SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1162

SEMANTIČKA SEGMENTACIJA PRIMJENOM JEZIČNIH UGRAĐIVANJA

Naomi Kombol

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1162

SEMANTIČKA SEGMENTACIJA PRIMJENOM JEZIČNIH UGRAĐIVANJA

Naomi Kombol

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 1162

Pristupnica:	**Naomi Kombol (0036533595)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Semantička segmentacija primjenom jezičnih ugrađivanja**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim zanimljivim primjenama. Međutim, standardno nadzirano učenje osjetljivo je na odabir taksonomije skupa za učenje. Ovaj problem možemo umanjiti gustom predikcijom jezičnih ugrađivanja. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće segmentacijske arhitekture utemeljene na konvolucijama i pažnji. Oblikovati segmentacijski postupak primjenom prednaučenog modela koji povezuje vizualne koncepte s jezičnim ugrađivanjima. Odabrati slobodno dostupne skupove slika te oblikovati podskupove za učenje, validaciju i testiranje. Odabrati prikladni model te validirati hiperparametre. Vrednovati naučene modele te prikazati i ocijeniti postignutu točnost. 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: 9. lipnja 2023.

Zahvaljujem majci i prijateljima na zavidnom pothvatu tjeranja me na odmor i zabavu tijekom ovog semestra. Zahvaljujem i mentoru prof. dr. sc. Siniši Šegviću na beskrajnom strpljenju s mojim mejlovima i istinskoj želji da podijeli svoje bogatstvo znanja kao i mag. ing. Petri Bevandić na svoj ukazanoj pomoći, a posebno što mi je pomogla u lovljenju blesavih grešaka po kodu.

SADRŽAJ

1. Uvod

Umjetna inteligencija danas uživa posebnu pozornost i rastuću važnost u svim aspektima naših života. Računalni vid, iako je tek dio te široke discipline, ima kao zadatak oponašati jednu od najosnovnijih sastavnica ljudskog života - naše oči. Omogućava kompjuterima izlučivanje i razumijevanje značajki slika te daljnji rad s njima.

Semantička segmentacija specifično se bavi klasificiranjem svakog piksela, no u praksi, modeli trenirani za ovakve zadatke vrlo su osjetljivi na izbor klasa. Prijenos znanja iz istih na, čak i samo malo, drugačije označene podatke zahtijeva ili ljudsku intervenciju ili rezultira padom performansi. Korištenje jezičnih ugrađivanja za te guste predikcije nastoji doskočiti tom problemu te povećati prijenosnu moć naučenih modela [15, 24]. Naime, kod klasifikacije onda više ne baratamo s čistim klasama, već njihovim apstraktnim reprezentacijama koje odražavaju njihovu suštinu npr. mreža trenirana za semantičku segmentaciju "Auta" od pozadine ne bi znala što učiniti ako joj predamo "Vozilo" kao upit, no ista ta mreža učena s jezično ugrađenim "Autom" vjerojatno bi izbacila nešto dovoljno dobro na upit "Vozilo" jer ta dva pojma bi trebala imati sličnu apstraktnu reprezentaciju.

Ovaj rad bavit će se korištenjem jezičnih ugrađivanja postojećeg CLIP modela iz implementacije [6], predtreniranog na podskupu LAION [1] skupa podataka, za semantičku segmentaciju s jezičnim ugrađivanjima. Slikovna okosnica tog, a i modela u radu, bit će ConvNext-Base [16]. Njega ćemo koristiti za ekstrahiranje značajki podslojeva te, povezujući te podslojeve sa naduzorkovateljima po uzoru na *single-scale* model iz [18], kombinirano ih naduzorkovati do početne rezolucije gdje ćemo, konačno, tu višekanalnu reprezentaciju pretvoriti u semantički segmentiranu sliku koristeći kosinusnu sličnost (objašnjeno kasnije) s jezično ugrađenim klasama. Razmotrit ćemo u kojoj mjeri je moguće maksimalno iskoristiti ConvNext-Base okosnicu tako što ćemo prvo trenirati samo parametre naduzorkovatelja, a tek kasnije odmrznuti cijelu mrežu. Također ćemo istražiti razlike između korištenja originalnih naziva klasa i kratkih opisa sastavljenih iz prvih nekoliko paragrafa stranica Wikipedije za spomenute klasa.

[1] https://laion.ai/blog/laion-5b/

2. Neuronske mreže

Umjetne neuronske mreže stvorene su na sliku bioloških mozgova čime se nastoji oponašati njihova sposobnost učenja i daljnjeg korištenja stečenog razumijevanja. Čine ih mnoštva međusobno povezanih umjetnih neurona koji omogućavaju distribuiranu paralelnu obradu podataka. Ovakvo modeliranje arhitekture odlika je konektivističkog pristupa razvoju umjetne inteligencije [25].

Postizanje prije spomenutih sposobnosti zahtijeva ugađanje parametara neuralne mreže na zadatak koji želimo da izvršava. Osnovni pregled postupka nadziranog učenja se sastoji od iterativnog predočavanja nasumičnih parova ulaznih podataka i očekivanih izlaza iz skupa za treniranje te podešavanja parametara mreže radi približavanja generiranog izlaza očekivanom izlazu. Razliku između generiranog i očekivanog izlaza mjerimo funkcijom gubitka. Optimizacija se najčešće provodi gradijentnim spustom koji, koristeći gradijente funkcije gubitka u ovisnosti o parametrima, ugađa mrežu.

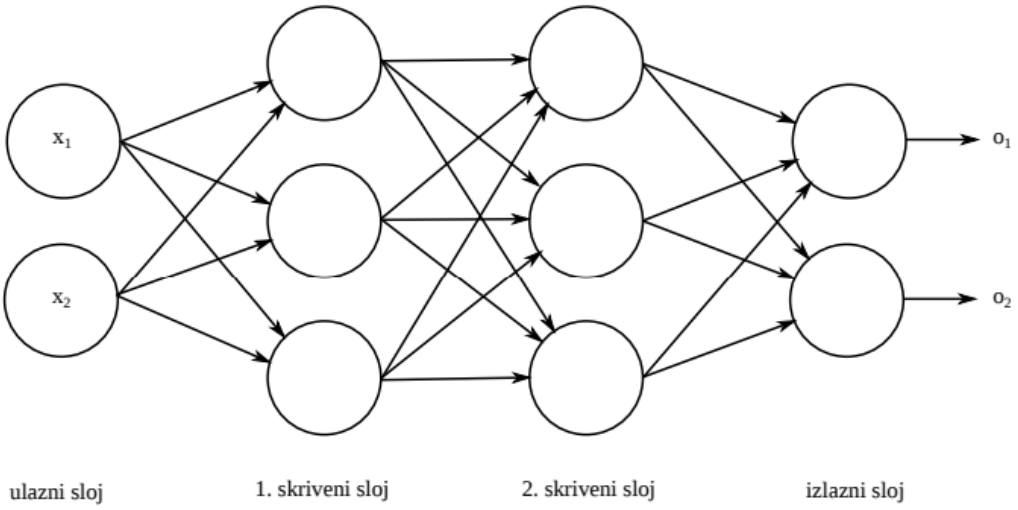
2.0.1. Duboki modeli

Duboki modeli pokazali su se kao moćan alat rješavanja zadataka umjetne inteligencije zbog svojeg superiornog kapaciteta, koji ostvaruju stavljanjem više od jednog skrivenog sloja (slika 2.1) između ulaznih i izlaznih, i općenitog napretka računalne snage. Prebacivanjem kalkulacija na grafičke kartice pomoću CUDA tehnologije [13], značajno se smanjilo vrijeme treniranja i pospješila mogućnost iskorištavanja sve većih skupova podataka.

2.1. Učenje neuronskih mreža

2.1.1. Funkcije gubitka

Funkcije gubitka su načini za kvantificiranje odstupanja predviđanja našeg modela od željenog izlaza. Njihovom minimizacijom približavamo model željenom ponašanju.



Slika 2.1: Primjer dubokog modela s 2 skrivena sloja koji ulazne, dvodimenzionalne podatke transformira u dvodimenzionalni izlaz (preuzeto iz [25])

Ovdje ćemo opisati jednu od najčešćih funkcija gubitka na primjeru mreža koje klasificiraju u C klasa i na izlazu daju C-dimenzionalni vektor predviđanja, no, prije toga, spomenuti ćemo softmaks funkciju zbog uloge u njenom izračunu.

Softmaks

Ako izlaz naše mreže kao aktivacijsku funkciju ima funkciju identiteta, onda, da bi taj izlaz nekako stavili u međusoban odnos vjerojatnosti, koristimo softmaks. Njegovom primjenom skaliramo C-dimenzionalni u ponovno C-dimenzionalni vektor čiji elementi leže u intervalu [0, 1] te čija suma iznosi jedan. Drugim riječima, skaliramo ga u vektor vjerojatnosti. Ako ovdje uzmemo da je taj izlazni vektor jednak $\vec{y} = [y_0, y_1, ...y_C]$ gdje svaki element predstavlja izlaz za pojedinu klasu, onda se pojedinačni elementi vektora nakon korištenja softmaksa računaju po sljedećem izrazu.

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{C} e^{y_j}} \quad for\ i = 1, 2, \ldots, C \tag{2.1}$$

Dodatna modifikacija softmaksa jest temperatura. Ovisno o svom iznosu, modificira raspodjelu vrijednosti elemenata vektora $\vec{y}$ prije provlačenja kroz softmaks tako da krajnja distribucija izgleda više uniformno ($T > 1$) ili da se više naglašavaju ekstremi ($T < 1$). Prva konfiguracija pospješuje vjerojatnost da se odabere neka opcija u koju je mreža manje sigurna, dok druga dodatno izoštrava sigurnost mreže.

$$Temperature\ Softmax(y_i) = \frac{e^{\frac{y_i}{T}}}{\sum_{j=1}^{C} e^{\frac{y_j}{T}}} \quad for\ i = 1, 2, \ldots, C \tag{2.2}$$

Unakrsna entropija

Unakrsna entropija vrlo je česta funkcija gubitka kod zadataka klasifikacije. Računa se po sljedećoj formuli gdje je x pojedina instanca, a t_i, za c od 1 do C, 1 ili 0 ovisno pripada li x klasi y_i ili ne. Bitno je da baratamo s vjerojatnostima, stoga se kod učenja, ako mreža kao prijenosnu funkciju za izlazne neurone koristi identitet, izlaz prvo provlači kroz funkciju softmaks.

$$CE(x) = -\sum_{i=1}^{C} t_i \log(p(y_i)) \tag{2.3}$$

2.1.2. Optimizacija modela

Optimizacija modela provodi se algoritamskim svođenjem funkcije gubitka na minimum. Iterativno računamo pogrešku te istu propagiramo unatrag (eng. *backpropagation*) tako podešavajući težine slojeva mreže.

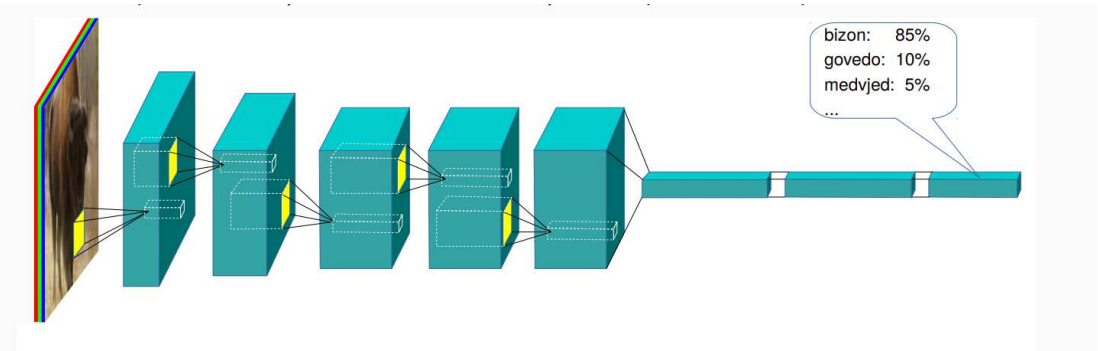
Gradijentni spust

Tijekom minimiziranja greške zapravo nastojimo pronaći, u idealnom slučaju, globalni minimum funkcije, no to nije uvijek moguće. Gradijentni spust je algoritam za iterativni pronalazak lokalnog minimuma funkcije. Funkcionira tako da računa gradijent funkcije gubitka, negira ga, te, u smjeru tog negativnog gradijenta, korigira parametre mreže. Korekciju određuje funkcija troška koja se računa na cijelom skupu podataka te mora biti diferencijalna. Tijekom izračuna, parametriziramo ju po parametrima mreže θ koje u epohi $t+1$ ažuriramo na temelju parametara iz epohe t. Sljedeći izraz, gdje η označava stopu učenja, a $\nabla_\theta Cost(\theta_t)$ gradijent funkcije troška po parametrima mreže, određuje princip ugađanja [23].

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta Cost(\theta_t) \tag{2.4}$$

Stohastički gradijentni spust

U praksi, postoji više načina za primjenu temeljne ideje gradijentnog spusta zanemarujući njeno čvrsto inzistiranje da se parametri ažuriraju tek nakon što mreža vidi sve uzorke. Stohastički gradijentni spust doskače problemu velikog memorijskog i vremenskog troška takve optimizacije tako što se parametri ažuriraju nakon predočenja svakog primjera mreži [23]. Ovo ponašanje reflektirano je u donjoj, modificiranoj jednadžbi ažuriranja parametara ($x^{(i)}$ i $y^{(i)}$ predstavljaju instance trening podatka i pripadne labele).



Slika 2.2: Primjer konvolucijske mreže s vizualizacijom klasifikacije slika. Prikazan je princip ekstrahiranja značajki konvolucijskim slojevima te konačno provlačenje istih kroz potpuno povezane slojeve za dobivanje klasnih predviđanja. (preuzeto iz [12]).

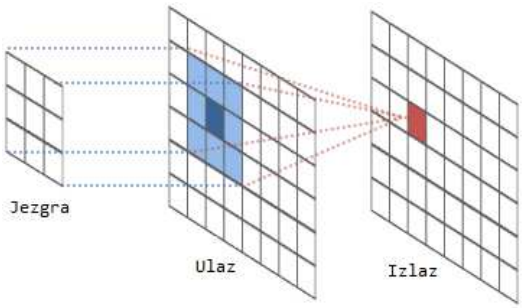
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta Cost(\theta_t; x^{(i)}, y(i)) \tag{2.5}$$

Optimizator Adam

Nadogradnja ideje stohastičkog gradijentnog spusta upravo je Adam (ime dolazi od izraza *"Adaptive moment estimation"*). Adam dodatno koristi prilagodljive stope učenja i koncept momenta. Za svaki parametar neuralne mreže pamti zasebnu stopu učenja koja se postupno mijenja na temelju promjena gradijenta funkcije gubitka u vremenu. Moment ovdje označava dodavanje djela prošle promjene parametra promjeni trenutne iteracije. Ovo pomaže kod izbjegavanja lokalnih minimuma jer daje dodatni zamah promjeni kako bi se parametar nastavio mijenjati u istom smjeru kao i do sad [23].

2.2. Konvolucijske neuronske mreže

Tradicionalne, potpuno povezane, neuralne mreže vrlo brzo postale su memorijski nezgrapne i manjkave za zadatke koji uključuju obradu slika: dijelom zbog broja potrebnih veza između slojeva, a djelom zbog velike osjetljivosti na prostorni položaj promatranih objekata. Nemaju sposobnost percipirati okolinu piksela, već i za najmanje pomake interesantnih značajki slike morale bi trenirati i s tom novom, malo modificiranom slikom kako bi ju mogle razumjeti. Konvolucijske neuralne mreže (CNN) zaobišle su manjak prostorne svijesti korištenjem konvolucija.



Slika 2.3: Vizualizacija računanja jednog elementa izlaza konvolucijskog sloja [19]

Konvolucijski sloj

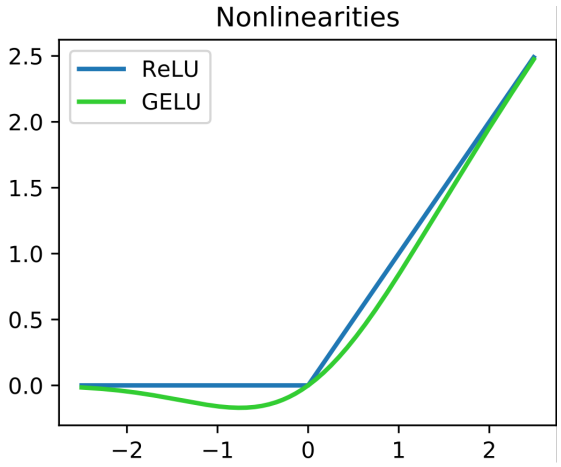
Konvolucijski slojevi rade na principu klizećeg prozora tj. pomicanja matrice jezgre preko matrice ulaznih podataka. Na svakoj mogućoj poziciji, izračunava se umnožak elemenata jezgre i odgovarajućeg ulaznog područja istog oblika, a zatim zbraja te umnoške (operacija konvolucije) što proizvodi točno jedan element izlaza (slika 2.3). Generiranje izlaza sloja uključuje mnogo takvih prolaza. Dimenzije konačnog izlaza ovise o željenom broju izlaznih kanala (što određuje i broj korištenih jezgri), koraku pomaka jezgre između računanja vrijednosti izlaza (npr. korak 2 označava da se jezgra pomiče za 2 elementa između računanja) te nadopunjavanju (eng. *padding*) originalne slike (u suprotnom se smanjuju dimenzije).

Ovakvi slojevi pogodni su za ekstrakciju značajki iz slika i razumijevanje njihove prostorne komponente što CNN-ovima omogućava učenje neovisno (do neke mjere) o prostornoj komponentni predmeta interesa. Ovo vidimo na slici 2.3, tj. da svaki elementa izlaza, ovisi o ciljnom element ulaza zajedno s nekolicinom piksela iz njegove okoline (ovisno o veličini jezgre).

Generalni rad konvolucijskih mreža oslanja se na ovakve slojeve i prikazan je na slici 2.2. Vidimo da se, postupnim sažimanjem slike i konvoluiranjem njenih značajki smanjuje dimenzionalnost širine i visine, no dobiva se dubinska dimenzionalnost što nam na kraju omogućava da se iz tih značajki, u ovom slučaju, klasificira originalna slika.

2.2.1. ConvNext arhitektura

Od njihove pojave do danas, mreže temeljene na slojevima pažnje postižu zavidne rezultate na zadatcima obrade slika i predstavljaju svojevrsnu revoluciju u području računalnog vida [8]. Donedavno su dominirale nad konvolucijskim modelima, no, zahvaljujući pažljivim promjenama u dizajnu mreže, ConvNext arhitektura vraća CNN-



Slika 2.4: Vizualizacija ReLu i GELU aktivacijskih funkcija. (slika preuzeta iz [22]).

ove u konkurenciju. Misao vodilja redizajna bila je "Kako odluke o dizajnu vizualnih transformera utječe na performance konvolucijskih mreža?" Poboljšanja vođena tom idejom uspješno pariraju u pogledu točnosti, skalabilnosti i robusnosti u svim bitnim mjerilima, a ConvNext svejedno zadržava jednostavnost implementacije i treniranja. Za bolje razumijevanje inovacija, u nastavku je razjašnjeno nekoliko bitnih koncepata.

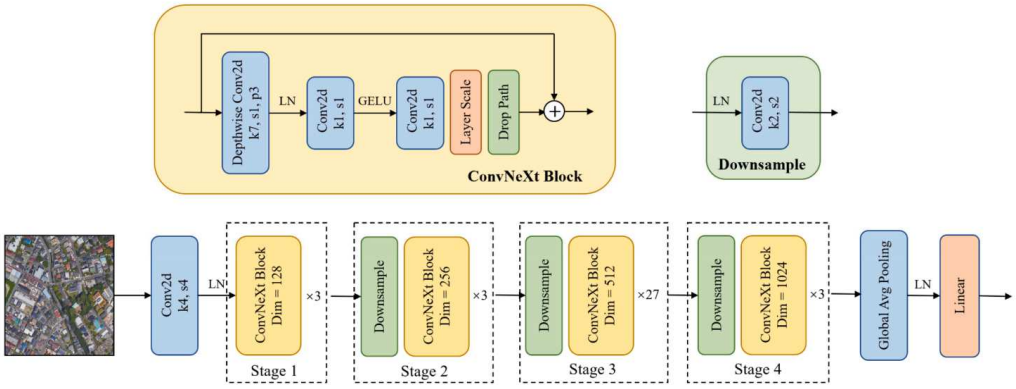
ReLu

Aktivacijske funkcije služe, u pravilu, za unošenje nelinearnosti u mreže. Primjenjuju se na izlaze individualnih neurona kako bi se isti modificirali na željene načine. ReLu (eng. *Rectified Linear Unit*, hrv. zglobnica) jedna je takva funkcija koja propušta vrijednost na izlazu neurona samo u slučaju da je veća od nule, u suprotnom postavlja ju na nulu (jednadžba 2.6). Grafički prikaz funkcije je na slici 2.4 i označen je plavom bojom.

$$\text{ReLU}(x) = \max(0, x) \tag{2.6}$$

GELU

GELU (eng. *Gaussian Error Linear Unit*) nastoji spojiti ReLu propusnost i ideju stohastičkog ispadanja neurona. Rješenje tog problema je da se svaka vrijednost na ulazu GELU funkcije množi s vrijednosti kumulativne funkcije raspodjele u točki izlaza za Gaussovu distribuciju srednje vrijednosti 0 i standardne devijacije 1 (jednadžba 2.7). Ulazi se tako skaliraju na osnovi toga koliko su veći od drugih ulaza. To osigurava da negativne vrijednosti i dalje donekle prolaze (slika 2.4), no da, ako su dovoljno male,



Slika 2.5: Opis ConvNext-Base arhitekture (slika preuzeta iz [5]). Veliki žuti pravokutnik gore lijevo prikazuje strukturu konvolucijskih blokova, gore desno je struktura blokova sažimanja, a dolje cjelokupni pregled modela.

množe se s toliko malim brojem da ispadaju [10].

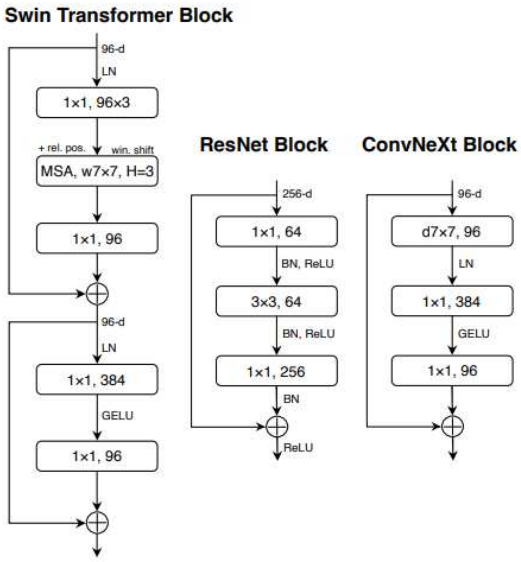
$$\text{GeLU}(x) = x \cdot P(X \leq x) = x \cdot \Phi(x), X \sim \mathcal{N}(0,1) \tag{2.7}$$

Kako je precizno evaluiranje GELU funkcije kompjuterski skupo, u [10] predložena je aproksimacija iz jednadžbe 2.8 koja ubrzava izvođenje GELU-a, no žrtvuje nešto točnosti.

$$\text{GeLU}(x) = \frac{x}{2}\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{2.8}$$

Dizajn ConvNext-a

Generalna arhitektura prati sliku 2.5 (iako postoje varijacije u broju filtera svake razine gledajući sve ConvNext arhitekture iz spomenutog rada). Smjernice preuzete iz dizajna ViT-ova očituju se u procesiranju slika na ulazu (plavi pravokutnik odmah poslije satelitske slike). Standardni 7x7 konvolucijski sloj koraka 2 s početka ResNet-ova zamijenjen je imitacijom *"patches"* [21] koncepta standardnih za transformatore. Ovdje je implementiran kao konvolucijski 4x4 sloj koraka 4. Nadalje implementiran je koncept invertiranog uskog grla. Vidimo da je središnji potpuno povezani sloj četiri puta širi od prijašnjeg konvolucijskog te finalnog potpuno povezanog sloja. Vidimo da ConvNext ima i povećanu jezgru (po uzoru na transformere), no pomicanjem konvolucijskog sloja ranije u bloku smanjujemo im kompleksnost (imaju manje kanala), stoga performanse ne pate previše (slika 2.6).



Slika 2.6: Dizajn blokova Swin transformera, Resnet-a i ConvNext-a preuzet iz [16]. Uočavamo sličnosti transformerskog i konvolucijskog bloka u veličini skrivenog konvolucijskog sloja pretkraj bloka kao i u 7x7 veličini jezgri slojeva za inicijalno ekstrahiranje značajki.

Mikro odluke dizajna sastoje se od zamijene ReLu aktivacijske funkcije s GELU, općenitog smanjenja broja aktivacijskih funkcija u modelu (u bloku je samo jedna, umjesto standardne prakse umetanja aktivacijskih funkcija iza svakog konvolucijskog sloja), smanjenja broja slojeva grupne normalizacije, zamjena istih sa slojevima normalizacije slojeva te implementacija poduzorkovanja između razina mreže u obliku 2x2 konvolucija koraka jednakog 2 [16].

Sve ove promjene i poboljšanja zajedno drastično pospješuju performance nove generacije konvolucijskih mreža te ih čine i više nego dovoljnim za potrebe ovog rada [16].

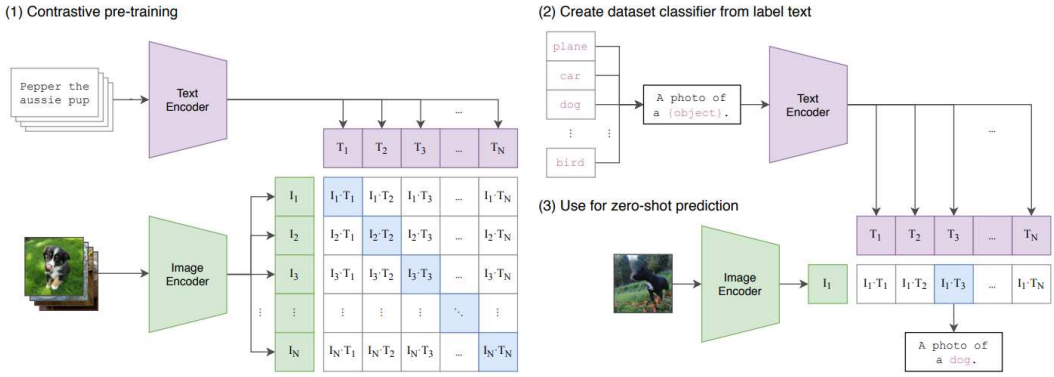
3. CLIP

CLIP (Contrastive Language-Image Pre-training) [6, 20] je multimodalni model učen na setu podataka slika i pripadnih tekstualnih opisa. Ključna ideja ovog pristupa jest da model razvije sposobnost reprezentacije slika i teksta na način koji dopušta njihovu usporedbu (slika 3.1) te, na temelju toga, ostvaruje veliku robusnost na primjene za koje čak i nije eksplicitno bio treniran.

Posebno zanimljiv je prijenos znanja na zadatke zero-shot i few-shot klasifikacije. Zero-shot klasifikacija označava zadatak svrstavanja u klase koje nisu bile viđene za vrijeme treniranja, a few-shot klasifikacija zadatak kad model mora svrstavati u klase za koje je vidio tek nekoliko primjera. Iskorištavanjem sposobnosti modela da ugradi riječi ili fraze i slike koje prije nije vidio, no čije apstraktne reprezentacije i dalje reflektiraju njihove stvarne odnose s onima na kojima je model uistinu učen, CLIP postiže zavidnu rezultate (princip rada vidimo na slici 3.1).

3.1. OpenCLIP

U radu je korištena *open-source* implementaciju CLIP arhitekture iz rada [6, 7]. Specifični korišteni model sastoji se od ConvNext-Base okosnice za kodiranje slika te transformera koji odgovara onom korištenom uz RN50x4 vizualni model iz [20] za ugrađivanje teksta. Tekstni model temelji se na mehanizmimam maskiranja i samopažnje pri kodiranju jezika. Oba korištena modela na izlazu daju 640-dimenzionalne vektore kao reprezentacije ulaza. Predtreniran je na podskupu LAION-5B (konkretno LAION Asthetic) seta podataka koji su dodatno bili prošireni korištenjem nasumičnih isječaka i brisanja te korištenjem stohastičke dubine [11] kod treniranja ugrađivača slika.



Slika 3.1: Princip rada CLIP-a preuzet iz [6]. Tijekom treninga (1) model ugrađuje svaku sliku i svaki tekst te uspoređuje parove na temelju kosinusne sličnosti s ciljem da na kraju odgovarajući parovi slika i teksta imaju najveću sličnost. Za potrebe klasifikacije, željene klase se jezično ugrađuju kao niz apstrakcija(2). Kod zadataka zero-shot klasifikacije (3), ugrađena slika se uspoređuje s listom potencijalnih opisa te se odabire onaj najsličniji (po kosinusnoj sličnosti).

4. Segmentacija

Temeljni problem računalnog vida je klasno svrstavanje svakog piksela slike na ulazu u sustav. Postoji više vrsta: obična semantička segmentacija koja pikselima svih instanci interesantnih objekata pridjeljuje istu oznaku, segmentacija instanci koja nastoji svaki individualni primjerak posebno označiti te panoptička segmentacija koja kombinira ova dva pristupa: unutar svake klase nastoji međusobno izdvojiti primjerke iste.

4.1. Semantička segmentacija

Popularan pristup semantičkoj segmentaciji baziran je na korištenju Koder-Dekoder arhitekture koja je predložena u [17]. Preskočne veze između spomenuta dva djela omogućavaju da ekstrakcijom značajki iz slojeva kodera minimiziramo gubitak informacija te njihovim kombiniranjem i naduzorkovanjem na izlazu dobivamo dobre predikcije za svaki piksel originalne scene (vidimo ovaj princip u kasnijem poglavlju, na slici 5.1).

Semantička segmentacija ima brojne primjene u medicini npr. analizi medicinskih slika, razvoju proširene i virtualne stvarnosti, agrikulturi te poboljšanju autonomne navigacije robota i vozila. Za sve ove primjene, duboke konvolucijske mreže pokazale su sposobnost postizanja vrhunskih rezultata [16].

4.2. CamVid

Treniranje i evaluacija u ovom radu odrađeni su koristeći skup podataka CamVid za semantičku segmentaciju. CamVid se sastoji od 701 ručno označene i validirane slike, dimenzija 720x960, tradicionalno raspodijeljene na 3 skupa [3, 4]. 367 slika i pripadnih oznaka svrstano je u skup za učenje, 101 u validacijski skup te 233 u skup za testiranje. Ovaj rad ipak specifično koristi slike upola manjih dimenzija (480x360) preuzete s [1]. Sadržaj slika obuhvaća prizore snimljene kamerom iz auta u prometu.

Slika 4.1: Primjer slika iz skupa CamVid. Lijevo je slika stvarnosti, a desno njen semantičko segmentirani par (u boji je radi lakšeg primjećivanja klasa).

Originalni set podataka također klasificira piksele u 32 distinktne klase, no u praksi je češći skup s tim brojem reduciranim na 11 + 1 klasa: nebo, zgrada, stup, cesta, nogostup, drvo, prometni znak ili simbol, ograda, auto, pješak i biciklist te ova dodatna klasa koja označava sve piksele koji ne spadaju u ranije navedene kategorije. Točno segmentirane slike zapisane su kao sive slike gdje intenziteti piksela odgovaraju njihovim klasama (s time da brojanje klasa kreće od 0: nebo je 0, zgrada 1, itd.). Zbog malene veličine skupa ovaj rad spaja skupove za trening i validaciju u trainval skup na kojem će modeli biti učeni, dok će se evaluacija provoditi na izvorno testnom skupu. Ovo je prihvaćena praksa kod ovakvih skupova [2].

5. Programsko rješenje

Koder-dekoder arhitektura popularno je rješenje za implementaciju općenite semantičke segmentacije [17, 2], stoga je ovdje korištena za ostvarenje semantičke segmentacije specifično primjenom jezičnih ugrađivanja. Zadatak je zahtijevao treniranje na skupu CamVid podataka te iskorištavanje postojećeg CLIP modela [6] (opisanog u poglavlju 3) , modificiranje ConvNext-Base mreže [16] istog modela kako bi mogla igrati ulogu kodera, izgradnju naduzorkovatelja/dekodera te uspješno povezivanje spomenutog kako bi na izlazu cjelokupna mreža davala klasna predviđanja za svaki piksel na temelju uspoređivanja izlaza koder-dekoder mreže i CLIP jezično ugrađenih klasa koje su joj, zajedno sa slikom, na ulazu proslijeđene.

Svi eksperimenti provedeni su na platformi Google Colab zbog mogućnosti korištenja moćnih grafičkih kartica besplatno te lakog baratanja i spremanja modela na Google Disk. Specifični GPU bio je Tesla T4 model s 15 GB memorije. Zbog prirode zadatka, maksimalna moguća veličina mini-grupe bila je 6 slika za naduzorkovatelj punih dimenzija, a 10 za modificirani (više o tome u opisu rezultata).

Vanjske biblioteke

Programska implementacija modela i treniranje izvedeni su Pythonom i njegovim bibliotekama Pytorch[1], TorchMetrics[2], Numpy[3] te ranije spomenutom *open-source* implementacijom CLIP modela open-clip-torch [4] u Pytorchu.

Pytorch je *open-source* biblioteka za strojno učenje. Nudi lako i intuitivno mnoštvo već gotovih implementacija funkcija gubitka, optimizatora, planera (eng. *scheduler*), klasa za dohvat i pripremu podataka za trening i evaluacije te, ono najkorisnije, podršku za automatsku diferencijaciju koja olakšava propagiranje pogreške unatrag i poboljšanje mreža. Također podržava izvođenje svega na CUDA sposobnim GPU-ovima što

[1] https://pytorch.org/

[2] https://torchmetrics.readthedocs.io/en/stable/

[3] https://numpy.org/

[4] https://github.com/mlfoundations/open_clip

znatno ubrzava period treniranja i evaluacije.

TorchMetrics prirodno je proširenje Pytorch biblioteke. Pruža mnoge gotove implementacije postojećih evaluacijskih metrika što znatno pojednostavljuje testiranje neuralnih mreža.

Numpy olakšava baratanje s matricama i vektorima. Ovdje je korišten u međukoracima procesiranja slika te evaluaciji.

Kosinusna sličnost

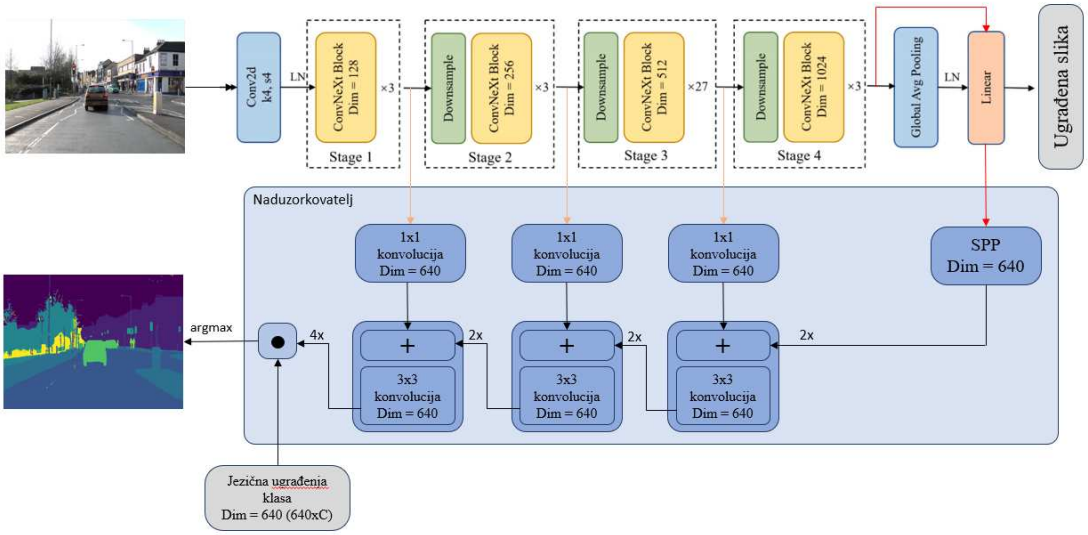
Radi čitljivijeg opisa izgradnje modela, kosinusna sličnost je također objašnjena ovdje. Ona je mjera za međusobnu sličnost dvaju vektora koja se računa po sljedećoj formuli. Dalje u radu koristit će se za kvantificiranje sličnosti jezičnih ugrađivanja i ugrađivanja pojedinih piksela.

$$Kosinusna\ slicnost(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} == \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.1}$$

5.1. Konstrukcija modela

Kako bi ConvNext-Base mogla funkcionirati kao koder, dodane su 4 preskočne veze na izlazima pojedinih razina. Razine su, između ostalog, određene činjenicom da se ulaz razine, poslije prolaza njome, smanjuje za određeni faktor (prvo 4x, zatim 2x, pa 2x i konačno dodatno 2x). Tri od njih direktno vode u naduzorkovatelj i na slici 5.1 označene su narančastim strelicama, dok posebna četvrta veza (crvene boje) iskorištava popuno povezani linearni sloj iz glave mreže (na slici 5.1 narančasti pravokutnik) da bi podatke dimenzije 1024 svela na podatke dimenzionalnosti 640.

To radimo upravo zato što želimo predviđanja na razini piksela, a ne cijele slike. Naime, na izlazu četvrtog sloja imali bi podatke dimenzija (N, C_{izlaz}, H, W) gdje je N broj uzoraka koji smo doveli na ulaz mreže, C_{izlaz} broj izlaznih kanala (u ovom slučaju 1024), a H i W visina i širina podataka (u odnosu na ulaznu sliku ove dimenzije su 32x manje). Kada bi te podatke normalno proveli kroz plavo označen sloj globalnog prosječnog sažimanja, dobili bi izlaz dimenzija (N, C_{izlaz}) koji bi prolaskom kroz linearni sloj bio dodatno smanjen na $(N, 640)$. Ovim postupkom izgubili bi mnogo informacija te bi bilo kakvi pokušaji konstrukcije semantično segmentirane originalne slike bili, u najmanju ruku, besmisleni. Zaobilaskom sloja sažimanja praktički svaki "piksel" izlaza četvrtog sloja kodiramo u 640-dimenzionalni vektor koji bi, da želimo



Slika 5.1: Modificirana ConvNext-Base arhitektura za semantičku segmentaciju opisana u 5.1 (originalna slika je preuzeta iz [5]).

segmentirati sliku 32x manjih dimenzija, čak i u tom trenutku mogli uspoređivati s jezičnim ugrađivanjima klasa.

Ipak, cilj nam je postići jednake dimenzije kao i ulazna slika, stoga taj izlaz provodimo dalje do najplićeg sloja naduzorkovatelja.

Naduzorkovatelj mreže

Ranije opisanim postupkom primjećujemo da imamo sve potrebne informacije ekstrahirane te da možemo krenuti s naduzorkovanjem. Cilj nam je tijekom cijelog procesa zadržati 640 kanala kako bi na kraju segmentaciju mogli provoditi na temelju kosinusne sličnosti jezičnih ugrađivanja i vizualnih značajki pojedinih piksela, stoga svi slojevi naduzorkovatelja ne mijenjaju tu dimenziju.

Implementirani naduzorkovatelj napravljen je po uzoru na single scale model iz [2]. Na slici 5.1 vidimo da podatci najniže rezolucije prolaze SPP (eng. *Spatial Pyramid Pooling*) slojem preuzetom iz [2] (originalno je opisan u [14]). Izlazi tog postupka bilinearno se naduzorkuju za faktor 2, zbrajaju sa značajkama izvučenim s razine za jednu ranije u koderu od posljednje te dodatno provlače kroz konvoluciju veličine jezgre 3x3 i koraka 1. Ovaj postupak se identično ponavlja (osim što se na kraju bilinearno naduzorkuje za faktor 4) dok ne dobijemo sliku (ili slike) originalnih dimenzija, ali sa 640 kanala.

Da bi dobili predviđanja na temelju kosinusne sličnosti za individualne piksele taj konačni izlaz prvo normaliziramo, a zatim matrično množimo s originalnim je-

zičnim ugrađivanjima (koja smo normalizirali u startu) tj. imamo izlaz dimenzija (N, C_{izlaz}, H, W) (ista značenja oznaka kao i ranije) i set vektora $(11, 640)$, gdje svaki redak predstavlja jezično ugrađivanje jedne od standardnih 11 klasa CamVida, uzimamo izlaz i matrično ga množimo s transponiranom matricom jezičnih ugrađivanja čime dobivamo konačni izlaz oblika $(N, C_{izlaz}, H, W) \times (11, 640)^T = (N, 11, H, W)$. Ovo možemo interpretirati kao gusta predviđanja 11 CamVid klasa za svaki piksel. Kada želimo dobiti segmentiranu sliku (ili set slika), samo uzmemo indeks sloja s najvećim predviđanjem za pojedini piksel (funkcija argmax u Pytorchu).

Jezična ugrađivanja s kojima uspoređujemo značajke piksela dobivamo iz netaknutog dijela originalnog CLIP modela tj. njegovog podmodela za ugrađivanje teksta. Korištena implementacija CLIP-a ima zgodnu metodu "encode_text()" koja na ulazu prima tokenizirane tekstove (open-clip-torch nudi i tokenizator, ovdje je tokenizer dohvaćen s "*open_clip.get_tokenizer('ViT-B-32-quickgelu')*") što trivijalizira generiranje jezičnih reprezentacija klasa.

Treniranje

Provedena su ukupno četiri eksperimenta: dva su bila isključivo treniranje naduzorkovatelja dok su parametri okosnice (modificirani ConvNext-Base) bili zamrznuti, a druga dva su bila treniranje potpune mreže. Za prvo spomenutu konfiguraciju proveo se trening jednom s jezično ugrađenim CamVid nazivima klasa (npr. izmišljena klasa "stolica" bila bi predana na ugrađivanje točno tako), a jednom s kraćim opisima klasa (npr. ista izmišljena klasa mogla je biti predana kao "stolica, dio namještaja za sjedenje, obično ima 4 noge" ili nešto slično) s ciljem kasnijeg uočavanja prednosti i nedostataka pojedinih ugrađivanja, a za treniranje cijele mreže sve se provelo samo s opisima klasa (jer je uočen manjak ikakvih razlika između krajnjih rezultata, stoga ovdje neće biti prikazan).

Sve mreže trenirane su Adam optimizatorom s postupnim kaljenjem stope učenja. Ovo je učinjeno kosinusnim kaljenjem (eng. *cosine annealing*) uz pomoć gotove implementacije u PyTorchu (klasa CosineAnnealingLR). Naduzorkovatelj je učen početnom stopom učenja 4×10^{-4} koja se postupno smanjivala do vrijednosti 10^{-6} i vrijednosti propadanja težina (eng. *weight decay*) od 10^{-4}. U eksperimentima gdje je i ConvNext-Base okosnica bila odmrznuta, njene početne i krajnje vrijednosti stope učenja bile su 4 puta manje od naduzorkovateljevih (od 10^{-4} do 2.5×10^{-7}) sa stopom propadanja težina od 2.5×10^{-5}. Raniji parametri preuzeti su iz [2]. Ovo radimo da maksimalno iskoristimo činjenicu da je okosnica već predtrenirana i smanjimo efekt

zaboravljanja [2].

U svim eksperimentima kao funkcija gubitka korištena je izvorna implementacija unakrsne entropije u Pytorchu s time da je ignore_index parametar bio postavljen na 11 (u CamVidu, 11 označava piksele koji ne spadaju u niti jednu od 11 klasa) kako mreža ne bi bila penalizirana za klasu u koju nema ni sposobnost klasificirati. Prije računanja gubitka, ulazni vektor predviđanja dijelio se s 0.01 što je osiguralo da se ugrađeni softmaks pytorch unakrsne entropije ponaša kao temperaturni softmaks. Ovo je učinjeno zbog prirode računanja kosinusnih sličnosti ugrađivanja piksela i teksta. Kako ime implicira, sličnost poprima vrijednosti između 0 i 1 što, iako će predviđanja u koje je mreža sigurna biti bitno mnogo puta veća od drugih, sprječava softmaks da ih značajno odvoji zbog malih veličina (ovo je dodatno ometano sličnostima između samih jezičnih ugrađivanja).

Za učenje, korišten je ranije spomenuti CamVid trainval skup smanjenih dimenzija 360x480. Treniranje modela provedeno je nad slučajnim isječcima veličine 224x224 uz dodatno rastresanje podataka nasumičnim horizontalnim zrcaljenjem s vjerojatnošću od 50%.

Broj epoha treniranja i neke specifičnosti bit će navedene kasnije u poglavlju s rezultatima radi preglednije prezentacije rezultata.

6. Metrike

Kvantificiranje kvalitete mreže konačni je korak verifikacije sposobnosti modela. Metrike su skupni naziv za takve metode te imaju nekoliko uvriježenih naziva za računanje koji će biti objašnjeni na početku na primjerima semantičke segmentacije. TP (od eng. *True positive*) označava broj korektno pozitivno označenih piksela tj. oni stvarno pripadaju promatranoj klasi, FP (eng. *False positive*) broj piksela koji su krivo označeni kao da joj pripadaju te FN (eng. *False negative*) broj piksela koji su krivo označeni kao da nisu dio klase.

Ovaj rad koristi točnost piksela, srednju točnost piksela te srednji omjer presjeka i unije za evaluaciju performansi.

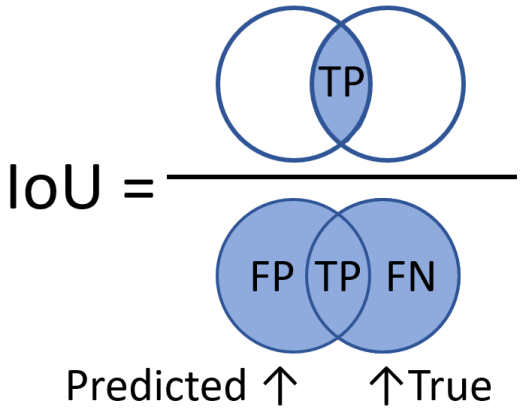
Točnost

Općenita točnost grubi je pokazatelj kvalitete modela jer je podložna zavaravajućem predstavljanju rezultata. Računa se kao omjer točno klasificiranih piksela i njihovog ukupnog broja (jednadžba 6.1) Nema mehanizam koji osigurava penaliziranje lošeg prepoznavanja određenih klasa ako je općenita distribucija piksela nerazmjerna npr. ako želimo segmentirati sliku na auto i pozadinu te sve naše slike sadrže vrlo male aute, model bi mogao naučiti sve piksele klasificirati kao pozadinu te i dalje postizati dobru točnost.

$$OverallAccuracy = \frac{Number\ of\ correctly\ classified\ pixels}{Total\ number\ of\ pixels} \tag{6.1}$$

Ovom problemu doskače se tako da se točnost računa za svaku klasu te na kraju uzima aritmetička sredina izračunatih točnosti. Ovu metodu zovemo srednja točnost (*Mean Accuracy* te se računa po 6.2

$$MeanAccuracy = \frac{1}{C} \times \sum_{i=0}^{C} \frac{Number\ of\ correctly\ classified\ pixels\ of\ class\ i}{Total\ number\ of\ pixels\ of\ class\ i} \tag{6.2}$$



Slika 6.1: Vizualizacija omjera presjeka i unije dvaju oblika (lijevo je predviđanje oblika, a desno stvarnost) s pripadnim oznakama FP, TP i FN područja (objašnjeno u tekstu).

IoU

U semantičkoj segmentaciji još je bitna metrika omjera presjeka i unije (eng. *Intersection over Union, IoU*). Matematički izraz za presjek preko unije za piksele neke klase dan je u 6.3. Intuitivno, IoU možemo razumjeti preko slike 6.1.

$$IoU = \frac{TP}{TP + FP + FN} \tag{6.3}$$

Spomenuta slika prikazuje samo kako se računa za jednu klasu, no kako želimo pouzdano kvantificiranje performansi na 11 CamVid klasa, koristit će se srednji omjer presjeka i unije računat preko tih 11 klasa (jednadžba 6.4, IoU_i je IoU računat za klasu i).

$$Mean\ IoU = \frac{1}{C} \times \sum_{i=1}^{C} IoU_i \tag{6.4}$$

7. Eksperimentalni rezultati

Rezultati su podijeljeni na eksperimente treniranja sa zamrznutom okosnicom i na one gdje se trenirala potpuna mreža.

Provedene su dvije vrste eksperimenata. Jedni s isključivim treniranjem naduzorkovatelja, a jedni s treniranjem cjelokupne mreže opisane u potpoglavlju 5.1. Treniranje naduzorkovatelja odvijalo se kroz 240 epoha s ranije opisanim optimizatorom, funkcijom gubitka i planerom. Broj epoha određen je eksperimentalno zbog otklanjanja problema divergiranja funkcije gubitka između 100. i 150. epohe kod pokušaja duljih treniranja. Treniranje potpune mreže pušteno je da se izvodi dodatnih 160 epoha (ukupno 400) po uzoru na rad [2] kako bi se maksimalno iskoristilo CLIP predtreniranje ConvNext-Base okosnice.

Između naduzorkovatelja u eksperimentima postoji samo jedna razlika u SPP sloju. SPP sloj kod treniranja samo naduzorkovatelja i jednog eksperimenta s cijelom mrežom inicijaliziran je s ranije spomenutih 640 ulaznih i 640 izlaznih kanala te 4 razine poduzorkovanja gdje je svaka imala 640 izlaznih kanala. Ostali parametri ostavljeni su kao zadani iz implementacije [2]. Kod treniranja cijele mreže drugi put, promijenjen je samo broj slojeva pojedinih razina na 160 jer se pokazalo da to daje nešto bolje rezultate. Ovo je omogućilo i povećanje veličine mini-grupe na 10 u tom eksperimentu što je vjerojatno dodatno doprinijelo rezultatu.

7.1. Osnovica

Prije prikaza rezultata, bitno je navesti osnovicu tj. performanse koje se može postići bez ikakve, ili barem s vrlo malo, intervencije, kao i SwiftNet single-scale modela po čijem uzoru je rađen naduzorkovatelj.

Najjednostavnijim pristupom bilinearnog naduzorkovanja 32x smanjenih ulaznih podataka koje dobijemo na izlazu iz modificiranog ConvNext-Base te segmentacijom takvih slika uz pomoć kosinusne sličnosti s jezičnim ugrađivanjima, postižemo, u slučaju kodiranja opisa klasa, srednji IoU od 23.91%. Takav pristup također postiže toč-

nost od 51.45% i srednju točnost od 46.31%. Interesantno je da, iako je točnost kod opisa bolja za otprilike 5.5 postotaka, srednja točnost je gora za približno 1 postotak, no mIoU je opet, za gotovo 3 postotka, veći (Tablica 7.1).

Tablica 7.1: Rezultati osnovnog pristupa na skupu CamVid. Izlaz iz modificiranog ConvNext-Base modela tj. 32x smanjeni ulazi dimenzionalnosti 640 bilinearno je naduzorkovan do originalne veličine ulaza te segmentiran.

Jezična ugrađivanja	Točnost [%]	Srednja točnost [%]	mIoU[%]
Imena klasa	45.71	47.35	21.19
Opisi klasa	51.45	46.31	23.93

Dodavanjem dva konvolucijska sloja jezgre 1x1 i koraka 1, kao i popratnih ReLu slojeva, na ranije spomenuti izlaz kodera, prateći ideju [9], dodan je svojevrsni adapter na izlaz kodera. Adapter je gotovo udvostručio ranije rezultate po pitanju srednjeg IoU. Sada imena klasa postižu 43.41%, a opisi 43.87%. Interesantno je da se gubi distinkcija u performansama ugrađivanja. Točnost jezičnih kodiranja je potpuno ista (80.78%), a srednja točnost ima tek male razlike, 59.65% za imena klasa te 60.84% za opise (Tablica 7.2).

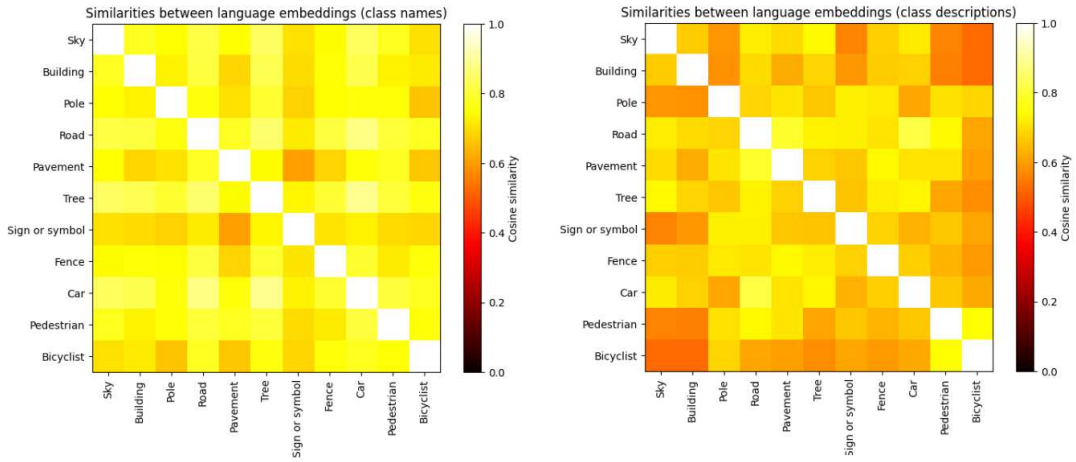
Tablica 7.2: Rezultati modificiranog osnovnog pristupa na skupu CamVid. Izlaz iz modificiranog ConvNext-Base modela tj. 32x smanjeni ulazi dimenzionalnosti 640 je prvo provučen kroz dva konvolucijska 1x1 sloja, zatim bilinearno naduzorkovan do originalne veličine ulaza te segmentiran.

Jezična ugrađivanja	Točnost [%]	Srednja točnost [%]	mIoU[%]
Imena klasa	80.78	59.65	43.41
Opisi klasa	80.78	60.84	43.87

SwiftNet single-scale model, treniranjem na ranije spomenutom duplo manjem skupu CamVid, postiže točnost od 89.96% i srednji IoU od 64.51% (Tablica 7.6). Kako je korištena implementacija priložena uz [2] i za evaluaciju, računanje srednje točnosti nije napravljeno.

7.2. Zamrznuta okosnica

Kod treniranja mreže sa zamrznutom okosnicom, vidimo da se za jednostavnije opise klasa (tj. sama imena) postiže marginalno bolji srednji IoU (63.16%), no razlika nije



Slika 7.1: Vizualizacija međusobnih kosinusnih sličnosti jezičnih ugrađivanja. Lijevo se nalaze sličnosti samih imena klasa, a desno opisa (iako su i dalje navedeni samo nazivi radi preglednosti).

značajna (iznosi 0.17% u korist ugrađivanja isključivo naziva). Mogli bismo to objasniti činjenicom da su ugrađivanja imena klasa međusobno sličnija te da se, igrom slučaja, na skupu za validaciju ta sličnost pogodno odražava na sposobnosti mreže. S druge strane, interesantno je da treniranje s opisima klasa postiže bolju srednju točnost (75.67% naspram 75.26%), iako ima manju općenitu točnost (89.68% naspram 91.19%) (tablica 7.3).

U tablici 7.4 stoji detaljan rastav presjeka preko unije za individualne klase mreže trenirane na nazivima klasa sa smrznutom okosnicom. Vidimo da se mreža muči s detekcijom, u pravilu, malih područja (stup, znak i ograda s redom 23.08%, 42.06% i 39.06% IoU), dok veća područja dobro lokalizira. Pješaci i biciklisti isto imaju IoU ispod 50%, no bolji su od ranije navedenih klasa. Najbolje prepoznaje područja ceste (90.32%).

Tablica 7.3: Rezultati pristupa na skupu CamVid. Treniran je isključivo naduzorkovatelj širine 640 dimenzija u svim djelovima i to na nazivima klasa (T označava trening skup podataka i tu je radi usporedbe)

Jezična ugrađivanja	Točnost [%]	Srednja točnost [%]	mIoU[%] za T	mIoU[%]
Imena klasa	91.19	75.26	73.90	63.16
Opisi klasa	89.68	75.67	73.88	62.99

7.3. Treniranje cijele mreže

Tablica 7.4: Rezultati pristupa na pojedinačnim klasama. Lijevo stoje samo nazivi klasa radi kompaktnije prezentacije, u sredini vrijednosti IoU za pojedine klase treniranja isključivo naduzorkovatelja dimenzije 640 na nazivima klasa, a desno klasni IoU za trening cijelog modela na opisima sa smanjenom dimenzijonalnošću SPP sloja naduzorkovatelja.

	Zamrznuta okosnica - nazivi	Cijeli model - uži SPP - opisi [%]
Klasa	IoU [%]	IoU [%]
Nebo	89.70	90.22
Građevina	80.07	78.74
Stup	23.08	31.81
Cesta	90.32	89.70
Nogostup	77.07	75.65
Drvo	71.71	72.92
Znak	42.06	45.11
Ograda	39.06	34.83
Auto	84.19	79.71
Pješak	47.56	47.23
Biciklist	49.96	55.48

Cijela mreža trenirana je samo na opisima klasa kako bi se u jezična ugrađenja, s kojima kosinusnom sličnošću uspoređujemo piksele, pohranio što bolji opis stvarnih semantičkih značenja klasa. Ovakav pristup pospješuje distinkciju između individualnih ugrađenja kao što vidimo u radu [24]. Najbolji postignuti mIoU iznosi 63.76% i to za mrežu sa smanjenom dimenzionalnošću SPP razina. Treniranje naduzorkovatelja pokazalo se boljim od treniranja cijele mreže (za 0.47 postotaka), što sugerira prenaučenost modela. Modificiranje okosnice, niti konvolucijskih dijelova naduzorkovatelja nije bila opcija, stoga je smanjen broj kanala individualnih razina SPP modula. U tablici 7.5 vidimo da je to donijelo poboljšanje u svim evaluacijskim metrikama (1.41 ekstra postotaka općenite točnosti, 1.37 srednje točnosti te 1.34 srednjeg IoU). Mreža s identičnim naduzorkovateljem kao primjeri iz ranijeg poglavlja, gdje broj kanala razina SPP-a odgovara broju krajnjih kanala (po uzoru na [2]), postiže marginalno bolji mIoU na trening skupu (79.08%) što potencijalno sugerira da je uistinu problem u prevelikom kapacitetu mreže.

Detaljan raspis IoU-a po klasama za mrežu smanjenog naduzorkovatelja stoji u tablici 7.4. Vidimo da treniranjem cijele mreže postižemo bolje rezultate za određene

problematične klase iz prošlog eksperimenata. IoU za stup skače na 31.81%. To je ujedno i najveće poboljšanje, i to od 8.73 postotaka. Lokalizacija znakova poboljšava se za 3.05 postotaka, no mIoU pada za čak 4.23 jedinica. Zanimljiva pogoršanja su još 4.48 manje postotaka kod lokalizacije autiju i 1.42 kod nogostupa, a poboljšanja: 5.52 postotaka bolji IoU za bicikliste. Ostale klase postižu podjednake performanse.

Tablica 7.5: Rezultati pristupa na skupu CamVid. Trenirana je cijela mreža na opisima klasa sa smanjenjem dimenzionalnosti SPP sloja naduzorkovatelja na 160 (T označava trening skup podataka i tu je radi usporedbe)

Varijanta SPP dijela	Točnost [%]	Srednja točnost [%]	mIoU[%] za T	mIoU[%]
640 kanala po razini	89.45	75.01	79.08	62.42
160 kanala po razini	90.86	76.38	78.75	63.76

Ukupni rezultati pokazuju općeniti trend smanjivanja sposobnosti lokalizacije objekata njihovim smanjenjem dimenzija: stupovi, ograde i znakovi prolaze najgore, zatim pješaci i biciklisti, a ostale klase postižu IoU od preko 70% u svim slučajevima.

Također se vide mane malog skupa treniranja (obje vrste eksperimenta gube od 11 do 17 postotka mIoU prijelazom s treninga na evaluacijski skup). Podjednake performanse treniranja cijele mreže i isključivo naduzorkovatelja mogli bismo objasniti veličinom slika skupa CamVid i činjenicom da naduzorkovatelj ima otprilike 6 puta manje parametara od ConvNext-Base modela (ovisno o verziji naduzorkovatelja).

Ovu slutnju dodatno podržava činjenica da SwiftNet single-scale model postiže slične rezultate. On čak ima manju točnost od oba pristupa; ne mnogo, manje od postotka. Kod srednjeg IoU postiže pak manje od 1 postotka poboljšanja (64.52%).

Tablica 7.6: Rezultati svih pristupa na skupu CamVid. Prvi odvojeni dio prikazuje najbolje rezultate za osnovne postupke, drugi dio najbolje rezultate pristupa iz rada, a treći uspješnost SwiftNet single-scale modela na istom setu podataka.

Pristup	Točnost [%]	Srednja točnost [%]	mIoU[%]
Bilinearno - opisi	51.45	46.31	23.93
Adapter - opisi	80.78	60.84	43.87
Zamrznuta okosnica - imena	91.19	75.26	63.16
Suženi SPP - opisi	90.86	76.38	63.76
SwiftNet - single scale	89.96	-	64.52

8. Zaključak

Jezična ugrađivanja otvorila su naizgled ogromna vrata za poboljšanje umjetne inteligencije općenito. Njihovo korištenje u segmentaciji koraca prema razvoju univerzalnih segmentera. Ovaj rad bavi se korištenjem CLIP jezičnih ugrađivanja za segmentaciju prizora s cesta iz skupa podataka CamVid.

Opisani su koraci i dijelovi treniranja neuronskih mreža: duboki modeli općenito, funkcije gubitka i optimizacijske metode. U središte pozornosti stavljeni su konvolucijski neuralni modeli tj. njihova najmodernija inačica: ConvNext arhitektura. Ona simbolizira budućnost konvolucijskih modela te ih ponovno vraća u utrku s transformerima po pitanju točnosti i moći. Predstavljen je i multimodalni CLIP model koji danas prednjači u zadatcima *zero-shot* i *few-shot* klasifikacije pomoću ugrađivanja slika i teksta u isti vektorski prostor čime se omogućava njihova usporedbu i zadržavanje odnosa iz stvarnog svijeta u apstraktnim reprezentacijama.

Nadalje su opisane vrste segmentacije te definirana semantička segmentacija kao i jedan vrlo čest skup za istu: CamVid. Iznesen je način za kombiniranje ugrađivačke moći CLIP modela i ConvNext-Base arhitekture tako što je predtrenirani ConvNext-Base model, uzet direktno iz OpenCLIP implementacije, nadograđen putom za naduzorkovanje te treniran za individualno kodiranje piksela u isti vektorski prostor u koji CLIP smješta slike i tekst kako bi se izvela semantička segmentacija na skupu CamVid podataka.

Praktični dio obuhvaća ispitivanje modela sa zamrznutom ConvNext-Base okosnicom i potpuno odmrznutih modela metrikama srednje točnosti i omjera presjeka i unije. Modeli s odmrznutom okosnicom performiraju marginalno najbolje. Mala razlika u performansama ne iznenađuje previše ako se sjetimo koliko je modificirani skup CamVid zapravo malen (iako je i dalje zanimljivo). Najbolji od njih (treniran na opisima klasa) postiže mIoU od 63.76%, dok najbolji model sa zamrznutom okosnicom, onaj s jezično ugrađenim nazivima klasa, postiže 63.16%. Iako je mIoU daleko manji od vrijednosti koje postižu druge mreže, konačni rezultat ovog pristupa može baratati s više od 11 standardnih CamVid klasa (iako će točnost predviđanja vjerojatno biti sma-

njena) te predstavlja važan korak prema povećanju robusnosti umjetne inteligencije.

Interesantno je za vidjeti moć CLIP modela, koji uz minimalnu intervenciju, postiže mIoU od oko 63% na skupu podataka CamVid iako originalni model nije treniran za segmentaciju. U budućnosti, želim istražiti kako se mreža ponaša ako smanjim dimenzionalnosti nekih dijelova naduzorkovatelja te time omogućim treniranje na skupu CamVid pune rezolucije. Kako korištenje funkcije gubitka koja u obzir uzima koliko je piksel blizu granice s drugom semantičkom cjelinom, te na temelju toga smatra točnost onih bližih bitnijom (kao u radu [2]), utječe na performanse mreže kao i isprobati treniranje na spoju nekoliko skupova podataka prizora s cesta u svrhu mogućeg poboljšanja rezultata. Također bih se željela pozabaviti odmrzavanjem samo dijelova okosnice (npr. posljednje razine).

LITERATURA

[1] Camvid - kaggle. URL https://www.kaggle.com/datasets/naureenmohammad/camvid-dataset.

[2] 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.

[3] Gabriel J. Brostow, Julien Fauqueur, i Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, xx(x):xx–xx, 2008.

[4] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, i Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. U *ECCV (1)*, stranice 44–57, 2008.

[5] Shenglong Chen, Yoshiki Ogawa, Chenbo Zhao, i Yoshihide Sekimoto. Large-scale individual building extraction from open-source satellite imagery via super-resolution-based instance segmentation approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 195:129–152, 2023. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2022.11.006. URL https://www.sciencedirect.com/science/article/pii/S0924271622002933.

[6] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, i Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning, 2022.

[7] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, i Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning

github implementation "openclip", 2022. URL https://github.com/mlfoundations/open_clip.

[8] 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. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

[9] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, i Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *CoRR*, abs/2110.04544, 2021. URL https://arxiv.org/abs/2110.04544.

[10] Dan Hendrycks i Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

[11] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, i Kilian Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016. URL http://arxiv.org/abs/1603.09382.

[12] Josip Krapac i Siniša Šegvić. Konvolucijski modeli. URL http://www.zemris.fer.hr/~ssegvic/du/du2convnet.pdf.

[13] Stephen Keckler, William Dally, Brucek Khailany, Michael Garland, i David Glasco. Gpus and the future of parallel computing. *Micro, IEEE*, 31:7 – 17, 11 2011. doi: 10.1109/MM.2011.89.

[14] Ivan Krešo, Josip Krapac, i Siniša Šegvić. Efficient ladder-style densenets for semantic segmentation of large images, 2019.

[15] Boyi Li, Kilian Q. Weinberger, Serge J. Belongie, Vladlen Koltun, i René Ranftl. Language-driven semantic segmentation. *CoRR*, abs/2201.03546, 2022. URL https://arxiv.org/abs/2201.03546.

[16] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, i Saining Xie. A convnet for the 2020s, 2022.

[17] Jonathan Long, Evan Shelhamer, i Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL http://arxiv.org/abs/1411.4038.

[18] Marin Oršić i Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern recognition*, 110:13, 2021. ISSN 0031-3203. doi: 10.1016/j.patcog.2020.107611.

[19] Sabina Pokhrel. Beginners guide to convolutional neural networks, 2019. URL https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, i Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

[21] René Ranftl, Alexey Bochkovskiy, i Vladlen Koltun. Vision transformers for dense prediction, 2021.

[22] Ringdongdang. Relu and gelu visualisation.

[23] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[24] Wei Yin, Yifan Liu, Chunhua Shen, Anton van den Hengel, i Baichuan Sun. The devil is in the labels: Semantic segmentation from sentences. *CoRR*, abs/2202.02002, 2022. URL https://arxiv.org/abs/2202.02002.

[25] Marko Čupić. Umjetna inteligencija, 2016. URL http://java.zemris.fer.hr/nastava/ui/ann/ann-20180604.pdf.

.

Sažetak

Računalni vid ima široku primjenu u stvarnom svijetu. Semantička segmentacija jedan je od njegovih temeljnih, ali i najzanimljivijih, zadataka. Ovaj rad predstavlja CLIP multimodalni sustav za ugrađivanje slika i teksta u isti vektorski prostor te ga iskorištava za generiranje jezičnih ugrađivanja koja se dalje koriste za semantičku segmentaciju. U tu svrhu prenamjenjuje se, modificira i nadograđuje naduzorkovateljem predtrenirani ConvNext-Base vizualni koder CLIP modela koji zatim individualne piksele kodira u isti vektorski prostor kao i CLIP model te, na temelju sličnosti tih kodiranja, ih klasificira. Korišten je skup CamVid podataka. Postupak izgradnje, rezultati i evaluacija pažljivo su i precizno opisani u radu. Dodatno su opisanu dijelovi i postupci treniranja dubokih modela, semantička segmentacija te korištene tehnologije i metrike.

Ključne riječi: računalni vid, semantička segmentacija, CLIP, CamVid, jezična ugrađivanja

Semantic segmentation by means of language embeddings

Abstract

Computer vision has found many applications in the real world. Semantic segmentation is one of its most basic components, but arguably one of the most interesting too. This paper presents the multimodal CLIP model for embedding images and text into the same vector space and uses it to generate language embeddings to further use in semantic segmentation. To that purpose, its ConvNext-Base visual tower is repurposed, modified and upgraded with an upsampling module so that it may embed individual pixels into the aforementioned space and exploit that for classification. All of this is done on the CamVid data set. The process of building such a model, evaluating it and the results are described with precision and detail. Additionally described are: deep neural network training, semantic segmentation and the used technologies and metrics.

Keywords: computer vision, semantic segmentation, CLIP, CamVid, language embedding