Sadržaj

Uvod

Kapljice kiše na automobilskom staklu ozbiljno ometaju vidljivost tijekom vožnje. U vrijeme velikog broja automobila na cestama, nužnost točnih i pouzdanih senzora sasvim je očita. Posebno, nepogodni vremenski uvjeti mogu ograničiti sigurnost na cestama te su stoga bitni brzi i precizni senzori koji se aktiviraju na određene događaje. Iz tog razloga ovaj rad se bavi proučavanjem i nalaženjem učinkovitog i točnog algoritma strojnog učenja za klasifikaciju kapljica kiše na automobilskom staklu uz pomoć kamere. Tako bi automobilski senzor na staklu znao prepoznati pada li kiša ili ne te na temelju toga uključiti brisače i izvršiti druge potrebne akcije.

Iako postoje razne vrste neuronskih mreža, rezidualne neuronske mreže posebno su se istaknule u postizanju izvanrednih rezultata pri rješavanju problema računalnog vida. Iz tog razloga su odabrane kao temeljni modeli proučavanja gore navedenog problema. U ovom radu detaljnije je opisana jedna od njih, mreža ResNet-18. U eksperimentalnom dijelu rada, arhitektura modela ResNet-18 se modificira u cilju postizanja što kvalitetnijeg modela za prepoznavanje kapljica. Kako su kapljice kiše vrlo male, povećanje fokusa na takve sitnije detalje u slici pokušalo se napraviti smanjenjem koraka unutar sloja sažimanja. No, time se gubi semantika prednaučenih težina. Kako bismo ju povratili, u određenim slojevima modela uvodi se dilatacija, tj. dilatirana konvolucija.

Model je treniran i evaluiran na dva skupa podataka. Jedan je manji i stvoren od tima znanstvenika, a drugi značajno veći dobiveni iz stvarnih situacija. Također, u sklopu rada, tehnikom vizualizacije gradijenata ulaza, prikazala se i pokušala objasniti logika zaključivanja modela.

1. Autonomna vozila i detekcija kapljica

Autonomna vožnja najviše je istraživano područje automobilske industrije danas. Cilj je automobile naučiti samostalnoj vožnji u kontroliranim i nekontroliranim uvjetima. Gotovo je nemoguće razviti automobile koji mogu nadmašiti ljudskog vozača bez snažnog i pouzdanog senzorskog sustava, pogotovo kada vožnja odvija u nepovoljnim klimatskim uvjetima poput kiše i snijega. [8]

Senzor kamere postao je temeljni i najvažniji dio sustava senzora koji se koristi u autonomnim vozilima. Putem kamere, automobil vizualno može razumjeti što se oko njega nalazi. Identičnu svrhu imaju i ljudske oči, stoga je pouzdanost kamere ključna za sigurniju i bolju vožnju.

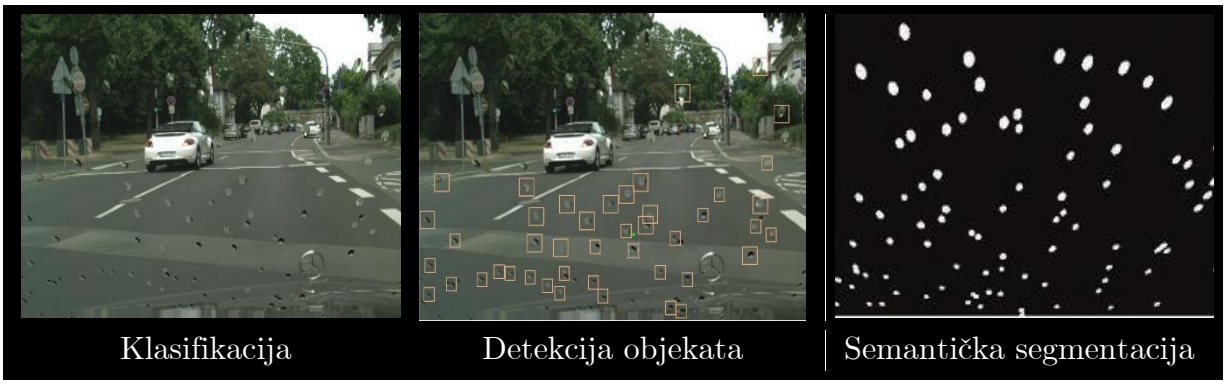
Postojeći sustavi pomoći vozaču rade iznimno dobro u povoljnim vremenskim uvjetima. Međutim, neizbježno je tijekom vožnje susresti se i s nepovoljnim klimatskim uvjetima, gdje se, zagađivači poput kišnih kapi, snježnih pahulja, prljavštine ili zemlje mogu zalijepiti na leću fotoaparata te se time vidljivost okoline poprilično smanjuje. Tako pozadina slike postaje zamućena i dolazi do gubitka informacija ili njihovog pogrešnog tumačenja. Posljedično, to može dovesti do niza krivih odluka koje služe kao pomoć vozaču kao što su detekcija pješaka, detekcija objekata, detekcija prometnih znakova, identificiraju trake, prepoznaju raskrižje i mnoge druge. Kako bi vožnja što kvalitetnija i sigurnija važno je da kamera daje jasne i potpune informacije.

Rješenje koje se nameće kako bi se izbjegli gore navedeni problemi jest osigurati nekompromisnu kvalitetu slike primljene od senzora kamere. To se može postići razvojem softvera za naknadnu obradu slika primljenih iz kamere i rekonstrukciju slika ako su one deformirane.

Kako bi se povratile izgubljene informacije, takve slike smanjene kvalitete najprije se moraju detektirati. Detektiranje kapljica kiše na kameri može se vršiti na

više pristupa. Neki od njih su klasifikacija slike, detekcija objekata i/ili semantička segmentacija. [8]

Ukratko, klasifikacija slike uključuje pridjeljivanje oznake klase ulaznoj slici. Detekcija objekata najčešće uključuje pridjeljivanje oznake klasa objektima na slici uz crtanje graničnog okvira oko jednog ili više detektiranih objekata. Semantička segmentacija je proces dodjeljivanja oznake klase svakom pikselu na slici.

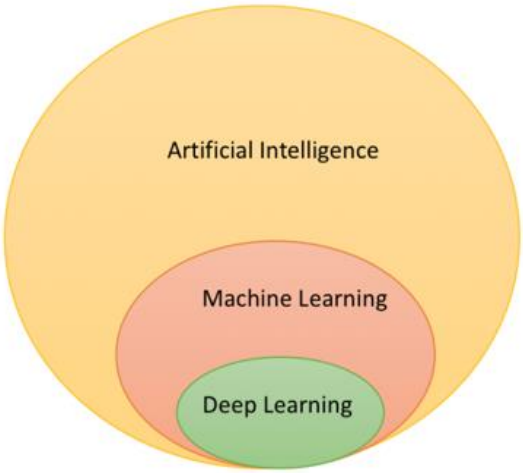


Slika 1.1 Vizualizacija izlaza slike uslijed primjene različitih pristupa detekcija kapljica – klasifikacije, detekcije objekta i semantičke segmentacije (Preuzeto s: https://insights.edag.com/en/rdnet-raindrop-detection-system-autonomous-driving)

U sklopu ovog rada fokus će biti upravo na klasifikaciji slika smanjene kvalitete, odnosno, slike će se klasificirati prema kriteriju jesu li na njoj prisutne kapljice kiše na zaštitnom staklu između stakla i kamere ili nisu. Klasifikacija će se vršiti uz pomoć dubokog modela strojnog učenja.

2. Duboki modeli

Kao što je prikazano na slici 2.1, duboko učenje (engl. *deep learning*) podpodručje je strojnog učenja, (engl. *machine learning*) koje pripada znanstvenoj grani umjetne inteligencije (engl. *artificial inteligence*), koje se bavi proučavanjem algoritmima inspiriranim biološkom strukturom i funkcioniranjem ljudskog mozga kako bi se strojevima omogućilo donošenje zaključaka nalik čovjekovu.



Slika 2.1 Prikaz dijela podgrana umjetne inteligencije (Preuzeto iz prvog poglavlja knjige: J.Moolayil, „Learn Keras for Deep Neural Networks“, 2019)

Razvoj grafičkih kartica (engl. *graphics processing unit – GPU)* uvelike je doprinio razvoju grane dubokog učenja jer je to omogućilo vrćenje do sad nepraktičnih algoritama za izvođenje. Svaki algoritam strojnog učenja, pa tako i dubokog, zahtjeva ove tri komponente koje će u nastavku poglavlja biti detaljno opisane:

1. Model $\big(h(x|\theta)\big)$
2. Optimizacijski postupak
3. Funkciju gubitka (θ)

2.1. Duboki konvolucijski modeli

Konvolucija je matematička operacija nad realnim brojevima kao argumentima funkcije. Operacije konvolucije se označava pomoću zvjezdice, kao u formuli (2.1).

$$s(t) = (x * w)(t) \tag{2.1}$$

Kako bismo dobili uvid što točno obavlja konvolucija, operacija konvolucije je opisana formulom (2.2) u kontinuiranoj domeni. U terminologiji konvolucijskih neuronskih mreža izlaz $s(t)$ naziva se mapom značajki (engl. *feature map*). Izraz $x(a)$ predstavlja ulaz, odnosno ulaznu funkciju, dok w označava jezgru (engl. *kernel*).

$$s(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da \tag{2.2}$$

Ukoliko se mjerenje obavlja svakog trenutka funkcija $s(t)$ postaje glatka. No, kao ulaz u konvolucijsku neuronsku mrežu (engl. *convolutional neural network – CNN*) često se predaje slika te se tada koristi diskretna verzija formule iz razloga što su podatci najčešće diskretizirani te ih ima konačan broj. Prikaz konvolucije u diskretnoj domeni opisan je formulom (2.3).

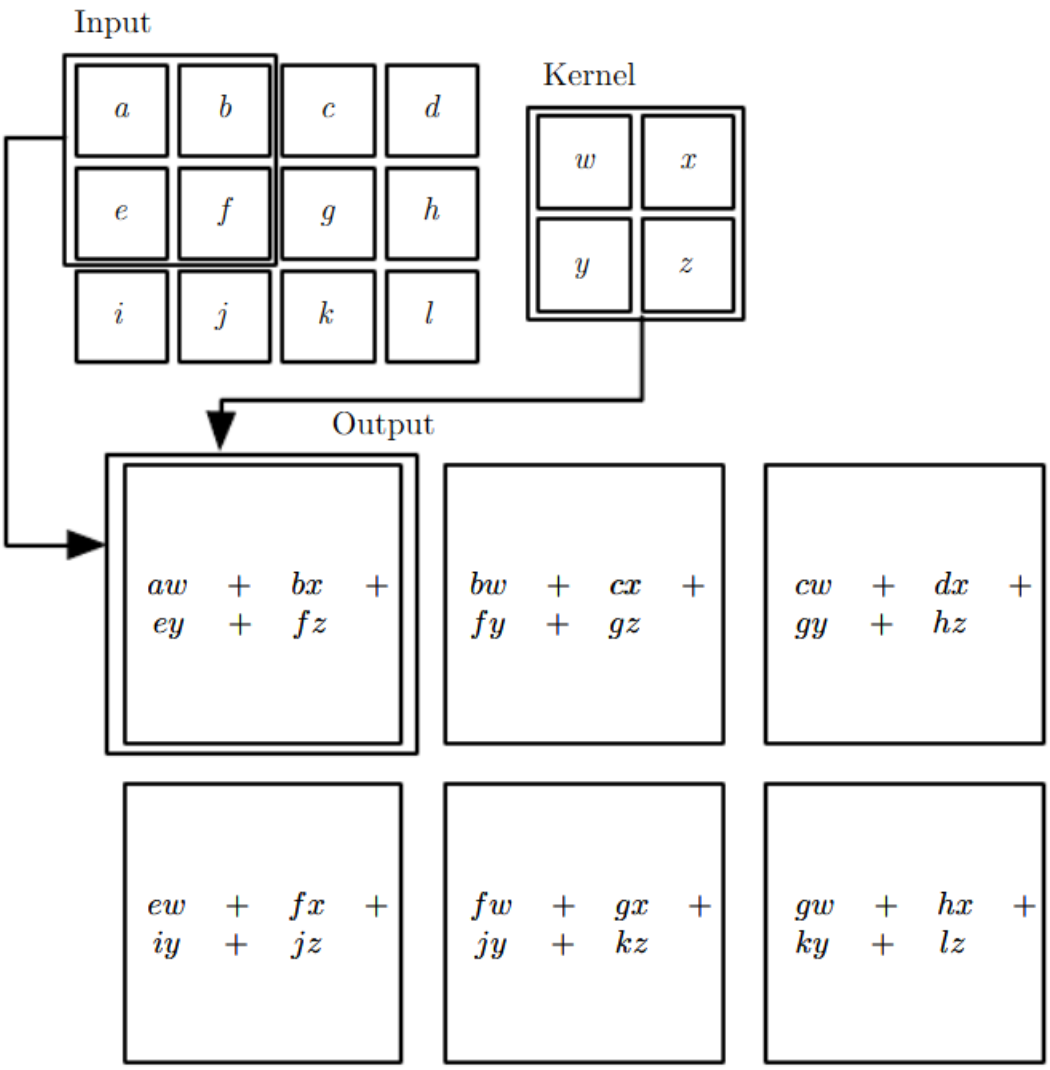
$$s(t) = \sum x(a)w(t-a) \tag{2.3}$$

Operacija konvolucije je komutativna, distributivna, asocijativna i asocijativna sa skalarnim množenjem. Zbog tih svojstava konvolucijske mreže često se koriste u područjima statistike , obrade signala, računalnog vida i drugim.[9]

Kako temeljni problem ovog rada spada u područje računalnog vida (engl. *computer vision*) te su ulazni podatci dvodimenzionalne slike, operacija konvolucije se izračunava po obje dimenzije. U tom slučaju će i jezgra biti dvodimenzionalna kvadratna matrica veličine kxk. Uobičajeno je da za vrijednost k izabere neparan broj. Tako opisana konvolucija nad 2D podatcima dana je formulom (2.4).

$$S(i,j) = (X * W)(i,j) = \sum\sum X(m,n)W(i-m, j-n) \tag{2.4}$$

Slika 2.2 grafički prikazuje operaciju dvodimenzionalne konvolucije čija je jezgra veličine 2. Jezgra preklopi dio ulazne matrice te se nad tim dijelom računa konvolucija. Zatim se jezgra pomakne za određeni korak. Kako je jezgra uvijek manjih dimenzija od dimenzija samog ulaza postavlja se pitanje što učiniti s rubnim vrijednostima slike, tj. onim dijelovima gdje jezgra ne preklala u cijelosti ulaznu matricu.



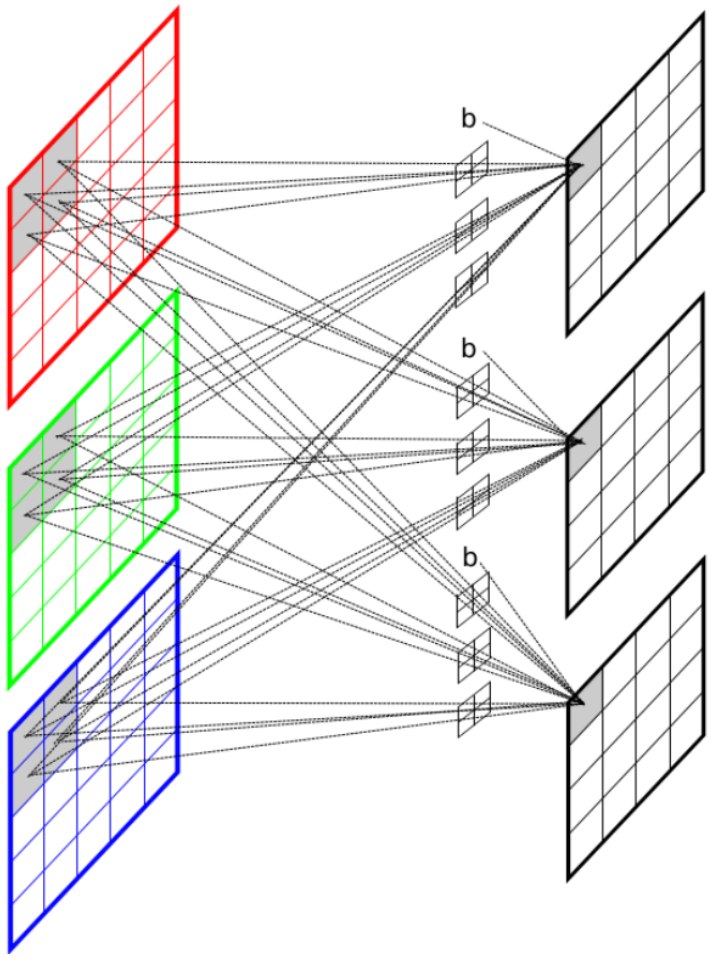
Slika 2.2 Prikaz 2D kovolucije, (Preuzeto s [9])

Postoje dva pristupa ovom problemu. Prvi pristup zove se nadopunjavanje (engl. *padding*) i radi tako da se ulaznu matricu proširi s konstantnim vrijednostima, najčešće nulama, veličine $(k-1)/2$ u svim smjerovima. Drugi pristup je ne vršiti operaciju konvolucije nad dijelovima slike koju jezgra nije u potpunosti preklopila.

Ovim pristupom smanjuje se dimenzija izlazne matrice u odnosu na ulaznu. Ako bismo voljeli izračunati dimenzije izlazne matrice to možemo učiniti pomoću formule (2.5). U njoj oznaka x označava širinu ulazne slike, k širinu jezgre, p širinu nadopunjavanja u jednom smjeru te s označava korak pomaka. Treba imati na umu da ova formula pretpostavlja kvadratnu strukturu ulaza i jezgre. Ukoliko to nije slučaj, odnosno ulaz nije jednake širine po svakoj dimenziji, dimenzija izlazne matrice dobije se tako da se formula primjeni zasebno za svaku dimenziju.

$$output_{shape} = \left[\frac{(w-k+2*p)}{s}\right] + 1 \tag{2.5}$$

Do sada smo objasnili kako radi konvolucija unutar jednog sloja. U praksi rijetko imamo samo jedan sloj konvolucije, najčešće ih ima više. Slika 2.3 ilustrira upravo to. U prvom sloju (lijevi dio slike) nalaze se tri kanala od kojih svaki ima svoju jezgru te, u ovom slučaju, svaki kanal prvog sloja ulazi u kanale idućeg sloja.



Slika 2.3 Prikaz rada konvolucije s više slojeva. (Preuzeto s [9])

Nad svakim kanalom posebno se vrši operacija konvolucije te se rezultante matrice međusobno zbroje i potom upisuju na odgovarajuće mjesto u mapi značajke sljedećeg sloja.

Također, valja napomenuti kako konvolucijskom sloju u dubokim konvolucijskim modelima često slijedi sloj sažimanja. On služi da smanji broj značajki s kojima sljedeći slojevi računaju. Operacija sažimanja analogna je operaciji konvolucije, u njoj također jezgra koja obavlja sažimanje preklopi ulaznu matricu te se nakon toga pomakne na sljedeći segment matrice. Uobičajeno se u sloju sažimanja odabire takav korak da se pritom ne događaju preklapanja ulazne matrice s jezgrom. Najčešće je korak jednak veličini jezgre.

Konvolucijski modeli imaju tri značajna svojstva u odnosu na ostale modele dubokog učenja koja mogu poboljšati sustav učenja. Oni su raspršena povezanost, dijeljenje parametara i ekvivarijantnost s obzirom na pomak.

U klasičnim neuronskim mrežama parametar jednog sloja ovisan je o svim parametrima prethodnog sloja, a to nije slučaj kod konvolucije. Svaki izlaz iz konvolucijskog sloja povezan je samo s onim parametrima koji su obuhvaćeni jezgrom, odnosno svojim susjedstvom. Te vrijednosti nazivaju se još i receptivno polje (engl. *receptive field*). Složenost operacija prešla je s $O(mn)$ na $O(kn)$, gdje je n veličina trenutnog sloja, m veličina prethodnog, a k veličina jezgre. Vrijedi $k \ll m$.

Druga razlika između konvolucijskih i neuronskih mreža je ta da neuronske mreže prilikom prolaska podatka kroz sloj svaku težinu koriste točno jednom. Konvolucijske mreže unutar svakog sloja dijele parametre. Svaka jezgra, kojih može biti više, unutar sloja ima vlastite parametre koji se operacijom konvolucije primjenjuju na svim lokacijama ulaznog podatka.

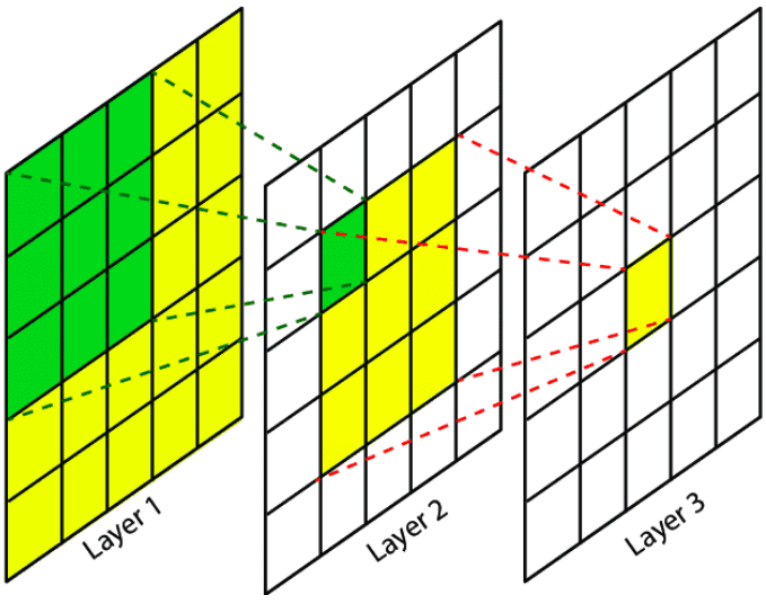
Ekvivarijantnost s obzirom na pomak svojstvo je konvolucijske mreže proizašlo iz postojanja svojstva dijeljenja parametara. Ekvivarijantnost je karakteristika koja obilježava uzročno-posljedičnu vezu između promjene na ulazu i promjene na izlazu. Dakle, ukoliko se dogodi pomak u podatcima koji ulaze u

konvoluciju, posljedično će se takav pomak izraziti i na izlazu. Na primjer, ukoliko želimo detektirati neko određeno svojstvo na slici, npr. rub. I sada ako cijelu sliku translatiramo u nekom smjeru, ponovno ćemo moći detektirati to svojstvo. Jedina razlika bit će u tome što će na izlazu iz konvolucije svojstvo biti na drugom mjestu na slici. U kontekstu klasifikacije, ovo je vrlo poželjno svojstvo jer omogućuje robusnu detekciju željenih svojstava. Također valja napomenuti kako konvolucija nije ekvivarijantna s obzirom na neke druge transformacije ulaza, npr. skaliranje ili rotaciju.

Postojanjem gore opisanih svojstva ubrzalo se vrijeme računanja zbog značajno manje parametara i time se postiže memorijska efikasnost.[9]

2.1.1. Receptivno polje i utjecaj dilatacije

Sve što ljudsko oko može vidjeti naziva se vidno polje. Receptivno polje neurona možemo gledati kao dio ukupnog vidnog polja. Pojam receptivno polje odnosi se na područje prostora koje je zahvaćeno svim prethodnim (dijelovima) slojeva, a ulaze u trenutni sloj. Drugim riječima, kojim informacijama pojedini neuron ima pristup i posljedično na njega mogu utjecati. [10]

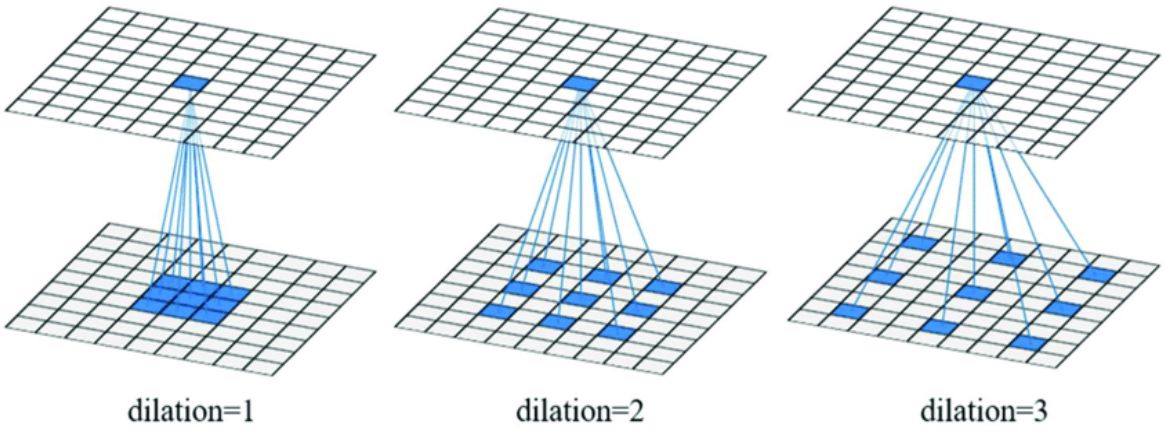


Slika 2.4 Receptivno polje konvolucijskog sloja (Preuzeto s [10])

Na slici 2.3 vidljivo je kako su na žutu regiju trećeg sloja utjecale žute regije prethodnih slojeva. Možemo također uočiti da se sa povećavanjem broja prethodnih slojeva opseg utjecajnih regija povećava. To znači da receptivno polje raste s dubinom slojeva.

Jedna od pretpostavki dubokih modela jest hijerarhijska struktura podataka nad kojima se uči. Tako svaki sljedeći sloj postaje sve specijaliziraniji u svojoj zadaći. Na primjer, ako želimo prepoznati nalazi li se na slici mačka, duboki model prvo bi mogao pronaći uši, zatim brkove pa šape i tako dalje. To ima smisla jer što se sloj nalazi dublje u modelu, to on ima veće receptivno polje. [10]

Već znamo da se receptivno polje povećava se s dubinom modela, no kojom mjerom će se ono širiti kroz svaki sloj ovisi o dilataciji (engl. *dilation*). Dilatacija je tehnika koja proširuje jezgru umetanjem rupa između njenih uzastopnih elemenata. Drugim riječima, dilatacija radi isto što i konvolucija uz preskakanje pojedinih **piksela/značajki** kako bi se pokrilo veće ulazno područje, odnosno receptivno polje. U sklopu konvolucije, dilatacija se smatra **faktorom/parametarom** l. Na temelju vrijednosti parametra dilatacije, $(l-1)$ **piksela/značajki** se preskaču unutar jezgre koja preklapa ulaznu matricu.[11]



Slika 2.5 Utjecaj dilatacije na receptivno polje (Preuzeto s [11])

Slika 2.5 prikazuje razliku između uobičajene konvolucije (dilatacija je postavljena na $l = 1$) i proširene, dilatirane konvolucije. Plavom bojom na donjoj matrici prikazano je područje jezgre koje prekriva ulaz, dok plavo područje na gornjoj matrici označava područje u koje se mapira izlaz (pomoću operacije konvolucije). Možemo vidjeti kako se povećanjem parametra dilatacije proširuje područje koje zahvaća jezgra, a samim time proširuje se receptivno polje. Na primjer, postavljajući $l = 2$, preskačemo svaki $l - 1 = 1$ piksel na ulazu kojeg jezgra zahvaća te time pokrivamo više informacija u svakom koraku.

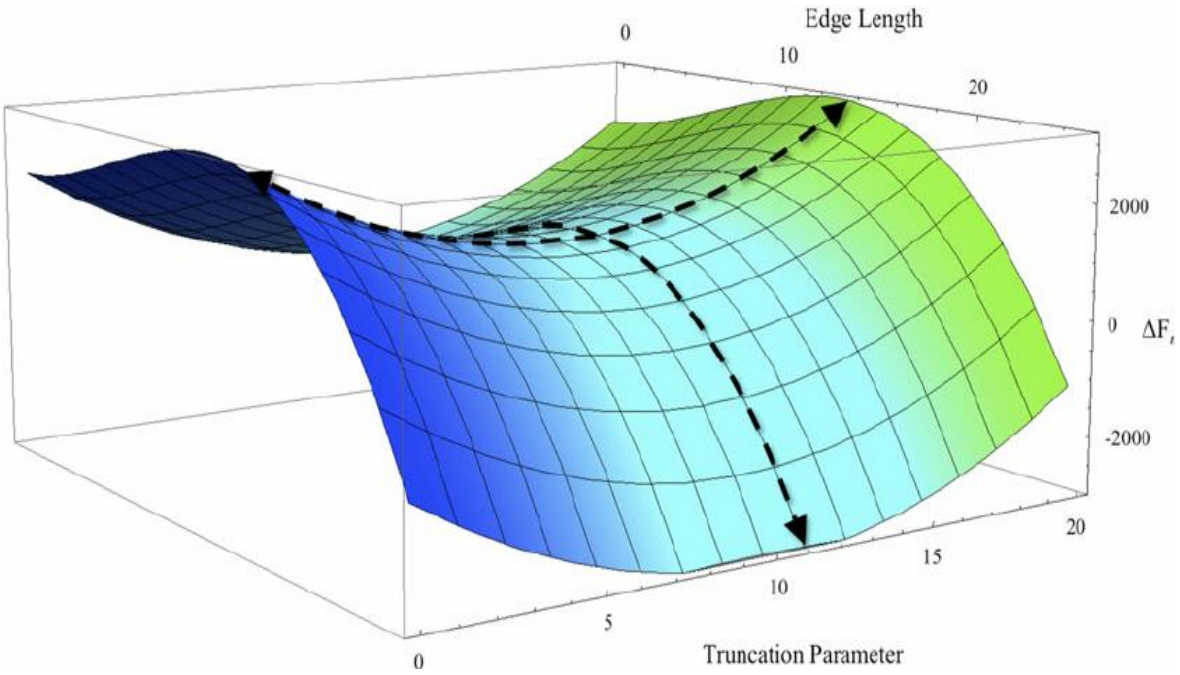
Takva proširena konvolucija pomaže proširiti pokriveno područje ulazne slike bez korištenja sloja sažimanja. Cilj je jezgrom preklopiti **veću površinu**/ **veće područje** kako bi se ostvarilo šire receptivno polje bez dodavanja dodatnih slojeva. Korištenjem ove metode dobiva se više informacija bez povećanja jezgre. [11]

2.2. Optimizacijski postupak

Modeli strojnog učenja uče pomoću optimizacijskih algoritama. Optimizacijski postupak je proces u kojem se iterativno trenira model koji rezultira maksimalnom ili minimalnom evaluacijom funkcije gubitka, ovisno o njenoj implementaciji. Traženje globalnih optimuma funkcije gubitka dovodi do maksimizacije učinkovitosti predviđanja modela. To je jedna od najvažnijih stavki u strojnom učenju za postizanje što boljih rezultata. Kako je proces učenja uobičajeno vremenski zahtjevan, optimizacijski postupak igra veliku ulogu u smanjenju vremena treniranja modela.

Traženje globalnih optimuma funkcije gubitka često zna biti vrlo težak zadatak zbog mogućih upadanja i nemogućnosti vraćanja iz regija lokalnih optimuma ili regija s vrlo malim gradijentima. Regije s malim gradijentima uzorkovane su takozvanim sedlom (engl. *saddle*). Sedlo je točka u kojoj je istovremeno prisutan i maksimum i minimum funkcije, ilustrirana je na slici 2.6. Iz slike možemo vidjeti da

ukoliko bismo iz točke sedla krenuli u jednom smjeru (sjeveroistočno ili jugozapadno iz naše perspektive), okarakterizirali bismo tu točku kao lokalni minimum funkcije. No, ukoliko bi iz te točke krenuli u drugom smjeru (sjeverozapadno ili jugoistočno iz naše perspektive), okarakterizirali bismo tu točku kao lokalni maksimum funkcije. Sedlaste regije specifične su po tome što je iznos gradijenata vrlo mali, blizu nuli u svim smjerovima i to otežava izlazak iz takvih regija.[12]

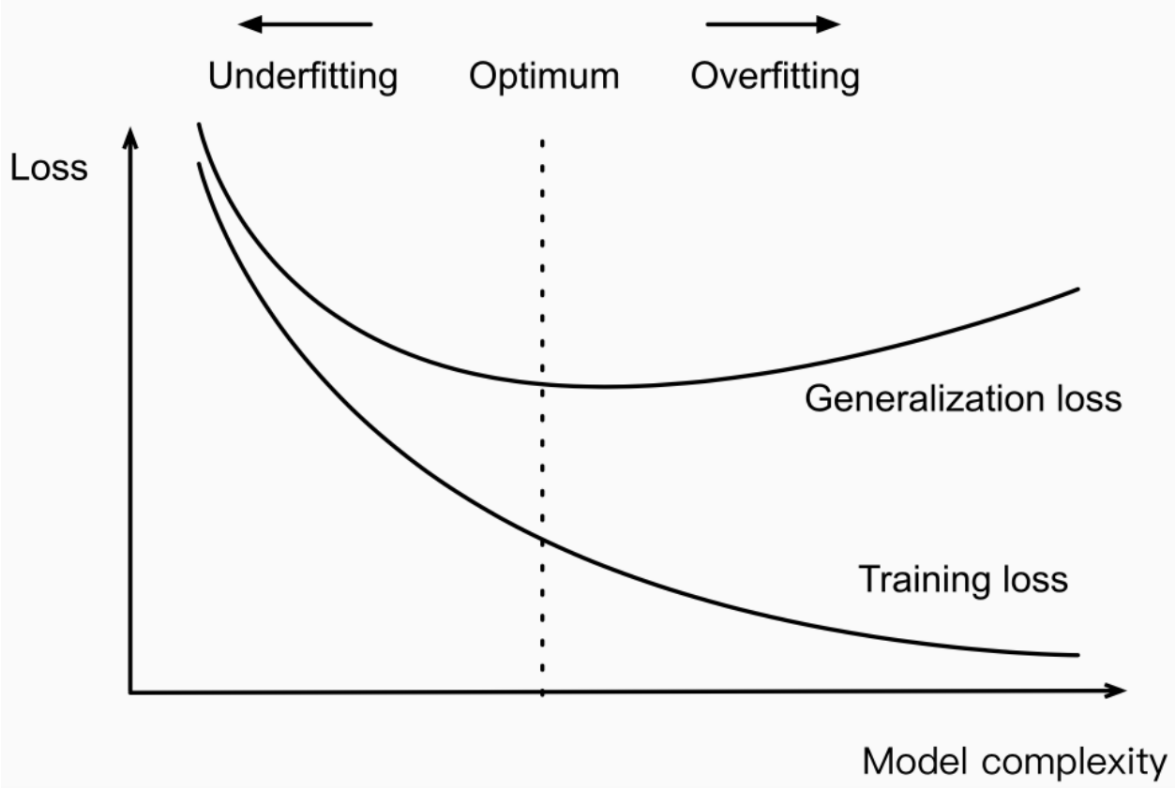


Slika 2.6 Prikaz sedla funkcije (Preuzeto s [12])

Modele optimiziramo tako da u svakoj iteraciji uspoređujemo rezultate modela dok se ne postignu optimalni. Usporedba se vrši tako da se nakon svake epohe učenja izračunava pogreška na skupu za učenje i validaciju. Model tijekom treniranja korigira svoje težine samo nad skupom za treniranje u svrhu smanjenja pogrešne. Usporedba nad ova dva skupa vrši se kako bi se dobila što realnija slika o modelovoj sposobnosti vršenja generaliziranih predikcija.

Slika 2.7 ilustrira takozvanu krivulju učenja koja prikazuje standardni tijek pogreške na skupu za učenje i validaciju u odnosu na broj epoha. Vidimo kako se

pogreška učenja uvijek smanjuje, to je i očekivano jer model korigira svoje težine u odnosu na taj skup i sve se više specijalizira za njega. Iz grafa također vidimo kako validacijska pogreška prvo pada, a zatim počinje rasti. Trenutak kada je najbolje zaustaviti treniranje modela je kada pogreška nad validacijskom skupu bude najmanja, a na grafu je prikazan iscrtanom crnom crtom (engl. *optimum*). To je trenutak kada se pretpostavlja da model najbolje generalizira. Stanje lijevo od tog trenutka zove se podnaučenost (engl. *underfitting*) i karakterizira ga to da model nije dovoljno prilagodio svoje težine ulaznim podatcima. Stanje desno naziva prenaučenost (engl. *overfitting*) i karakterizira to da se model previše specijalizirao za podatke za učenje te je izgubio na svojim generalizacijskim sposobnostima, odnosno radi loše predikcije na neviđenim podatcima.[13]



Slika 2.7 Tijek pogreške na trening i validacijskom skupu tijekom treniranja modela (Preuzeto s [13])

Optimizacijski algoritmi dijele se u dvije glavne kategorije koje se danas široko koriste. One su deterministički i stohastički algoritmi.

1. Deterministički algoritam koristi pravila za premještanje jednog rješenja na drugo. Ta pravila su jasno definirana, deterministička.
2. Stohastički algoritam je algoritam koji eksplicitno koristi slučajnost (vjerojatnost) za pronalaženje optimuma ciljne funkcije. Stohastičnost se odnosi na varijabilni proces u kojem ishod uključuje određenu slučajnost i nesigurnost. Takav algoritam funkcionira na ravnoteži između istraživanja prostora za pretraživanje rješenja i iskorištavanja već naučenog znanja o prostoru nad kojim se pretražuje. Odabir sljedećih lokacija pretraživanja odabire se stohastički, odnosno vjerojatno se temelji na tome koja su područja nedavno pretražena.[12]

2.2.1. Adam optimizator

Adam (engl. Adaptive Moment Estimation) je kratica za estimator adaptivnog momenta. To je stohastički optimizacijski algoritam temeljen na tehnici gradijentnog spusta (engl. *gradient decent*). U sebi ima ugrađen efekt momenta koji je fizikalna ekvivalencija inercije koja je u ovom slučaju primijenjena na gradijent. Ideja algoritma je korak učenja mijenjati u ovisnosti o prošlim podatcima, uzimajući u obzir eksponencijalni težinski prosjek (engl. *exponentially weighted average*) gradijenata. Na taj način, uvažavanjem informacija iz prošlosti, Adam prilagođava brzinu učenja. [14]

U nastavku je dan pseudokod Adam algoritma.

Pseudokod 2 Algoritam Adam

Ulaz: Korak učenja ϵ

Ulaz: Faktori odumiranja za procjene momenata, ρ_1 i ρ_2 iz $[0, 1)$ (preporučene vrijednosti 0.9 i 0.999)

Ulaz: Mala konstanta δ za numeričku stabilizaciju (preporučena vrijednost 10^{-8})

Ulaz: Početni parametri $\boldsymbol{\theta}$

Inicijaliziraj varijable prvog i drugog momenta: $\mathbf{s} = 0, \mathbf{r} = 0$

Inicijaliziraj vremenski korak: $t = 0$

dok uvjet završetka nije zadovoljen **radi**

  Uzorkuj mini grupu veličine m iz skupa za učenje $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ s pripadajućim referentnim vrijednostima $\mathbf{y}^{(i)}$

  Izračunaj gradijent: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$

  $t \leftarrow t + 1$

  Osvježi pristranu procjenu prvog momenta. $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1)\mathbf{g}$

  Osvježi pristranu procjenu drugog momenta: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2)\mathbf{g} \odot \mathbf{g}$

  Popravi pristranost procjene prvog momenta: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1-\rho_1^t}$

  Popravi pristranost procjene drugog momenta: $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1-\rho_2^t}$

  Izračunaj promjenu: $\delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}$

  Osvježi parametre: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$

kraj dok

Način na koji se računa moment je sljedeći: prvo se izračuna gradijent pomoću funkcije gubitka u zadanom trenutku, pa se zatim prvo spremaju eksponencijalno smanjujući prosjeci gradijenata iz prošlih koraka (korišten za procjenu prvog momenta), a zatim i eksponencijalno smanjujući prosjeci kvadratnih gradijenata iz prošlih koraka (korišten za procjenu drugog momenta). Računanje prvog momenta dano je formulom (2.6), a drugog (2.7). Momenti u prvom (nultom) koraku inicijalizirani su na nulu.

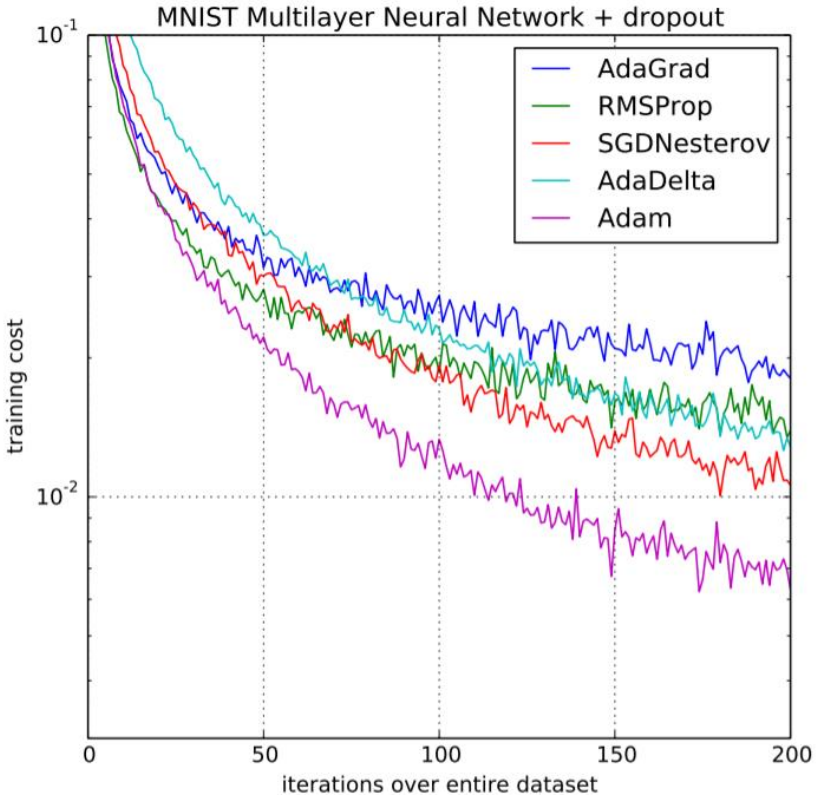
$$s_t = \rho_1 * s_{t-1} + (1 - \rho_1) * g_t \tag{2.6}$$

$$r_t = \rho_2 * r + (1 - \rho_2) * g_t^2 \tag{2.7}$$

Kada su izračunate procjene prvog i drugog momenta računa se njihova pristranost označena simbolima $\widehat{s_t}$ i $\widehat{m_t}$. Krajnji parametar θ ažurira se po formuli (2.8).

$$\theta_{t+1} = \theta_t - \epsilon \frac{\widehat{s_t}}{\sqrt{m_t} + \delta} m_t \tag{2.8}$$

Ovim algoritmom kontrolira se brzina gradijenta spuštanja na takav način da postoji minimalna oscilacija kada se dosegne globalni minimum, a pritom radimo dovoljno velike korake kako bismo prošli lokalne minimume na putu. Dakle, kombinirajući značajke gore navedenih metoda kako bi se učinkovito dostigao globalni minimum.[14]



Slika 2.8 Usporedba performansi Adam optimizatora i drugih. (Preuzeto s [14])

Uspoređujući performanse različitih optimizacijskih postupaka, treniranih nad istim modelom sa poznatim MNIST skupom podataka, Adam optimizator pokazuje najbolje performanse uspoređujući se s drugim algoritmima. Na slici 2.8 jasno je prikazano kako Adam nadmašuje ostatak optimizatora sa značajnom razlikom u smislu troškova obuke (nizak) i izvedbe (visoki).[14]

2.3. Funkcija gubitka

Svaki model strojnog učenja mora imati definiranu funkciju gubitka (engl. *loss function*) koja se koristi prilikom faze učenja i validacije. Preko nje određuje se koliku grešku model proizvodi prilikom učenja. Tijekom učenja, funkcije gubitka koriste se u optimizacijskim algoritmima. Cilj je gotovo uvijek minimizirati ju. U pravilu, što je gubitak manji, to je model bolji. U sklopu ovo rada koristit će se gubitak unakrsne entropije (engl. *cross-entropy loss*) te ćemo ga pobliže objasniti u nastavku.

Entropija H je mjera nesigurnosti izlaza u odnosu na dani ulaz x. Formula prema kojoj se može izračunati entropija za slučajnu varijablu x iz diskretnog skupa stanja X s vjerojatnošću $p(x)$ je prikazana s izrazom (2.9). Unakrsna entropija je mjera razlike između dvije distribucije vjerojatnosti za danu slučajnu varijablu x ili skup događaja, u našem slučaju dani su ulazni podatci, slike. Opisana je formulom (2.10) gdje su P i Q dva diskretna distribucijska skupa, a $P(x)$ označava vjerojatnost varijable x u P, odnosno Q za $Q(x)$.

$$H(X) = -\sum_{x} p(x)\log\left(p(x)\right) \tag{2.9}$$

$$H(P,Q) = -\sum_{x} P(x)\log\left(Q(x)\right) \tag{2.10}$$

Rezultat će biti pozitivan broj i bit će jednak entropiji distribucije ako su dvije distribucije vjerojatnosti identične. Što je mjera nesigurnosti manja, to dana slučajna varijabla sadrži više informacija i obrnuto.

Gubitak unakrsne entropije koristi se u klasifikacijskim modelima. U zadacima klasifikacije, znamo ciljnu distribuciju vjerojatnosti P jer nam je unaprijed poznata vjerojatnost svake oznake klase za svaki primjer u skupu podataka. P predstavlja očekivanu vjerojatnost. S druge strane Q predstavlja očekivanu vjerojatnost, odnosno vjerojatnost svake klase za dati primjer koju je predvidio naš model. Dakle, naš model nastoji aproksimirati ciljnu distribuciju vjerojatnosti Q.

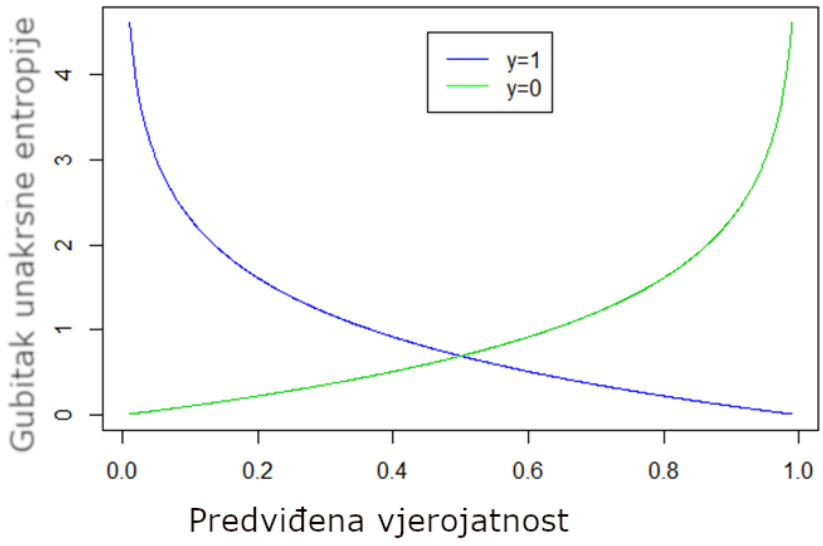
$$L_{CE}(x) = -\sum_{i=1}^{n} P(x)\log_e\big(Q(x)\big), \qquad za\ n\ klasa \tag{2.10}$$

Matematička formula gubitka unakrsne entropije prikazana je izrazom (2.10). Kako bismo ga uklopili u optimizacijski algoritam, potrebno je pogledati petu liniju pseudokoda koja izgleda ovako:

5: **Izračunaj gradijent:** $\mathbf{g} \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(\mathbf{x}^{(i)};\boldsymbol{\theta}),\mathbf{y}^{(i)})$

U Njoj izraz $L(f(x^{(i)};\theta),y^{(i)})$ predstavlja funkciju gubitka. Kako bi se formule poklapale treba samo simbol za ciljnu vjerojatnosnu distribuciju P zamijeniti izrazom $f(x^{(i)};\theta)$, a predviđenu vjerojatnosnu distribuciju Q sa $y^{(i)}$ što predstavlja izlaz modela.

Grafička interpretacije dana je slikom 2.9 gdje je prikazan jedan pozitivan i jedan negativan uzorak. Za pozitivan uzorak (y=1) vjerojatnost predviđanja raste kada se gubitak smanjuje. To je baš ono što i želimo, smanjiti gubitak kako bi se vjerojatnost točnog predviđanja modela povećala.[15]

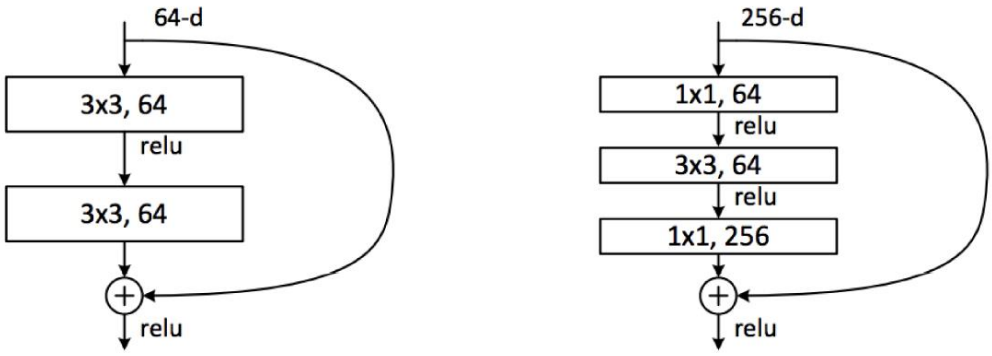


Slika 2.9 Kretanje gubitka unakrsne entropije u ovisnosti o predviđenoj vjerojatnosti pojave događaja (Preuzeto s [15])

3. ResNet-18 arhitektura

ResNet je skraćenica za rezidualnu mrežu (engl. *residual network*) nastala 2015. godine od strane Microsoftovih istraživača. Arhitektura ove mreže imala je za cilj omogućiti učinkovito funkcioniranje velike količine konvolucijskih slojeva. Međutim, dodavanje više dubokih slojeva mreži često rezultira degradacijom izlaza. Ovo je poznato kao problem nestajućeg gradijenta (engl. *vanishing gradient*) gdje se neuronske mreže, dok se treniraju povratnom propagacijom, oslanjaju na propuštanje gradijenta, spuštajući funkciju gubitka kako bi pronašli minimizirajuće težine. Zbog prisutnosti više slojeva, ponovno množenje kroz slojeve rezultira sve manjim i manjim gradijentom čime on "nestaje" što dovodi do zasićenja performansi mreže ili čak smanjenja njene performanse.[2]

Rezidualne neuronske mreže karakterizira posjedovanje rezidualnih jedinica. One koriste skakačke veze koje se uglavnom nazivaju prečacima ili identitetskim vezama (engl. *skips*). Skakačke veze prvenstveno funkcioniraju skačući preko jednog ili više slojeva tvoreći prečace između tih slojeva. Ulaz u rezidualnu jedinicu se grana u dvije grane te se jedna od njih provlači dalje kroz mrežu, druga ostaje jednaka ulazu u spomenutu rezidualnu jedinicu. Grane se ponovno spoje tako da se izlazi obiju grana zbroje. Kod zbrajanja grana treba obratiti pažnju da su dimenzije izlaza grana međusobno kompatibilne.

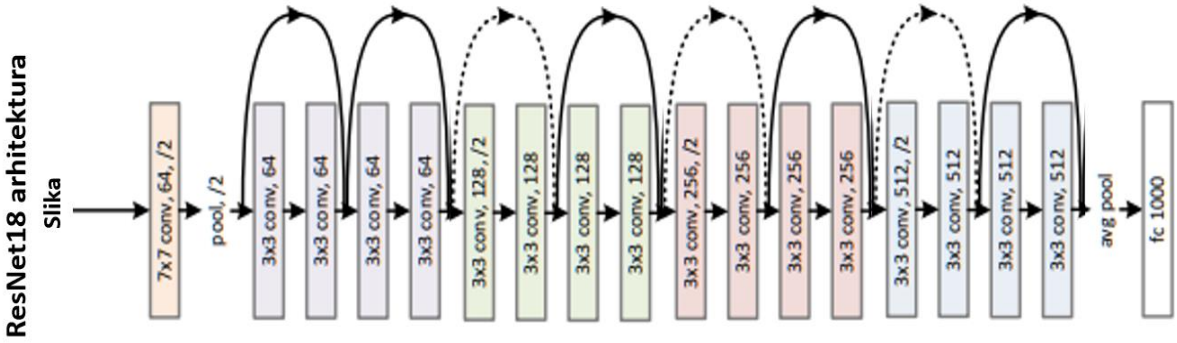


Slika 3.1 Prikaz rezidualnih jedinica u ResNet-18 arhitekturi (Preuzeto s [2])

Dva primjera rezidualnih jedinica u ResNet-18 arhitekturi ilustrirana su na slici 3.1. Lijevo je prikazana osnovna rezidualna jedinica, a desno je prikazana rezidualna jedinica s uskim grlom. Ovako prikazane preskočne veze naziva se i „prečac vezom identiteta“ (engl. *identity shortcut connection*). Zovu se tako jer je grana koja je preskočila određen broj slojeva ostala jednaka ulazu, odnosno nad njom je primijenjena samo funkcija identiteta.

Cilj uvođenja prečica bio je riješiti prevladavajući problem nestajućeg gradijenta s kojim se suočavaju duboke mreže. Prečice uklanjaju problem nestajanja gradijenta tako što ponovno koriste aktivacije dobivene iz prethodnog sloja. Mapiranje identiteta u drugoj grani ne čini ništa osim preskakanja određenih slojeva, što rezultira upotrebom aktivacija prethodnih slojeva. Ovaj proces preskakanja veze sažima mrežu, i stoga, mreža uči brže. Proces je popraćen sa povećanjem širine slojeva modela porastom dubine kako bi se i preostali dijelovi mreže mogli trenirati i istraživati veći prostor značajki.

Arhitektura modela ResNet-18 prikazana je na slici 3.2 u nastavku. U eksperimentalnom dijelu ovog rada koristit će se upravo ova mreža pa će biti detaljnije objašnjena kako bi se lakše pratile promjene nad arhitekturom u eksperimentima. [2]



Slika 3.2 Prikaz arhitekture mreže ResNet-18 (Preuzeto s [2])

ResNet18 mreža sastoji se od 18 slojeva, na ulaz prima trokanalnu RGB sliku. Prvi sloj u mreži je konvolucijski s filterom širine $7x7$, korakom 2 i uzastopnim popunjavanjem nulama širine 3. On na ulaz prima ulaznu RGB sliku, a kao izlaz konvolucije vraća matricu jednake dimenzije ulaznoj slici, no dubine 64. Slijedne ga redom normalizacije, ReLu prijenosna funkcija te sažimanje maksimalnom vrijednošću s filterom širine $3x3$ i korakom 2.

Idućih 16 slojeva podijeljeni su u 4 rezidualna bloka. Svaki blok sastoji se od dva slijedno povezana rezidualna sloja. Osnovni blokovi prikazani su punom strelicom, dok su blokovi koji uz osnovni blok rade i poduzorkovanje (engl. *downsample*) prikazani isprekidanom strelicom. Kod osnovnog bloka ulaz se dijeli na dvije grane, prečica (prva grana) ostaje jednaka ulazu, a druga grana prolazi kroz dva konvolucijska sloja između kojih se primjenjuje aktivacijska funkcija ReLu. Nakon što druga grana prođe kroz konvolucijske slojeve, prva i druga grana se zbroje te se zbroj dovodi na ulaz sljedećeg rezidualnog bloka i tako se slijedno izvode ostala 3 bloka. Također, možemo primijetiti kako se svakim rezidualnim blokom dubina mreže udvostruči u odnosu na dubinu prethodnog bloka. Tako dubina zadnjeg, četvrtog, sloja iznosi 512. Izlaskom iz četvrtog sloja, prvo se obavlja sažimanje srednjom vrijednošću na koje se nastavlja zadnji sloj mreže, potpuno povezani sloj. Na samom izlazu mreže očekujemo dobiti vektor (mapu značajki) duljine 1000.

Autori mreže tvrde kako slaganje slojeva ne bi trebalo degradirati performanse mreže, jer bismo jednostavno mogli slagati mapiranja identiteta (sloj koji ne radi ništa) na trenutnoj mreži, a rezultirajuća arhitektura radila bi isto. To ukazuje da dublji model ne bi trebao proizvesti grešku na skupu za treniranje veću od svojih plićih kolega.

Postoji više inačica ove mreže, neke od njih su ResNet-34, ResNet-50, ResNet-101 i ResNet-152. Broj u nazivu ovisi o broju slojeva u mreži, no svaka od arhitektura ima po četiri rezidualna bloka. Rezidualni blok svake arhitekture razlikuje se po broju slojeva u njemu. Tako u ResNet18 arhitekturi, kao što smo gore i vidjeli, svaki rezidualni blok sadrži po dva rezidualna podbloka, svaki sastavljen od

dva konvolucijska sloja između grananja. Na slici 3.3 prikazan je odnos ResNet-18 i ostalih ResNet arhitektura.[2]

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

Slika 3.3 Usporedba arhitektura različitih ResNet modela (preuzeto s [2])

Također, u Tablici 3.1 u nastavku možemo vidjeti koliko svaka od ovih arhitektura ima parametara. Možemo uočiti kako ResNet-18 ima oko 5 puta manje parametara od ResNet-152 arhitekture.

Tablica 3.1 Broj parametara mreže u ovisnosti o dubini modela

Number of Layers	Number of Parameters
ResNet 18	11.174M
ResNet 34	21.282M
ResNet 50	23.521M
ResNet 101	42.513M
ResNet 152	58.157M

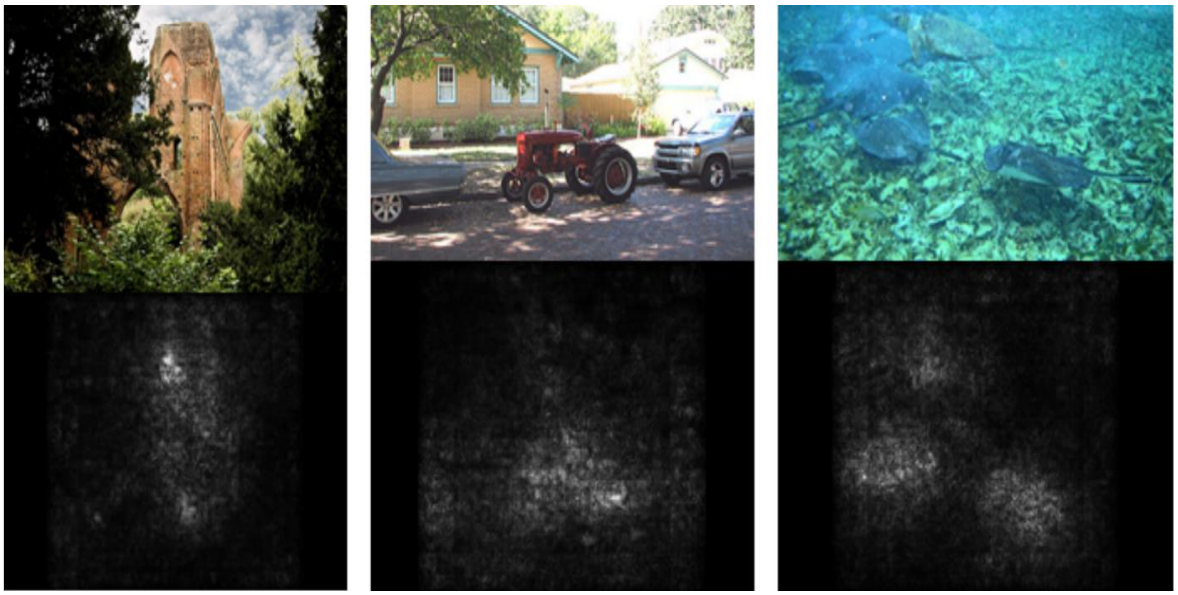
4. Tehnike vizualizacije gradijenata modela – interpretacija zaključivanja modela

Vizualizacijske tehnike za interpretaciju modela strojnog učenja vrlo su popularne i potrebne za razumijevanje načina na koji model donosi pojedine odluke. Korisne su, na primjer, u slučajevima kada model pogrešno klasificira određeni ulazni podatak i kada želimo vidjeti zašto je model donio odluku kakvu je. Tako možemo pogledati ulazni podatak i značajke koje su dovele do konačne odluke i iz toga izvući korisne zaključke. Ovim tehnikama želimo potvrditi na koji način naš model zaključuje, na temelju smislenih značajki ili samo zbog neke vrste šuma u podatcima.

Postoji mnogo radova koji koriste mape istaknutih mjesta (engl. *saliency maps*) za baš to.[7] One ističu određena područja slike, obično prikazana kao toplinska karta. Cilj istaknutih područja je prikazati stupanj važnosti piksela za donošenje odluke mreže. Stupanj važnosti izražava se ili bojom ili jakosti te boje. Što je boja izraženija, to piksel ima veći utjecaj na modelovu odluku. Odnosno, u kontekstu toplinske karte, vrućina odgovara regijama koje imaju veliki utjecaj na konačnu odluku modela.[7]

4.1. Vizualizacija gradijenata modela

Na slici 4.1 prikazane su ulazne slike (gore) s pripadajućim istaknutim područjima (dolje) koje su dobivene iz rada iz 2014. godine o Vizualizaciji klasifikacijskih modela.[7] Mape su dobivene tako što je napravljen unaprijedi prolaz slika kroz konvolucijski klasifikacijski model te su unatražnijim prolazom izračunati gradijenti po slojevima koji su prikazani na slici kao mapa istaknutih područja. U sklopu rada navedeno je kako je istrenirani model točno klasificirao sve tri slike, dakle točno ih je označio kao slika crkve, traktora i divovske raže. Model je svoje zaključke donosio



Slika 4.1 Primjer ulaznih slika i pripadajućih mapa istaknutih područja (Preuzeto s [7])

najviše na temelju bijelo označenih regija slika. Dakle, pikseli visoke uočljivosti kod slike traktora bili su baš na području slike gdje se on i nalazio, a ne na područjima slike s automobilima. Iz tog razloga je model i klasificirao sliku kao sliku traktora.

U eksperimentalnom dijelu ovog rada vizualizirat ćemo mape istaknutih područja kako bismo donijeli što više zaključaka o radu modela.

5. Korišteni skupovi podataka

Eksperimentalni dio ovog rada koristi čak dva skupa podataka. Jedan manji, „umjetno stvoren“ od strane istraživačkog tima nazvan DeRaindrop i drugi značajno veći u kojem se nalaze „stvarni“ primjeri, primjeri nastali u stvarnim situacijama od strane većeg broja korisnika zvan RT-Raindrop.

5.1. DeRaindrop

Prvi skup podataka proizašao je iz članka [1] koji se bavi problemom vizualnog uklanjanja kišnih kapi sa slike, transformirajući sliku koja sadrži kišne kapi u čistu sliku. To su postigli kombiniranjem diskriminativne i generativne mreže. Generativna mreža više pažnje posvećuje regijama slike s kišnim kapima i njene okoline, dok će diskriminativna mreža moći pristupiti lokalnoj konzistentnosti restaurirane (kišne) okoline. Ukupno ima 1168 parova slika, odnosno 2336 slika, Na slici 6.1 demonstrirani su rezultati članka. Slike su podijeljene u trening, validacijski i testni skup. Slijedno sadrže 1722, 116, 498 slika.

Lijeve slike prikazuju ulazne slike koje su degradirane kišnim kapima, dok desne prikazuje željene rezultate, odnosno ulazna slika koja je lišena kišnih kapi. U njima je vidljivo da je većina kišnih kapi uklonjena sa slike te se strukturni detalji slike obnavljaju. Skup podataka je stvoren od strane istraživačkog tima koji se bavio radom. Možemo primijetiti kako lijeva i desna slika čine svojevrstan par, pozadina slike je ista, dok su u jednom (lijevoj) slici prisutne kišne kapi, a u drugoj nisu. Parovi slika nastali su tako da je istraživački tim koristio dva jednaka stakla koje je reprezentiralo staklo automobila. Jedno je bilo poprskano vodom kako bi se stvorio efekt kapljica, a drugo je ostalo čisto kako bi se, za potrebe tog rada, stvorila željena rekreirana slika. Također, podatci su labelirani tako da klasa 0 označava prisutnost kiše, dok klasa 1 označava stanje bez kiše.

Slika 5.1 Demonstracija rada uklanjanja kapljica sa slike (Preuzeto s [1])

Za potrebe ovog rada, navedeni skup podataka činio se idealan kao početna točka za razvoj našeg klasifikacijskog problema. Parovi slika koji sadrže jednaku pozadinu, s jedinom razlikom u prisutnosti kapljica kiše čine se odličan izbor, prema tomu model bi trebao postati otporan na pozadinu prilikom donošenja odluke i fokusirati se na pronalaženje jedine razlike među klasama, što su kapljice kiše.

5.2. RT-Raindrop

Drugi skup podataka predstavlja podatke dobivene iz stvarnog svijeta, prilikom vožnje. U skupu su prisutne 3 klase, 0, 1 i 2. Podatci su bili prvotno podijeljeni podijeljeni na skupove za treniranje i validaciju koji zajedno sadrže 630.784 slika. Raspodjela po klasama je sljedeća: 280.832, 233.317, 116.635 za klase 0, 1, 2. Kako ovaj skup na početku nije imao testni podskup, dio podataka iz trening seta nasumično se odabrao da bude testni. Konačna podjela podataka po skupovima za treniranje, validaciju i testiranje su: 468.837, 15.563 i 29.711. Primjećujemo da zbroj nije jednak 630.784, već je manji za broj primjeraka klase 2 koja je izostavljena uz treniranja. Razlog je objašnjen u nastavku.

Primjerci klase 0 prikazani su na slici 6.2 u nastavku. Primjercima je zajedničko suho vrijeme bez padalina kao i pregledno okruženje, bez okolnog prometa. Na nekim primjerima prisutno je tmurnije nebo, kao da će uskoro krenuti padati oborine. Najznačajnija razlika između ove i sljedeće klase je izostanak okolnog prometa, ceste se čine gotovo prazne.

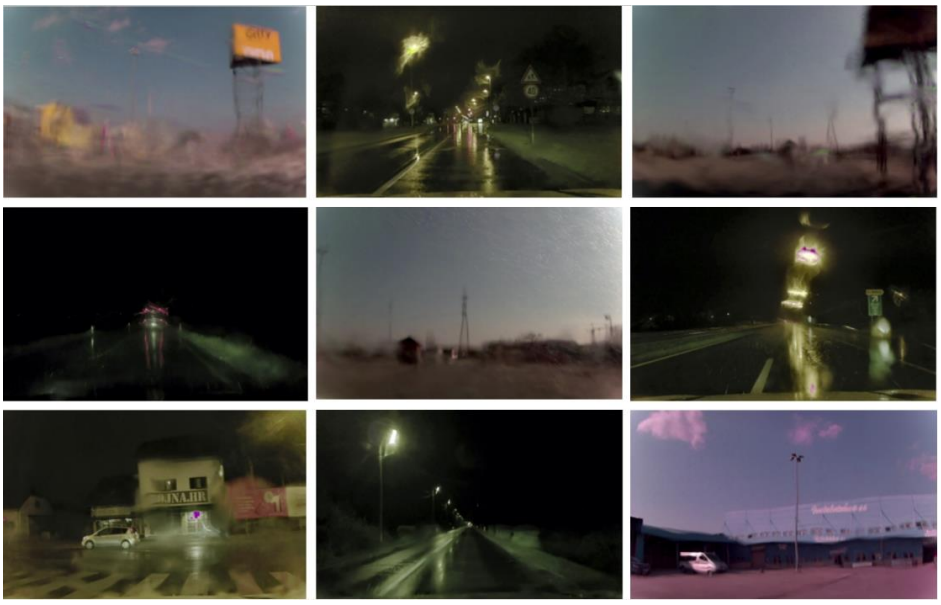
Slika 5.2 Primjerci klase 0 drugog skupa podataka

Primjerci klase 1 prikazani su na slici 6.3 u nastavku. Zajedničko im je tmurnije vrijeme, smanjena vidljivost kao i prisustvo većeg broja okolnog prometa.

Ukoliko su na primjerima prisutne kapljice kiše, one su puno većih dimenzija od onih iz skupa DeRaindrop. Također su prisutne u manjem broju, nekad se na prvu ni ne primijete, već izgledaju kao velika mrlja na fotografiji.

Slika 5.3 Primjerci klase 1 drugog skupa podataka

Primjerci klase 2 prikazani su na slici 6.4 u nastavku. Zajedničke su im mutne slike, vrlo izražena smanjena vidljivost. Primjerci ove klase izbačeni su iz eksperimentalnog dijela ovog rada iz razloga što nadilaze razinu rada, odnosno ne spadaju u rješavanje izvornog problema.



Slika 5.4 Primjerci klase 2 drugog skupa podataka

5.3. Augmentacije nad skupovima podataka

Augmentacija je proces umjetnog povećanja skupa podataka na način da se generiraju novi podatci iz postojećih podataka. To je često korištena tehnika kojom se povećava ulazni skup podataka. Nad ulaznim podatcima vrše se određene transformacije čiji je cilj napraviti što robusniji algoritam strojnog učenja. Neke od čestih transformacija slika su rotacija slike, okretanje slike oko svoje vertikalne i horizontalne osi, uzimanje centralnog ili nasumičnog isječka slike, promjena veličine slike i druge.

U sklopu eksperimentalnog dijela ovog rada koristit ćemo izrezivanje centralnog (engl. *center crop*) i nasumičnog (engl. *random crop*) isječka. Kao što im i imena govore, centralni isječak dobije se uzimanjem isječka zadane veličine iz centra slike, a nasumični isječak dobije se nasumičnim odabirom dijela slike. Također, koristit će se i promjena veličine slike (engl. *resize*). U nastavku, na slici 5.5 prikazana je originalna slika s njenom augmentacijom korištenjem centralnog i nasumičnog isječka.

Slika 5.5 Prikaz nasumičnog isječka (dolje lijevo) i nasumičnog isječka (dolje desno).

6. Korištene tehnologije

Eksperimentalni dio ovog rada pisan je u programskom jeziku Python. Najvažnija programska okruženja korištena za modeliranje mreže i dohvaćanje podataka su Pytorch i TensorFlow. Oni će biti pobliže opisani u nastavku.

6.1. Python

Python je skriptni jezik opće namjene visoke razine kojeg je stvorio Guido van Rossum 1990. godine. Ime je dobio po BBC-ovoj televizijskoj seriji „Monty Python's Flying Circus“. Python je jezik opće namjene, što znači da se može koristiti za stvaranje niza različitih programa i nije specijaliziran za bilo kakve specifične probleme. Ova svestranost, zajedno s njegovom lakoćom za početnike, učinila ga je jednim od najčešće korištenih programskih jezika danas. Istraživanje koje je provela industrijska analitičarska tvrtka RedMonk pokazala je da je to bio drugi najpopularniji programski jezik među programerima 2021. [3]

Interpretabilan je, objektno orijentirani programski jezik s dinamičkom semantikom. Njegove ugrađene strukture podataka visoke razine, u kombinaciji s dinamičkim tipkanjem i dinamičkim uvezivanjem, čine ga vrlo atraktivnim za brzi razvoj aplikacija. Obično se koristi za razvoj web stranica i softvera, automatizaciju zadataka, analizu podataka i vizualizaciju podataka. Koristi ga se i kao skriptni jezik ili jezik za povezivanje postojećih komponenti zajedno. Python-ova jednostavna sintaksa koja se lako uči naglašava čitljivost i stoga smanjuje troškove održavanja programa. Budući da je relativno jednostavan za naučiti, Python su usvojili mnogi „ne-programeri“, kao što su računovođe i znanstvenici, za razne svakodnevne zadatke. Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda. Python-ov interpreter uz opsežne standardne biblioteke, dostupni su u izvornom ili binarnom obliku bez naknade i mogu se slobodno distribuirati. [4][5]

6.2. Pytorch

PyTorch je besplatni radni okvir otvorenog koda (engl. *machine learning framework*) namijenjen za strojno učenje. Temeljen je na programskom jeziku Python i biblioteci Torch. Razvili su ga timovi za umjetnu inteligenciju iz Facebook Inc.-a 2016. godine. To je jedna od poželjnih platformi za istraživanje dubokog učenja. Okvir je izgrađen kako bi se ubrzao proces između izrade prototipa istraživanja i implementacije rješenja. [6]

6.3. TensorFlow

TensorFlow je besplatni radni okvir otvorenog koda namijenjen za numeričko računanje i strojno učenje. Stvorio ga je tim Brain iz Google-a koji je prvobitno pušten u javnost 2015. godine. Temeljen je na programskom jeziku Python i namijenjen da čini strojno učenje i razvoj neuronskih mreža bržim i lakšim. Olakšava proces prikupljanja podataka, treniranje modela, i posljedično predviđanje i pročišćavanja budućih rezultata.[16]

6.3.1. TFRecord format datoteke

TFRecord format je jednostavan format za pohranjivanje niza binarnih zapisa. Ako radite s velikim skupovima podataka, korištenje formata binarne datoteke za pohranu podataka može imati značajan utjecaj na performanse učitavanja podataka i kao posljedicu na vrijeme treniranja vašeg modela. Binarni podaci zauzimaju manje prostora na disku, potrebno im je manje vremena za kopiranje i mogu se puno učinkovitije čitati s diska. To je osobito istinito ako su vaši podaci pohranjeni na rotirajućim diskovima, zbog mnogo niže performanse čitanja/pisanja u usporedbi sa SSD-ovima.

Međutim, čista izvedba nije jedina prednost formata datoteke TFRecord. Optimizirana je za korištenje s Tensorflow-om na više načina. Za početak, olakšava kombiniranje više skupova podataka i integrira učitavanje i pred obradu podataka.

Posebno za skupove podataka koji su preveliki da bi se u potpunosti pohranili u memoriju ovo je prednost jer se samo podaci koji su potrebni u tom trenutku (npr. serija) učitavaju s diska i zatim obrađuju.

Dakle, postoji mnogo prednosti korištenja TFRecords. Ali tamo gdje ima prednosti, moraju postojati i nedostatci, a u slučaju TFRecords-a loša strana je to što svoje podatke prvo morate pretvoriti u ovaj format, a dostupna je samo ograničena dokumentacija o tome kako to učiniti.

TFRecord datoteka pohranjuje podatke kao niz binarnih stringova. To znači da se prvo mora odrediti struktura podataka prije nego što ih se zapiše u datoteku. Tensorflow nudi dvije komponente u tu svrhu: *tf.train.Example* i *tf.train.SequenceExample*. Svaki uzorak podataka mora se pohraniti u jednu od ovih struktura, zatim ga serijalizirati i koristiti *tf.python_io.TFRecordWriter* za pisanje na disk. [16]

U sklopu ovog rada drugi skup podataka dobiven je u TFRecord formatu. U nastavku je prikazan isječak koda koji predstavlja strukturu svakog podatka.

```
1   feature = {
2           'image/video': tf.io.FixedLenFeature([], tf.string),
3           'image/encoded': tf.io.FixedLenFeature([], tf.string),
4           'image/format': tf.io.FixedLenFeature([], tf.string),
5           'image/frame': tf.io.FixedLenFeature([], tf.int64),
6           'label/rain': tf.io.FixedLenFeature([], tf.int64),
7       }
```

Dakle, svaki podatak sadrži sljedeće atribute: „image/video“, „image/encoded“, „image/format“, „image/frame“ i „label/rain“. Za potrebe ovog rada, odnosno za treniranje i evaluiranje modela, najbitniji su „**image/encoded**“ i „**label/rain**“ atributi. Image/encoded atribut sadrži, kao što i sam naziv govori, enkodiranu sliku u bitovima. Da bi se ta slika predala modelu potrebno ju je prvotno dekodirati. Label/rain atribut sadrži stvarnu klasu slike. Ona predstavlja željeno predviđanje modela.

7. Eksperimenti

Svi eksperimenti u sklopu rada vrtili su se na grafičkoj kartici GeForce GTX 1070 koja sadrži 8119 MiB RAM-a.

Cilj ovog rada je oblikovati algoritam strojnog učenja koji na temelju dobivene ulazne slike određuje jesu li u njoj prisutne kapljice kiše ili ne. Kako se takav algoritam sastoji od modela, funkcije gubitka i optimizacijskog postupka, potrebno ih je odrediti. Za model smo se odlučili koristiti arhitekturu modela ResNet, u većini eksperimenata ResNet-18. Korišteni su Adam optimizacijski postupak i funkcija gubitka unakrsne entropije. Kako su optimizacijski postupak i funkcija gubitka svojstveni za sve eksperimente, u nastavku ih nećemo posebno isticati.

Učenje nad isječcima slike smanjilo bi receptivno polje modela i tako model natjeralo da odluku donosi na temelju malenih detalja slike. To je upravo ono što želimo, jer su kapi malene.

7.1. Eksperimenti provedeni nad skupom podataka DeRaindrop

7.1.1. Osnovni eksperimenti – različite veličine isječaka slika

Rezultati prvog niza eksperimenata prikazani su u tablici 7.1. Svi modeli koriste ResNet-18 arhitekturu te su trenirani i evaluirani nad skupom podataka DeRaindrop. Svi su koristili ImageNet inicijalizaciju težina te su trenirani s jednakim brojem epoha i korakom učenja. U eksperimentu se mijenjala samo veličina isječka koji se predaje ulazu modela. Isprobano je učenje nad isječcima slika veličina 32x32, 64x64, 128x128, 224x224 i 480x480. Za početak, isječci su dobiveni izrezivanjem centra svake slike (engl. *center crop*) na željenu dimenziju pomoću Pytorch-eve

funkcije.[1] Važno je napomenuti kako su slike validacijskog i testnog skupa kroz mrežu provučene u njihovim originalnim dimenzijama, dakle 480x720. Zajednički hiperparametri su:

- Broj epoha: 25
- Korak učenja: 0.0003
- ImageNet inicijalizacija

Tablica 7.1 Performanse modela treniranog na skupu podataka DeRaindrop sa različitim dimenzijama isječaka slike

Broj treniranja	**Veličina isječka**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
1.	32x32	90,18%	50,00%	50,40 %
2.	64x64	94,65%	86,21%	74,50%
3.	128x128	95,99%	99,14%	82,33%
4.	224x224	97,50%	98,20%	82,30%
5.	480x480	98,20%	100%	85,94%

7.1.2. Smanjenje koraka u prvom sloju sažimanja

Daljnja pretpostavka bila je da bi se važnost detalja mogla pojačati tako da korak prvog sloja sažimanja (sloj nakon korijenske konvolucije) postavimo na 1 umjesto zadanog (engl. *default*) 2. Korijensku konvoluciju s jezgrom 7x7 i korakom 2 ostavljamo takvu kakva je. Tada se dovodi u pitanje valjanost prednaučenih težina jer se smanjenjem koraka sažimanja izbacuje poduzorkovanje s kojim su prednaučene težine trenirane.

[1] Dokumentacija o implementaciji centralnog isječka zadane veličine
https://pytorch.org/vision/stable/generated/torchvision.transforms.CenterCrop.html

Ulaznoj slici, nakon izlaska iz korijenske konvolucije, za duplo se smanji rezolucija. U originalnoj implementaciji nakon prvog sloja sažimanja, s korakom 2, rezolucija izlaza je ponovno za duplo smanjena. Već nakon ovog sloja rezolucija podatka je četiri puta manja (/4) u odnosu na rezoluciju ulazne slike/isječka. Postavljanjem koraka sažimanja na 1, rezolucija izlaznog podatka sloja sažimanja jednaka je ulaznoj rezoluciji, dakle, ne smanjuje se i time se u tom koraku ne vrši predviđeno poduzorkovanje. Rezolucija takve implementacije nakon sloja sažimanja je dva puta manja (/2) u odnosu na rezoluciju ulazne slike/isječka. Kada se ulazni isječak propagira kroz mrežu sve do izlaza zadnjeg, četvrtog, rezidualnog bloka, njegova rezolucija se smanji za /32 u originalnoj implementaciji, odnosno /16 u implementaciji gdje je postavljen korak 1 u prvom sloju sažimanja.

Radi toga, isprobana je inicijalizacija težina ne samo s prednaučenim (ImageNet inicijalizacija), već je testirana i nasumična inicijalizacija. Broj epoha i korak učenja ostali su isti. Rezultati spomenutih eksperimenata prikazani su u tablici 7.2. Zajednički hiperparametri su:

- Broj epoha: 25
- Korak učenja: 0.0003

Tablica 7.2 Performanse modela treniranog na skupu podataka DeRaindrop sa različitim korakom prvog sloja sažimanja i početnom inicijalizacijom težina

Broj treniranja	**Veličina isječka**	**Korak prvog sažimanja**	**Inicijalizacija težina**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
6.	32x32	2	nasumična	72,90%	57,76%	52,81%
7.	32x32	1	nasumična	76,77%	50,00%	50,50%
8.	32x32	1	prednaučena	90,82%	52,59%	53,01%
9.	64x64	2	nasumična	87,75%	67,24	61,45%
10.	64x64	1	nasumična	88,63%	93,10%	76,91%

11.	64x64	1	prednaučena	93,55%	88,79%	76,91%
12.	128x128	2	nasumična	90,12%	68,10%	65,26%
13.	128x128	1	nasumična	90,74%	89,14%	75,30%
14.	128x128	1	prednaučena	96,63%	99,14%	83,13%
15.	224x224	2	nasumična	92,16%	87,07%	75,30%
16.	224x224	1	nasumična	93,42	96,55%	83,13%
17.	224x224	1	prednaučena	97,85%	99,14%	83,13%
18.	480x480	2	nasumična	96,63%	97,41%	83,33%
19.	480x480	1	nasumična	95,95%	94,65%	83,33%
20.	480x480	1	prednaučena	96,60%	97,45%	83,45%

Promatrajući rezultate dosadašnjih treniranja primjećujemo kako su isječci veličine 32x32 i 64x64 premaleni te da se mreža puno bolje snalazi u donošenju odluka kada je trenirana nad slikama većih dimenzija. Značajnijih razlika u performansama modela treniranim nad većim isječcima (128x128, 224x224, 480x480) i nema. Najveća razlika na testnom setu između njih iznosi, uz isti set hiperparametara, 3,64%.

Prethodna pretpostavka o smanjenju koraka prvog sloja sažimanja radi povećanja utjecaja detalja pokazala se uspješom. Ako usporedimo rezultate modela treniranih u prvom nizu eksperimenata s njihovim parovima u drugom nizu (modeli trenirani s istim hiperparametrima, jedina razlika u koraku u sloju sažimanja), vidimo kako su se performanse svih modela povećale smanjenjem koraka prvog sloja sažimanja na 1. Uz to, ImageNet inicijalizacija težine pokazala se boljom u svim eksperimentima u odnosu na one nasumično inicijalizirane.

Iz rezultata je vidljivo da je najbolji rezultat postignut u 20. treniranju nad slikama veličine 480x480, no odlučili smo kako će se kao referentni model u daljnjim eksperimentima koristiti model dobiven 14. treniranjem treniran nad isječcima

veličine 128x128. Razlika u točnosti na testnom skupu između njih manja je od 0,4%, a 14. model uči nad manjim ulaznim podatcima te time zahtjeva manju memoriju i samo treniranje je brže.

7.1.3. Performanse modela ResNet-18 vs. ResNet-34 vs. ResNet-50

Idućim nizom eksperimenata zanimalo nas je kakve bi rezultate davali modeli s dubljim arhitekturama kao što su ResNet-34 i ResNet-50. U tablici 7.3 prikazani su rezultati treniranja. Oba modela imala su iste hiperparametre kao najbolji/referentni model ResNet-18. Zajednički hiperparametri su:

- Broj epoha: 25
- Korak učenja: 0.0003
- ImageNet inicijalizacija
- Korak prvog sloja sažimanja: 1
- Veličina isječka: 128x128

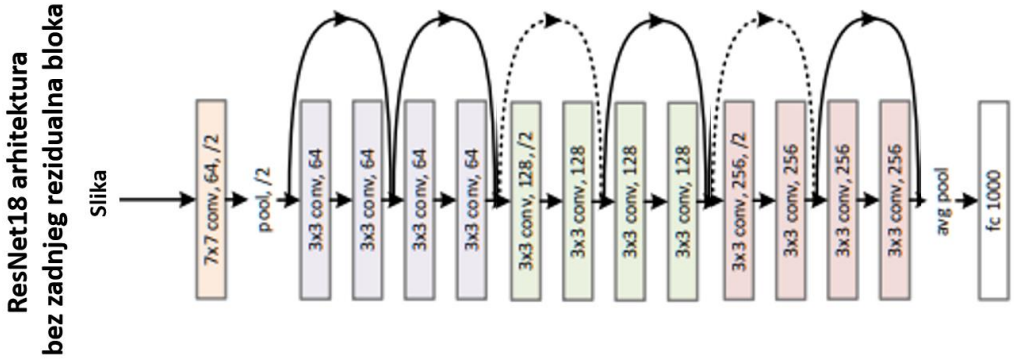
Tablica 7.3 Performanse modela različitih arhitektura trenirane s istim hiperparametima

Broj treniranja	**Arhitektura modela**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
14.	ResNet-18	96,63%	99,14%	83,13%
21.	ResNet-34	95,93%	98,28%	84,54%
22.	ResNet-50	96,45%	96,55%	82,53%

Performanse modela ResNet-34 u odnosu na referentni ResNet-18 povećale su se za 1,41%, a ResNet-50 smanjile za 0,60%. Ako se prisjetimo 3. poglavlja o ResNet arhitekturi, znamo kako ResNet-34 arhitektura ima skoro pa duplo više parametara od ResNet-18 arhitekture. To povlači dulje vrijeme prolaska kroz jednu epohu, a time i dulje vrijeme treniranja kao i veće memorijsko zauzeće modela. Uzevši u obzir i povećanje točnosti na testnom skupu od svega 1,41%, odluka je nastaviti koristiti ResNet-18 arhitekturu za daljnja treniranja.

7.1.4. Performanse modela vs. manji broj rezidualnih blokova

Iz ovih eksperimenata zaključili smo kako povećanjem broja slojeva ne dobivamo značajno na performansama modela. U idućem eksperimentu zanimalo nas je što bi bilo kada bismo smanjili broj slojeva, odnosno kada bismo maknuli zadnji, od ukupno četiri rezidualna bloka. Implementacijski je to napravljeno tako da je umjesto četvrtog rezidualnog bloka postavljena matrica identiteta koja dobiveni ulaz samo prepušta dalje idućem sloju modela, a on je potpuno povezani sloj. Također, bilo je potrebno promijeniti i očekivane ulazne dimenzije potpuno povezanog sloja. Na slici 7.1 prikazana je ResNet-18 arhitektura bez zadnjeg rezidualnog bloka.



Slika 7.1 ResNet18 arhitektura bez zadnjeg rezidualnog bloka

U tablici 7.4 prikazani su rezultati eksperimenta nad modelom bez zadnjeg bloka. Model je treniran nad istim hiperparametrima kao najbolji/referentni model ResNet-18. Iz prikazanih rezultata vidimo kako se modelova sposobnost zaključivanja značajno smanjila micanjem zadnjeg rezidualnog bloka i kako ta akcija nije povoljna za poboljšanje performansi modela.

Tablica 7.4 Performanse modela različitih arhitektura trenirane s istim hiperparametima

Broj treniranja	**Arhitektura modela**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
14.	ResNet-18	96,63%	99,14%	83,13%
23.	ResNet18 – bez 4. bloka	96,34%	91,38%	76,31%

Dosadašnjim eksperimentima zaključili smo kako mreža bolje zaključuje kada je trenirana nad isječcima veće dimenzije. To povlači činjenicu da joj je potrebno više informacija, detalja pa tako i šire receptivno polje. Iduća stvar koja je pomogla performansi modela je smanjenje koraka u sloju sažimanja. Time model može više pažnje posvetiti detaljima slike, jer je dio piksela prilikom pomicanja okvira koji se sažima zahvaćen u više okvira. No, na taj način nismo očuvali semantiku ImageNet inicijalizacije i pritom smo smanjili modelovo receptivno polje. Vidjeli smo da se smanjenjem dubine gube performanse modela. Također, u svim eksperimentima ImageNet inicijalizacija pokazala se boljom od nasumične.

Iz svih ovih zapažanja nameće se napraviti takvu strukturu modela da on čuva semantiku ImageNet težina uz korištenje smanjenog koraka u prvom sloju sažimanja, a da pritom receptivno polje ostane jednako originalnoj implementaciji. To ćemo napraviti povećavanjem parametara dilatacije u konvolucijskim slojevima rezidualnih blokova. Korijensku konvoluciju i dalje ostavljamo istom, nećemo ju mijenjati.

7.1.5. Uvođenje dilatirane konvolucije u rezidualne blokove

Da bi bila moguća promjena parametara dilatacije unutar arhitekture modela, bilo je potrebno izmijeniti izvorni kod (engl. *source code*). Korištena je Pytorch-eva izvorna implementacija modela ResNet-18 koja se može pronaći na njihovoj službenoj GitHub stranici. [2] Potrebne promjene nad izvornim kodom kako bi se postigla implementacija željene struktura modela bile su sljedeće:

1. Kod pozivanja modela, postaviti dodatni argument *dilation* koji označava na koju će vrijednost biti postavljena dilatacija unutar svake 2D konvolucije unutar rezidualnih blokova. Također, ukoliko se argument ne preda,

[2] Link na Pytorch-ev GitHub repozitorij koji sadrži programsku implementaciju originalnog modela ResNet-18: https://github.com/pytorch/vision/tree/main/torchvision/models

pretpostavlja se da je željena dilatacija jednaka izvornoj, 1. Primjer poziva modela s ugrađenim parametrom za dilataciju prikazan je u nastavku.

```
network = resnet18(pretrained=True, progress=True, dilation=2)
```

2. U klasi *BasicBlock* potrebno je zakomentirati dio koji vraća grešku (engl. *raise error*) ukoliko je dilatacija postavljena na više od 1. Spomenuti dio koda koji je potrebno zakomentirati prikazan je u nastavku.

```
class BasicBlock(nn.Module):
    def __init__():
        ...

        if dilation > 1:
            raise NotImplementedError("Dilation > 1 not supported in BasicBlock")

        ...
```

3. U funkciji *make_layer* potrebno je zakomentirati dio koji, ukoliko se mijenja parametar dilatacije u arhitekturi modela, postavlja korak konvolucije (engl. *stride*) na 1, a dilataciju dodatno povećava za veličinu tog koraka. U kontekstu arhitekture ResNet-18, koraci unutar konvolucijskog sloja mogu biti 1 ili 2. To znači da bi se, ukoliko želimo postaviti dilataciju na 2, u određenim konvolucijskim slojevima koji sadrže korak=2, dilatacija sama postavila na 4. Mi bismo voljeli da u svim slojevima dilatacija bude jednaka, onakva kakvu smo ju izvorno postavili, jednaka 2. Spomenuti dio koda koji je potrebno zakomentirati prikazan je u nastavku.

```
def _make_layer():
        ...
        if dilate:
            self.dilation *= stride
            stride = 1
        ...
```

4. I kao zadnji korak, kako bismo očuvali ImageNet inicijalizaciju težina unutar funkcije *conv3x3,* koja služi za konstruiranje 2D konvolucijskih slojeva unutar rezidualnih blokova s jezgrom 3x3, postavljamo nadopunjavanje (engl. *padding*) na istu vrijednost kao i vrijednost dilatacije. Time postižemo da izlazna matrica tog konvolucijskog sloja bude jednakih dimenzija kao što je u izvornoj arhitekturi pretpostavljeno (kada je dilatacija postavljena na 1).

Spomenuta funkcija prikazana je u nastavku. Žutom bojom označen je dio koda koji je potrebno izmijeniti.

```
def conv3x3(in_planes: int, out_planes: int, stride: int = 1, groups: int = 1,
            dilation: int = 1) -> nn.Conv2d:
    """3x3 convolution with padding"""
    return nn.Conv2d(
        in_planes,
        out_planes,
        kernel_size=3,
        stride=stride,
        padding=dilation,
        groups=groups,
        bias=False,
        dilation=dilation,
    )
```

U tablici 7.5 nalazi se primjer jednog tako implementiranog konvolucijskog sloja unutar rezidualnog bloka. Lijevo je prikazana originalna struktura sloja (sa dilatacijom=1), a desno je prikazana struktura s dilatacijom=2. U tablici vidimo kako se postavljanjem dilatacije paralelno postavlja i pripadajuće nadopunjavanje na istu vrijednost dilatacije. Također, možemo primijetiti kako se korak u oba slučaja nije mijenjao.

Tablica 7.5 Struktura konvolucijskog sloja unutar rezidualnog bloka s postavljenom dilatacijom na 1 i 2

Konvolucija sa dilatacijom 1	**Konvolucija sa dilatacijom 2**
(conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), dilation=(1, 1), bias=False)	(conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(2, 2), dilation=(2, 2), bias=False)

U sljedećoj tablici, 7.6, prikazane su performanse modela treniranih sa dilatacijom 2 i 4 u odnosu na referentni model ResNet-18 s dilatacijom 1. Modeli su trenirani nad istim hiperparametrima kao najbolji/referentni model ResNet-18. Unaprijed nismo mogli znati što očekivati zbog toga što je ovakvom implementacijom napravljeno da dilatacija s jedne strane čuva semantiku modela

učenog na ImageNet-u, a s druge strane povećava receptivno polje koje možda neće biti povoljno za detekciju malih objekata.

Tablica 7.6 Performanse modela s različitim parametrom dilatacije unutar rezidualnih slojeva trenirane s istim setom hiperparametara

Broj treniranja	**Arhitektura modela**	**Dilatacija u rezidualnim blokovima**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
14.	ResNet-18	1	96,63%	99,14%	83,13%
24.	ResNet-18	2	96,51%	90,52%	75,50%
25.	ResNet-18	4	96,10%	95,69%	76,51%

Iz prikazanih rezultata ispostavilo se kako povećanje dilatacije nije prikladno za razmatrani problem. Unatoč očuvanju ImageNet semantike, povećavanje receptivnog polja nije pogodno za detekciju malih objekata poput kapljica.

7.1.6. Traženje optimalne kombinacije hiperparametara

Sada kada znamo optimalnu arhitekturu mreže za zadani problem pozabavit ćemo se pronalaskom kombinacije idealnih hiperparametara. Prethodnim treniranjima zaključili smo kako modelu pašu:

- Prednaučena (ImageNet) inicijalizacija težina
- Korak u prvom sloju sažimanja: 1
- Dilatacija: 1

Njih ćemo tako zamrznuti te će promatrani hiperparametri biti:

- Broj epoha: 30, 50, 75, 100, 150
- Stopa učenja: 0.001, 0.0001, 0.0003, 0.00005

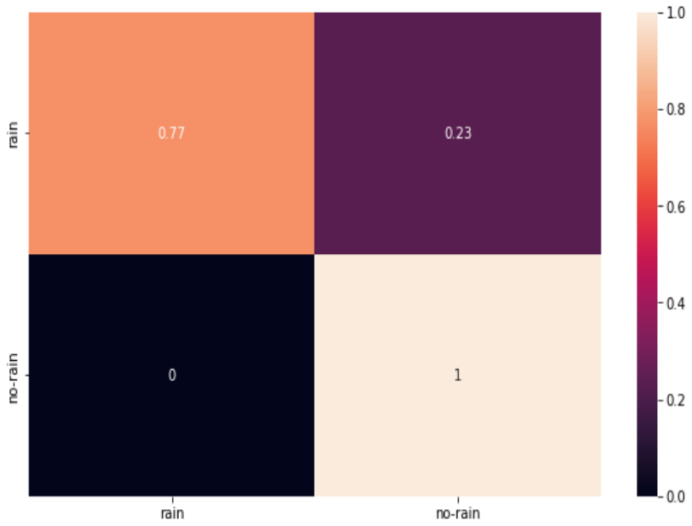
Rezultati nekih treniranja na temelju kojih smo došli do zaključaka o optimalnom setu hiperparametara prikazani su u tablici 7.7.

Broj treniranja	Broj epoha	Stopa učenja	Točnost (Trening)	Točnost (Validacija)	Točnost (Test)
14.	25	0.0003	96,63%	99,14%	83,13%
26.	30	0.0003	96,75%	96,55%	78,51%
27.	50	0.0003	97,21%	95,68%	77,91%
28.	30	0.001	93,15%	96,55%	88,15%
29.	50	0.001	94,71%	93,96%	78,30%
30.	75	0.001	96,98%	94,82%	80,73%
31.	30	0.00005	97,56%	98,27%	82,93%
32.	50	0.00005	98,03%	93,10%	78,11%

U njoj vidimo kako modelu više paše veći korak učenja, u protivnom gubi na sposobnosti generalizacije. Također, ukoliko model pustimo da se trenira duži period, kroz veći broj epoha, također mu se smanjuje sposobnost generalizacije. Optimalni hiperparametri, prema provedenim dosadašnjim mjerenjima su:

- Broj epoha: 30
- Stopa učenja: 0.001
- Prednaučena (ImageNet) inicijalizacija težina
- Korak u prvom sloju sažimanja: 1
- Dilatacija: 1

Najbolji model postiže točnost na testnom skupu od 88,15%. U nastavku, na slici 7.2, je prikazana konfuzijska matrica modela na testnom skupu. Matrica pokazuje kako model odlično raspoznaje slike na kojima nisu prisutne kapljice kiše, no muči se s dijelom slika gdje su one prisutne.



Slika 7.2 Normalizirana Matrica konfuzije najboljeg modela (28. treniranje) na testnom skupu.

Malo detaljniji prikaz performansa modela nalazi se u tablici 7.7 u nastavku. U njoj, osim dosadašnje mjere vrednovanja modela, točnosti, prikazane su i preciznost (engl. *precision*), odziv (engl. *recall*) i f1-rezultat (engl. *f1-score*) modela na testnom skupu. Također, prikazane metrike odvojene su po klasama.

Kako je preciznost mjera koja gotovi o tome koliko primjeraka pojedine klase je model ispravno klasificirano, vidimo kako su slike koje je model označio kao slike s prisutnim kapljicama gotovo pa uvijek takve slike. No, to nije slučaj sa slikama za koje model tvrdi da nemaju prisutne kapljice. Čak 19% tako označenih slika testnog skupa je pogrešno klasificirano. S druge strane, na temelju odziva, mjere koja govori koliko od ukupno primjera pojedine klase je ispravno označeno, vidimo kako model sve slike u kojima nisu prisutne kapljice kiše tako i označava. Tek 77% od svih primjeraka na kojima je označena prisutnost kapljica kiše je pronađeno.

Tablica 7.7 Preciznost, opoziv, f1-rezultat i točnost najboljeg modela (28. treniranje) na testnom skupu DeRaindrop podataka

	Preciznost	**Odziv**	**F1-rezultat**
Rain	99%	77%	87%
No rain	81%	100%	89%

Kao i po matrici konfuzije, na temelju analize tablice 7.7 vidljivo je kako se model poprilično dobro nosi sa slikama u kojima nisu prisutne kapljice kiše, gotovo 100%. S druge strane modelova sposobnost ispravne detekcije slika s kapljicama je manja. To nam i nije najpoželjnija situacija, jer upravo takve slike su nam u kontekstu autonomne vožnje najvažnije kako bismo znali vršiti prave akcije i obratiti veću pozornost jer su oko nas prisutni nepovoljniji vremenski uvjeti.

7.1.7. Dodatne augmentacije nad ulaznim podatcima

Sad kada smo našli optimalne hiperparametre voljeli bismo još isprobati mogu li se modelove performanse dodatno poboljšati različitim augmentacijama. Do sada su sva treniranja rađena nad centralnim isječcima veličine 128x128. Time gubimo veliki dio informacija koji se krije u ostalim dijelovima slike koji su modelu možda relevantniji za donošenje odluka. Korištena je PyTorch-eva implementacija idućih augmentacije nad podatcima:

- Nasumični isječak veličine 128x128 (engl. *random crop*)[3]
- Nasumični isječak nasumične veličine na kojeg se nastavlja promjena veličine isječka na 128x128 (engl. *random resized crop*)[4]

Tablica 7.8 Performanse modela trenirane različitim tehnikama augmentacije podataka

Broj treniranja	**Augmentacija podataka**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
28.	Centralni isječak	93,15%	96,55%	88,15%
33.	Nasumični isječak	90,36%	99,14%	84,34%
34.	Nasumični isječak + promjena veličine	95,93%	90,52%	85,14%

[3] Dokumentacija o korištenoj implementaciji nasumičnog isječka zadane veličine https://pytorch.org/vision/main/generated/torchvision.transforms.RandomCrop.html
[4] Dokumentacija o korištenoj implementaciji nasumičnog isječka nasumične veličine koji se zatim skalira na zadanu veličinu https://pytorch.org/vision/main/generated/torchvision.transforms.RandomResizedCrop.html

Iz priloženih rezultata u tablici 7.8 vidimo kako odabirom nasumičnih isječaka ne postižemo bolje performanse modela. Rezultati na prvu začuđuju, no kada promatramo skup podataka DeRaindrop primjećujemo kako su na slikama u kojima su prisutne kapljice kiše, one prisutne na cijeloj slici. Uzimanjem uvijek istog isječka svake slike, kod svakog para slike sa i bez kapljica prisutna je jednaka pozadina pa model na ovako malom broju epoha ne uspijeva naučiti tražiti najbitnije diskriminativne karakteristike među klasama. Kako bismo dali poštenu priliku ovim novim augmentacijama pokušali smo ih trenirati na više epoha. No i time nismo uspjeli nadmašiti rezultate dobivene s centralnim isječkom. Dobiveni rezultati prikazani su tablicom 7.9.

Tablica 7.9 Performanse modela s augmentacijama trenirane na više epoha

Broj treniranja	**Augmentacija podataka**	**Broj epoha**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
28.	Centralni isječak	30	93,15%	96,55%	88,15%
35.	Nasumični isječak	50	88,73%	97,41%	86,95%
36.	Nasumični isječak	75	89,95%	98,28%	80,52%
37.	Nasumični isječak + promjena veličine	50	93,67%	86,21%	77,51%
38.	Nasumični isječak + promjena veličine	75	95,00%	72,41%	67,89%

7.1.8. Evaluacija modela na skupu podataka RT-Raindrop

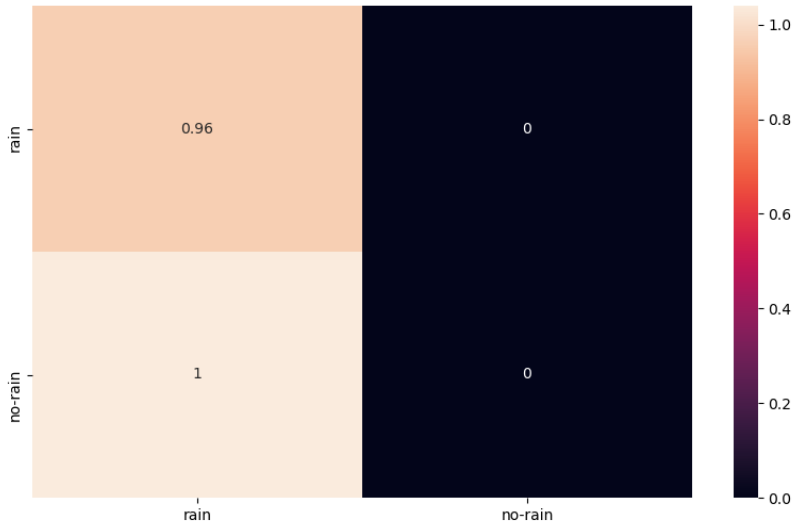
Do sada smo evaluacije (i treniranja) vršili isključivo na DeRaindrop skupu podataka. U idućoj tablici prikazana je točnost modela, dobivenim 28. treniranjem, na RT-Raindrop skupu podataka. Već smo u poglavlju 5. objasnili razlike između primjeraka ta dva skupa podataka te kako „logika“ podjele po klasama nije ista.

Podsjetimo se, DeRaindrop skup podataka dijeli slike na temelju prisutnosti kapljica kiše, a RT-Raindrop skup najjednostavnije rečeno, dijeli primjerke na temelju uvjeta na cesti – povoljni ili manje povoljni/nepovoljni uvjeti. U toj klasi koja predstavlja nepovoljne uvjete nalaze se i slike na kojima su prisutne kapljice, no u njoj se nalaze i primjerci koji ne sadržavaju kišne kapi, već neke druge oblike nepovoljnih uvjeta koji su učestaliji, poput gužve na cesti i tmurnog neba.

Tablica 7.10 Točnost modela na RT-Raindrop skupu podataka.

Broj treniranja	**Točnost (RT-Raindrop)**
28.	45%

Točnost modela treniranog na skupu podataka DeRaindrop nema dobru sposobnost generalizacije nad skupom RT-Raindrop. Štoviše, model je gori od nasumičnog bacanja novčića. Ipak se model previše specijalizirao za jednu funkciju, detekciju prisutnosti mnoštvo sitnijih kapljica i ne može dobro razaznati povoljne od nepovoljnih uvjeta. Velike razlike u performansama modela na skupu podataka DeRaindrop, na kojem je učio, i skupu podataka RT-Raindrop upućuju na pomak u domeni među skupova. To i nije tako začuđujuće jer, kao što smo već i rekli, naizgled ti skupovi izgledaju različito te je logika podjele klasa različita. U skupu RT-Raindrop, kada bi se i na slici nalazile kapljice kiše, one bi naizgled izgledale vrlo različito od onih DeRaindrop skupa. Naime, u njemu su kapljice, ukoliko su prisutne, mnogo veće dimenzije i na slici nije jasno označen rub te kaplje, već na slici one izgledaju kao neke mrlje. Te rezultate pokazuju i normalizirana konfuzijska matrica na slici 7.3 i tablica 7.11 koja prikazuje odnos preciznosti, odziva i f1-rezultata mreže evaluirane na testnim podatcima skupa RT-Raindrop. Dublji uvid u zaključivanje modela dobit ćemo promatrajući ih.



Slika 7.3 Normalizirana Matrica konfuzije najboljeg modela (28. treniranje) na testnom skupu RT-Raindrop podataka

Matrica konfuzije kao i tablica 7.9 govore nam kako je model za svaku sliku skupa RT-Raindrop misli kako su na njoj prisutne kapljice kiše. Dakle, da je takav model služio kao senzor brisačima, prema ovim rezultatima oni se nikad ne bi ugasili, što je vrlo loše.

Tablica 7.11 Preciznost, opoziv, f1-rezultat i točnost najboljeg modela (28. treniranje) nad testnim skupom RT-Raindrop podataka.

	Preciznost	**Odziv**	**F1-rezultat**
Rain	48%	100%	65%
No rain	0%	0%	0%

7.2. Eksperimenti provedeni nad skupom podataka RT-Raindrop

Rezultati prvog eksperimenata prikazani su u tablici 7.12. Hiperparametri modela jednaki su najboljem modelu treniranim na skupu podataka DeRainDrop s dvije razlike. Jedna razlika je broj epoha koji je u ovom slučaju 5. Druga razlika je

ta što su originalne slike ovog skupa podataka veće početne rezolucije u odnosu na DeRaindrop, sve slike su dimenzija 1080x1920. Također, raspodjela kapljica, na slikama koje ih sadrže, nije po cijeloj slici, već na njenom manjem dijelu i većinski na rubovima. Iz tih razloga, augmentacija nad ulaznom slikom nije bila izrezivanje centralnog isječaka, već samo promjena rezolucije cijele slike na dimenziju 128x128. Radi iznimno velikog broja podataka ovog skupa i dugog trajanja vrćenja jedne epohe, čak 24 sata, odabran je tako mali broj. Iz tablice vidimo kako je model, na trening skupu i u tih 5 epoha naučio gotovo sve primjerke. Na temelju validacijskog i testnog skupa zaključujemo kako se model prenaučio na skupu za treniranje i time je izgubio sposobnost generalizacije. Validacijski i testni skup evaluirani su na način da se ulazna slika skalirala na dimenziju 512x512 te tako provukla kroz model iz razloga što bi grafička kartica ostala bez memorije u slučaju prolaska slike potpune rezolucije.

Tablica 7.12 Performanse modela treniranog na skupu podataka RT-Raindrop

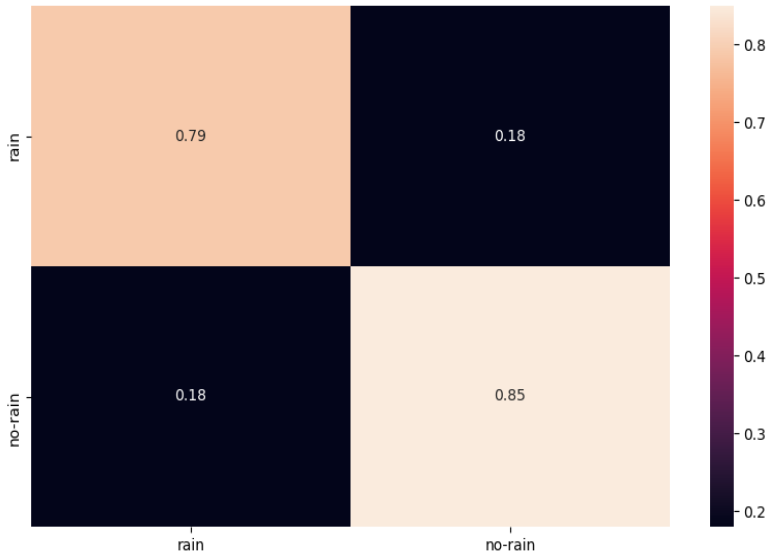
Broj treniranja	**Augmentacija podataka**	**Broj epoha**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
38.	Promjena veličine na 128x128	5	99,06%	80,02%	62,49%

Zbog loše performanse modela na testnom skupu, odlučili smo dotrenirati model tako da u njega ubacujemo nasumične isječke veće veličine, rezolucije 224x224. To smo napravili u dva navrata, jednom smo model dotrenirali na samo jednu, a drugi put na dvije dodatne epohe. Rezultati su prikazani u tablici 7.13 ispod. Iz rezultata vidljivo je kako je povećanje isječaka dovelo do značajnog poboljšanja performansi modela. Razlika točnosti modela dotreniranog na jednoj, odnosno dvije epohe na testnom skupu je vrlo mala. Pritom se točnost na testnom i validacijskom skupu povećala treniranjem na više epoha. Daljnji rezultati i ostale metrike u nastavku prikazane su nad modelom dobivenim 40. treniranjem.

Tablica 7.13 Performanse modela dotreniranog na jednoj epohi s ulaznim podatcima većih dimenzija u odnosu na 38. treniranje.

Broj treniranja	**Augmentacija podataka**	**Broj epoha**	**Točnost (Trening)**	**Točnost (Validacija)**	**Točnost (Test)**
39.	Nasumični isječak veličine 224x224	1	88,62%	70,55%	83,51%
40.	Nasumični isječak veličine 224x224	2	93,27%	90,62%	82,16%

U nastavku, na slici 7.4, je prikazana normalizirana konfuzijska matrica modela na testnom skupu. Matrica pokazuje kako model podjednako dobro raspoznaje slike obiju klasa, no ipak je malo bolji u točnom raspoznavanju slika na kojima nisu prisutne kapljice kiše, odnosno slikama dobrih vremenskih uvjeta.



Slika 7.4 Normalizirana matrica konfuzije modela (40. treniranje) na testnom skupu RT-Raindrop podataka.

Kako bismo dodatno analizirali performanse modela, u nastavku u tablici 7.14 prikazane su preciznost, odziv, f1-rezultat po klasama. Već smo rekli da je preciznost mjera koja gotovi o tome koliko primjeraka pojedine klase je model ispravno klasificirano, a odziv mjera koja govori koliko od ukupno primjera pojedine klase je ispravno označeno. F1-rezultat je težinska kombinacija te dvije mjere. Sve tri mjere

govore nam kako model podjednako pogađa/griješi po klasama. Preciznost nam govori kako je od svih primjeraka testnog skupa koje je model klasificirao kao slike kiše/nepovoljnih vremenskih uvjeta, njih 81% bio ispravno određeno. Na isti način, 83% slika za koje je model mislio kako su slike bez kiše/povoljnih uvjeta je bilo ispravno klasificirano. Odziv nam govori isto. 82% svih slika prve klase, kao i druge ispravno su klasificirane.

Tablica 7.14 Preciznost, opoziv, f1-rezultat i točnost modela (40. treniranje) nad testnim skupom RT-Raindrop podataka.

	Preciznost	**Odziv**	**F1-rezultat**
Rain	81%	82%	82%
No rain	83%	82%	83%

Prema svim gornjim evaluacijama, model radi vrlo dobro. Bilo bi dobro dodatno unaprijediti preciznost modela u detekciji loših vremenskih uvjeta.

7.2.1. Evaluacija modela na skupu podataka DeRaindrop

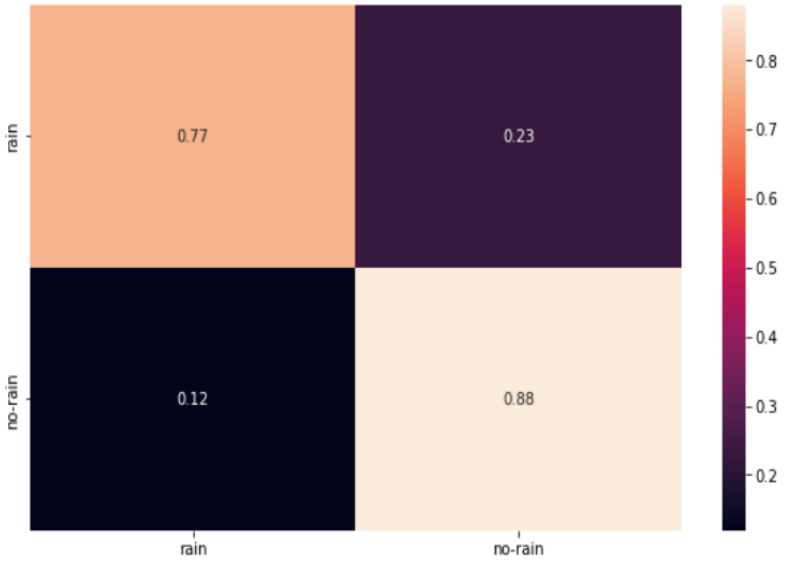
Kao što smo to napravili i s modelom treniranim na skupu DeRaindrop, tako ćemo i model treniran na skupu podataka RT-Raindrop evaluirati na DeRaindrop-u. U nastavku, u tablici 7.15 prikazana je točnost modela na tom skupu. Za razliku od rezultata u poglavlju 7.1.8, ovi su mnogo bolji. Točnost modela na testnom skupu podataka DeRaindrop na kojem model nije učio je čak 82%. Zanimljivo je kako je točnost na ovom skupu jednaka točnosti na testnom skupu RT-Raindrop na kojem je i treniran.

Tablica 7.15 Točnost modela (40. treniranje) na skupu podataka RT-Raindrop.

Broj treniranja	**Točnost (DeRaindrop)**
38.	82%

Daljnjom analizom, na temelju matrice konfuzije na slici 7.5 i tablice preciznosti, odziva i f1-rezultata, detaljnije ćemo proučiti zaključivanje modela.

Matrica pokazuje kako model vrlo dobro raspoznaje slike na kojima nisu prisutne kapljice kiše, a malo slabije prepoznaje slike gdje su one prisutne.



Slika 7.5 Normalizirana Matrica konfuzije najboljeg RT-Raindrop modela (40. treniranje) na testnom skupu DeRaindrop podataka.

Ukoliko detaljnije promotrimo tablicu 7.16, vidjet ćemo kako je model čak 86% slika, koje je klasificirao kao slike s kišom, ispravno klasificirao. No, nešto manji postotak, od 79%, slika za koje je smatrao da ne pada kiša je ispravno klasificirao. Odziv iznosi 77% za klasu koja predstavlja kišne iliti nepovoljne vremenske uvjete i 88% za klasu sa suhim, povoljnim uvjetima. Razlika između odnosa među klasama između preciznosti i odziva govori nam kako je veći broj primjeraka testnog skupa svrstan u povoljne vremenske uvjete. Zato je postotak točnih predikcija modela manji (preciznost) u odnosu na ukupni postotak točno određenih primjera (odziv) klase koja ne posjeduje kišne kapi.

Tablica 7.14 Preciznost, opoziv, f1-rezultat i točnost najboljeg RT-Raindrop modela (40. treniranje) nad testnim skupom DeRaindrop podataka.

	Preciznost	**Odziv**	**F1-rezultat**
Rain	86%	77%	81%
No rain	79%	88%	83%

7.3. Vizualizacija logike zaključivanja finalnog modela

Do sada smo analizirali performanse modela isključivo na temelju brojčanih metrika. U ovom dijelu ćemo pokušati prikazati i objasniti logiku zaključivanja modela. Prikazat ćemo mreži „najbolje“ i „najgore“ primjere, odnosno primjere koje je mreža s velikom sigurnošću točno odredila i one za koje je s velikom sigurnošću odredila suprotno.

7.3.1. Vizualizacija logike modela treniranog s DeRaindrop

Na sljedećim slikama prikazani su neki od mreži najlakših primjera, odnosno točno klasificiranih primjera. Prikazuju se originalna slika (lijevo) i pripadajući gradijent ulaza (desno). Prva slika, 7.6 prikazuje slike koje sadrže kapljice kiše i model ih je s velikom sigurnošću tako klasificirao. Vidimo kako se gradijenti najveće vrijednosti, što svjetliji dijelovi slike, nalaze baš na mjestima gdje se nalaze i kapljice kiše.

Slika 7.6 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najlakše primjere koji sadrže kapljice kiše.

Na idućoj, 7.7 slici, prikazani su primjeri koji ne sadrže kišne kapi i koji su modelu bili lagani za klasifikaciju. Za razliku od gradijenata ulaza koji sadrže kapljice kiše, gradijenti ulaza bez kiše po iznosu su manji i imaju manji broj žarište.

Slika 7.7 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najlakše primjere koji ne sadrže kapljice kiše.

Na idućim slikama prikazani su oni najteži primjeri u testnom DeRaindrop skupu s njihovim gradijentima ulaza. Prva slika, 7.8, prikazuje slike koje su označene da sadrže kapljice kiše, a model ih je s velikom razlikom u sigurnosti označio da ne sadrže. To su većinski slike koje na prvu sadrže vrlo malo kapljica i te kapljice nemaju jasno označeni rub, teško se prepoznaju.

Slika 7.8 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najteže primjere koji sadrže kapljice kiše.

Slijedeća slika, 7.9, prikazuje slike koje su označene da ne sadrže kapljice kiše, a model ih je označio da sadrže. Takvih slika u testnom skupu je samo jedna. I očekivano je da je takvih primjera vrlo malo, jer smo već prije zaključili kako model vrlo dobro predviđa one primjere koji u sebi nemaju prisutne kapljice kiše.

Slika 7.9 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najteže primjere koji ne sadrže kapljice kiše.

7.3.2. Vizualizacija logike modela treniranog s RT-Raindrop

Na sljedećim slikama, kao u pod poglavlju iznad, prikazani su neki od mreži najlakših primjera, odnosno točno klasificiranih primjera. Prve četiri slike su slike iz skupa podataka RT-Raindrop, dok su iduće četiri slike skupa DeRaindrop. Na svakoj slici prikazuju se originalna ulazna slika (lijevo) i pripadajući gradijent ulaza (desno).

Prva slika, 7.10 prikazuje slike nepovoljnih vremenskih uvjeta i model ih je s velikom sigurnošću tako klasificirao. Primjećujemo kako su na prikazanim slikama prisutne kapljice kiše, no u mnogo manjem broju u odnosu na najlakše slike skupa DeRaindrop koje su bile prikazane na slici 7.6. Također, gradijenti ulaza raštrkani su po cijeloj slici, ne samo na dijelovima na kojima su kapljice i manjeg su iznosa.

Slika 7.10 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najlakše primjere otežanih vremenskih uvjeta koji sadrže kapljice kiše.

Na idućoj, 7.11 slici, prikazani su primjeri povoljnih vremenskih uvjeta, bez kišnih kapi koji su modelu bili lagani za klasifikaciju. Za razliku od gradijenata ulaza nepovoljnih uvjeta, gradijenti ulaza povoljnih su po iznosu još manji, s podjednakim brojem žarišta raspoređenim po cijeloj slici. Hvataju se na cestu, okolni krajolik, arhitekturu i promet. Gradijent se najmanje prima za dijelove koji prikazuju nebo.

Slika 7.11 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najlakše primjere povoljnih uvjeta bez kiše.

Na iduće dvije slike vizualizirani su oni najteži primjeri u testnom RT-Raindrop skupu s njihovim gradijentima ulaza. Prva slika, 7.12, prikazuje slike koje su označene da prikazuju otežane vremenske uvjete, a model ih je s velikom razlikom u sigurnosti označio da ne sadrže. To su većinski slike koje na prvu, kao i kod DeRaindrop skupa, sadrže vrlo malo ili ništa kapljica i te kapljice nemaju jasno označeni rub, teško se prepoznaju.

Prva i treća pod slika sadrže automobile. U prvoj je nebo sivkaste boje, dok je u trećoj modre. Naizgled, na trećoj slici prevladavaju povoljni vremenski uvjeti u pogledu pada li u tom trenutku kiša ili ne. Na prvoj pod slici malo je teže golim okom odrediti pada li kiša ili ne, jer se na prvu kapljice kiše ne vide, ali je nebo sivo što upućuje na veliku mogućnost kiše. Kada gledamo gradijente ulaza te slike, vidimo kako su najveće vrijednosti prisutne na dijelu slike koji prikazuje žutu kuću, a ona definitivno ne bi trebala odlučivati o vremenskim uvjetima. U drugoj pod slici, prisutne su kišne kapi ili nekakve mrlje u obliku kapljica u gornjem lijevom kutu slike, a cesta je pritom prazna. Takve primjere praznih cesta najčešće su bili prisutni u prvoj klasi, klasi povoljnih vremenskih uvjeta. Gradijenti najvećih iznosa nalaze se u središnjem dijelu slike, na cesti. A gradijenti ulaza u gornjem lijevom kutu slike koji sadrži kapljice ili mrlje oblika kapljica su vrlo malog iznosa.

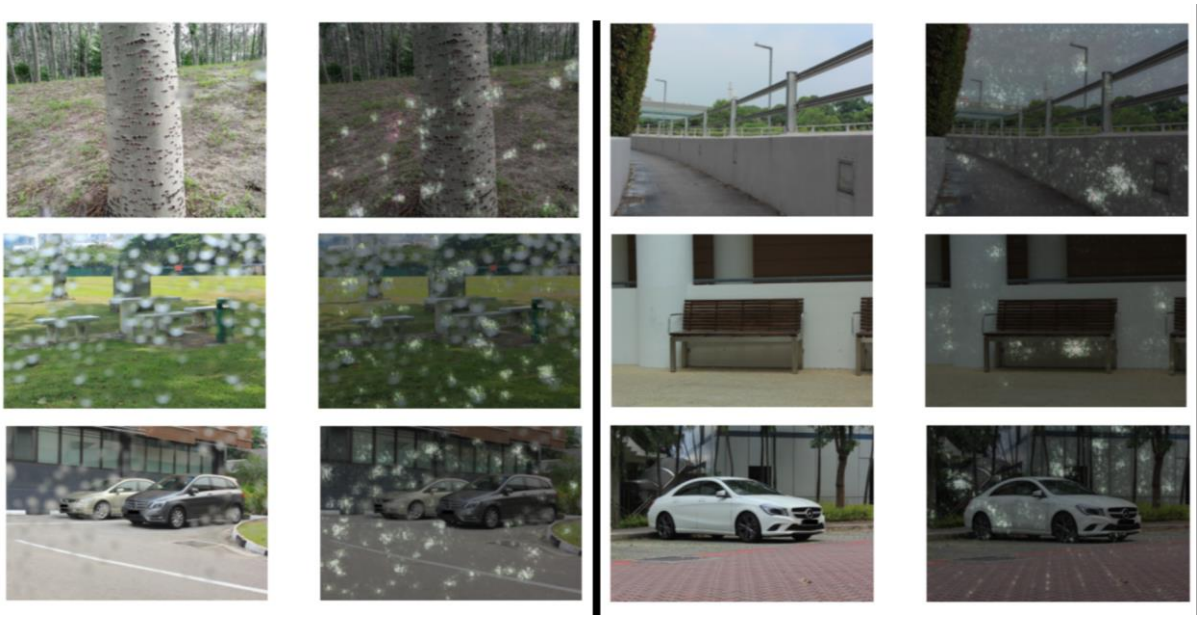
Slika 7.12 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najteže primjere otežanih vremenskih uvjeta.

Slijedeća slika, 7.13, prikazuje slike koje su označene da prikazuju povoljne vremenske uvjete, a model ih je označio da prikazuju nepovoljne. Većinom su to slike na kojima je nebo tmurno i cesta mokra. Kao i na velikoj većini prikazanih slika skupa RT-Raindrop u ovom poglavlju, gradijenti ulaza najprisutniji su na sredini i donjoj polovici slike. Iz toga zaključujemo da model veći fokus stavlja na cestu, a ne na nebo.

Slika 7.13 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najteže primjere koji prikazuju povoljne vremenske uvjete bez kiše.

S obzirom da je model treniran s RT-Raindrop skupom pokazao vrlo dobre performanse na skupo DeRaindrop, prikazat ćemo nekoliko mreži najlakših i najtežih primjeraka iz skupa DeRaindrop.

Na slici 7.14, lijevo su prikazani najlakši primjerci klase koja označava kišne uvjete, dok su desno prikazani primjerci bez kiše s pripadajućim gradijentima.



Slika 7.14 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najlakše primjere sa kišnim kapima (lijevo) i bez kišnih kapi (desno) iz skupa DeRaindrop.

Promatrajući primjerke slika koje sadrže kišne kapi, iznenađujuće je vidjeti kako se gradijenti ulaza lijepe baš na područja gdje se nalaze kapi. No, i na ovim primjercima, najviše na prvoj i drugoj pod slici s lijeve strane, primjećujemo kako su gradijenti ulaza iznosom najjači na donjoj polovici slike.

Na idućoj slici, s lijeve strane prikazani su najteži primjerci klase koja označava kišne uvjete, dok su desno prikazani primjerci bez kiše s pripadajućim gradijentima. Primjerci lijeve klase razlikuju se od primjeraka iste klase prikazanih na gornjoj slici 7.14 u tome što su prisutne kapljice kiše značajno manjih dimenzija. Na prvoj pod slici se golim okom one niti ne vide.

Slika 7.15 Vizualizacija ulazne slike i njenog gradijenta ulaza za mreži najteže primjere sa kišnim kapima (lijevo) i bez kišnih kapi (desno) iz skupa DeRaindrop.

Već smo prije rekli kako su skupovi DeRaindrop i RT-Raindrop vizualno vrlo različiti te im se domene isto razlikuju. Tu potvrdu smo dobili evaluirajući model, koji je treniran na skupu DeRaindrop, na skupu podataka RT-Raindrop. No, u obrnutoj situaciji model treniran RT-Raindrop skupom, a evaluiran na DeRaindrop-u se dobro snašao u klasifikaciji primjeraka. To nam govori kako je skup podataka DeRaindrop specijaliziran za precizniju i užu podjelu podataka prema jednom

kriteriju, a on je prisutnost detekcija prisutnosti kapljica kiše. S druge strane, evaluacijom modela treniranog na skupu RT-Raindrop, možemo zaključiti kako je proces donošenja odluke mnogo kompleksniji i samo vizualizacijom gradijenata ulaza, u ovom trenutku, ga ne možemo odrediti. Također, možemo zaključiti kako postoji određeno preklapanje primjeraka skupa DeRaindrop s onima iz RT-Raindrop-a, da je DeRaindrop podskup skupa podataka RT-Raindrop.

8. Budući rad

Nastavak na ovaj projekt mogao bi se odnositi na dodatno podešavanje hiperparametara modela treniranog na skupu podataka RT-Raindrop kako bi se dobili još bolji rezultati.

Kao moguća nadogradnja i potencijalno rješenje što boljoj generalizaciji vremenskih uvjeta bilo bi konstruirati i istrenirati model koji uči na oba skupa podataka. Kako je skup RT-Raindrop značajno veći od DeRaindrop-a, predlažem kako bi se za treniranje nasumično uzeo podskup RT-Raindrop skupa kako bi se omjer količine podataka među skupovima smanjio.

S obzirom na temu ovog rada koja se bavi klasifikacijom kapljica kiše između scene i kamere, trebalo bi se dovesti u pitanje valjanosti skupa podataka RT-Raindrop u sklopu obrađene teme. Kako bi se prikupile stvarne slike kapljica između scene i kamere, jedna opcija je ručno probrati slike iz RT-Raindrop skupa i ručno ih klasificirati na temelju prisutnosti kišnih kapljica. Druga opcija bila bi pokušati naći drugi skup podataka dobiven iz stvarne uporabe koji po domeni više uklapa zadanom problemu.

Zaključak

Kapljice kiše na automobilskom staklu ozbiljno ometaju vidljivost tijekom vožnje povećavajući vjerojatnost nesreća. Stoga je bitno na vrijeme prepoznati kapljice kiše na staklu auta kako bi senzor dalje mogao uključiti brisače i druge potrebne akcije.

Konvolucijska neuronska mreža poput ResNet-a i na ovom problemu iskazala kao vrlo dobar izbor. Postignuti su rezultati od 88,15% točnosti na testnom setu skupa DeRaindrop i 82,13% na skupu RT-Raindrop. Potvrdila se pretpostavka da se smanjenjem koraka u prvom sloju sažimanja mreže povećava važnost detalja. Također, uvođenjem dilatacije koja čuva semantiku ImageNet inicijalizaciju težina ne dovodi do poboljšanja učinkovitosti modela. Vizualno promatrajući skupove podataka DeRaindrop i RT-Raindrop primijetili da su vrlo različiti te da se dijelovi logike podjele skupa DeRaindrop pojavljuju se i u skupu RT-Raindrop. Te potvrde dobili smo i evaluacijama modela koji su trenirane na jednom, a evaluirani na drugom skupu. Dok model treniran na DeRaindrop-u za sve primjerke testnog seta RT-Raindrop smatra da primjerci sadrže kišne kapi, RT-Raindrop model se mnogo bolje nosi s predviđanjem klase primjeraka iz skupa DeRaindrop.

Potencijalno rješenje u povećanju generalizacijskih sposobnosti među skupovima bilo bi trenirati model na oba skupa. Pritom bi bilo dobro paziti na omjere u broju podataka između skupova kako jedan ne bi nadvladao nad drugim.

Vizualizirani su gradijenti ulaznih podataka kako bi se dodatno prikazalo ponašanje finalnih modela. U modelu treniranom na skupu DeRaindrop, gradijenti najvećeg iznosa hvatali su se za piksele na kojima su bile kapljice kiše. Dakle, one su najviše doprinijele donošenju odluke. Prilikom analize najtežih slika, uočeno je da se mreža teško nosi sa slikama s vrlo malim brojem kapljica kao i velikim kapljicama bez definiranog ruba. Gradijenti ulaza modela treniranog na skupu RT-Raindrop svoju odluku ne donose isključivo na temelju kapljica. Oni su se pretežito hvatali za cestu, okolni promet i arhitekturu.

Literatura

[1] R. Qian, R. T. Tan, W. Yang, J. Su, J. Liu, „Attentive Generative Adversarial Network for Raindrop Removal from a Single Image“,

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,“ IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[3] Stephen O'Grady, „The RedMonk Programming Language Rankings: June 2021“ https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/ (Pristupljeno 6.6.2022.)

[4] Python,“ What is Python? Executive Summary“, https://www.python.org/doc/essays/blurb/ (Pristupljeno 6.6.2022)

[5] Coursera, „What Is Python Used For? A Beginner’s Guide“ https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

[6] A.Paszke, „PyTorch: An Imperative Style, High-Performance Deep Learning Library“, https://arxiv.org/pdf/1912.01703.pdf

[7] Simonyan, K., Vedaldi, A., and Zisserman, A., *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*, arXiv e-prints, 2013. Poveznica: https://arxiv.org/abs/1312.6034

[8] M.N.Venugopal „An AI based raindrop detection system for Autonomous Driving“, EDAG

[9] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. MIT Press, 2016., str. 326-341. Poveznica: http://www.deeplearningbook.org/

[10] Lin, H.; Shi, Z.; Zou, Z. Maritime *Semantic Labeling of Optical Remote Sensing Images with Multi-Scale Fully Convolutional Network*. Remote Sens. 2017, 9, 480. Poveznica: https://doi.org/10.3390/rs9050480

[11] CUI, X., ZHENG, K., GAO, L., ZHANG, B., YANG, D. and REN, J. 2019. *Multiscale spatial-spectral convolutional network with image-based framework for hyperspectral imagery classification*. Remote sensing [online], 11(19), article 2220. Poveznica: https://doi.org/10.3390/rs11192220

[12] A. Kathuria, *Intro to optimization in deep learning: Gradient Descent*. 2018. Poveznica: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[13] A.Kumar, Overfitting & Underfitting in Machine Learning, Dana Analytics, 2022. Poveznica: https://vitalflux.com/overfitting-underfitting-concepts-interview-questions/

[14] Kingma, D. P. and Ba, J., *Adam: A Method for Stochastic Optimization*, arXiv e-prints, 2014. Poveznica: https://arxiv.org/abs/1412.6980

[15] D. Rafaeli, *Cross Entropy*,The Maverick Meerkat, 2020. Poveznica: https://themaverickmeerkat.com/2020-11-15-Cross-Entropy/

[16] Abadi, M., *TensorFlow: A system for large-scale machine learning*, arXiv e-prints, 2016. Poveznica: https://arxiv.org/abs/1605.08695

Privitak

U ovom poglavlju nalaze se tablice koje prikazuju arhitekturu modela ResNet-18 sloj po sloj s pripadajućim iznosom receptivnog polja u pojedinom sloju. Veličina receptivnog polja računala se po sljedećoj formuli:

$$rp_i = rp_{i-1} + (k_i - 1) * d_i * f_{i-1}, \qquad rp_0 = 1$$

$$f_i = f_{i-1} * s_i, \qquad f_0 = 1$$

$rp_i - receptivno\ polje\ u\ sloja\ i$

$k_i - veličina\ jezgre\ u\ poeraciji\ konvolucije\ ili\ sažimanja\ u\ sloju\ i$

$d_i - faktor\ dilatacije\ u\ opercaiji\ konvolucije$

$s_i - korak\ u\ poeraciji\ konvolucije\ ili\ sažimanja\ u\ sloju\ i$

$f_i - faktor\ širenja\ receptivnog\ polja$

Također, treba napomenuti kako je maksimalna moguća veličina receptivnog polja upravo veličina/rezolucija ulazne slike. Ne može biti veće od čitave slike. Tablice u nastavku prikazuju receptivno polje neovisno o rezoluciji ulazne slike, tako da je primjenjivo na slikama različitih rezolucija. Na mjestima gdje je u tablici napisano da je receptivno polje veće od rezolucije ulazne slike, čitatelj ima mogućnost zamijeniti taj broj s potpunom rezolucijom slike. Generalno, receptivno polje se nikada ne može umanjiti. Jednom kada mreža vidi cijelo receptivno polje ulaznog podatka u nastavku uvijek ima receptivno polje jednako potpunoj rezoluciji ulazne slike.

Kako je slika 2D podatak, odnosno matrica s dvije dimenzije, u svim tablicama ispod receptivno polje računato je samo za jednu dimenziju, druga se računa na isti način pritom vodeći računa o rezoluciji druge dimenzije.

Tablica 0.1 Originalna arhitektura modela ResNet-18 s prikazanim iznosom receptivnog polja po slojevima.

Broj sloja	**Operacija**	**Jezgra**	**Korak**	**Dilatacija**	**Nadopunjavanje**	**Receptivno polje**	**f**
0	Ulazna slika					1	1
1	conv	7	2	1	3	7	2
2	BN	-	-	-	-	7	2
3	relu	-	-	-	-	7	2
4	pool	3	2	-	1	11	4
5	conv	3	1	1	1	19	4
6	BN	-	-	-	-	19	4
7	relu	-	-	-	-	19	4
8	conv	3	1	1	1	27	4
9	BN	-			-	27	4
10	conv	3	1	1	1	35	4
11	BN	-	-	-	-	35	4
12	relu	-	-	-	-	35	4
13	conv	3	1	1	1	43	4
14	BN	-	-	-	-	43	4
15	conv	3	2	1	1	51	8
16	BN	-	-	-	-	51	8
17	relu	-	-	-	-	51	8
18	conv	3	1	1	1	67	8
19	BN	-	-	-	-	67	8
20	conv	1	2	1	0	67	16
21	BN	-	-	-	-	67	16
22	Conv	3	1	1	1	99	16
23	BN	-	-	-	-	99	16
24	relu	-	-	-	-	99	16
25	Conv	3	1	1	1	131	16
26	BN	-	-	-	-	131	16
27	Conv	3	2	1	1	163	32
28	BN	-	-	-	-	163	32
29	relu	-	-	-	-	163	32
30	Conv	3	1	1	1	227	32
31	BN	-			-	227	32
32	Conv	1	2	1	0	227	64
33	BN	-	-	-	-	227	64
34	Conv	3	1	1	1	355	64
35	BN	-	-	-	-	355	64
36	relu	-			-	355	64
37	Conv	3	1	1x1	1	483	64
38	BN	-	-	-	-	483	64
39	Conv	3	2	1	1	611	128
40	BN	-	-	-	-	611	128
41	relu	-	-	-	-	611	128
42	Conv	3	1	1	1	867	128
43	BN	-			-	867	128
44	Conv	1	2	1	0	867	256
45	BN	-			-	867	256
46	Conv	3	1	1	1	1379	256
47	BN	-	-	-	-	1379	256
48	relu	-	-	-	-	1379	256
49	Conv	3	1	1	1	1891	256
50	BN	-	-	-	-	1891	256
51	fc	-	-	-	-	dimenzija ulazne slike	

Tablica 0.2 Arhitektura modela ResNet-18 sa smanjenim korakom u prvom sloju sažimanja (promjena u odnosu na originalnu arhitekturu označena sivo u tablici) s prikazanim iznosom receptivnog polja po slojevima

Broj sloja	**Operacija**	**Jezgra**	**Korak**	**Dilatacija**	**Nadopunjavanje**	**Receptivno polje**	**f**
0	Ulazna slika					1	1
1	conv	7	2	1	3	7	2
2	BN	-	-	-	-	7	2
3	relu	-	-	-	-	7	2
4	pool	3	1	-	1	11	2
5	conv	3	1	1	1	15	2
6	BN	-	-	-	-	15	2
7	relu	-	-	-	-	15	2
8	conv	3	1	1	1	19	2
9	BN	-			-	19	2
10	conv	3	1	1	1	23	2
11	BN	-	-	-	-	23	2
12	relu	-	-	-	-	23	2
13	conv	3	1	1	1	27	2
14	BN	-	-	-	-	27	2
15	conv	3	2	1	1	31	4
16	BN	-	-	-	-	31	4
17	relu	-	-	-	-	31	4
18	conv	3	1	1	1	39	4
19	BN	-	-	-	-	39	4
20	conv	1	2	1	0	39	8
21	BN	-	-	-	-	39	8
22	Conv	3	1	1	1	55	8
23	BN	-	-	-	-	55	8
24	relu	-	-	-	-	55	8
25	Conv	3	1	1	1	71	8
26	BN	-	-	-	-	71	8
27	Conv	3	2	1	1	87	16
28	BN	-	-	-	-	87	16
29	relu	-	-	-	-	87	16
30	Conv	3	1	1	1	119	16
31	BN	-			-	119	16
32	Conv	1	2	1	0	119	32
33	BN	-	-	-	-	119	32
34	Conv	3	1	1	1	183	32
35	BN	-	-	-	-	183	32
36	relu	-			-	183	32
37	Conv	3	1	1x1	1	247	32
38	BN	-	-	-	-	247	32
39	Conv	3	2	1	1	311	64
40	BN	-	-	-	-	311	64
41	relu	-	-	-	-	311	64
42	Conv	3	1	1	1	439	64
43	BN	-			-	439	64
44	Conv	1	2	1	0	439	128
45	BN	-			-	439	128
46	Conv	3	1	1	1	695	128
47	BN	-	-	-	-	695	128
48	relu	-	-	-	-	695	128
49	Conv	3	1	1	1	951	128
50	BN	-	-	-	-	951	128
51	fc	-	-	-	-	dimenzija ulazne slike	

Tablica 0.3 Arhitektura modela ResNet-18 s faktorom dilatacije=2 u konvolucijama rezidualnih blokova (promjena označena sivo u tablici) s prikazanim iznosom receptivnog polja po slojevima.

Broj sloja	**Operacija**	**Jezgra**	**Korak**	**Dilatacija**	**Nadopunjavanje**	**Receptivno polje**	**f**
0	Ulazna slika					1	1
1	conv	7	2	2	3	1	2
2	BN	-	-	-	-	7	2
3	relu	-	-	-	-	7	2
4	pool	3	2	-	1	7	4
5	conv	3	1	2	1	11	4
6	BN	-	-	-	-	27	4
7	relu	-	-	-	-	27	4
8	conv	3	1	2	1	27	4
9	BN	-			-	43	4
10	conv	3	1	2	1	43	4
11	BN	-	-	-	-	59	4
12	relu	-	-	-	-	59	4
13	conv	3	1	2	1	59	4
14	BN	-	-	-	-	75	4
15	conv	3	2	2	1	75	8
16	BN	-	-	-	-	91	8
17	relu	-	-	-	-	91	8
18	conv	3	1	2	1	91	8
19	BN	-	-	-	-	123	8
20	conv	1	2	1	0	123	16
21	BN	-	-	-	-	123	16
22	Conv	3	1	2	1	123	16
23	BN	-	-	-	-	187	16
24	relu	-	-	-	-	187	16
25	Conv	3	1	2	1	187	16
26	BN	-	-	-	-	251	16
27	Conv	3	2	2	1	251	32
28	BN	-	-	-	-	315	32
29	relu	-	-	-	-	315	32
30	Conv	3	1	2	1	315	32
31	BN	-			-	443	32
32	Conv	1	2	1	0	443	64
33	BN	-	-	-	-	443	64
34	Conv	3	1	2	1	443	64
35	BN	-	-	-	-	699	64
36	relu	-			-	699	64
37	Conv	3	1	2	1	699	64
38	BN	-	-	-	-	955	64
39	Conv	3	2	2	1	955	128
40	BN	-	-	-	-	1211	128
41	relu	-	-	-	-	1211	128
42	Conv	3	1	2	1	1211	128
43	BN	-			-	1723	128
44	Conv	1	2	1	0	1723	256
45	BN	-			-	1723	256
46	Conv	3	1	2	1	1723	256
47	BN	-	-	-	-	2747	256
48	relu	-	-	-	-	2747	256
49	Conv	3	1	2	1	2747	256
50	BN	-	-	-	-	3771	256
51	fc	-	-	-	-	dimenzija ulazne slike	

Tablica 0.42 Arhitektura modela ResNet-18 s faktorom dilatacije=2 u konvolucijama rezidualnih blokova i korakom prvog sloja sažimanja=1 (promjena označena sivo u tablici) s prikazanim iznosom receptivnog polja po slojevima.

Broj sloja	**Operacija**	**Jezgra**	**Korak**	**Dilatacija**	**Nadopunjavanje**	**Receptivno polje**	**f**
0	Ulazna slika					1	1
1	conv	7	2	2	3	7	2
2	BN	-	-	-	-	7	2
3	relu	-	-	-	-	7	2
4	pool	3	1	-	1	11	2
5	conv	3	1	2	1	19	2
6	BN	-	-	-	-	19	2
7	relu	-	-	-	-	19	2
8	conv	3	1	2	1	27	2
9	BN	-			-	27	2
10	conv	3	1	2	1	35	2
11	BN	-	-	-	-	35	2
12	relu	-	-	-	-	35	2
13	conv	3	1	2	1	43	2
14	BN	-	-	-	-	43	2
15	conv	3	2	2	1	51	4
16	BN	-	-	-	-	51	4
17	relu	-	-	-	-	51	4
18	conv	3	1	2	1	67	4
19	BN	-	-	-	-	67	4
20	conv	1	2	1	0	67	8
21	BN	-	-	-	-	67	8
22	Conv	3	1	2	1	99	8
23	BN	-	-	-	-	99	8
24	relu	-	-	-	-	99	8
25	Conv	3	1	2	1	131	8
26	BN	-	-	-	-	131	8
27	Conv	3	2	2	1	163	16
28	BN	-	-	-	-	163	16
29	relu	-	-	-	-	163	16
30	Conv	3	1	2	1	227	16
31	BN	-			-	227	16
32	Conv	1	2	1	0	227	32
33	BN	-	-	-	-	227	32
34	Conv	3	1	2	1	355	32
35	BN	-	-	-	-	355	32
36	relu	-			-	355	32
37	Conv	3	1	2	1	483	32
38	BN	-	-	-	-	483	32
39	Conv	3	2	2	1	611	64
40	BN	-	-	-	-	611	64
41	relu	-	-	-	-	611	64
42	Conv	3	1	2	1	867	64
43	BN	-			-	867	64
44	Conv	1	2	1	0	867	128
45	BN	-			-	867	128
46	Conv	3	1	2	1	1379	128
47	BN	-	-	-	-	1379	128
48	relu	-	-	-	-	1379	128
49	Conv	3	1	2	1	1891	128
50	BN	-	-	-	-	1891	128
51	fc	-	-	-	-	dimenzija ulazne slike	

Sažetak

Raspoznavanje slika važno je područje računalnog vida s mnogim zanimljivim primjenama. Jedna od takvih primjena je i detekcija smanjene vidljivosti uslijed kapljica na zaštitnom staklu kamere. Ovaj rad proučava rješavanje tog problema prikladno oblikovanim konvolucijskim modelima, točnije korištenjem modela s ResNet-18 arhitekturom. Raznim postupcima, poput učenja na isječcima, uvođenjem dilatirane konvolucije te smanjenjem koraka u prvom sloju sažimanja, pokušava se dobiti što bolja sposobnost generalizacije. Fokus je stavljen na prepoznavanja sitnih detalja slike i očuvanje originalne ImageNet semantike mreže. U sklopu rada korištena su dva skupa podataka od kojih su oblikovani podskupovi za učenje, validaciju i testiranje. Kreiran je klasifikacijski algoritam strojnog učenja te su vrednovani naučeni modeli i prikazane su njihove metrike kao i vizualizirana logika zaključivanja modela pomoću gradijenata ulaza.

Ključne riječi: detekcija kapljica, dilatirana konvolucija, ResNet-18, vizualizacija mapa istaknutih područja

Summary

Image recognition is an important area of computer vision with many interesting applications such as the detection of reduced visibility due to drops on the protective glass of the camera. This paper studies the solution to this problem with appropriately designed convolutional models, i.e. using ResNet-18 models. The best possible model was deducted by a combination of various procedures, such as learning on clips, introducing dilated convolution and reducing stride in the first pooling layer. The focus was on recognizing the fine details of the image and preserving the original ImageNet semantics of the network. Subsets for learning, validation and testing were formed from two data sets which were used in the paper. The classification algorithm of machine learning was created and the learned models were evaluated, along with their metrics, as well as the visualized logic of the model inference using input gradients to do so.

Keywords: raindrop detection, dilated convolution, ResNet-18, saliency maps visualization