformera u računalnom vidu. U jako malo vremena arhitektura bazirane na pažnji uspjele su stići konvolucijske mreže, a u nekim slučajevima ih i prestići. Usprkos tim obećavajućim rezultatima još uvijek možemo zamijetiti neke probleme kao što je problem s resursima i problem s potrebnom količinom podatka. DeiT je riješio problem s količinom podataka donekle, no još uvijek su potrebne veće količine podataka kako bi dobivao rezultate usporedive s onima dobivenim pomoću konvolucijskih modela.

5.7. Mjere konzistentnosti greške

U ovome ulomku bit će ukratko opisano kako se predlaže mjerenje greške konzistentnosti te kako te mjere mogu pomoći pri usporedbi različitih arhitektura [24].

5.7.1. Kohenova kapa

Kohenova kapa je mjera koja se koristi kako bi se iskazalo koliko se slažu dva sustava u svojim odlukama. Smatra se kao robusnija metoda od uspoređivanja samo koliko se

često slažu dva sustava u odluci.

$$c_{obs_{i,j}} = \frac{e_{i,j}}{n} \quad [24] \tag{5.4}$$

$$c_{exp_{i,j}} = p_i p_j + (1 - p_i)(1 - p_j) \quad [24] \tag{5.5}$$

$$\kappa_{i,j} = \frac{c_{obs_{i,j}} - c_{expi,j}}{1 - c_{expi,j}} \quad [24] \tag{5.6}$$

U jednadžbi (Jednadžba 5.6), $c_{obs_{i,j}}$ označava izmjereno preklapanje grešaka dok $c_{expi,j}$ označava očekivano preklapanje grešaka. u jednadžbi (Jednadžba 5.4), $e_{i,j}$ označava slaganje dva sustava za određeni razred, a n je ukupan broj primjera koje klasificiraju. U jednadžbi (Jednadžba 5.5), p_i i p_j su vjerojatnosti, dobivene iz binomne razdiobe, da će sustav i i sustav j klasificirati primjer u određeni razred.

Problem sa kohenovom kapom je da ne uzima u obzir dodatne detalje o pogrešci klasifikacije slike već jesu li se dva sustava složila te kolika je vjerojatnost da su se dva sustava slučajno složila. Također je teško interpretirati odakle dolaze različitosti i sličnosti između sustava [24].

5.7.2. Jensen-Shannonova udaljenost

Jedan od načina kako dobiti bolju usporedbu dva sustava je uspoređivati njihove matrice zabune. Matrica zabune je matrica koja pokazuje koliko je sustav primjera iz svakog razreda točno klasificirao te za primjere iz razreda koje nije točno klasificirao, prikazuje u koje ih je razrede krivo klasificirao.

U ovome ulomku koristit ćemo ImageNet skup podataka koji ima 1000 različitih razreda. Pošto bi matrica zabune bila prevelika za skup podataka od 1000 različitih razreda i pošto je teško skupiti dovoljno podataka od ljudi kako bi se popunila ta matrica zabune, u radu [24] se radi grupiranje razreda koristeći WordNet hijearhije [16] kako bi se 1000 različitih razreda smanjilo na 16 različitih razreda. Tako se dobiva 16 različitih razreda: avion, medvjed, bicikl, ptica, brod, boca, automobil, stolica, sat, pas, slon, tipkovnica, nož i kamion. Što znači da 1000 razreda grupiramo u 16 razreda.

Koristeći matricu zabune može se doći do mjera koje su bolje od Kohenova kape te je moguće i uspoređivati greške s ljudskim greškama. Generira se distribucija vjerojatnosti greške tako da se prvo broj grešaka za svaki razred podijeli s ukupnim brojem grešaka (Jednadžba 5.7).

$$p_i = \frac{e_i}{\sum_i^C e_i}, \forall i \in \{1, 2, ..., C\} \quad [24] \tag{5.7}$$

Te se onda računa Jensen-Shannonova udaljenost po idućem izrazu (Jednadžba 5.8).

$$JS(p, q) = \sqrt{\frac{D(p\|m) + D(q\|m)}{2}} \quad [24] \tag{5.8}$$

Gdje je $m = (p_i + q_i)/2$, p i q su distribucije vjerojatnosti greške dva sustava, a D (Jednadžba 5.9) je Kullback-Leibler divergencija.

$$D(p\|q) = \sum_i p_i \log(\frac{p_i}{q_i}) \quad [24] \tag{5.9}$$

Jensen-Shannova udaljenost se koristi za mjerenje sličnosti između dvije distribucije vjerojatnosti.

Razredna Jensen-Shannonova udaljenost

Razredna Jensen-Shannonova udaljenost (eng. *class-wise JS distance*) uspoređuje koji razredi su krivo klasificirani, za dan broj razreda (Jednadžba 5.10). CM je matrica zabune u kojoj se skuplja 1000 razreda u 16 prije određenih razreda.

$$e_i = \sum_j CM_{i,j}, \forall j \neq i \quad [24] \tag{5.10}$$

Za svaki razred se prolazi kroz matricu zabune te se zbroji broj primjera koji su krivo klasificirani. Možemo zamijetiti da je broj grešaka 16 te tako ne dobivamo potpunu informaciju o svim pogreškama mreže.

Međurazredna Jensen-Shannonova udaljenost

Međurazredna Jensen-Shannonova (eng. *inter-class JS distance*) (Jednadžba 5.11) udaljenost pokazuje u koje razrede je klasificirano sustav primjere kada ih je klasificirano krivo.

$$e_{i,j} = CM_{i,j} \tag{5.11}$$

Ova metrika sadržava u sebi punu distribuciju grešaka klasifikacije te se pokazalo da ova vrijednost ne korelira sa Kohenovom kapom kao što je to slučaj kod razredne Jensen-Shannove udaljenosti. Možemo zamijetiti da više ne zbrajamo greške za svaki razred da dobijemo 16 grešaka nego uzimamo broj grešaka za svako polje matrice zabune te tako dobivamo 240 tipova grešaka.

6. AtResNet (ResNet s pažnjom)

ViT i DeiT su arhitekture koje uopće ne koriste konvolucijske slojeve, a postižu impresivne rezultate. To su arhitekture koje su pokazale da se može bez konvolucijskih slojeva dobiti dobar klasifikator slika, no znači li to da nam konvolucijski slojevi uopće ne trebaju? Osmišljena je arhitektura koja koristi ResNet i koder *transformera* kako bi se pokazalo da pažnja i konvolucijski slojevi mogu ići zajedno te da je to nešto vrijedno daljnjeg istraživanja. Ova arhitektura je nazvana AtResNet te će se u daljnjem radu koristiti taj naziv.

6.1. ResNet

ResNet je duboka razidualna mreža koja koristi konvolucijske slojeve. Koristimo ResNet kao bazu za novu arhitekturu AtResNet. Prikaz slojeva ResNeta i veličina izlaza poslije svakog sloja je prikazana na slici (Slika 6.1)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Slika 6.1: Tablica koja prikazuje arhitekturu ResNeta kako je implementirana u *torchvision* biblioteci. [5]

6.2. Dodavanje razrednog tokena

ResNetu nadodajemo razredni token na identičan način kao što to radimo kod ViTa ili DeiTa (Slika 4.1 i Slika 5.1). Izlaz tog razrednog tokena ćemo koristiti za klasifikaciju pa ćemo zato ukloniti iz ResNeta posljednji potpuno povezani sloj i sloj globalnog sažimanja (eng. *Average Pool*). Na kraj, dodat ćemo potpuno povezani sloj koji će primati tenzor veličine razrednog tokena te će veličina izlaza biti jednaka onoj koliko razreda ima podatkovni skup nad kojim treniramo (10 u našem slučaju pošto se koristi podatkovni skup CIFAR-10 [12]).

6.3. Dodavanje pažnje

Nakon dodavanja razrednog tokena slijedi dodavanje pažnje u ResNet. Na slici (Slika 6.1) dodajemo koder identičan onomu iz arhitekture ViT (Slika 4.1) između slojeva conv2_x i conv3_x, conv3_x i conv4_x, conv4_x i conv5_x te poslije sloja conv5_x (Slika 6.1). Ulaz takvom sloju pažnje je razredni token i izlaz iz prijašnjeg konvolucijskog sloja. Tenzor izlaza iz konvolucijskog sloja se mijenja korištenjem izravnavanja (eng. *flatten*). Zadnje dvije dimenzije koje predstavljaju dužinu i širinu izravnamo te dobivamo

tenzor dimenzija BxCxN gdje je B veličina grupe, C broj kanala, a N je veličine HxW.

Želimo izbjeći kvadratnu složenost paženje stoga samo od razrednog tokena radimo tenzor upita (eng. *query*) dok od izlaza iz konvolucijskih slojeva radimo tenzore ključa (eng. *key*) i tenzor vrijednost (eng. *value*). Kako bi mogli pomnožiti tenzor upita i tenzor ključa, posljednja dimenzija mora biti ista što postižemo razdvajanje potpunog povezanog sloja, s kojim dobivamo tenzore upita, ključa i vrijednost iz ulaza, na 3 dijela. Prvom potpuno povezanom sloju veličina i izlaza i ulaza su jednake veličini razrednog tokena. Drugom i trećem potpuno povezanom sloju veličina izlaza je jednaka veličini razrednog tokena dok je veličina ulaza jednaka veličini 3. dimenzije izravnatg tenzora dobivenog iz konvolucijskog sloja. S takvim postupkom pažnje gdje se upit radi samo od razrednog tokena smanjujemo složenost s kvadratne na linearnu.

6.4. Rezultati

Treniran je AtResNet-18 model koji se sastoji od modificiranog ResNet-18 modela. Dostignuta je točnost od 95.13% nad testnim podskupom podatkovnog skupa CIFAR-10. Originalnom ResNetu se dodaje još 4.8M parametara. Rezultati usporedbe AtResNet-18 se mogu vidjeti u tablici (Tablica 8.2). AtResNet-18_p je modificirani ResNet-18 kojem su dodana 4 kodera. Dimenzija razrednog tokena je 256 te koristimo pažnju sa 8 glava.

Uočavamo malo manju točnost AtResNet-18_p s obzirom na ResNet-18_p. Daljnjim eksperimentiranjem nad arhitekturom vjerujemo da se taj rezultat može dostići, a i prestići. Uspoređivanjem broja slika u sekundi zaključujemo da AtResNet-18 usporava ResNet-18, ali ta razlika nije toliko drastična zbog linearne složenosti pažnje u AtResNetu. Zauzeće CUDA memorije je veće, ali opet ta razlika nije toliko drastična.

Ovim rezultatom pokazujemo da ova arhitektura ima potencijala te da čak ova prvotna verzija modela postiže dovoljno dobre rezultate. AtResNet je lako primjenjiv na sve ostale verzije ResNeta te jedno od mogućih poboljšanja je dodavanje više kodera ili promjena mjesta kodera u arhitekturi.

7. Programska implementacija

7.1. Korištene tehnologije

Za izradu. treniranje i evaluiranje modela korišten je programski jezik *Python* [25] te biblioteka *Pytorch* [18]. Također je korištena biblioteka *timm* [27] koja omogućava jednostavnu implementaciju modela kao što je ViT, ali i za implementaciju augmentacija nad podacima. Za vizualizaciju značajki korištena je modificirana implementacija iz repozitorija *Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet* [28] repozitorija [13] te repozitorija [10] za vizualizaciju pažnje.

Koristi se modificirani repozitorij rada [22] za treniranje DeiTa, a i ResNeta. Repozitorij koristi biblioteke Pytorch i timm kod učitavanja podataka, augmentacije podataka, treniranja i evaluiranja mreža.

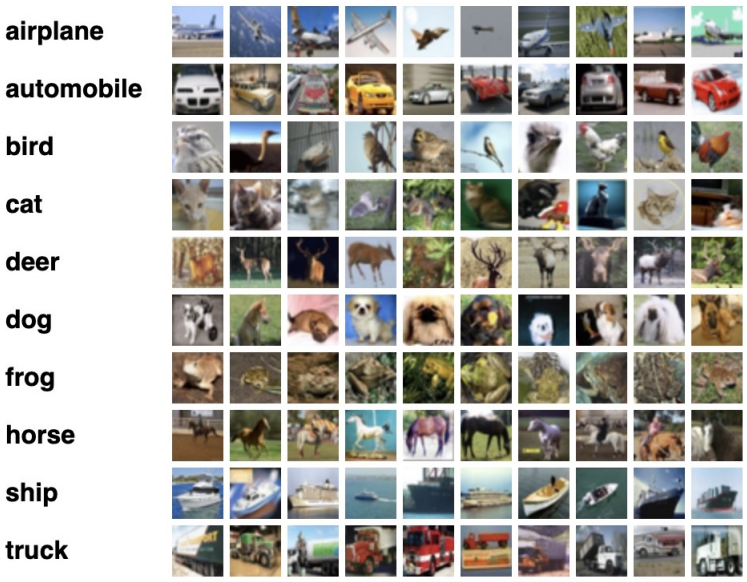
Za uspoređivanje DeiTa s ostalim mrežama i čovjekom korišten je modificirani repozitorij rada *Are Convolutional Neural Networks or Transformers more like human vision?* [24]. Proces testiranja i uspoređivanja mreža proveden je na besplatnoj platformi Google Colaboratory koja nudi besplatno korištenje grafičke kartice NVIDIA TESLA K80.

7.2. Skup podataka

Korišten je skup podataka CIFAR-10 za treniranje i evaluaciju DeiTa. Taj skup podataka je odabran zbog manje količine slika, ali i zbog manje veličine slika tako da je jednostavno baratati s njim, a učenje ne zahtijeva toliko vremena. CIFAR-10 je skup podataka koji se sastoji od 50000 slika u podskupu za treniranje i 10000 slika u podskupu za testiranje. Svaka slika je označena brojem od 0 do 9 u kojem svaki broj označava jedan razred (Slika 7.1) te su sve slike veličine 32x32.

Također je korišten i skup podataka dostupan preko repozitorija rada *Are Convolutional Neural Networks or Transformers more like human vision?* [24] za uspoređivanje

DeiTa s drugim mrežama i čovjekom.



Slika 7.1: Primjer slika i popis razreda CIFAR-10 skupa podataka. [12]

7.3. Arhitekture mreža

7.4. ResNet-18

Korišten je ResNet-18 [7] kao model učitelj pri treniranju DeiTa koji postiže točnost od 69.758% [5] na ImageNet skupu podataka. ResNet-18 je duboka rezidualna mreža koja koristi konvolucijske slojeve. Implementacija modela i predtrenirane težine su preuzete iz biblioteke *torchvision*. Modelu je promijenjen zadnji sloj gdje zamijenjujemo veličinu izlaza iz posljednjeg potpuno povezanog sloja s 1000 na 10 zbog činjenice da mrežu treniramo na skupu podataka CIFAR-10 u kojem svaka slika može biti jedan od deset različitih razreda.

7.5. DeiT

U radu su predstavljene 3 arhitekture DeiTa, jako mala (eng. *tiny*), mala (eng. *small*) i normalna (eng. *base*). Jako malu arhitekturu ćemo označavati sa DeiT-T, malu sa DeiT-S dok će normalna biti DeiT-B. Mala i normalna arhitektura se neće koristiti u svrhu ovoga radim osim u svrhu uspoređivanja rezultata. Svi rezultati koji se odnose na te arhitekture su preuzeti iz rada *Training data-efficient image transformers & dis-*

tillation through attention [23]. Za ovaj rad koristi će se jako mala arhitektura koja je i arhitektura s najmanje parametara.

Ta arhitektura je odabrana zbog orgraničenih resursa te i zbog toga što svrha ovog rada nije postići najbolje rezultate na skupu podataka CIFAR-10 nego pokazati da je arhitektura utemeljena na slojevima pažnje konkurentna s arhitekturama utemeljenim na konvolucijskim slojevima.

Razlike u modelima su prikazane u tablici (Tablica 7.1).

Ime	Dubina	Broj glava	Veličina vektora	Broj parametara
Tiny [23]	12	3	192	5M
Small [23]	12	4	384	22M
Base [23]	12	12	768	86M

Tablica 7.1: Usporedba osnovnih DeiT arhitektura

7.6. Postupak treniranja

Rezultat ovog rada su istrenirani modeli DeiT i ResNet-18 nad skupom podataka CIFAR-10. Modeli su trenirani metodom opisanom u radu *Training data-efficient image transformers & distillation through attention* te se temelje na implementaciji iz rada [23]. Svi rezultati treniranja su dobiveni treniranjem sa grupama veličine 128 nad 100 epoha.

7.6.1. Uvećanje podataka

Za uvećavanje podataka koristi se funkcija `build_function` iz biblioteke timm. Slike iz CIFAR-10 su inače 32x32, ali se pretvaraju u slike veličine 224x224 na kojima model onda uči. Nad slikama se odvijaju slučajne augumentacije: slučajno mijenjanje svijetline, kontrasta i zasićenja boje, *RandAugument* [2], slučajno brisanje [30], slučajno horizontalno i vertikalno okretanje slike te slučajna promjena veličine slike. Tada se primjenjuje i normalizacija slike s prosječnim vrijednostima piksela svakog kanala u ImageNet skupu podataka.

7.6.2. Treniranje s cutmix metodom

Eksperimentiranjem, se pokazalo da korištenjem metode *cutmix* [29] dobivaju se bolji rezultati nad ImageNet i CIFAR-10 skupom podataka. Modelu se poboljšava sposob-

nost generalizacije te samim time i točnost nad skupom za testiranje. U metodi *cutmix* slučajno se odabiru dijelovi slika iz skupa za treniranje te se postavljaju na druge slike. Stvarne labele tih slika su sada netočne pošto su se slike izmiješale te zato za vrijednost labele uzimamo više razreda koji proizlaze iz svih slika koje su spojene kako bi nastala nova slika te je vrijednost za tu labelu proporcionalna veličini dijelova slike iz promatranog razreda s obzirom na veličinu cijele slike.

Kada se trenira ovom metodom kao funkciju gubitka koristimo unakrsnu entropiju s mekim labelama koja se razlikuje od unakrsne entropije po tome da labele ne mogu biti samo 0 i 1 nego mogu imati vrijednosti od 0 do 1.

Slika 7.2: Primjer primjene *cutmix* metode. Sada bi oznake za sliku bile: Pas 0.6, Mačka:0.4. [29]

7.6.3. Optimizator

Kao optimizator koristi se AdamW [14] te se hiperparametar stope učenja mijenja pomoću propadanja kosinusom s ponovnim pokretanjem.

7.6.4. Treniranje ResNet-18 i AtResNet

Preuzima se arhitektura ResNet-18 s težinama koje su dostupne preko biblioteke torchvision. Težine su dobivene treniranjem ResNet-18 nad skupom podataka ImageNet.

Prije treniranja slike se obrađuju postupkom augmentacije podataka opisanim u prethodnom ulomku te se koristi *cutmix* metoda pri treniranju. Optimizator je opisan u prijašnjem ulomku.

7.6.5. Treniranje DeiT

Nakon opisa treniranja ResNet-18 i nakon dobivenog naučenog modela nad skupom podataka Cifar-10, slijedi treniranje DeiTa. Treniranje DeiTa je kompliciranije te osim same arhitekture ViT ono, u procesu učenja, sadrži i prethodno naučeni ResNet-18. DeiT se počinje trenirati s težinama preuzetim iz rada [23] koje su dobivene treniranjem nad ImageNet skupom podataka koristeći RegNetY-16GF [19] kao model učitelj.

DeiT se trenira s prethodno opisanim postupkom augmentacije podataka te je postupak identičan kao i kod treniranja ResNet-18 arhitekture. Ono po čemu se razlikuje treniranje DeiTa je funkcija gubitka. Odabiremo učenje s čvrstim oznakama, gdje su obje funkcije gubitka u *hard-label lossu*, unakrsna entropija s mekim labelama.

8. Eksperimenti

8.1. Rezultati eksperimenata

8.1.1. Naši modeli

U našim eksperimentima trenirali smo modele DeiT-T i ResNet-18. Rezultati i usporedba modela nalaze se u sljedećoj tablici (Tablica 8.1). Kod DeiT-T oba modela su predtrenirana nad ImageNetom postupkom destilacije. Model učitelj kod predtreniranog modela je RegNetY-16GF [19] treniran na ImageNet skupom podataka. DeiT-T$_p$ je treniran nad CIFAR-10 skupom podataka kao da je ViT koristeći samo razredni token bez destilacije.

Ime modela	Predtreniran na ImageNetu	Broj parametara	Model učitelj	Cifar-10
DeiT-T$_p$	Da	5M	Nema	96.32%
DeiT-T$_{dp}$	Da	5M	ResNet-18$_p$	97.10%
ResNet-18$_p$	Da	11M	Nema	95.32%
ResNet-18	Ne	11M	Nema	92.1%
AtResNet-18$_p$	Da	16M	Nema	95.13%

Tablica 8.1: Svi modeli koji imaju oznaku $_p$ su trenirani iz preuzetih težina koje su dobivene treniranjem modela na ImageNetu. Oznaka $_d$ označava da je model treniran postupkom destilacije znanja koristeći čvrste oznake nad skupom podataka CIFAR-10.

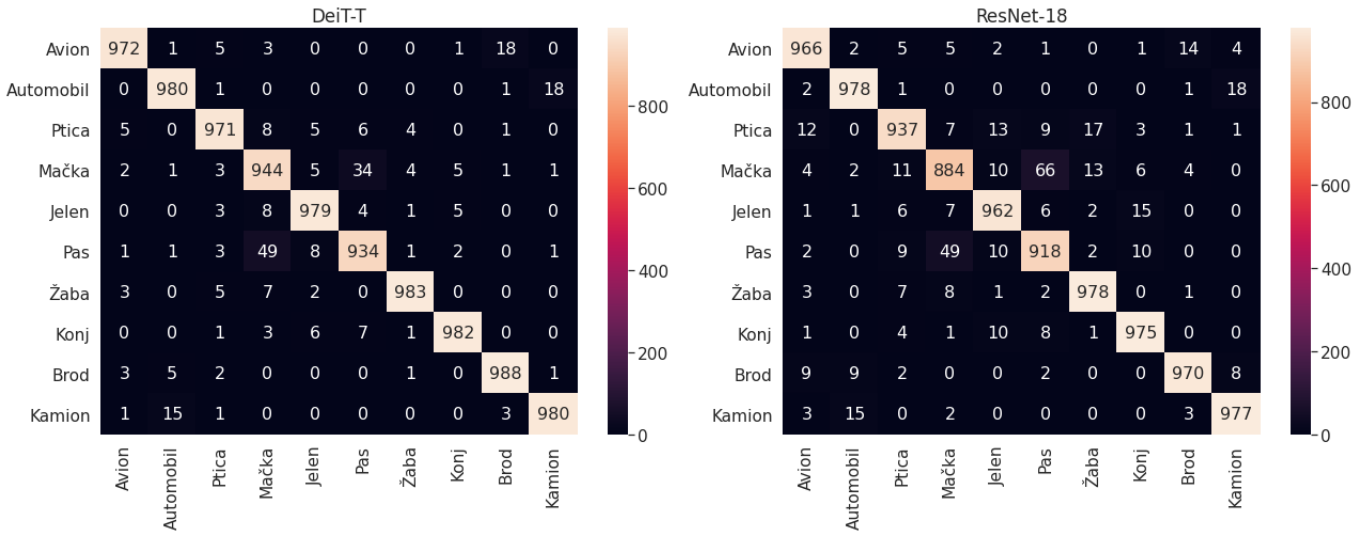
Dva modela koji daju najbolje rezultate su DeiT-T$_{dp}$ i ResNet-18$_p$ te ćemo u daljnjem radu njih uspoređivati s ostalim modelima i međusobno.

Ime modela	Broj parametara	Cifar-10	img/sec	Veličina na disku	CUDA alokacija memorije
DeiT-T$_{dp}$	5M	97.1%	143.25	88.7MB	5001MB
ResNet-18$_p$	11M	95.32%	227.03	179.1MB	2782MB
AtResNet-18$_p$	16M	95.13%	170.81	268.5MB	3193MB

Tablica 8.2: Usporedba DeiT-T$_{dp}$, ResNet-18$_p$ i AtResNet-18$_p$. Broj slika po sekundi mjeri se algoritmom opisanom u radu [17] dok se veličine alocirane CUDA memorije računa kao maksimalna veličina alocirane memorije na uređaju tijekom inferencije minus alocirana memorija prije inferencije. Analiza alokacije memorije se provodila s grupama veličine 128 slika.

DeiT-T$_{dp}$ posjeduje bolju točnost te manje parametara što znači i manju veličinu na disku dok ResNet-18$_p$ je 1.63x brži od DeiT-T$_{dp}$ te ResNet-18$_p$ alocira 1.5 puta manje CUDA memorije. Dobivene metrike za brzinu i alokaciju memorije su možda neočekivane kada se gleda samo broj parametara, no zbog već spomenutog nedostatka pažnje, a to je kvadratna vremenska i prostorna složenost, to su rezultati koji nas zapravo nisu iznenadili. Ono što je iznenađujuće je da iako DeiT-T$_{dp}$ ima dvostruko manje parametara od ResNet-18$_p$a ipak postiže bolje rezultate. Destilacija znanja se najčešće koristi kako bi se s većim modelom naučio manji model te bi tada manji model imao slabije performanse, ali veću brzinu te bi zauzimao manje prostora. To u ovom slučaju nije tako jer DeiT postiže bolje rezultate, ali je sporiji i zahtjeva više resursa za treniranje i zaključivanje.

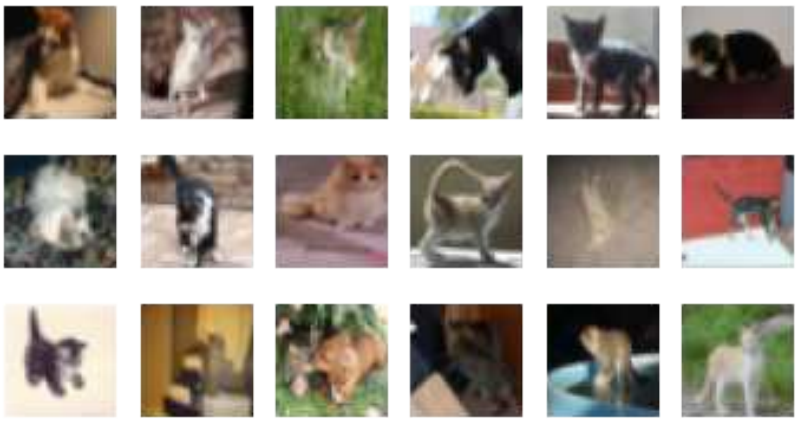
Iz slike (Slika 8.1) se lako uočava da DeiT-T$_{dp}$ klasificira svaki razred bolje od ResNet-18$_p$ te to pogotovo uočavamo kod razreda "Mačka" gdje DeiT-T$_{dp}$ klasificira 944/1000 slika mačke točno dok ResNet-18$_p$ klasificira 884/1000 točno. Razred "Mačka" je primjer razreda kojeg je teško klasificirati u ovom skupu podataka te to može dati indikacije da arhitektura koja koristi pažnju bolje klasificira tako teške primjere (Slika 8.2).



(a) Matrica zabune za DeitT$_{dp}$

(b) Matrica zabune za ResNet-18$_p$

Slika 8.1: Matrice zabune za DeitT$_{dp}$ i ResNet-18$_p$.



Slika 8.2: Slike u prvom redu je ResNet-18$_p$ krivo klasificirao, a DeiT-T$_{dp}$ točno. U drugom redu su prikazane slike za koje je DeiT-T$_{dp}$ bio netočan, a ResNet-18$_p$ točan. U zadnjem redu se nalaze slike koje su krivo klasificirane od strane oba modela.

8.1.2. Usporedba rezultata s ostalim modelima

U tablici (Tablica 8.3) su prikazani rezultati naših modela, ali i ostalih modela preuzetih iz drugih radova.

Kao što smo mogli i očekivati točnost naših arhitektura je manja od ostalih, no tu moramo zamijetiti manju količinu parametara naših modela.

Modeli DeiT-B$_p$, ViT-B/16, ViT-L/16 i EfficientNet-B7 su trenirani nad CIFAR-10

tako što su trenirani na ImageNet skupu podataka te onda i na CIFAR-10. Na takav način smo dobili i rezultate za naše modele ResNet-18$_p$ i DeiT-T$_p$.

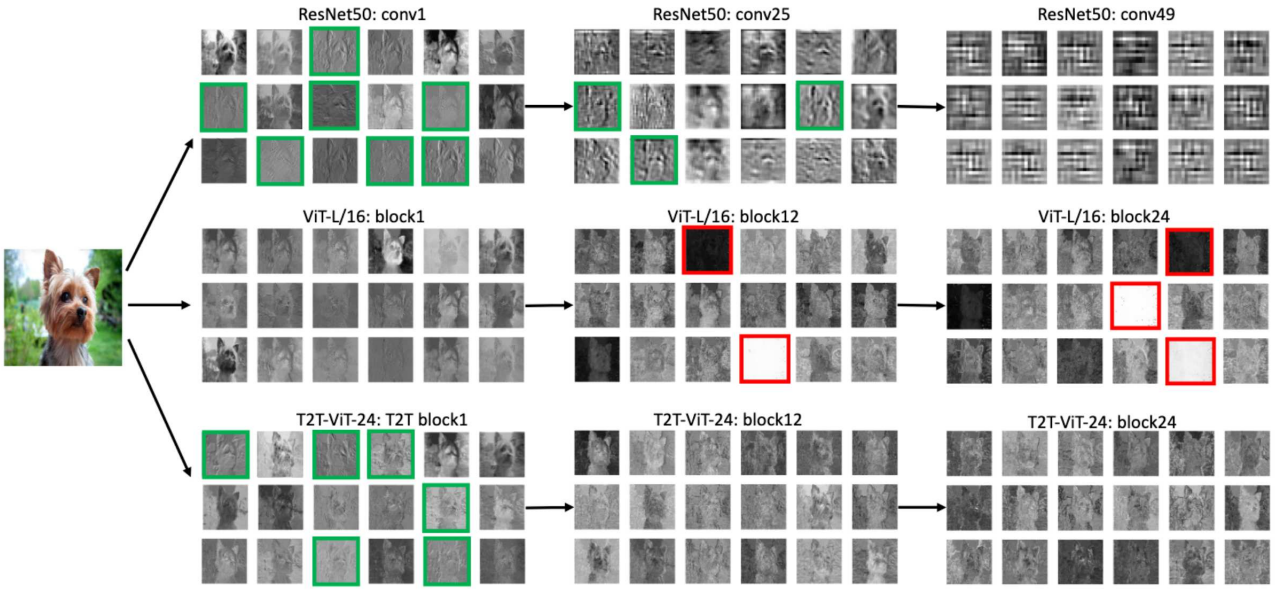
Oba naša modela su puno brža što se tiče vremena obrade slike po sekundi zbog puno manjeg broja parametara koje sadržavaju.

Ime modela	Broj parametara	Cifar-10	img/sec
DeiT-T$_{dp}$	5M	97.1%	143.25
ResNet-18$_p$	11M	95.32%	227.03
DeiT-B$_p$ [23]	86M	99.1%	90.57
ViT-B/16 [4]	86M	98.1%	90.63
ViT-L/16 [4]	307M	97.9%	30.93
EfficientNet-B7 [21]	66M	98.9%	27.12

Tablica 8.3: Točnost klasifikacijskih modela na CIFAR-10.

8.1.3. Usporedba značajki

U radu *Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet* [28] je predstavljena usporedba vizualiziranih značajki ViT-L/16, ResNet-50 te T2T-ViT-24, modela koji je bio rezultat spomenutog rada. Poznato je bilo da ViT dobiva kompetitivne rezultate samo nad velikim podskupovima podataka, većim od ImageNet-a koji ima oko 1.2M slika te se uspjelo pokazati pomoću vizualizacije podataka zašto je to tako. Pokazalo se da ViT zanemaruje lokalne strukture niske razine te općenito ne uči dovoljno dobro strukturu, no sadržava globalne relacije zbog globalnog receptivnog polja [28].

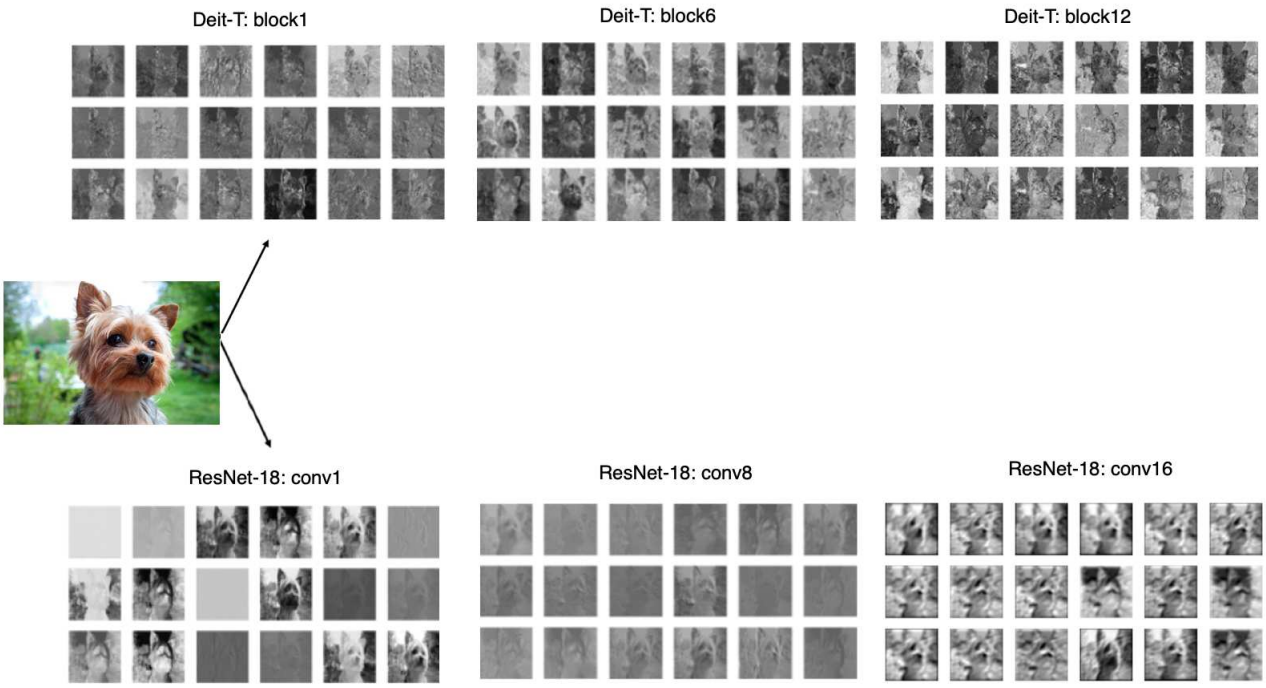


Slika 8.3: Vizualizacija značajki. Zelenom bojom označene su značajke koje su naučile strukturu niske razine dok su crvenim označene značajke koje su ili potpuno bijele ili crne što znači da nisu korisne. Slike je preuzeta iz rada [28]. Vizualizacija se provodi prikazivanjem ulazne slike nakon što prođe kroz određeni sloj te se uzima samo prvi kanal dobivene slike u slučaju ResNeta.

Iz slike (Slika 8.3) možemo uočiti da ni ResNet50 ni T2T-ViT-24 nemaju ni jednu značajku označenu crvenom bojom dok ih ViT ima više. To je zato što je ViT neefikasan u učenju strukture te je zato potrebna ogromna količina podataka kako bi ViT pružio slične rezultate kao konvolucijske mreže slične veličine.

Usporedit ćemo sada naše modele ResNet-18_p i DeiT-T_{dp}. Nažalost zbog ograničenih resursa ViT nismo mogli istrenirati nad ImageNet skupom podataka te ćemo tako uzeti vizualizaciju značajki iz slike (Slika 8.3).

Postupak generiranja vizualizacija značajki napravljen je po programskoj implementaciji dostupnoj uz rad [28] te je ona modificirana kako bi se dobili rezultati za DeiT-T_{dp} i ResNet-18. Uzeta je ista slika psa kao i u slici (Slika 8.3) kako bi se prikazala vizualizacija. Iako su modeli sa slika (Slika 8.3) trenirani na ImageNetu eksperimentiranjem je utvrđeno nema većih razlika između vizualizacija značajki kod ResNet-18_p i DeiT-T_{dp} kada su trenirane samo na ImageNet-u i kada su trenirane dodatno i na CIFAR-10 skupu podataka.



Slika 8.4: Vizualizacija značajki DeiT-T$_{pd}$ i ResNet-18$_p$. Slika se skalira na veličinu 1024x1024 kao i u Slika 8.3 kako bi lakše analizirali dobivene značajke.

Iz Slika 8.4 vidimo ne samo da DeiT-T nema problema kojih ViT ima (Slika 8.3). DeiT-T nema beskorisnih značajki koje su potpuno bijele ili crne što je još jedan dokaz da je DeiT-T efikasniji od ViT i da mu nije potreba toliko količina podataka. Zamjećujemo da se značajke ViTa i DeiT-T dosta razlikuju iako je arhitektura vrlo slična. Značajke DeiT-T su između onih od konvolucijske mreže i ViTa što ukazuje na tvrdnju iz rada [22] kako DeiT sadrži karakteristike i *transformera* i konvolucijske mreže.

Također ćemo vizualizirati pažnju (Slika 8.5) metodom predstavljenom u radu [4]. Vidimo da je DeiT dobro prepoznao strukturu psa na slikama (Slika 8.5) te obraća pažnju na psa dok pozadinu zanemaruje.

Slika 8.5: Vizualizacija pažnje DeiT-T$_{dp}$. Dohvaćaju se vjerojatnosti poslije provođenje operacije *softmax*, a prije množenja s tenzorom V u pažnji te se uzima srednja vrijednost vjerojatnosti od svih glava. Uzima se samo prvi red matrice vjerojatnosti te se taj red pretvori u matricu dimenzija PxP gdje je P dimenzija slike podijeljena s veličinom kvadratnog okna. Tako dobivena matrica se skalira na veličinu slike i množi sa slikom. Tamo gdje su veće vjerojatnosti, tamo će slika biti svjetlija.

8.2. Usporedba ResNet, ViT i DeiT s obzirom na čovjeka

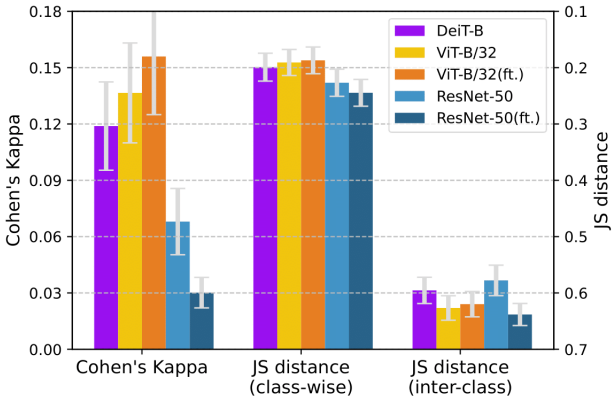
Često postavljano pitanje u umjetnoj inteligenciji je kako ona dolazi do predikcija te koliko se taj način razlikuje od onoga promatranog kod ljudi. To pitanje je tema rada *Are Convolutional Neural Networks or Transformers more like human vision?* [24] koji pokušava dati odgovor na pitanje koji tip arhitekture je sličniji čovjeku u raspoznavanju slika. Fokus je na usporedbu ResNeta i ViTa te se koriste određene metode i metrike, koje ćemo opisati u idućem ulomku, kako bi se došlo do odgovora na to pitanje.

U malo prije spomenutom radu spominje se da je većina usporedbi temeljeno na točnosti ili nekim drugim lako mjerljivim metrikama s kojima se često uspoređuju

dvije mreže, no to je možda i nije pravi pristup. Raspoznavanje slika je kompleksan zadatak te je to zadatak kojeg ljudi jako dobro rješavaju, puno bolje od umjetne inteligencije. Možda ne bi bilo loše probati naučiti od ljudi te pokušati primijeniti metode koje koriste ljudi u umjetnoj inteligenciji. Također se navodi da iako ima jedan način da model bude točan, postoji mnogo različitih načina na koje model može biti kriv. To je ključno razmišljanje ovoga rada jer se metode za mjerenje sličnosti baziraju na analizi pogrešaka mreže. Uspoređivanje grešaka mreža može dovesti do zanimljivih spoznaja i zaključaka.

8.2.1. Rezultati

Uzeti su rezultati iz rada [24] te su dodani rezultati DeiT-B/16 arhitekture kako bi se napravila usporedba. Arhitekture korištene u ovoj analizi su: ResNet-50, ViT-B/32 te DeiT-B/16 (/32 i /16 označavaju veličinu kvadratnih okna na koje se slika dijeli). ResNet-50 i Deit-B/16 su pretranirane na ImageNet skupu podataka ILSVRC-2012, a ViT-B/32 na skupu podataka ImageNet-21k. [20]. Također je korišten skup podataka *Stylized ImageNet* koji sadrži slike iz ImageNet skupa podataka u kojima su teksture i oblici promijenjeni [6]. ResNet-50(ft.) i ViT-B/32(ft.) označavaju mreže koje su dodatno trenirane koristeći augmentacije slika: slučajna rotacija, slučajno brisanje dijelova slike, *Sobel* filter, zamućivanje, distorcija boja i dodavanje šuma. [24]



Slika 8.6: Konzistentnost grešaka na *Stylized ImageNet* skupu podataka. Slika je preuzeta iz rada [24] te su dodani rezultati za DeiT-B.

Na slici (Slika 8.6) se nalaze tri već spomenute metrike s kojima ćemo analizirati pet modela. Kohenova kapa upućuje na to koliko se slažu dva sustava u svojim odlukama. Kohenova kapa se dobila uspoređivajući rezultate mreža s ljudskim rezultatima dostupnima u repozitoriju *Are Convolutional Neural Networks or Transformers more*

like human vision? rada. U inicijalnoj analizi [24] zaključuje se po Kohenovoj kapi da je ViT sličniji ljudima nego ResNet zbog veće vrijednosti Kohenove kape. To je i zaključak kojeg možemo navesti i za DeiT-B koji je isto tako konzistentniji s ljudima od ResNeta, no ne toliko koliko je ViT-B/32.

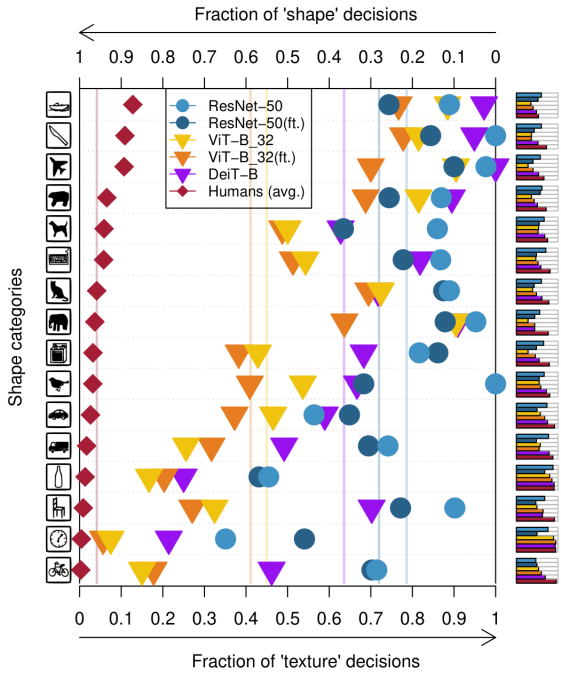
Sada se uspoređuje razredna JS udaljenost (*engl. JS distance (class-wise)*. Manja vrijednost razredne JS udaljenosti je indikator veće konzistentnosti između promatranih sustava. Dobivamo iste rezultate kao i kod Kohenove kape gdje je ViT najkonzistentniji s obzirom na ljude, slijedi ga DeiT pa onda ResNet.

Zadnja vrijednost koju ćemo proučavati je vrijednost međurazredne JS udaljenosti (*engl. JS distance (inter-class)* te se u inicijalnoj analizi [24] navodi iznenađujuća činjenica da je udaljenost veća kod ResNeta nego kod ViTa. Iz slike se također može očitati da je udaljenost kod DeiTa također viša nego kod ViTa, ali još uvijek je manja od ResNeta. Što je indikator da ResNet možda i posjeduje neka svojstva koja ga čine sličnijim čovjeku, no dodatnim eksperimentima pokazalo se da dodatnim treniranjem mreža s augumentacijama podataka (ft. mreže) smanjuje sličnost ResNeta s ljudima dok kod ViTa povećava tu sličnost.

Ono na što ćemo se fokusirati su rezultati dobiveni za DeiT-B. DeiT-B je *transformer* arhitektura učena destilacijom znanja gdje je konvolucijska mreža učitelj te je zato negdje između *transformera* i konvolucijske mreže gdje je klasifikator povezan s destilacijskim tokenom sličniji konvolucijskoj mreži dok je klasifikator povezan s razrednim tokenom sličniji *transformeru* [23]. Prijašnjom analizom možemo potvrditi te pretpostavke o sličnosti. Analizom se pokazalo da je DeiT u svim mjerama između ViTa i ResNeta što indicira da je DeiT stvarno između konvolucijske mreže i *transformera* te da posjeduje svojstva od obje arhitekture.

Slika (Slika 8.7) predstavlja sklonost različitih modela da predikciju donesu na temelju oblika ili teksture. Eksperimentima se pokazalo da konvolucijske mreže daju prednost teksturi kod klasifikacije objekta dok se ljudi više pouzdaju u oblik [1]. Zato je analizirana razlika između klasifikacije ResNeta i ViTa u usporedbi s čovjekom [24]. Sklonost obliku se računa nad skupom podataka *Stylized ImageNet* u kojem se nalaze slike objekata gdje se oblik i tekstura objekta ne podudaraju. Provodimo klasifikaciju slika iz skupa podataka *Stylized ImageNet* sa svakim modelom te uspoređujemo udio slika u kojima je ispravno klasificiran oblik nasuprot onih gdje je ispravno klasificirana tekstura. Iz slike (Slika 8.7) je uočljivo da je ViT puno bliži čovjeku što se tiče sklonosti obliku te opet vidimo slučaj kao i u prijašnjoj analizi (Slika 8.6) gdje se DeiT nalazi između ResNeta i ViTa. U radu [24] se isto daje opaska na to da sklonost obliku objašnjava veću vrijednost razredne JS udaljenosti i Kohenove kape kod

ViTa, no ne objašnjava veću vrijednost međurazredne JS udaljenosti kod ResNeta. Dodatnim treniranjem ResNeta s augmentacijama dobivamo veću sklonost obliku, a manju međurazrednu JS udaljenost. To je zanimljiva činjenica jer je očekivan rezultat da povećanjem sklonosti obliku objekta dobivamo veću sličnost s ljudima što nije kod ResNeta nije slučaj. Tesitranje sklonosti obliku uzima u obzir samo primjere gdje su ili oblik ili tekstura točno klasificirani što znači da ne sadržava većinu netočnih klasifikacija koja je vidljiva kod ResNeta. U radu [24] se baš ta činjenica spominje kao moguće objašnjenje veće vrijednosti međurazredne JS udaljenosti, a manje vrijednosti JS udaljenosti i Kohenove kape kod ResNeta.



Slika 8.7: Sklonost obliku objekta za ResNet, ViT i DeiT. Vertikalne linije predstavljaju srednje vrijednosti. [24]

9. Zaključak

Cilj ovog rada bio je istražiti primjenu sloja pažnje u računalnom vidu, specifično za raspoznavanje slika. Za tu svrhu odabrana je arhitektura ViT koja sadrži slojeve pažnje te je ona korištena u eksperimentima.

Istražene su prednosti korištenja pažnje u računalnom vidu kao što je manja induktivna sklonost, globalno receptivno polje, odlična skalabilnost. Također su istražene i negativne strane pažnje, kvadratna vremenska i prostorna složenost pažnje te potreba za velikom količinom podataka za treniranje. Proučene su arhitekture koje sadrže sloj pažnje te su detaljnije proučene arhitekture ViT i DeiT. Istraženi su razlozi zašto se pažnja ne koristi toliko često u računalnom vidu koliko u obradi prirodnog jezika. Proveli smo eksperimenti u kojima su naučene mreže DeiT-T i ResNet-18 te su se rezultati usporedili s onima iz originalnog rada koji je predstavio DeiT [23]. Rezultati su potvrdili činjenicu da se DeiT-T može koristiti za svrhu raspoznavanja slika. Potvrdili smo i probleme kod arhitekture bazirane na slojevima pažnje kao što su kvadratna vremenska i prostorna složenost pažnje. Daljnjim eksperimentima istraženi su načini kako pažnja funkcionira u računalnom vidu te su uspoređene arhitekture ResNet-18 i DeiT-T međusobno, ali i s rezultatima drugih arhitekture predstavljenim u različitim radovima [11] [28].

Usporedio se način na koji ResNet, DeiT i ViT raspoznaju slike s obzirom na čovjeka te su potvrđene neke pretpostavke o DeiT arhitekturu koje je imao rad koji je originalno predložio tu arhitekturu [23]. Uspoređivanjem rezultata za DeiT i rezultata iz rada [24] potvrđena je prepostavka da je DeiT po načinu raspoznavanja slike između konvolucijske mreže i transformer arhitekture.

LITERATURA

[1] Nicholas Baker, Hongjing Lu, Gennady Erlikhman, i Philip Kellman. Deep convolutional networks do not classify based on global object shape. *PLOS Computational Biology*, 14:e1006613, 12 2018. doi: 10.1371/journal.pcbi.1006613.

[2] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, i Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, i Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, i Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[5] Facebook. Torchvision models. https://pytorch.org/vision/stable/models.html, 2021.

[6] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, i Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2019.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition, 2015.

[8] Geoffrey Hinton, Oriol Vinyals, i Jeff Dean. Distilling the knowledge in a neural network, 2015.

[9] Sepp Hochreiter i Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] Eunkwang Jeon. Vit-pytorch. https://github.com/jeonsworld/ViT-pytorch, 2020.

[11] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, i Mubarak Shah. Transformers in vision: A survey, 2021.

[12] Alex Krizhevsky, Vinod Nair, i Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[13] Yunpeng Chen Li Yuan. T2t-vit. https://github.com/yitu-opensource/T2T-ViT, 2021.

[14] Ilya Loshchilov i Frank Hutter. Decoupled weight decay regularization, 2019.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, i Jeff Dean. Distributed representations of words and phrases and their compositionality. U C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, i K. Q. Weinberger, urednici, *Advances in Neural Information Processing Systems*, svezak 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[16] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38 (11):39–41, Studeni 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748.

[17] Marin Oršić i Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition*, 110:107611, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2020.107611. URL https://www.sciencedirect.com/science/article/pii/S0031320320304143.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, i Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. U *Advances in Neural Information Processing Systems 32*, stranice 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-l
pdf.

[19] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, i Piotr Dollár. Designing network design spaces, 2020.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, i Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[21] Mingxing Tan i Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[22] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, i Hervé Jégou. https://github.com/facebookresearch/deit, 2021.

[23] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, i Hervé Jégou. Training data-efficient image transformers and distillation through attention, 2021.

[24] Shikhar Tuli, Ishita Dasgupta, Erin Grant, i Thomas L. Griffiths. Are convolutional neural networks or transformers more like human vision?, 2021.

[25] Guido Van Rossum i Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, i Illia Polosukhin. Attention is all you need, 2017.

[27] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[28] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, i Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet, 2021.

[29] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, i Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

[30] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, i Yi Yang. Random erasing data augmentation, 2017.

Sažetak

Raspoznavanje slika važno je područje računalnog vida kojim dominiraju duboke konvolucijske mreže godinama, no nova arhitektura mreža temeljena na slojevima pažnje postaje sve popularnija u dubokom učenju osobito na području obrade prirodnog jezika. U ovom radu istražen je mehanizam pažnje i arhitekture koje ga koriste, istraženo je kako implementirati takve arhitekture za raspoznavanje slika te koje su prednosti i nedostatci. Implementirana je jedna takva arhitektura te su nad njom provedeni eksperimenti kako radi takva mreža te usporedbe s drugim popularnim mrežama u području raspoznavanja slika u računalnom vidu.

Ključne riječi: pažnja, računalni vid, raspoznavanje slika, transformer.

Image Classification using self-attention

Abstract

Image recognition is an important area of computer vision dominated by deep convolutional networks for years but the new attention-based network architecture is becoming increasingly popular in deep learning especially in the field of natural language processing. In this paper, the mechanism of attention and the architectures that use it are investigated, it is investigated how to implement such architectures for image recognition and what are the advantages and disadvantages. One such architecture has been implemented and experiments on how such a network works and comparisons with other popular networks in the field of computer image recognition have been performed.

Keywords: attention, computer vision, image classification, transformer.