

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2095

Konvolucijski modeli za klasifikaciju videa

Antonio Borac

Zagreb, lipanj 2020.

Zagreb, 2. ožujka 2020.

Grana: **2.09.04 umjetna inteligencija**

DIPLOMSKI ZADATAK br. 2095

Pristupnik: **Antonio Borac (0036492829)**

Studij: **Računarstvo**

Profil: **Računarska znanost**

Zadatak: **Konvolucijski modeli za klasifikaciju videa**

Opis zadatka:

Klasifikacija videa neriješen je problem računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme najbolji rezultati u tom području postižu se dubokim konvolucijskim modelima. Ovaj rad razmatra nadzirane pristupe za klasifikaciju radnji osoba, gdje je svaki video skupa za učenje označen semantičkim razredom radnje koju video prikazuje.

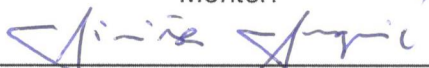
U okviru rada, potrebno je proučiti konvolucijske arhitekture za klasifikaciju videa. Oblikovati klasifikacijski model za raspoznavanje videa iz podatkovnog skupa UCF-101. Validirati hiperparametre, prikazati i ocijeniti ostvarene rezultate te provesti usporedbu s rezultatima iz literature. Predložiti pravce budućeg razvoja.

Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 13. ožujka 2020.

Rok za predaju rada: 30. lipnja 2020.

Mentor:



Prof. dr. sc. Siniša Šegvić

Djelovođa:



Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za
diplomski rad profila:



Doc. dr. sc. Marko Čupić

*Zahvaljujem mentoru prof. dr.
sc. Siniši Šegvicu na pruženoj pomoći tijekom studiranja i pri izradi ovog rada. Hvala
mojoj obitelji i prijateljima na strpljenju i podršci tijekom cijelog školovanja.*

SADRŽAJ

1. Uvod	1
2. Duboko učenje	3
2.1. Konvolucijski modeli	7
2.1.1. Konvolucijski sloj	7
2.1.2. Potpuno povezani sloj	8
2.1.3. Sloj sažimanja	9
2.1.4. Normalizacija po grupama	9
2.1.5. Slojevi za trodimenzionalne podatke	10
3. Konvolucijski modeli za klasifikaciju videa	11
3.1. Kralježnička arhitektura (engl. <i>backbone</i>)	11
3.1.1. Konvolucijski modeli sa rezidualnim vezama	12
3.1.2. MobileNets	13
3.2. Modeli sa sažimajućim značajkama	16
3.3. SlowFast arhitektura	18
4. Programska izvedba	21
4.1. Keras	21
4.2. Keras generator podataka	22
4.3. Modeli sa sažimajućim značajkama	23
4.4. SlowFast model	24
5. Skupovi podataka za učenje	26
5.1. UCF-101	26
6. Eksperimenti	28
6.1. Metrike	28
6.2. Povećanje skupa za učenje	28

6.3. Rezultati	29
6.3.1. Modeli sa sažimajućim značajkama	29
6.3.2. SlowFast model	31
6.3.3. Slike rezultata	32
7. Zaključak	35
Literatura	37

1. Uvod

Duboko učenje je od svojih početaka znatno napredovalo. Za takav napredak su najviše zaslužni intenzivni razvoj sklopovlja koji podržava izrazito paralelnu obradu podataka i razvoj algoritama koji efikasno koriste takvo sklopovlje. U posljednjih nekoliko godina izgrađeni su brojni modeli koji uspješno rješavaju problem klasifikacije slika, na kojem postižu bolje rezultate od čovjeka. U istraživanjima su uglavnom zastupljeni modeli koji se koriste za klasifikaciju slika, prepoznavanje govora i analizu teksta. Uspjesi postignuti na tim problemima naveli su stručnjake da pokušaju riješiti složenije probleme poput klasifikacije videa metodama dubokog učenja.

Klasifikacija videa neriješen je problem računalnog vida s mnogim zanimljivim primjenama. Iako je video niz od N sličica odnosno okvira, klasifikacija videa nije samo klasifikacija N sličica. Prilikom klasifikacije videa u obzir se mora uzeti i odnos između pojedinih okvira videa. Obično pretpostavljamo da su susjedni okviri u videu korelirani i modeli za klasifikaciju videa moraju uzeti tu činjenicu u obzir. U posljednje vrijeme najbolji rezultati u području klasifikacije videa postižu se dubokim konvolucijskim modelima. Prije nego što su se masovno počeli koristiti konvolucijski modeli, istraživanja na području klasifikacije videa su vodila u smjeru histograma slikovnih riječi, te su se za samu klasifikaciju često koristili SVM modeli [10].

U okviru rada ćemo objasniti česte pojmove u području dubokog učenja i opširno ćemo opisati ključne komponente nužne za razvoj dubokih modela za klasifikaciju videa. Detaljno ćemo razmotriti dvije arhitekture koje ostvaruju dobre rezultate na skupovima namijenjenima za učenje modela za klasifikaciju videa. Osim samih arhitektura, opisat ćemo i kralježničke modele (engl. *backbone*) koje koristimo. Razmotrit ćemo kako pristupiti videu kao podatku za učenje. Izgradit ćemo opisane modele i evaluirati ostvarene rezultate.

U izgradnji naših modela nam uvelike pomažu programski okviri visokog nivoa poput Kerasa [1] i PyTorch-a [2]. Za izgradnju naših modela smo odlučili koristiti programski okvir Keras. U okviru rada ćemo dati kratak pregled navedenog programskog okvira i prikazat ćemo kako jednostavno i brzo izgraditi željene modele.

Iz naših eksperimenata ćemo izvesti zaključak u kojem ćemo sažeto iznijeti postignute rezultate. Usporedit ćemo dobivene rezultate s rezultatima iz literature i dati prijedloge za eventualno poboljšanje. Navest ćemo prijedloge za daljnji rad temeljen na izvedenim eksperimentima u sklopu ovog rada.

2. Duboko učenje

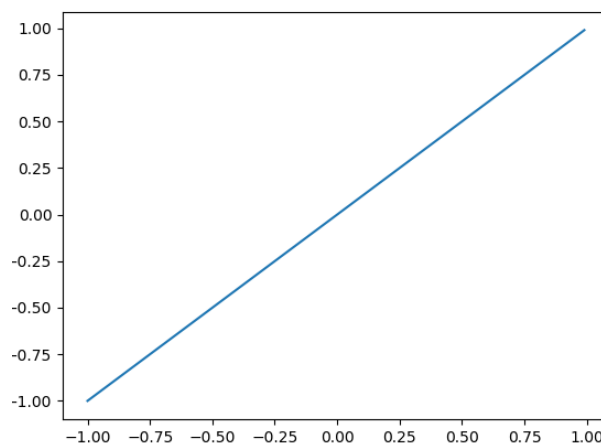
U ovom poglavlju ćemo se upoznati sa ključnim slojevima na kojima se baziraju naši modeli. Razumijevanje načina rada tih slojeva je veoma važno za razumijevanje arhitekture koje razmatramo u okviru našeg rada.

Za početak, upoznajmo se sa osnovnim pojmovima dubokog učenja.

Aktivacijska funkcija

Kako bi konvolucijskim modelima mogli modelirati nelinearne funkcije, možemo koristiti različite nelinearne aktivacijske funkcije. Najpoznatije aktivacijske funkcije koje se koriste su linearna aktivacijska funkcija, softmax funkcija, sigmoidalna funkcija i funkcija zglobnice.

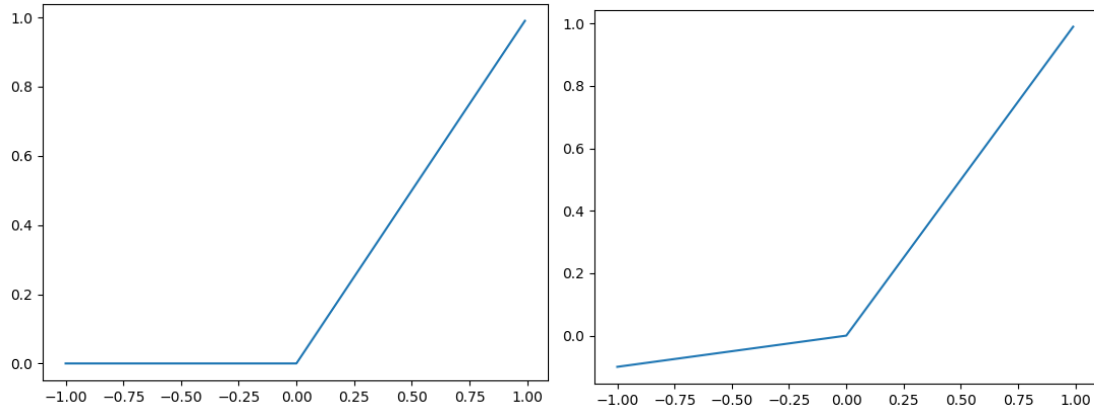
Linearna aktivacijska funkcija je definirana formulom $h(x) = x$.



Slika 2.1: Linearna aktivacijska funkcija.

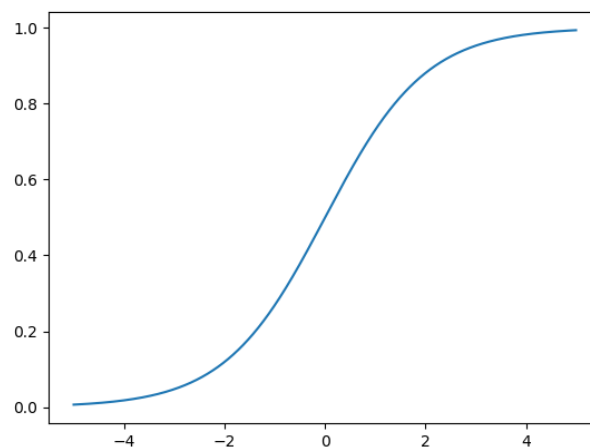
Funkcija zglobnica (engl. *rectified linear unit*, *ReLU*) je definirana formulom $h(x) = \max(0, x)$. Uz popularnu ReLU funkciju, primjenu nalazi i propusna zglobnica (engl. *Leaky ReLU*) koja je definirana formulom $h(x) = \max(\alpha x, x)$, $\alpha \geq 0$. Propusna zglobnica je pokušaj rješavanja problema umiruće zglobnice (engl. *dying ReLU*) koji se

pojavlja se kod funkcije zglobnice kad se u modelu ne koristi sloj normalizacije po grupama. Problem kod zglobnice je što joj je derivacija jednaka 0 za $x < 0$. U tom slučaju dolazi do potencijalno neželjenog gašenja neurona.



Slika 2.2: ReLU(lijevo) i LeakyReLU(desno).

Sigmoidalna funkcija je definirana formulom $h(x) = \frac{1}{1+\exp^{-x}}$. Prednost sigmoidalne aktivacijske funkcije je što joj je kodomena skup $< 0, 1 >$, pa se može interpretirati kao vjerojatnost. Sigmoida ima neželjeno svojstvo - za velike pozitivne i negativne vrijednosti ulazi u područje zasićenja. Gradijent funkcije u tom području postaje malen i neuron više ne može učiti. Taj problem u literaturi nazivamo problemom nestajućih gradijenata. Upravo taj problem je jedan od problema koji su ograničavali duboke modele. Problem je umanjen korištenjem aktivacijske funkcije zglobnice i sloja normalizacije po grupama.



Slika 2.3: Sigmoidalna aktivacijska funkcija.

Funkcija softmax je definirana formulom $h(x)_i = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)}$ za $i = 1, \dots, K$ i $z = (z_1, \dots, z_k) \in R^K$. Možemo primjetiti da je $h(x)_i$ vrijednost u skupu $< 0, 1 >$ i

da je $\sum_i^K h(x)_i$ jednaka 1, pa vrijednost $h(x)_i$ možemo interpretirati kao vjerojatnost pripadnosti x i -tom razredu.

Primjena aktivacijske funkcije zglobnice na aktivacije skrivenih slojeva je jedan od ključnih faktora koji je omogućio uspješno učenje dubokih modela.

Funkcija gubitka

Funkcija gubitka predstavlja mjeru troška nekog događaja. U kontekstu dubokog učenja predstavlja mjeru odstupanja stvarnog izlaza modela od željenog izlaza za dani ulaz. Cilj učenja modela je smanjiti vrijednost funkcije gubitka za dane ulaze, odnosno optimizirati funkciju gubitka. Postoje brojne funkcije gubitka koje se koriste u dubokom učenju ovisno o problemu koji se rješava. Tako u regresijskim modelima često susrećemo srednju kvadratnu pogrešku (engl. *Mean Absolute Error*), dok u klasifikacijskim problemima često susrećemo unakrsnu entropiju (engl. *Cross Entropy Loss*), funkciju koju ćemo i mi optimizirati u našim eksperimentima. Tu pogrešku računamo na sljedeći način:

$$L_i = - \sum_j^K y_{i,j} \log p_{i,j} \quad (2.1)$$

gdje je K broj razreda, $y_{i,j}$ 1 ako i -ti primjer pripada j -tom razredu inače 0, $p_{i,j}$ vjerojatnost pripadnosti i -tog primjera j -tom razredu. $p_{i,j}$ je izlaz `softmax` funkcije, a y_i je jedinični vektor koji ima vrijednost 1 na indeksu razreda kojem primjer pripada, pod uvjetom da su razredi poredani.

Optimizacijski postupak

Optimizacijskim postupkom nastojimo korigirati parametre neuronskih mreža kako bi mreža predstavljala što bolju aproksimaciju funkcije koju želimo modelirati. Naime, za razliku od klasičnog optimizacijskog problema kod kojih je poznata distribucija koju modeliramo, u dubokom učenju to najčešće nije slučaj. Optimiramo neku mjeru gubitka J u nadi da ćemo indirektno optimirati i originalnu mjeru P . Taj postupak je najčešće iterativan. Česti optimizacijski postupci koje koristimo su Adam, stohastički gradijentni spust i Adagrad. Postoji mnogo optimizacijskih postupaka koji se koriste ovisno o problemu koji se rješava.

Stohastički gradijentni spust je iterativna metoda za optimizaciju funkcije. Ideja koja se krije iza ove metode je da za informaciju o daljnjem smjeru u kojem trebamo ići sa parametrima funkcije koristimo gradijent funkcije. Prva derivacija derivabilne funkcije pokazuje smjer najbržeg porasta vrijednosti funkcije gubitka. Želimo se kre-

tati u smjeru suprotnom od smjera najvećeg porasta kako bi smanjili vrijednost funkcije gubitka u sljedećoj iteraciji. Bitno je istaknuti da se samo ažuriranje težina može obaviti nakon svakog primjera koje model vidi, nakon nekoliko primjera i nakon svih primjera skupa za treniranje. Ako ažuriranje obavljamo nakon svakog primjera, govorimo o online učenju, pri čemu je gradijent samo aproksimacija pravog gradijenta. Tada je formula za stohastički gradijentni spust

$$w = w - \eta \nabla f(x_i) \quad (2.2)$$

gdje je w težina koju ažuriramo, η stopa učenja. Kad ažuriranje obavljamo nakon svih primjera, govorimo o offline učenju, te formula tada postaje

$$w = w - \eta \sum_{i=0}^N \nabla f(x_i) \quad (2.3)$$

gdje je N broj primjera u skupu za učenje. U tom slučaju govorimo o učenju s grupama. Kad ažuriranje obavljamo nakon nekoliko primjera, govorimo o učenju s mini grupama. U slučaju kad za učenje koristimo cijeli skup za treniranje, gradijent u lokalnim i globalnim optimuma funkcije gubitka iznosi 0, dok u slučaju kad koristimo manje primjera to ne mora biti slučaj.

Optimizacijski postupak Adam je također iterativan postupak kao i stohastički gradijentni spust. Pripada grupi algoritama koji koriste moment za učenje modela. Algoritam prati eksponencijalni pomični prosjek gradijenata i kvadrata gradijenata. Korekcija se obavlja prema eksponencijalnom pomičnom prosjeku gradijenata a ne prema izračunatom gradijentu u određenom koraku.

Regularizacija

Prilikom optimizacije funkcije gubitka, često dolazi do problema da model previše dobro nauči klasificirati uzorke iz skupa za učenje, odnosno da "nauči napamet" skup za učenje. Model tad ne modelira željenu funkciju već se ponaša kao memorija koja pamti podatke iz skupa za učenje. Ako na ulaz modela dovedemo neki uzorak koji model do tad nije vidio, model će se loše ponašati. Taj problem želimo ublažiti brojnim tehnikama regularizacije. U sklopu naših eksperimenata, između ostalog koristimo L2 regularizaciju, sloj normalizacije po grupama i povećanje skupa podataka za učenje. L2 regularizacija na funkciju gubitka modela dodaje regularizacijski faktor koji ovisi o normi vektora težina.

2.1. Konvolucijski modeli

Konvolucijske neuronske mreže su specijalizacija modela sa potpuno povezanim slojevima. Za razliku od modela s potpuno povezanim slojevima, koriste konvolucijske slojeve u skrivenim slojevima mreže. Konvolucijski slojevi modeliraju lokalnu interakciju i dijele parametre, što im omogućava korištenje manje parametara u odnosu na modele s potpuno povezanim slojevima [19]. Pokazali su se kao dobar pristup širokom spektru problema računalnog vida. Danas, većina od modela koji ostvaruju najbolje rezultate na skupovima za klasifikaciju slika su upravo konvolucijski modeli [3]. U nastavku ćemo opisati te slojeve, kako ti slojevi rade te za što se koriste.

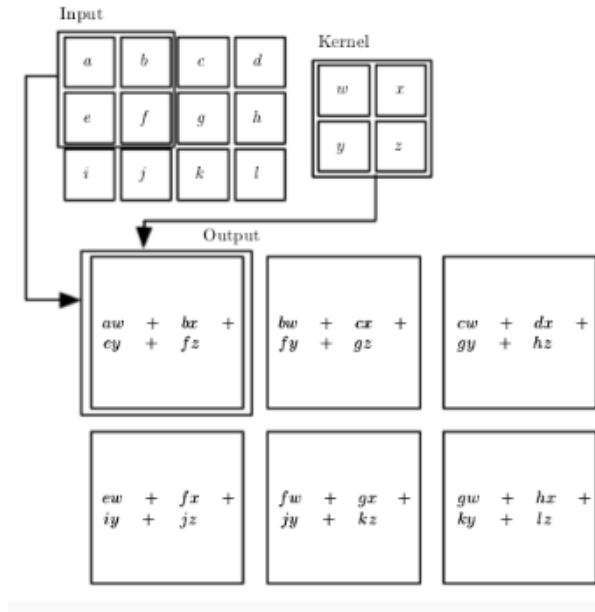
2.1.1. Konvolucijski sloj

Konvolucijski slojevi predstavljaju najvažnije slojeve u konvolucijskim modelima. Naziv su dobili po operaciji konvolucije¹ koju provode nad ulaznim podacima. Sama operacija je diferencijabilna, što ju čini iznimno privlačnom za korištenje u modelima koji su optimizirani sa algoritmima učenja koji koriste gradijente.

Operacija konvolucije se provodi uz pomoć filtra. Filtar je matrica težina kojom se množe pripadajući dijelovi ulazne mape značajki kako bi se dobio element izlazne mape značajki. Princip rada filtra je dobro prikazan na slici 2.4. Svaki konvolucijski sloj sadrži filtarij kojim se obavlja operacija konvolucije. Operaciju je najjednostavnije objasniti na jednom primjeru. Ulaz u sloj je slika dimenzija $32 \times 32 \times 3$. Koristimo filtari sa 16 filtra dimenzija 3×3 . Jezgru po ulazu pomičemo za 1 u horizontalnom i vertikalnom smjeru. Filtari operaciju konvolucije provodi nad svim kanalima ulazne mape značajki, tako da je zapravo dimenzija svakog filtra $3 \times 3 \times 3$. Dubina izlazne mape značajki će biti jednaka broju filtara koji koristimo, što je u našem slučaju jednako 16. Prostorne dimenzije izlaznog sloja računamo pomoću sljedeće formule: $W_{out} = (W_{in} - F + 2P)/S + 1$, gdje je W_{in} prostorna dimenzija ulaza, F dimenzija jezgre, P broj nula koje nadodajemo oko podataka po dimenziji koju računamo što je bitno za dimenzije izlazne mape značajki, S pomak po ulazu a W_{out} prostorna dimenzija izlaza. Prema toj formuli, primjerice širina izlazne mape značajki u našem slučaju postaje $W_{out} = (32 - 3 + 2 * 0)/1 + 1 = 30$. Dimenzija izlazne mape značajki je $30 \times 30 \times 16$. Primjetimo da smo u ovom slučaju izabrali P jednak 0, odnosno ne koristimo nadopunjavanje ulazne mape značajki, što je dovelo do smanjenja prostornih dimenzija izlazne mape značajki. U literaturi, takva postavka nadopunjavanja

¹U strojnom učenju pod operaciju konvolucije podrazumijevamo operaciju unakrsne korelacije [19].

je često nazvana "valid". Često se koristi i nadopunjavanje nulama koje bi dovelo do toga da prostorne dimenzije izlazne mape značajki ostanu jednake prostornim dimenzijama ulazne mape značajki. U literaturi, takvu postavku nadopunjavanja često zovemo "same".



Slika 2.4: Prikaz operacije konvolucije. Slika je preuzeta iz [6].

Na slici 2.4 možemo vidjeti kako se odvija operacija konvolucije. Jezgru filtra pomičemo po ulaznoj mapi značajki. Svaki element prozora ulazne mape značajki množimo sa poklapajućim elementom jezgre. Dobivene brojeve zbrojimo i pomičemo jezgru za pomak s kako bi izračunali sljedeći element izlazne mape značajki.

2.1.2. Potpuno povezani sloj

U potpuno povezanom sloju je svaka aktivacija prethodnog sloja povezana sa svakim neuronom trenutnog sloja. Ako broj aktivacija u prethodnom sloju označimo sa n , izlaz j -tog neurona potpuno povezanog sloja tada je:

$$x_j = \sum_i^n x_{i,j-1} w_{i,j-1} + b_j \quad (2.4)$$

, gdje je b_j prag za j -ti neuron. Vidljivo je da je na ovaj način moguće modelirati samo linearne funkcije, pa često na izlaz neurona dodajemo neku nelinearnu funkciju koju nazivamo aktivacijska funkcija. Tada izlaz j -tog neurona uz nelinearnu aktivacijsku funkciju ρ postaje:

$$x_j = \rho\left(\sum_i^n x_{i,j-1} w_{i,j-1} + b_j\right) \quad (2.5)$$

. Ako je n broj izlaza prethodnog sloja, a m broj neurona u trenutnom sloju i ako uzmemo u obzir sve pragove za svaki neuron, tada će potpuno povezani sloj ukupno imati $m \cdot n + m$ težina.

U konvolucijskim modelima potpuno povezane slojeve često susrećemo "na kraju" modela, gdje se koristi sa aktivacijskom funkcijom softmax kako bi izlaz modela dao vjerojatnost klasifikacije ulaznog podatka za svaki semantički razred skupa podataka. Izbjegavamo ih koristiti u početnim dijelovima konvolucijskog modela jer unose jako puno parametara.

2.1.3. Sloj sažimanja

Sloj sažimanja najčešće se koristilo kako bi se smanjile prostorne dimenzije ulazne mape značajki što je vodilo do smanjenja broja parametara i utjecalo na smanjivanje pretreniranosti modela. Danas se sloj sažimanja koristi za smanjenje memorijskog i računskog opterećenja prilikom učenja i evaluiranja modela.

Svaki sloj sažimanja ima definiran prozor sažimanja koji provodi definiranu operaciju nad dijelom ulazne mape značajki. Postoji niz operacija koje je moguće provoditi nad prozorom ulazne mape značajki, od kojih najčešće susrećemo operaciju maksimuma i aritmetičke sredine. Sloj sažimanja maksimumom kao rezultat daje maksimum iz trenutnog prozora iz ulazne mape značajki. Bitno je napomenuti da dubina izlazne mape značajki ostaje ista dubini ulazne mape značajki, što nije nužno bio slučaj kod konvolucijskog sloja gdje je dubina izlazne mape značajki jednaka broju jezgri. Širinu izlazne mape značajki računamo pomoću sljedeće formule: $w_{out} = (w_{in} - p) / s + 1$, gdje je p širina prozora sažimanja, w_{in} širina ulazne mape značajki, s pomak prozora po širini. Analogno računamo i visinu izlazne mape značajki.

U praksi u konvolucijskim modelima često susrećemo slojeve globalnog sažimanja. Obično ih susrećemo na kraju konvolucijskog modela gdje je dubina mape značajki veća od širine i visine. Efektivno, prozor kod globalnog sažimanja je jednak prostornim dimenzijama mape značajki, što znači da je izlaz ovog sloja jednak $1 \times 1 \times D$, gdje je D dubina mape značajki.

2.1.4. Normalizacija po grupama

Normalizacija po grupama (engl. *batch normalization*) je široko rasprostranjena tehnika koja omogućava brže i stabilnije treniranje dubokih modela [17]. Tehnika radi tako da stabilizira distribuciju ulaza sloja koju kontroliraju srednja vrijednost i varijanca ulaza. Kod dubokih modela, istoremena korekcija težina u svim slojevima stvara

problem jer su slojevi povezani - izlaz jednog sloja predstavlja ulaz drugog sloja. Dolazi do efekta promjene ulaznih podataka za drugi sloj što se naziva kovarijacijski pomak. To želimo izbjeći, izlaz svakog sloja bi trebao ponovno biti normaliziran što sljedećem sloju osigurava stabilnije ulaze. Normalizacija se radi na temelju izlaza koje taj neuron ima kad na ulaz mreže dovedemo sve primjere iz mini grupe [19]. Računa se srednja vrijednost odziva neurona i varijanca na temelju ulaznih podataka. Pomoću tih vrijednosti normaliziramo odziv za svaki ulaz iz mini grupe, skaliramo izlaz i dodajemo pomak. Formula koja prikazuje taj postupak je $y = \alpha y' + \beta$, gdje je y' normalizirani izlaz, y konačni izlaz, α parametar za skaliranje i β fiksni pomak. Parametri α i β se uče tijekom faze učenja modela.

Smatralo se da normalizacija po grupama efikasno pomaže pri učenju modela zato jer smanjuje interni kovarijacijski pomak. Rad [17] otkirva da je najveći doprinos sloja normalizacije po grupama taj da značajno zaglađuje funkciju cilja. Takvo zaglađivanje funkcije cilja vodi do stabilnijeg ponašanja gradijenata što dalje vodi do bržeg učenja.

2.1.5. Slojevi za trodimenzionalne podatke

Iako se najčešće susrećemo sa operacijama konvolucije i sažimanja koje koriste dvodimenzionalne prozore i jezgre, moguće je susresti i višedimenzionalne prozore i jezgre. U okviru ovog rada se susrećemo sa konvolucijskim slojevima i slojevima sažimanja koji imaju trodimenzionalnu jezgru. U tom slučaju, na ulaz primamo tenzor sljedećih dimenzija: `(batch_size, sp1, sp2, temp, channels)`. U slučaju 2d konvolucijskog sloja jezgra se pomiče u horizontalnom i vertikalnom smjeru po ulaznoj mapi značajki i pri svakom pomaku kreira jedan element izlazne mape značajki. Takvim pomicanjem jezgre nastaje jedna matrica za svaku jezgru konvolucijskog sloja. U slučaju trodimenzionalnih jezgri, osim što se jezgre pomiču u horizontalnom i vertikalnom smjeru, pomiču se i po dodatnoj dimenziji. Rezultat djelovanja svake jezgre bit će po jedan volumen. Operacija koja se provodi se pri tome ne mijenja, osim što za jedan element izlazne mape značajki računamo konvoluciju trodimenzionalne jezgre sa trenutnim dijelom ulaznog volumena. Ista stvar vrijedi i za slojeve sažimanja.

Ovakvi slojevi se koriste za sekvencijalne² podatke kao što je video.

²Sekvencijanim podacima nazivamo sve podatke kod kojih je bitan poredak.

3. Konvolucijski modeli za klasifikaciju videa

3.1. Kralježnička arhitektura (engl. *backbone*)

U okviru našeg rada, "backbone" arhitekturom smatramo dio mreže koji koristimo za ekstrakciju značajki. U našim eksperimentima ćemo koristiti modele koji su predtrenirani na Imagenet skupu podataka kako bi ubrzali proces učenja cjelokupnog modela. Imagenet je skup podataka koji se koristi za klasifikaciju slika koji sadrži preko 14 milijuna podataka i koji je označen sa preko 20 000 razreda. Ako prilikom treniranja modela težine inicijaliziramo nasumičnim vrijednostima, izlazi modela na samom početku treniranja će biti slučajni. Tek nakon treniranja modela kroz određen broj epoha će težine biti postavljene na određene vrijednosti uz koje izlaz mreže ima smisla. Predtreniranje je postupak kojim ubrzavamo taj proces. Druga velika prednost predtreniranja modela je što je model moguće naučiti sa manje primjera za učenje [14] unatoč tome što distribucije težina na problemu na kojem se težine predtreniraju i na problemu koji se rješava nisu iste. Ako težine inicijalno postavimo na vrijednosti dobivene treniranjem na nekom bliskom problemu, modelu će vjerojatno trebati manje vremena da nauči težine koje su dobre za trenutni problem koji rješavamo od nasumične inicijalizacije.

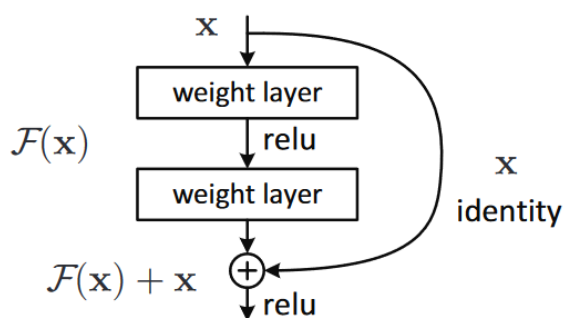
Iako predtreniranje na skupovima poput Imageneta nije nužno¹ za dobre rezultate modela, u praksi se pokazuje da predtreniranje na Imagenetu i kasnije ugađanje na skupu nad kojim obavljamo treniranje ubrzava konvergenciju. U radu ćemo koristiti ekstraktore značajki koji su trenirani na Imagenet skupu podataka koje ćemo ugađati u nadi da ćemo dobiti dobre rezultate.

¹Rad [8] pokazuje da predtreniranje nije nužno i da se jednako dobri rezultati mogu dobiti slučajnom inicijalizacijom na problemima poput detekcije ako model učimo na jako velikom skupu podataka. U slučaju kad imamo jako malo podataka za učenje, predtreniranje je ključno za postizanje dobrog rezultata [15].

3.1.1. Konvolucijski modeli sa rezidualnim vezama

Duboki konvolucijski modeli sa rezidualnim vezama su prisutni od 2015. te su od samog predstavljanja postali jako popularni. Koriste se na gotovo svim područjima računalnog vida. Velik problem kod učenja dubokih modela predstavljao je problem nestajućih gradijenata. Taj problem je uglavnom riješen normalizacijom ulaza i korištenjem slojeva normalizacije. Duboki modeli počinju konvergirati korištenjem navedenih slojeva, ali se uočava drugi problem - problem degradacije točnosti. Povećanjem broja slojeva modela dolazi do zasićenja točnosti, pa daljnjim povećanjem broja slojeva i do degradacije točnosti. Iako bi očekivali da do smanjenja točnosti dolazi zbog pre-naučenosti, pokazuje se da to nije slučaj. Dodavanjem slojeva na model koji pokazuje dobre rezultate dolazi do povećanja pogreške na skupu za učenje. Rad [7] problemu degradacije pristupa uvođenjem rezidualnih veza.

Ako pretpostavimo da niz slojeva može aproksimirati neku složenu funkciju $H(x)$, možemo pretpostaviti da slojevi mogu također aproksimirati neku složenu funkciju $F(x) = H(x) - x$. Ako model aproksimira funkciju $F(x)$, originalna funkcija koju je potrebno modelirati postaje $F(x) + x$. Rad [7] pretpostavlja da je takvo mapiranje lakše naučiti nego originalno mapiranje $H(x)$. Formulacija $F(x) + x$ može biti ostvarena sa unaprijednom neuronskom mrežom sa preskočnim vezama. Preskočne veze preskaču jedan ili više slojeva u modelu. U okviru rada, preskočne veze preskaču niz od nekoliko slojeva, na izlaz niza slojeva se dodaje doprinos preskočne veze. Takvu konstrukciju nazivamo rezidualna jedinica. Preskočne veze ne unose nove parametre u model i ne povećavaju složenost.

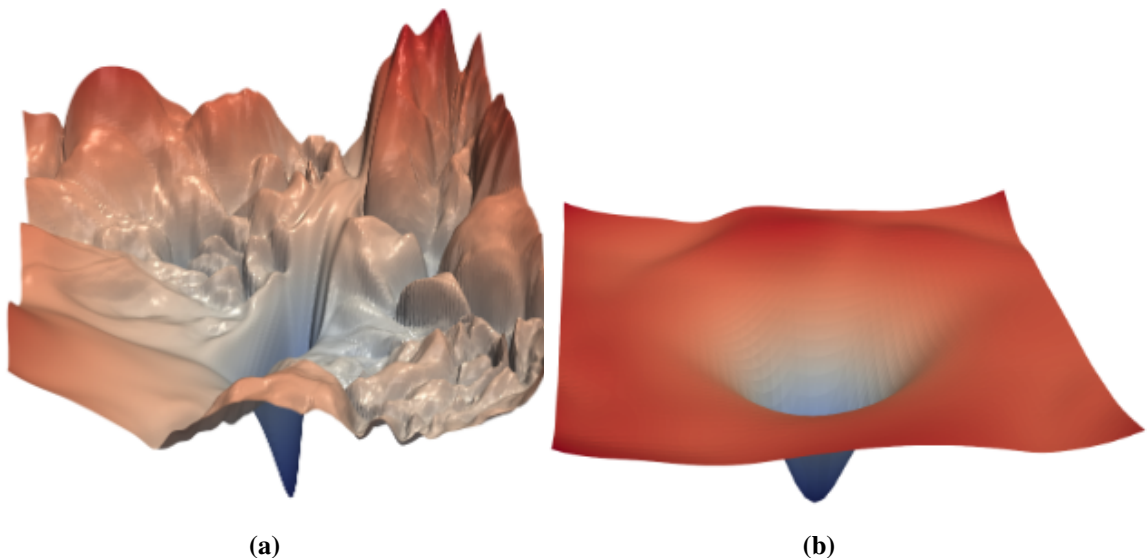


Slika 3.1: Prikaz rezidualne jedinice. Prikaz je preuzet iz [7].

Ovakav postupak ima mnogo prednosti. Osim što dodatno umanjuje problem nestajućih gradijenata, omogućava gradnju kompleksnijih ovisnosti u mreži. Omogućava direktnu propagaciju informacija kroz slojeve, odnosno omogućava učenje slojeva koji direktno ovise o "plićim" dijelovima modela.

Predlažu se dva načina slaganja rezidualnih jedinica. Prvi način je osnovni način (engl. *basic*) gdje se u svakoj rezidualnoj jedinici koriste dva konvolucijska sloja sa jezgrom 3×3 . Drugi način je usko grlo (engl. *bottleneck*) gdje se koriste tri konvolucijska sloja, redom 1×1 , 3×3 i 1×1 , gdje su 1×1 slojevi zaduženi za smanjivanje i potom povećavanje dimenzija kako bi 3×3 sloj imao manje ulazne i izlazne dimenzije. Drugi način se koristi u dubljim rezidualnim arhitekturama kako bi se kontrolirao broj parametara. Nakon svakog konvolucijskog sloja dolazi sloj normalizacije po grupi i zglobnica kako bi se smanjio utjecaj nestajućih gradijenata.

Na arhitekturi se kontinuirano radi uz mnoga poboljšanja. Poznato je da neke arhitekture u praksi pokazuju dobre rezultate ali nam nije odmah jasno zašto. Jedna od takvih arhitektura je i rezidualna arhitektura. Preskočne veze zaglađuju funkciju cilja što pomaže pri učenju modela.



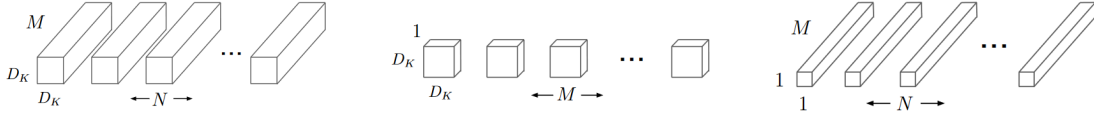
Slika 3.2: Na slici a je prikazana funkcija cilja bez korištenja preskočnih veza dok je na slici b prikazana funkcija cilja nakon ubacivanja preskočnih veza u model. Prikazi su preuzeti iz [12].

3.1.2. MobileNets

MobileNets predstavljaju razred efikasnih modela za mobilne i ugradbene sustave namijenjene za računalni vid. Baziraju se na arhitekturi koja koristi dubinski separabilne konvolucije za izgradnju efikasnih dubokih modela [9].

Arhitektura se ističe po tome što obične konvolucijske slojeve mijenja sa dva konvolucijska sloja pri čemu prvi sloj ima ulogu nezavisnog filtriranja pojedinih mapa značajki, a drugi ulogu kombiniranja značajki između različitih kanala ulaza. Kombi-

naciju takva dva konvolucijska sloja nazivamo dubinski separabilna konvolucija. Prvi konvolucijski sloj je grupni konvolucijski sloj sa grupom veličine 1, a drugi točkasti konvolucijski sloj (engl. *pointwise convolutional layer*). U MobileNet arhitekturi, grupni konvolucijski sloj primjenjuje zasebnu jezgru za svaki ulazni kanal. Točkasti konvolucijski sloj koristi jezgru 1×1 kojim kombinira izlaze grupnog konvolucijskog sloja.



Slika 3.3: Standardna konvolucija (lijevo) je zamijenjena grupnom konvolucijom (sredina) i točkastom konvolucijom (desno). Slika je preuzeta iz [9].

Standardna konvolucija ima sljedeću složenost:

$$D_K * D_K * M * N * D_F * D_F \quad (3.1)$$

gdje je N broj ulaznih kanala, M broj izlaznih kanala, $D_k \times D_k$ dimenzija jezgre (uz pretpostavku da koristimo kvadratnu jezgru) i $D_f \times D_f$ prostorne dimenzije izlazne mape značajki. Kod grupnog konvolucijskog sloja imamo zasebnu jezgru za svaki ulazni kanal. Tada je njegova složenost:

$$D_K * D_K * M * D_F * D_F \quad (3.2)$$

Dubinski konvolucijski sloj je efikasan ali radi samo filtraciju ulaznih kanala, ne i njihovu kombinaciju. Točkasti konvolucijski sloj radi linearnu kombinaciju izlaza dubinskog konvolucijskog sloja sa 1×1 konvolucijom. Ukupna složenost dubinski separabilne konvolucije je:

$$D_K * D_K * M * D_F * D_F + M * N * D_F * D_F \quad (3.3)$$

U MobileNet arhitekturi svi su konvolucijski slojevi dubinski separabilni osim prvog konvolucijskog sloja. Prvi konvolucijski sloj prati 13 blokova dubinski separabilnih slojeva. Iza svakog konvolucijskog sloja dolazi sloj normalizacije po grupi i aktivacijska funkcija zglobnica.

MobileNet uvodi dva hiperparametra kojima se omogućuje da model bude još manji i brži iako time gubi na točnosti što ovisno o problemu koji rješavamo može biti prihvatljivo. Prvi hiperparametar je multiplikator dubine $\alpha \in (0, 1]$ koji smanjuje broj

ulaznih kanala u konvolucijski sloj, odnosno dubinski separabilna konvolucija tada ima sljedeću složenost:

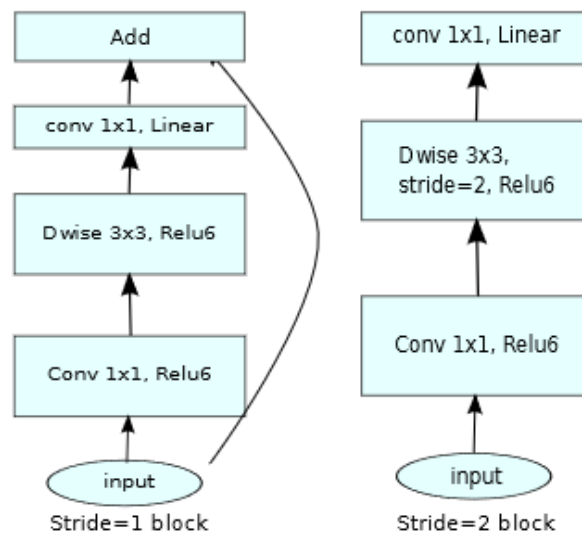
$$D_K * D_K * \alpha M * D_F * D_F + \alpha M * \alpha N * D_F * D_F \quad (3.4)$$

Ovaj parametar smanjuje složenost sloja i smanjuje broj parametara za faktor oko α^2 . Drugi hiperparametar je multiplikator rezolucije $\rho \in (0, 1]$ koji smanjuje prostorne dimenzije mapi značajki. Dubinski separabilna konvolucija tada ima složenost:

$$D_K * D_K * M * \rho D_F * \rho D_F + M * N * \rho D_F * \rho D_F \quad (3.5)$$

MobileNet ima nekoliko novijih verzija koja nose određena poboljšanja u odnosu na originalnu arhitekturu. Tako se MobileNetV2 arhitektura bazira na invertiranoj rezidualnoj strukturi.

Arhitektura koristi Relu6 aktivacijsku funkciju koja je slična zglobnici. Računa se pomoću formule $y = \min(\max(0, x), 6)$. Koristi se zbog robusnosti kod izračuna sa malom preciznosti.



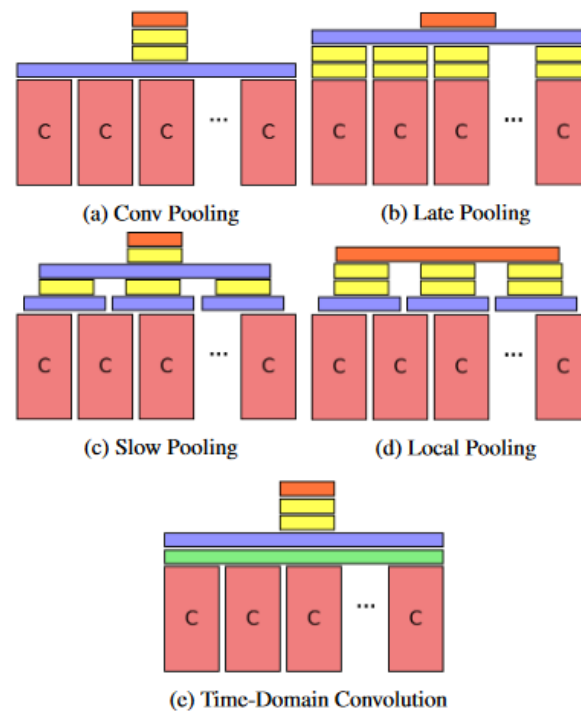
Slika 3.4: Prikaz invertiranog rezidualnog bloka sličnog rezidualnim blokovima koji su uvedeni u MobileNetV2 arhitekturu kao značajno poboljšanje u odnosu na MobileNet arhitekturu. Slika lijevo označava niz slojeva koji se koristi kad je pomak jednak 1 ili je dubina izlaznog sloja jednaka dubini ulaznog sloja. U tom slučaju se koristi preskočna veza. Slika desno označava slučaj kad je pomak jednak 2 ili je dubina izlazne mape značajki različita od dubine ulazne mape značajki. Tada se ne koristi preskočna veza. Primjetimo invertirani bottleneck dizajn, gdje se redom koriste konvolucije 1x1, 3x3 i 1x1. Prikaz je preuzet iz [16].

Na slici 3.4 je prikazana osnovna gradivna jedinica MobileNetV2 arhitekture. Prvi konvolucijski sloj 1x1 se koristi za ekspanziju dimenzija, a zadnji konvolucijski sloj

1×1 se koristi za ponovno sažimanje dimenzija. Primjetimo da je ovo obrnuto od bottle-neck arhitekture koja se koristi u rezidualnoj arhitekturi. Za prve dvije konvolucije se koristi aktivacija ReLU6 dok se za zadnju konvoluciju koristi linearna aktivacijska funkcija. Korištenje linearne aktivacije je ključno kako bi spriječili nelinearnosti da unište previše informacija, što se primjerice može dogoditi kod aktivacijske funkcije zglobnice.

3.2. Modeli sa sažimajućim značajkama

Prva arhitektura za klasifikaciju videa koju ćemo implementirati u sklopu našeg rada je arhitektura sažimajućih značajki (engl. *feature pooling architecture*). Iz svakog okvira videa se uz pomoć bloka konvolucijskih slojeva ekstrahiraju značajke. Potom se raznim kombinacijama slojeva sažimanja kombiniraju izlazi ekstraktora značajki za svaki pojedini okvir. Rad [13] predlaže nekoliko različitih načina kako kombinirati značajke. Na slojeve potrebne za kombinaciju značajki se dodaje klasifikator koji daje konačan izlaz modela. Conv Pooling arhitektura na izlaz bloka konvolucijskih slojeva postavlja



Slika 3.5: Prikaz različitih feature-pooling arhitektura. Crvena, plava, žuta, narančasta i zelena boja redom označavaju konvolucijske slojeve, sloj sažimanja, potpuno povezani sloj, softmax sloj, vremensku konvoluciju. Prikaz je preuzet iz [13].

sloj sažimanja maksimumom po vremenskoj domeni. Late Pooling na blok konvolucijskih slojeva dodaje dva potpuno povezana sloja. Nakon drugog potpuno povezanog sloja se dodaje sloj sažimanja po vremenskoj domeni te se računa izlaz modela. Slow pooling prvo kombinira značajke manjeg vremenskog prozora, na koji se dodaje potpuno povezani sloj te sloj sažimanja po vremenskoj domeni. Na taj sloj se dodaje još jedan potpuno povezan sloj i konačno se računa izlaz modela. Local Pooling je sličan prethodnoj arhitekturi. Razlika je u tome što ima samo jedan sloj sažimanja na koji se dodaju dva potpuno povezana sloja te se računa izlaz modela. Time-Domain Convolution koristi vremensku konvoluciju za kombinaciju značajki pojedinih okvira. Na taj sloj se dodaje sloj sažimanja i dva potpuno povezana sloja nakon kojih se računa izlaz modela.

U arhitekturama se kao sloj sažimanja koristi sloj sažimanja maksimumom. To nije nužno samo po sebi, ali se pokazalo da sloj sažimanja prosjekom i potpuno povezani sloj kao sloj sažimanja imaju problema sa učenjem zbog velikog broja gradijenata koje generiraju. U sklopu ovog rada smo odlučili isprobati Conv Pooling arhitekturu jer se pokazalo da od svih gore navedenih arhitektura postiže najbolje rezultate [13].

3.3. SlowFast arhitektura

Druga arhitektura koju razmatramo i implementiramo u sklopu našeg rada je Brzo-Spora (engl. *SlowFast*) arhitektura. Glavna pretpostavka na kojoj se bazira ova arhitektura je da su u videu sporiji pokreti izgledniji od brzih pokreta, što je i istina ako uzmemo u obzir da je većina svijeta oko nas koju percipiramo statična. Ako razmatramo brid nekog objekta koji se pomiče u proizvoljnom smjeru, možemo percipirati samo kretanje koje je okomito na taj brid, iako je moguće da postoji komponenta kretanja u tangencijalnom smjeru s obzirom na promatrani brid. To se događa zato jer pokret percipiramo pomoću promjene intenziteta piksela koja se u tangencijalnom smjeru nije promijenila. Problem nazivamo problemom otvora kamere. Kako naše oko prostorne



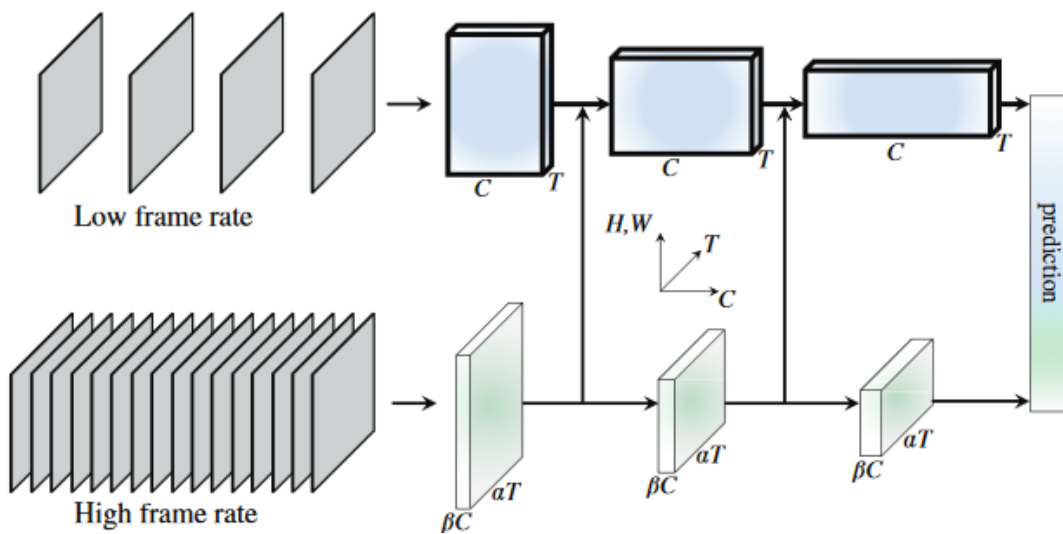
(a) Prikaz scene prije pomicanja objekta.

(b) Prikaz scene nakon pomicanja objekta u desno.

Slika 3.6: Crni objekt sa slike b se pomaknuo u lijevo u odnosu na sliku ???. Taj pomak je moguće percipirati u području desnog kruga, ali ne i u području lijevog i donjeg kruga. Taj problem nazivamo problemom otvora kamere.

i vremenske značajke različito percipira, rad pretpostavlja da bi model za klasifikaciju videa mogao jednako tako različito tretirati prostorne i vremenske značajke videa. Konkretno, ako razmatramo problem klasifikacije videa, kategorička prostorna semantika prikazanog sadržaja se sporo mijenja. Odnosno, na videu gdje je prikazano mahanje ruke, ruka ostaje ruka tijekom cijelog videa - objekt se sporo mijenja, dok se sama radnja mahanja ruke odvija relativno brzo. Za raspoznavanje ruke kao objekta nam je potrebno puno manje okvira videa nego za raspoznavanje mahanja ruke kao pokreta. Upravo tu činjenicu želimo iskoristiti u našoj arhitekturi. Ljudsko oko ima dvije vrste stanica, od kojih je 80% Parvocelularnih stanica (P-stanice) i 15 – 20% Magnocelularnih stanica (M-stanice). M-stanice su zadužene za brze vremenske promjene i nisu osjetljive na boju i prostorne detalje, dok su P-stanice zadužene za prostorne detalje i slabo reagiraju na brze promjene. Analogno tim spoznajama, u sklopu modela imamo dvije staze, jednu koja će obrađivati okvire uzrokovane visokom frekvencijom i drugu koja će obrađivati okvire uzorkovane nižom frekvencijom [5]. Prvu stazu nazivamo "Brza" staza, dok drugu nazivamo "Spora" staza.

"Spora" staza služi za hvatanje semantičke informacije iz videa, za koju je dovoljan jedan ili nekoliko okvira iz videa. "Brza" staza služi za hvatanje brzog pokreta u videu, promjene koje se odvijaju između gotovo svakog okvira. Ta staza koristi okvire koji su uzorkovani frekvencijom koja je α puta veća od frekvencije korištene za "Sporu" stazu. "Brza" staza je dizajnirana tako da ima β puta manje kanala (dubina mape značajki) od "Spore" staze kako bi imala slabiju mogućnost percipiranja prostornih informacija što je zadatak "Spore" staze. Staza je namjerno tako dizajnirana kako bi ubrzali izračun, na izračune u "Brznoj" stazi otpada 20% ukupnog izračuna modela. Na slici 3.7 vidimo prikaz kako izgleda SlowFast arhitektura.



Slika 3.7: Prikaz SlowFast arhitekture. Gornja stazu nazivamo Spora staza, a donju Brza staza prema frekvenciji uzorkovanja okvira. Informacije iz Brze staze ugrađujemo u Sporu stazu pomoću lateralnih veza u različitim dijelovima arhitekture. Tenzori se transformiraju sa 3D konvolucijskim slojevima. Iako je arhitektura namijenjena za obradu okvira iz videa, na ulaz pojedinih staza je moguće postaviti značajke iz nekog drugog modela. Prikaz je preuzet iz [5].

Kako bi dali modelu priliku da ugradi informacije iz jedne staze u drugu, koristimo lateralne veze. Originalni rad predlaže nekoliko različitih načina kako obaviti ugradnju informacija. U okviru naših eksperimenata ćemo isprobati sljedeći način. Koristit ćemo jednosmjernu vezu koja ugrađuje informacije iz "Brze" staze u "Sporu" stazu. Pokazuje se da dvosmjerna veza radi jednako dobro kao i jednosmjerna veza pa smo izabrali jednostavniju opciju. Ako je dimenzija mape značajki "Spore" staze T, S^2, C , tada je dimenzija "Brze" staze $\alpha T, S^2, \beta C$, gdje je T broj okvira, S prostorna dimenzija i C broj kanala mape značajki. Za ugradnju koristimo vremensku konvoluciju sa $2\beta C$ jezgri dimenzija $\{5, 1^2\}$, pri čemu je veličina jezgre iskazana kao $\{T, S^2\}$ gdje je T veličina jezgre po vremenskoj osi, S^2 veličine jezgre po prostornim osima. Za ope-

raciju konvolucije se koristi pomak α . U našim eksperimentima koristimo ResNet18 prilagođen za trodimenzionalne podatke. Nakon svakog rezidualnog bloka dodajemo lateralnu vezu koja ugrađuje informacije iz "Brze" staze u "Sporu" stazu. Iza zadnjeg rezidualnog bloka umjesto lateralne veze na svaku stazu dodajemo sloj globalnog sažimanja prosjekom. Dobivena dva vektora iz svake staze konkatenujemo i dovodimo na ulaz potpuno povezanog sloja sa brojem neurona jednakom broju semantičkih razreda skupa podataka sa "softmax" aktivacijskom funkcijom kojeg koristimo za klasifikaciju.

4. Programska izvedba

4.1. Keras

Keras je programski okvir visoke razine za strojno učenje. Odlikuje ga jednostavnost uporabe, modularnost i proširivost. Radi na brojnim platformama poput Tensorflowa, Theana i PlaidMLa. Omogućuje brzo i jednostavno eksperimentiranje sa dubokim modelima. Uz PyTorch je jedan od popularnijih okvira za strojno učenje. Tensorflow 2.0 je usko vezan uz Keras i velika većina Tensorflow modela je pisana upravo u Kerasu. Keras iznosi dva programska sučelja za kreiranje modela.

Programsko sučelje `tf.keras.Sequential` grupira niz slojeva u instancu razreda `tf.keras.Model`. Pogodno je za implementaciju modela koji imaju jedan ulaz i jedan izlaz. U sljedećem isječku vidimo model koji se sastoji od dva potpuno povezana sloja i koji je izgrađen ovim sučeljem.

```
1 model = tf.keras.Sequential
2
3 model.add(tf.keras.layers.Dense(15, input_shape=(10,)))
4 model.add(tf.keras.layers.Dense(8))
```

Model se izrađuje dodavanjem slojeva modelu pomoću `model.add` funkcije. Prilikom kreiranja prvog sloja u modelu, potrebno je predati i `input_shape` argument. Pri dodavanju ostalih slojeva nije potrebno predavati dimenzije podataka već ih Keras sam računa.

Funkcijsko programsko sučelje je fleksibilnije od `tf.keras.Sequential` sučelja. Za razliku od navedenog sučelja, dopušta kreiranje modela sa više ulaza i izlaza. Potrebno je definirati ulazni sloj za svaki ulaz sa potrebnim dimenzijama. U sljedećem isječku koda vidimo model koji je izgrađen sa dfunkcijskim sučeljem.

```
1 inp_1 = tf.keras.layers.Input(shape=(224,224,3))
2 inp_2 = tf.keras.layers.Input(shape=(2,))
3
4 conv_1 = tf.keras.layers.Conv2D(4, kernel_size=[5,5])(inp_1)
```

```

5 d_1 = tf.keras.layers.Dense(3)(inp_2)
6
7 model = tf.keras.Model(inputs=[inp_1, inp_2], outputs=[conv_1, d_1])

```

Svaki Keras model ima metodu `summary()` koja prikazuje veličinu mape značajki nakon svakog sloja i broj parametara, a u modelima izgrađenim sa funkcijskim programskim sučeljem i slojeve s kojim je neki sloj povezan, što su informacije koje su veoma korisne prilikom procesa pronalaženja grešaka u kodu.

4.2. Keras generator podataka

Prilikom implementacije naših modela, mnogo problema smo imali sa pripremom podataka za treniranje pomoću Kerasa. Keras nudi mnogobrojne generatore podataka koji omogućavaju jednostavno učitavanje skupova podataka koji sadrže slike, ali ne nudi generatore koji učitavaju video podatke. Zbog toga smo morali sami pristupiti implementaciji takvog generatora. Podatke smo pokušali učitati u model na dva načina.

Prvi način koji smo isprobali je uzorkovati potreban broj okvira iz videa prije samog treniranja. Ovaj je pristup dobar jer ubrzava učenje - nije potrebno trošiti vrijeme na uzorkovanje okvira tijekom treniranja, te omogućava pohranjivanje ekstrahiranih značajki iz video okvira pomoću kraljezničkog modela treniranog na drugom problemu. Mana ovog pristupa je što generira veliki broj datoteka na disku, što nam je uz činjenicu da smo na Google Disku imali ograničen prostor za pohranu, predstavljalo velik problem.

Drugi način koji smo isprobali je uzorkovati okvire iz videa tijekom samog treniranja. Prije samog treniranja smo učitali liste sa putanjama do video podataka i napravili smo train/valid/test podjele na temelju tih podataka. Prilikom dohvata podatka tijekom treniranja, učitali smo traženi video i iz njega pomoću `cv2` paketa uzorkovali željeni broj okvira iz videa. Augmentaciju podataka smo obavljali tako da smo na svaki niz okvira primjenili slučajnu transformaciju. Iako je ovaj način dohvata podataka sporiji od prvog načina, pokazalo se da je za naše eksperimente bolji.

4.3. Modeli sa sažimajućim značajkama

Razvili smo arhitekturu sa sažimajućim značajkama pomoću `tf.keras.Sequential` sučelja. Kako bismo primjenili isti kralježnički model za ekstrakciju iskoristili smo `tf.keras.layers.TimeDistributed` sloj. Taj sloj će primijeniti sloj koji omeđuje na svaki vremenski okvir. Ako želimo iskoristiti potpuno povezani sloj sa 10 neurona na svaki okvir iz videa, pomoću `tf.keras.layers.TimeDistributed` sloja ćemo to napraviti na sljedeći način: `TimeDistributed(Dense(10))`. Bitno je primjetiti da su težine između potpuno povezanih slojeva dijeljene, odnosno da se na svaki okvir videa primjenjuje isti potpuno povezani sloj. U okviru rada smo implementirali Conv Pooling arhitekturu iz rada [13], gdje se operacija sažimanja provodi nakon zadnjeg konvolucijskog sloja kroz okvire videa. U sljedećem isječku je prikazana implementacija te arhitekture.

```
1 base_model = BaseModel(weights='imagenet', input_shape=(224,224,3),
   include_top=False)
2
3 model = Sequential()
4 model.add(TimeDistributed(base_model, input_shape=(frames,224,224,3))
   )
5 model.add(TimeDistributed(Flatten()))
6 model.add(GlobalMaxPooling1D())
7 model.add(Dense(101, kernel_regularizer=tensorflow.keras.
   regularizers.l2(0.1)))
8 model.add(Dense(classes, kernel_regularizer=tensorflow.keras.
   regularizers.l2(0.1), activation='softmax'))
```

Iz kralježničkog modela smo uzeli sve slojeve do zadnjeg konvolucijskog sloja uključivo, pri čemu je model prethodno treniran na Imagenet skupu podataka. Tijekom treniranja naših modela, dopustili smo pretreniranim težinama iz kralježničkih modela da se dodatno prilagode problemu koji rješavamo. Izvodili smo eksperimente i tako da smo fiksirali predtrenirane težine, ali smo na taj način ostvarivali nešto lošiji rezultat.

4.4. SlowFast model

Ovu arhitekturu smo implementirali pomoću Kerasovog funkcijskog sučelja. Prvi sloj u modelu je sloj kojim uzorkujemo okvire koji su dovedeni na ulaz kako bismo dobili "Sporu" stazu. U sljedećem isječku koda vidimo funkciju koju koristimo za uzorkovanje okvira. Funkcija `prepare_input` vraća Kerasov `lambda` sloj koji obavlja željenu operaciju. Kao argument prima faktor α koji je objašnjen u poglavlju 3.3.

```
1 def prepare_input(alpha):
2     def func(x):
3         b = tensorflow.identity(x[:, ::alpha])
4         return b
5     return Lambda(func)
```

Nakon ovog sloja, efektivno imamo dvije staze za obradu podataka. Staze obrađujemo zasebnim slojevima uz iznimku slojeva koje koristimo za spajanje podataka iz jedne staze u drugu. U našim eksperimentima spajamo informacije iz brze staze u sporu. Spajanje provodimo uz pomoć konvolucijskog sloja koji radi sa 3D podacima koji na ulaz prima mapu značajki Brze staze. Taj sloj ima $2\beta C$ jezgri, svaka jezgra je dimenzija $5 \times 1 \times 1$ uz pomak α kao što je objašnjeno u poglavlju [?]. Izlaz iz tog konvolucijskog sloja konkatenujemo na mapu značajki Spore staze. Primjerice, ako je mapa značajki Spore staze dimenzija $4 \times 56 \times 56 \times 256$, tada je dimenzija mape značajki Brze staze $32 \times 56 \times 56 \times 32$. Izlaz konvolucijskog sloja koji koristimo za spajanje je dimenzija $4 \times 56 \times 56 \times 64$, te je izlaz samog sloja za spajanje nakon konkatencije $4 \times 56 \times 56 \times 320$.

Pošto koristimo ResNet18 kao kralježničku arhitekturu, navedeno spajanje obavljamo nakon svakog ResNet gradivnog bloka. Izuzetak je zadnji ResNet gradivni blok, kad spajanje obavljamo nešto drugačije. Mapu značajki sa svake staze dovodimo na globalni sloj sažimanja prosjekom, što vidimo na sljedećem isječku koda.

```
1 glob_av_s = GlobalAveragePooling3D()(stem5_slow)
2 glob_av_f = GlobalAveragePooling3D()(stem5_fast)
3
4 conc = Concatenate(axis=-1)([glob_av_s, glob_av_f])
5 do = Dropout(dropout)(conc)
6 fc = Dense(_CLASSES, use_bias=True, activation="softmax")(do)
```

Mapa značajki dobivena nakon zadnjeg rezidualnog bloka za "Sporu" stazu je u liniji 1 prethodnog isječka dovedena na ulaz od sloja globalnog sažimanja prosjekom, dok je u liniji 2 mapa značajki dovedena na ulaz sloja globalnog sažimanja prosjekom za "Brzu" stazu. Izlaze tih slojeva potom konkatenujemo na liniji 4, obavljamo dropout na

liniji 5 i dovodimo na potpuno povezani sloj sa brojem neurona jednakom broju razreda skupa podataka na liniji 6. Koristimo softmaks aktivacijsku funkciju kao aktivacijsku funkciju zadnjeg potpuno povezanog sloja. Izlaz modela shvaćamo kao vjerojatnost pripadnosti ulaznog podatka pojedinom razredu.

Nakon svakog konvolucijskog sloja koje koristimo u ovoj arhitekturi, dolazi sloj normalizacije po grupama. Za aktivacije konvolucijskih slojeva koristimo aktivacijsku funkciju zglobnicu.

5. Skupovi podataka za učenje

Danas postoji mnogo standardnih skupova podataka za koji se koriste za evaluaciju modela za klasifikaciju videa. U najpoznatije skupove ubrajamo UCF-101, Kinetics-400, Kinetics-600, Charades, YouTube-8M. Neki od skupova su doista impresivni, YouTube-8M skup sadrži preko 8 milijuna videa ukupnog trajanja preko 500 000 sati, pri čemu je svaki video iz skupa označen s jednim od 4803 razreda.

5.1. UCF-101

Skup sadrži kratke video isječke koji predstavljaju razne ljudske aktivnosti [20]. Isječci su označeni sa ukupno 101 razredom pri čemu je svaki isječak označen samo jednim razredom. Aktivnosti koje pronalazimo u ovom skupu su doista raznovrsne, od raznih sportova poput raftinga, do radnji kao što su šišanje kose i nanošenje šminke. Skup ukupno sadrži 13 320 snimaka ukupnog trajanja preko 27 sati, pri čemu se 9537 videa nalazi u skupu za treniranje te 3783 videa u testnom skupu.

Glavne prednosti koje ovaj skup ima nad prethodnim skupovima poput skupova UCF-50 i HMDB51 su povećanje broja semantičkih razreda u skupu i povećanje varijacije među razredima korištenjem raznovrsnijih ljudskih aktivnosti [20]. Najveća mana skupa je što je iz jednog videa gdje se primjerice pokazuje osoba koja četka kosu, izvađeno i do 7 video isječaka. Skup je organiziran tako da su isječci unutar jednog razreda podijeljeni u 25 grupa. Unutar svake grupe se nalazi 4-7 isječaka koji imaju slične značajke poput pozadine i subjekta koji vrše aktivnost. Zbog činjenice da je većina aktivnosti unutar jedne grupe snimano istom kamerom sa sličnim ambijentalnim uvjetima, imamo relativno malenu varijaciju među podacima jednog razreda [11].

Danas se pojavljuje sve manje modela koji prikazuju rezultate na ovom skupu jer postoje noviji skupovi koji nemaju probleme skupa UCF-101¹, ali zbog računalnih

¹Između ostalih, Kinetics-400 smatramo standardnim skupom za evaluaciju modela za klasifikaciju videa

ograničenja će ovaj skup dobro poslužiti za ispitivanje naših modela.



Slika 5.1: Prikaz svih razreda skupa UCF-101. Prikaz je preuzet iz [20].

6. Eksperimenti

U okviru ovog poglavlja opisujemo eksperimente koje smo provodili u sklopu ovog rada, te koje smo rezultate ostvarili. Provodili smo klasifikaciju videa iz UCF-101 skupa podataka pomoću dvije različite arhitekture: arhitektura sa sažimajućim značajkama i SlowFast arhitektura.

6.1. Metrike

Metrike predstavljaju objektivne mjere za evaluaciju performansi naših modela. Najpopularnija metrika koja se koristi za evaluaciju modela za klasifikaciju je točnost klasifikacije. Točnost klasifikacije računamo na sljedeći način:

$$\text{Točnost} = \frac{\text{Broj točnih predviđanja modela}}{\text{Ukupan broj primjera}} \quad (6.1)$$

Kako bismo mogli koristiti točnost kao metriku, naš skup podataka mora biti balansirani, odnosno moramo imati podjednak broj primjera unutar svakog razreda. Skup koji koristimo u našim eksperimentima je dobro balansirani.

Osim točnosti klasifikacije, u literaturi često susrećemo i top-k točnost. Top-k točnost se razlikuje od obične točnosti po tome što se točnim predviđanjem smatra slučaj kad je točan razred unutar k najvećih vjerojatnosti izlaza modela.

Rezultate koje smo postigli u sklopu naših eksperimenata evaluiramo pomoću metrika top-1 točnosti i top-3 točnosti.

6.2. Povećanje skupa za učenje

Kako radimo sa relativno malim skupom podataka, u našim eksperimentima smo odlučili koristiti povećanje skupa za učenje (engl. *data augmentation*) kako bismo poboljšali rezultate klasifikacije i smanjili mogućnost pretreniranja modela. Za povećanje skupa smo koristili instancu Kerasovog ImageDataGenerator, koju smo kreirali sa že-

ljenim transformacijama. U sljedećem isječku je prikazan proces kreiranja instance ImageDataGeneratora koji koristimo za povećanje skupa za učenje.

```
1 data_aug = tensorflow.keras.preprocessing.image.ImageDataGenerator(  
2     zoom_range=.2,  
3     horizontal_flip=True,  
4     rotation_range=6,  
5     width_shift_range=.15,  
6     height_shift_range=.15)
```

Stvoreni objekt smo predali generatoru podataka. Na kraju svake epohe smo pomoću objekta kreirali nasumičnu transformaciju za svaki video što je vidljivo na sljedećem isječku.

```
1 self.prepared_aug = []  
2 for i in range(self.file_count):  
3     self.prepared_aug.append(self.augmentation.  
4         get_random_transform(self.target_shape))
```

Tijekom treniranja, iz svakog videa smo uzorkovali željeni broj okvira, te smo na svaki okvir iz videa primjenili istu pripremljenu transformaciju.

```
1 f = [self.augmentation.apply_transform(fr, self.prepared_aug[i])  
2     for fr in f]
```

6.3. Rezultati

U ovom poglavlju prikazujemo rezultate naših eksperimenata. Svi eksperimenti su provedeni na Google Colab platformi. Google Colab kod izvodi na virtualnom stroju na kojem nam je dostupno 12 GB radne memorije. Za treniranje nam je dostupna i grafička kartica. Ovisno o tome koji je virtualni stroj dostupan, na raspolaganje možemo dobiti Tesla K80 grafičku karticu i Tesla P100 grafičku karticu. Kako bi mogli trenirati modele neovisno o grafičkoj kartici koju dobijemo na raspolaganje sa istom veličinom mini grupe, koristimo veličinu mini grupe 8 za sve eksperimente.

6.3.1. Modeli sa sažimajućim značajkama

Eksperimentirali smo sa različitim kralježničkim arhitekturama kako bismo pokušali dobiti što bolji rezultat. Originalni rad [13] koristi nešto starije arhitekture, AlexNet i GoogleNet. Izabrali smo novije arhitekture, ResNet18, MobileNet i MobileNetV2 kako bi povećali šansu uspjeha.

Koristili smo modele koji su predtrenirani na Imagenet skupu podataka kako bi ubrzali treniranje. Koristili smo Adam optimizacijski postupak sa stopom učenja 10^{-3} . Model smo trenirali 20 epoha, pri čemu smo koristili rano zaustavljanje kako pohranili model koji najbolje generalizira.

UCF-101 definira tri podjele skupa na skup za treniranje i testiranje koje se koriste za prijavu rezultata. Podjele su napravljene tako da su svi podaci iz skupa nasumično podijeljeni u skup za treniranje i skup za testiranje pri čemu 70% podataka iz skupa ide u skup za treniranje. Postupak je ponovljen tri puta. Rezultati se obično prijavljuju kao prosjek točnosti na testnom skupu iz svake od navedenih podjela. Zbog ograničenog vremena dostupnog za treniranje na Google Colabovim grafičkim karticama, rezultate prijavljujemo na samo prvoj od navedene tri podjele. Svi eksperimenti u ovom radu su provedeni sa 32 okvira uzorkovanih iz videa. Okvire smo uzorkovali drugom metodom koja je opisana u 4.2. Uzorkovali smo pomoću cv2 paketa na način da smo koristili korak uzorkovanja koji je izračunat kao ukupni broj okvira u videu/željeni broj okvira. U literaturi [13] su provedeni eksperimenti i gdje je uzorkovano samo prvih 30 sekundi videa. Tablica 6.1 prikazuje postignute rezultate. Prvi stupac tablice označava koja je kralježnička arhitektura u modelima sažimajućih značajki korištena. Drugi stupac prikazuje ostvarenu top-1 točnost, a treći stupac ostvarenu top-3 točnost. U tablici je vidljivo da najbolju top-1 točnost ostvaruje model koji koristi MobileNetV2 kao kralježničku arhitekturu.

Tablica 6.1: Prikaz ostvarene top-1 i top-3 točnosti na testnom skupu UCF-101 skupa korištenjem različitih kralježničkih arhitektura u Conv Pooling arhitekturi sažimajućih značajki.

backbone	top-1 točnost	top-3 točnost
ResNet18	66.65%	81.60%
MobileNet	70.54%	86.63%
MobileNetV2	71.73%	83.75%

Proveli smo i eksperiment sa 10 okvira uzorkovanih iz videa na MobileNetV2 arhitekturi i postigli smo top-1 točnost od 68.76% i top-3 točnost od 80.85%, što je slabiji rezultat od modela koji na ulazu prima 32 okvira. Ovaj rezultat je očekivan jer model koji na ulazu prima 32 okvira vidi više konteksta iz jednog videa što je prednost za klasifikaciju.

Rezultati ostvareni modelima sa sažimajućim značajkama su blizu rezultata koji su

postignuti u originalnom radu. U originalnom radu je na istom skupu sa Conv Pooling arhitekturom postignut rezultat od 80.8% uz prijavljenu top-1 točnost na test skupu koja je dobivena trostrukom unakrsnom validacijom. Mi najbolji rezultat ostvaruje modelom koji koristi MobileNetV2 kao backbone, što smo i očekivali s obzirom na to da je arhitektura već nadmašila rezultat MobileNet arhitekture na problemu klasifikacije slika [16]. Pretpostavljamo da bi daljnjim podešavanjem hiperparametara uspjeli ostvariti bolji rezultat. Pri evaluaciji bi trebalo koristiti sve tri predložene podjele kao što je napravljeno u radu, ako je to moguće. U originalnom radu je prijavljena točnost od 80.8% sa kralježničkim arhitekturama sa većim brojem parametara. Također, prijavljen je rezultat od 87.6% uz korištenje optičkog toka. U sklopu budućih eksperimenata bi svakako bilo zanimljivo ispitati i kako bi se naši modeli ponašali uz korištenje optičkog toka. U tom slučaju bi očekivali poboljšanje rezultata. Modele naučene sa optičkim tokom bi bilo zanimljivo usporediti sa tehnikama koje su predložene u radu [18] koje nadmašuju rezultate modela sa optičkim tokom na problemu semantičke segmentacije okvira u budućnosti. Kako bi mogli trenirati modele sa većim mini-grupama u budućim eksperimentima bi bilo zanimljivo isprobati tehniku pohranjivanja gradijenata koja bi omogućila našim modelima niže memorijsko zauzeće. Takve bi modele mogli trenirati sa većim veličinama mini grupe što bi vjerojatno dovelo i do stabilnijeg učenja.

6.3.2. SlowFast model

Testirali smo performanse SlowFast modela na testnom skupu UCF-101 skupa. Implementirali smo ResNet18 verziju prilagođenu radu s 3D podacima. Za razliku od prethodnog modela, ovaj model nismo predtrenirali na Imagenet skupu podataka kao što je i predloženo u originalnom radu [5]. Koristili smo Adam optimizacijski postupak sa stopom učenja 10^{-4} . Model smo trenirali 40 epoha, pri čemu smo kao i kod prethodnog modela koristili rano zaustavljanje kako bi pohranili model koji najbolje generalizira. Ostvarili smo od 42.95% top-1 točnost i 61.77% top-3 točnost na testnom skupu.

SlowFast arhitektura je ostvarila lošiji rezultat što nismo očekivali, te pretpostavljamo da je moguće da je rezultat nastao uslijed lošeg izbora hiperparametara ili zbog previđene greške u implementaciji. Trebali bi nastaviti eksperimentirati sa hiperparametrima kako bi poboljšali rezultat.

6.3.3. Slike rezultata

U nastavku dajemo nekoliko uzoraka iz testnog skupa sa pripadajućim predikcijama modela s kojim smo ostvarili najbolji rezultat, MobileNetV2.

Za video podatke prikazane slikama 6.1a i 6.1b vidimo da model sa velikom sigurnošću tvrdi da je na videim prikazana radnja sviranja klavira i radnja penjanja po umjetnoj stijeni. Radnje su specifične i nisu slične ostalim radnjama u skupu pa model nema većih problema sa prepoznavanjem točnih radnji. Za video prikazan slikom 6.1c vidimo da model točno predviđa da je na videu prikazana radnja mješanja hrane ali sa manjom sigurnošću. Razlog tomu je da model ima problema sa razlikovanjem radnji koje se događaju pod sličnim uvjetima. Za video prikazan slikom 6.2b model točno tvrdi da je na videu prikazano udaranje čekićem. Ovaj rezultat je zanimljiv jer čekić nije lako uočiti i jer video sadrži dvije osobe što nije specifično za samu radnju.



(a) Model je predvidio točan razred "Playing Piano", izlaz modela je 0.9938 za navedeni razred.



(b) Model je predvidio točan razred "RockClimbingIndoor" sa izlazom modela 0.9987.



(c) Model je predvidio točan razred "Mixing" sa izlazom modela 0.4043. Drugi najveći izlaz je za razred "BlowingCandles" sa vrijednosti 0.375.



(d) Model je predvidio točan razred "Hammering" sa izlazom modela 0.3520.

Slika 6.1: Prikaz nekoliko podataka iz testnog skupa na kojima model ostvaruje dobre rezultate.

Na slici koja prikazuje video prikaz ljuljanja na ljuljački 6.2a vidimo da je model pogrešno procijenio da je na videu prikazana šetnja psa. Na videu se stvarno nalazi pas i osoba što je specifično za radnju šetnje psa, dok se osoba na ljuljački jako brzo pomiče što otežava prepoznavanje točne radnje. Za video prikazan slikom 6.2b model pogrešno tvrdi da je na videu prikazano šišanje. Drugi izlaz modela je sušenje kose. Model smatra da je okolina u kojem se odvija radnja udaranja čekićem specifičnija za šišanje ili sušenje kose. Model ima problema sa raspoznavanjem između kriket-kuglanja i kriketa kao što uočavamo na slici 6.2c. Analizom podataka iz skupa smo ustanovili da su radnje obično snimane u sličnim okolnostima i ne razlikuju se znatno.



(a) Model je pogrešno predvidio "WalkingWithDog" sa izlazom modela 0.5115, stvarni razred je "Swing".



(b) Model je predvidio razred "Haircut" sa izlazom modela 0.3753. Drugi izlaz sa najveći izlaz je za razred "BlowDryHair" sa vrijednosti 0.2365. Točan razred je "Hammering".



(c) Model je predvidio razred "CricketBowling" sa izlazom modela 0.9991 dok je točan razred "CricketShot".

Slika 6.2: Prikaz nekoliko podataka iz testnog skupa na kojima model ne ostvaruje dobre rezultate.

7. Zaključak

Računalni vid obiluje nerješnim problemima kojima se sve češće pristupa korištenjem metoda dubokog učenja. Zahvaljujući rapidnom razvoju hardvera i tehnika koje se koriste u dubokom učenju, takve metode pokazuju sve bolje rezultate. U ponekim problemima takvi modeli čak i nadmašuju performanse čovjeka. Na Imagenet problemu je ljudska top-5 pogreška klasifikacije bila 5.1%, dok su već modeli iz 2017. ostvarivali top-5 pogrešku klasifikacije od 3.57% [4].

U sklopu ovog rada smo se upoznali sa kovolucijskim modelima za klasifikaciju videa. Upoznali smo koji skupovi postoje za klasifikaciju videa i pobliže smo se upoznali sa skupom UCF-101 na kojem smo provodili eksperimente. Naučili smo koristiti Keras programski okvir za strojno učenje. Napravili smo sustav za klasifikaciju videa pomoću navedenog programskog okvira. Izdvojili smo dvije konkretne arhitekture s kojima smo se detaljno upoznali i koje smo u konačnici implementirali. Objasnili smo ključne pojmove koji se kriju iza učenja takvih modela. Objasnili smo kako rade i kako se konstruiraju konvolucijski modeli. Ukratko smo opisali kralježničke arhitekture koje smo koristili u sklopu naših eksperimenata. Predstavili smo dvije mogućnosti kako pristupiti videu kao podatku za učenje, pri čemu smo opisali svoja iskustva sa navedenim mogućnostima i dali preporuke za korištenje ovisno o računalnim resursima dostupnim prilikom učenja.

Sa modelima sa sažimajućim značajkama smo ostvarili maksimalni rezultat od 71.73% sa MobileNetV2 kralježničkom arhitekturom. Sa ResNet18 i MobileNet arhitekturama smo ostvarili nešto lošiji rezultat, što smo i očekivali jer MobileNetV2 i na problemima poput klasifikacije slika ostvaruje bolji rezultat. Sa SlowFast arhitekturom smo postigli rezultat od 42.95% točnosti klasifikacije na testnom skupu. Očekivali smo da će rezultat SlowFast arhitekture nadmašiti rezultate ostvarene sa modelima sa sažimajućim značajkama. Smatramo da je rezultat lošiji od rezultata ostvarenog sa modelima sa sažimajućim značajkama zbog greške u implementaciji ili zbog lošeg izbora hiperparametara. Proveli smo eksperiment sa 10 okvira uzorkovanih iz jednog videa i uvjerali se da modeli ostvaruju lošiji rezultat u odnosu na ostale modele koji koriste 32

okvira iz videa. Rezultat je očekivan jer modeli koji na ulazu primaju manje okvira iz videa vide potencijalno manje konteksta videa.

U budućim eksperimentima bi svakako bilo poželjno ispitati kako bi se ponašali modeli uz uvođenje optičkog toka u modele. U okviru modela sa sažimajućim značajkama bi se mogli provesti eksperimenti sa ostalim načinima sažimanja. Zanimljivo bi bilo vidjeti i kako bi se modeli ponašali na većim skupovima poput skupa Kinetics-400.

LITERATURA

- [1] Module: tf.keras | TensorFlow Core v2.2.0, May 2020. URL https://www.tensorflow.org/api_docs/python/tf/keras. [Online; accessed 1. Jun. 2020].
- [2] PyTorch, Jun 2020. URL <https://pytorch.org>. [Online; accessed 1. Jun. 2020].
- [3] Papers with Code - Image Classification, Jun 2020. URL <https://paperswithcode.com/task/image-classification>. [Online; accessed 3. Jun. 2020].
- [4] Samuel F. Dodge i Lina J. Karam. A study and comparison of human and deep learning recognition performance under visual distortions. *CoRR*, abs/1705.02498, 2017. URL <http://arxiv.org/abs/1705.02498>.
- [5] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, i Kaiming He. Slowfast networks for video recognition. *CoRR*, abs/1812.03982, 2018. URL <http://arxiv.org/abs/1812.03982>.
- [6] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL <http://arxiv.org/abs/1512.03385>.
- [8] Kaiming He, Ross B. Girshick, i Piotr Dollár. Rethinking imagenet pre-training. *CoRR*, abs/1811.08883, 2018. URL <http://arxiv.org/abs/1811.08883>.
- [9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, i Hartwig Adam. Mobilenets: Ef-

- ficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL <http://arxiv.org/abs/1704.04861>.
- [10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, i L. Fei-Fei. Large-scale video classification with convolutional neural networks. U *2014 IEEE Conference on Computer Vision and Pattern Recognition*, stranice 1725–1732, 2014.
- [11] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, i Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. URL <http://arxiv.org/abs/1705.06950>.
- [12] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, i Tom Goldstein. Visualizing the loss landscape of neural nets. U S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, i R. Garnett, urednici, *Advances in Neural Information Processing Systems 31*, stranice 6389–6399. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf>.
- [13] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, i George Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015. URL <http://arxiv.org/abs/1503.08909>.
- [14] Maxime Oquab, Leon Bottou, Ivan Laptev, i Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. U *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [15] Marin Orsic, Ivan Kreso, Petra Bevandic, i Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. U *CVPR*, 2019.
- [16] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, i Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL <http://arxiv.org/abs/1801.04381>.

- [17] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, i Aleksander Madry. How does batch normalization help optimization? U S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, i R. Garnett, urednici, *Advances in Neural Information Processing Systems 31*, stranice 2483–2493. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf>.
- [18] Josip Saric, Marin Orsic, Tonci Antunovic, Sacha Vrazic, i Sinisa Segvic. Warp to the future: Joint forecasting of features and feature motion. U *CVPR*, June 2020.
- [19] Sinisa Segvic. Duboko učenje, Jun 2020. URL <http://www.zemris.fer.hr/~ssegvic/du>. [Online; accessed 3. Jun. 2020].
- [20] Khurram Soomro, Amir Roshan Zamir, i Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. URL <http://arxiv.org/abs/1212.0402>.

Konvolucijski modeli za klasifikaciju videa

Sažetak

Klasifikacija videa neriješen je problem računalnog videa s mnogim zanimljivim primjenama. U posljednje vrijeme najbolji rezultati u tom području postižu se pristupima temeljenim na dubokim konvolucijskim modelima. Za ovaj rad posebno su zanimljivi nadzirani pristupi gdje je svaki video označen razredom radnje koju video predstavlja. U okviru rada proučio sam postojeće pristupe klasifikaciji videa. Objasnio sam teorijsku podlogu bitnu za razumijevanje eksperimenata koji se provode. Izrađena implementacija modela temelji se na radnom okviru Keras. Izgradio sam dvije konkretne arhitekture za klasifikaciju videa. Opisao sam i arhitekture na kojima se arhitekture od interesa baziraju. Prikazao sam i ocijenjenio ostvarene rezultate i predložio pravce za budući razvoj. **Ključne riječi:** duboko učenje, klasifikacija videa, konvolucijski modeli za klasifikaciju videa, Keras, UCF-101, SlowFast, sažimanje značajki

Convolutional models for video classification

Abstract

Video classification is an unsolved problem of computer vision with many interesting applications. Lately, best results in that area are obtained using approaches based on deep convolutional models. This thesis considers supervised approaches where every video is labeled with class of action the video represents. I examined existing approaches to video classification which were interesting for this paper. I explained the theory necessary for understanding of the experiments. Model implementation is based on Keras framework. I have built two distinct architectures for video classification. Additionally, I have explained the architectures on which the architectures of interest are based. I reported the results and compared them with results from literature and listed possible improvements and directions for future work.

Keywords: deep learning, video classification, convolutional models for video classification, Keras, UCF-101, SlowFast, feature-pooling