SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 485

Detekcija kapljica kiše na zaštitnom staklu između scene i kamere

Josip Srzić

Zagreb, srpanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 485

Pristupnik:	**Josip Srzić (0036522082)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Detekcija kapljica kiše na zaštitnom staklu između scene i kamere**

Opis zadatka:

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. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje tenzorima i slikama. Proučiti i ukratko opisati postojeće konvolucijske arhitekture. Odabrati slobodno dostupni skup slika te oblikovati podskupove za učenje, validaciju i testiranje. Oblikovati klasifikacijsku arhitekturu te uhodati učenje, validiranje i zaključivanje. Vrednovati naučene modele te prikazati postignutu točnost. Komentirati memorijsku i računsku učinkovitost modela. Predložiti pravce za budući rad. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 10. lipnja 2022.

Zahvaljujem mentoru prof. dr. sc. Siniši Šegviću na strpljenju, ohrabrenjima i savjetima. Zahvaljujem svojoj obitelji i bližnjima na neprestanoj podršci.

SADRŽAJ

1. Uvod

Smanjena vidljivost ceste i okolnog područja za vrijeme padanja kiše je stvaran problem s kojim se susreću brojni vozači automobila. Jedan od načina ublažavanja tog problema je korištenje pametnih sustava za detekciju kapljica kiše na vjetrobranskom staklu uz pomoć kamere, koji mogu pravovremeno intervenirati paljenjem brisača ili nekim drugim akcijama. Navedena problematika mogla bi se proširiti i na ostale domene kao što je detekcija kiše na nadzornim kamerama pametnih domova i brojne druge.

Nagli razvoj računalnog vida, osobito u posljednjem desetljeću, omogućio je strojevima da obavljaju zadatke klasifikacije slika gotovo dobro kao i ljudi. Pri tome zapažen uspjeh postiže jedan poseban razred umjetnih neuronskih mreža koji se naziva konvolucijske neuronske mreže. Danas je prisutan veliki broj stručnjaka i tvrtki koji ulažu puno vremena i resursa kako bi razvili konvolucijske arhitekture koje bi imale što bolje moguće performanse. Zahvaljujući konceptu prenesenog učenja (engl. *transfer learning*) i internetu, svatko tko želi može preuzeti prednaučeni model te ga dodatno prilagoditi i istrenirati na svom problemu. Također, postoje usluge poput Google Colaba i Kagglea koje omogućuju korisnicima da svoje modele treniraju na udaljenim poslužiteljima u oblaku koji su najčešće bolje opremljeni za izvođenje skupih računskih operacija od prosječnog osobnog računala. Iz svih tih razloga, rad na područjima strojnog i dubokog učenja te njihova praktična primjena dostupnija je nego ikad prije.

Ovaj rad je motiviran upravo tim činjenicama. Cilj je bio pronaći javno dostupan skup podataka koji se sastoji od slika koje sadrže kapljice kiše i slika bez njih te izgraditi klasifikator koji će za danu nepoznatu ulaznu sliku odrediti nalazi li se na njoj kiša ili ne. Rad nastoji reproducirati i nastaviti eksperimente iz [2]. Odabrana arhitektura za ovaj zadatak je obitelj dubokih konvolucijskih modela s rezidualnim vezama (engl. *ResNet*), koji su se dosad pokazali jako uspješnim na raznim klasifikacijskim zadacima.

U poglavljima 2 i 3 opisani su neki osnovni pojmovi iz strojnog i dubokog učenja koji će pomoći lakšem praćenju daljnjeg teksta. U poglavlju 4 opisani su detalji

korištene arhitekture. U poglavlju 5 je predstavljen korišteni razvojni okvir PyTorch te programsko ostvarenje. U poglavljima 6 i 7 opisan je korišteni skup podataka te rezultati eksperimenata nad njima. U poglavlju 8 se iznosi zaključak rada.

2. Strojno učenje

U današnje doba ljudi raspolažu s ogromnim količinama podataka. Jedna od glavnih zadaća strojnog učenja je na neki način pretočiti te podatke u znanje. Formalno, strojno učenje jest programiranje računala na način da optimiziraju neki kriterij uspješnosti temeljem podatkovnih primjera ili prethodnog iskustva [1]. Svaki algoritam strojnog učenja sastoji se od tri komponente:

- model,
- funkcija gubitka,
- optimizacijski postupak.

Premda postoje popularna rješenja za svaku od ovih komponenata, njihov odabir prilikom implementacije uvijek treba obaviti s obzirom na problem o kojem se radi. To se posebno odnosi na odabir modela.

Ulaz u algoritam strojnog učenja nazivamo primjer. Svaki primjer okarakteriziran je nizom značajki, odnosno skupom nekakvih atributa na temelju kojih se izvode zaključci. Izlaz algoritma strojnog učenja ovisi o vrsti strojnog učenja koju promatramo. Postoje tri glavne paradigme:

- **nadzirano učenje (engl. *supervised learning*):** učenje na parovima koji se sastoje od značajki i odgovarajućih oznaka, bavi se problemima poput klasifikacije i regresije.

- **nenadzirano učenje (engl. *unsupervised learning*):** učenje na neoznačenim podacima, a cilj je u njima pronaći neke pravilnosti, bavi se problemima grupiranja, procjene gustoće, smanjenja dimenzionalnosti.

- **podržano učenje (engl. *reinforcement learning*):** izrada inteligentnih agenata koji interagiraju sa svojom okolinom putem akcija te dobivaju neku vrstu nagrade (pozitivnu ili negativnu) koju nastoje maksimizirati, jako široko područje primjene

Ovaj rad se bavi klasifikacijom slika, stoga se u nastavku daje malo širi opis nadziranog učenja.

2.1. Nadzirano učenje

U nadziranom učenju svaki ulazni primjer ima pripadajuću oznaku (engl. *label*). Primjeri se najčešće slažu u matricu koja se naziva **matrica dizajna**. Svaki redak predstavlja jedan primjer, a svaki stupac jednu značajku (indeks primjera je u superskriptu, a indeks značajke u supskriptu):

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_n^{(N)} \end{bmatrix}$$

Oznake primjera se slažu u vektor (svaki redak sadrži oznaku koja pripada primjeru s istim indeksom retka u matrici $\mathbf{X}$):

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

Kod klasifikacije vrijednost oznake y predstavlja indeks razreda kojem pripada odgovarajući primjer. U slučaju regresije najčešće vrijedi $y \in \mathbb{R}$, odnosno y je neki realni broj.

Proces nadziranog učenja odvija se u dvije faze:

- **treniranje:** Model prima označene podatke na ulazu te prilagođava svoje parametre kako bi izlaz modela za svaki primjer bio što bliže oznaci za taj primjer.

- **predikcija:** Na ulaz modela dovode se primjeri koje dosad nije vidio. Model na temelju svojih parametara računa izlaz koji predstavlja predikciju za svaki primjer. Izlaz modela se uspoređuje s pripadnim oznakama primjera kako bi se odredila uspješnost predikcije. Kod klasifikacije za najčešću mjeru uspješnosti predikcije uzima se točnost (engl. *accuracy*), odnosno udio ispravno klasificiranih primjera u ukupnom broju primjera.

Formalnije, cilj nadziranog učenja je što preciznije aproksimirati funkciju h koja se naziva **hipoteza**

$$h : X \to Y \tag{2.1}$$

koja element iz prostora primjera preslikava u element iz prostora oznaka (indeks razreda kod klasifikacije ili brojčana vrijednost kod regresije).

Važno svojstvo dobrog modela u nadziranom učenju je **generalizacija**. To je sposobnost modela da radi točne predikcije na dosad ne viđenim primjerima. Do slabe generalizacije može doći ako se model previše prilagodi podacima za treniranje. Model tada postiže veliku točnost na skupu za treniranje, ali ne uspijeva dovoljno dobro aproksimirati funkciju h za sve elemente domene. To je ujedno glavni problem strojnog učenja, a naziva se **prenaučenost**. Do prenaučenosti često dolazi ako koristimo model koji je presložen s obzirom na problem za koji se koristi, pogotovo ako raspolažemo s relativno malim skupom podataka.

Kako bismo prepoznali i umanjili učinak prenaučenosti, korisna je metoda **unakrsne provjere**. Ukupni skup podataka dijeli se na: skup za učenje, skup za evaluaciju i skup za testiranje. Nakon treniranja na skupu za učenje, model obavlja predikciju nad skupom za evaluaciju (pritom ne ažurirajući svoje parametre) te prikazuje uspješnost. Taj postupak ponavljamo više puta za razne kombinacije modela i/ili hiperparametara[1]. Cilj je pronaći model optimalne složenosti. Nakon što odaberemo najbolji model iz skupa kandidata, provjeravamo i prikazujemo njegovu uspješnost na skupu za testiranje.

Još jedan način na koji možemo umanjiti prenaučenost je primjenom raznih metoda **regularizacije**. Regularizacija je skup metoda koje umanjuju prenaučenost dodavanjem dodatne informacije u algoritam s namjerom smanjenja složenosti. Njihova karakteristika je da pritom ne smanjuju empirijsku grešku (grešku na skupu za učenje) te da se mogu primijeniti na sve dijelove algoritma strojnog učenja: model, funkciju gubitka, optimizacijsku metodu pa čak i na same podatke korištenjem raznih augmentacija.

[1] parametri koji se ne uče već ručno odabiru, oni određuju način učenja parametara modela

2.2. Model

Model u algoritmu strojnog učenja se definira kao skup hipoteza zadanih izrazom 2.1 te parametriziranih s $\boldsymbol{\theta}$.

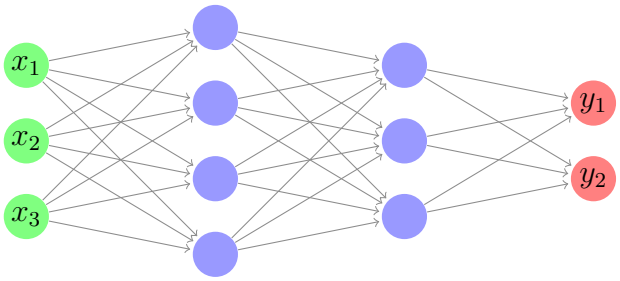
$$H = \{h(\mathbf{x}; \boldsymbol{\theta})\}_{\boldsymbol{\theta}} \tag{2.2}$$

Svaki ovako definiran model je izravno određen svojim skupom parametara, tj. skup parametara $\boldsymbol{\theta}$ se izravno preslikava u hipotezu h. Ovakva definicija omogućava da na učenje modela gledamo kao na potragu za najboljom funkcijom iz skupa funkcija H, odnosno kao na optimizacijski problem [18].

2.2.1. Umjetne neuronske mreže

Jedan popularan model u algoritmima strojnog učenja u današnje doba su umjetne neuronske mreže. Umjetna neuronska mreža je skup jednostavnih međusobno povezanih procesnih elemenata (neurona) koji služe distribuiranoj paralelnoj obradi podataka [3]. Glavna inspiracija načinu rada umjetnih neuronskih mreža dolazi iz neurona u biološkom mozgu. Umjetni neuroni primaju, obrađuju i odašilju signale dalje u mrežu analogno komunikaciji bioloških neurona putem sinaptičkih veza. S vremenom se način korištenja i organizacije umjetnih neurona značajno udaljio od analogije sa živčanim sustavom, ali nam i dalje može dati dobru intuiciju. Umjetne neuronske mreže su glavni predstavnik konektivističkog pristupa u području umjetne inteligencije. Jedna od posljedica toga je da često nećemo moći objasniti *zašto* je model donio određenu predikciju, ali zato možemo rješavati neke probleme koji su se pokazali presloženima za simbolistički pristup.

Svaka umjetna neuronska mreža ima svoju arhitekturu. Neuroni se u tipičnim arhitekturama slažu u slojeve. Sloj koji prima podatke iz okoline naziva se ulazni sloj (engl. *input layer*). Sloj koji određuje konačni izlaz mreže naziva se izlazni sloj (engl. *output layer*). Između njih može se nalaziti nula ili više skrivenih slojeva (engl. *hidden layer*). U najčešćem slučaju, ulazi svaki neuron i-tog sloja su pobude iz *svakog* neurona $(i-1)$-tog sloja. Takve slojeve (u širem smislu i mreže) nazivamo potpuno povezani (engl. *fully connected*) slojevi. Ako još uzmemo za pretpostavku da u mreži ne postoje povratne ni cikličke veze, tada pričamo o **unaprijednim slojevitim potpuno povezanim neuronskim mrežama** (engl. *feedforward multi-layered fully connected neural network*). Jedan primjer takve mreže prikazan je na slici 2.1.



Slika 2.1: Arhitektura jedne unaprijedne slojevite potpuno povezane neuronske mreže s 2 skrivena sloja. Arhitekturu mreže možemo sažeto iskazati kao 3x4x3x2.

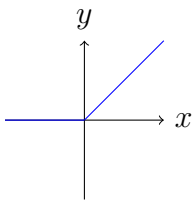
Pobuda i-tog neurona u l-tom sloju računa se prema formuli:

$$z_i^{(l)} = \mathbf{w}^{(l-1)} \cdot \mathbf{a}^{(l-1)} + b_i^{(l)} \tag{2.3}$$

gdje $\mathbf{w}^{(l-1)}$ označava vektor težina veza koje slijede $(l-1)$-ti sloj, $\mathbf{a}^{(l-1)}$ vektor *aktivacija* neurona u $(l-1)$-tom sloju, a $b_i^{(l)}$ prag (engl. *bias*) neurona. Kada bismo ovako definirane pobude izravno prenosili kroz mrežu, nepovoljno bismo se ograničili na samo linearne ovisnosti izlaza mreže o ulaznim podacima. To proizlazi iz činjenice da bi se na izlazu neuronske mreže našla kompozicija linearnih funkcija, za koju znamo da je onda također linearna funkcija. Da bi povećali ekspresivnost naših modela, od ključne su važnosti **prijenosne funkcije**:

$$a_i^{(l)} = \varphi(z_i^{(l)}) \tag{2.4}$$

Pomoću njih u model dodajemo nelinearnosti koje nam omogućuju izgradnju znatno kompleksnijih funkcija. Neki primjeri često korištenih prijenosnih funkcija su: funkcija skoka (engl. *step function*), sigmoidalna funkcija, tangens hiperbolni. Prijenosna funkcija koja se empirijski pokazala najbolja jest **zglobnica** (engl. *Rectified Linear Unit - ReLU*) prikazana na slici 2.2. Neke od njenih prednosti su smanjenje vjerojatnosti pojave problema umirućeg gradijenta (engl. *vanishing gradient*) i smanjenje vremenske složenosti prilikom učenja zbog jednostavnog izračuna vrijednosti funkcije te gradijenta.



Slika 2.2: Prijenosna funkcija ReLU definirana s $f(x) = \max(0, x)$

2.3. Funkcija pogreške

Funkcija pogreške je numerička procjena uspješnosti našeg modela na označenom skupu podataka, odnosno udaljenosti između očekivanog i stvarnog izlaza iz modela. Cilj algoritma strojnog učenja je pronaći parametre modela koji će minimizirati funkciju pogreške na skupovima za učenje i provjeru. Koriste se različite funkcije pogreške u ovisnosti o tome bavimo li se klasifikacijom ili regresijom. U klasifikacijskim problemima popularan izbor je unakrsna entropija, a kod regresije srednje kvadratno odstupanje/srednja kvadratna pogreška.

2.3.1. Unakrsna entropija (engl. *Cross-Entropy Loss*)

Kod klasifikacije česta je praksa da se izlazni sloj neuronske mreže provlači kroz $softmax$ funkciju

$$softmax(j, \hat{\boldsymbol{y}}) = \frac{e^{\hat{y}_j}}{\sum_k e^{\hat{y}_k}} \tag{2.5}$$

gdje je j indeks nekog razreda, a $\hat{\mathbf{y}}$ vektor koji sadrži aktivacije neurona zadnjeg sloja. Softmax nam omogućava da izlaz iz mreže promatramo kao vjerojatnosnu distribuciju.

Unakrsna entropija je funkcija koja mjeri udaljenost između dvije vjerojatnosne distribucije. Ako za jednu distribuciju uzmemo izlaz funkcije $softmax$, a za drugu vektor koji sadrži 1 na indeksu razreda kojem pripada primjer i 0 na ostalim (engl. *one-hot encoded vector*), onda nam unakrsna entropija može dati dobru procjenu koliko je naš model daleko od ispravne predikcije. Ako to napravimo za sve primjere iz skupa ili jedne grupe (engl. *mini-batch*), izraz za ukupnu pogrešku glasi

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=0}^{N}\sum_{k=0}^{K} y_{ik} \ln \hat{y}_{ik} \tag{2.6}$$

gdje je N broj ulaznih primjera u model, a K broj razreda.

2.3.2. Srednja kvadratna pogreška (engl. *Mean Squared Error*)

Srednja kvadratna pogreška je podrazumijevana funkcija pogreške kad se radi o regresijskim problemima. Ona mjeri srednju vrijednost kvadratne udaljenosti između stvarne i predviđene vrijednosti izlaza.

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2 \tag{2.7}$$

Učinak kvadrata je da će veće udaljenosti izlaza i predikcije više pridonositi ukupnoj pogrešci nego male udaljenosti, odnosno model se dodatno kažnjava ako radi jako krive procjene.

2.4. Optimizacijski postupak

Optimizacijski postupak je algoritam koji nastoji aproksimirati optimalan skup parametara modela, odnosno pronaći parametre za koje će funkcija pogreške modela biti minimalna. Većina optimizacijskih postupaka se oslanja na računanje gradijenata funkcije gubitka po svakom parametru. Jedna od prednosti modela baziranih na umjetnim neuronskim mrežama je uporaba vrlo jednostavnih operacija (zbrajanje, množenje) te jednostavno derivabilnih prijenosnih funkcija (npr. ReLU) koje zauzvrat čine računanje gradijenta puno jednostavnijim. Usprkos tome, današnji duboki modeli znaju imati i po više milijuna parametara, stoga zadatak optimizacijskog postupka nije nimalo trivijalan i zahtjeva efikasnu izvedbu. Tu je od velike pomoći algoritam **unazadne propagacije (engl. *backpropagation*)**. On radi unazadni prolaz nad grafom računskih operacija, pritom rekurzivno primjenjujući lančano pravilo kako bi efikasno izračunao gradijent svakog parametra u mreži. Gledano s visoke razine, tipični koraci prilikom obrade jednog (ili mini-grupe) primjera kod algoritma strojnog učenja baziranog na umjetnim neuronskim mrežama jesu:

1. unaprijedni prolaz (engl. *forward pass*): primjer ulazi u neuronsku mrežu, računaju se i propagiraju aktivacije neurona od ranijih prema dubljim slojevima sve do izlaza, pritom se međuvrijednosti često spremaju (engl. *cache*) za kasnije korištenje prilikom unazadne propagacije

2. unazadni prolaz: računaju se gradijenti počevši od izlaza mreže pa sve do ulaza

3. ažuriranje parametara: izračunati gradijenti koriste se kako bi se na neki način poboljšali parametri mreže, sami izračun novih parametara ovisi o odabiru optimizatora

2.4.1. Gradijentni spust (engl. *Gradient descent*)

Gradijentni spust je optimizacijski postupak koji nastoji iterativno poboljšavati parametre modela i tako smanjiti vrijednost funkcije gubitka. Ovisno o veličini ulaznog skupa nakon kojeg se obavlja ažuriranje parametara, postoje tri vrste gradijentnog spusta:

- **stohastički gradijentni spust (engl. *Stochastic Gradient Descent*)** — obrađuje se jedan po jedan ulazni primjer, vrijednost funkcije gubitka nije stabilna već sadrži sitne fluktuacije, računski zahtjevno jer se nad svakim primjerom treba obaviti puno operacija i ne iskorištava se vektorizacija
- **gradijentni spust nad grupom (engl. *Batch Gradient Descent*)** — suprotan ekstrem u odnosu na SGD, parametri se ažuriraju tek nakon prolaska kroz cijeli skup podataka (jedne epohe), postoji opasnost od zapinjanja u lokalnom minimumu, nije vrlo memorijski učinkovit
- **gradijenti spust nad mini-grupom (engl. *Mini-batch Gradient Descent*)** — najčešće korišteni oblik gradijentnog spusta, parametri se ažuriraju nakon prolaska kroz svakih m primjera, gdje je $1 < m < N$ neki hiperparametar, svojevrsni kompromis između prethodna dva pristupa, učinkovito iskorištava memoriju zbog vektorizacije i dobro konvergira ka globalnom minimumu funkcije gubitka

Neovisno o varijanti gradijentnog spusta, ažuriranje težina modela računa se prema

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \nabla_{\boldsymbol{W}} \mathcal{J}(\boldsymbol{W}, \boldsymbol{b}), \tag{2.8}$$

a ažuriranje pragova prema

$$\boldsymbol{b} \leftarrow \boldsymbol{b} - \eta \nabla_{\boldsymbol{b}} \mathcal{J}(\boldsymbol{W}, \boldsymbol{b}). \tag{2.9}$$

Hiperparametar η naziva se stopa učenja (engl. *learning rate*). Njegov odabir ima veliki utjecaj na uspješnost gradijentnog spusta [6]. Ako je stopa učenja premala, učenje će trajati predugo i postoji rizik zapinjanja u lokalnom minimumu funkcije gubitka. Ako je prevelika, proces učenja postaje nestabilan. Tipična vrijednost koja se uzima kao početna je 0.001.

Proširenje ovog algoritma koje može znatno ubrzati učenje jest **gradijenti spust sa zaletom** (engl. *Gradient Descent with Momentum*). Umjesto običnog gradijenta kao u osnovnom slučaju, ovdje se koristi eksponencijalno umanjujući prosjek prethodnih

gradijenata koji se računa prema:

$$V_{dW} \leftarrow \beta V_{dW} + (1-\beta)\nabla_{\boldsymbol{W}}\mathcal{J}(\boldsymbol{W},\boldsymbol{b}) \tag{2.10}$$

$$V_{db} \leftarrow \beta V_{db} + (1-\beta)\nabla_{\boldsymbol{b}}\mathcal{J}(\boldsymbol{W},\boldsymbol{b}) \tag{2.11}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta V_{dW} \tag{2.12}$$

$$\boldsymbol{b} \leftarrow \boldsymbol{b} - \eta V_{db} \tag{2.13}$$

Parametar β određuje utjecaj prethodno akumuliranog prosjeka. Što je veći, to će prethodni prosjek imati veći utjecaj na vrijednost V_{dW}. Posljedica zaleta je smanjenje oscilacije gradijenta u smjerovima koji ne vode do globalnog minimuma te povećanje iznosa "skokova" smjeru globalnog minimuma.

2.4.2. RMSProp (engl. *Root Mean Square Propagation*)

Ideja iza ovog optimizacijskog postupka je slična kao kod gradijentnog spusta sa zaletom. Cilj je stabilizirati učenje i bolje ga usmjeriti prema minimumu.

$$S_{dW} \leftarrow \beta S_{dW} + (1-\beta)(\nabla_{\boldsymbol{W}}\mathcal{J}(\boldsymbol{W},\boldsymbol{b}))^2 \tag{2.14}$$

$$S_{db} \leftarrow \beta S_{db} + (1-\beta)(\nabla_{\boldsymbol{b}}\mathcal{J}(\boldsymbol{W},\boldsymbol{b}))^2 \tag{2.15}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \frac{\eta}{\sqrt{S_{dW}}}\nabla_{\boldsymbol{W}}\mathcal{J}(\boldsymbol{W},\boldsymbol{b}) \tag{2.16}$$

$$\boldsymbol{b} \leftarrow \boldsymbol{b} - \frac{\eta}{\sqrt{S_{db}}}\nabla_{\boldsymbol{b}}\mathcal{J}(\boldsymbol{W},\boldsymbol{b}) \tag{2.17}$$

Ako prosjek kvadrata gradijenata nekog parametra $S_{d\theta}$ postane jako velik, dijeljenje stope učenja s korijenom od $S_{d\theta}$ osigurat će stabilizaciju gradijenta.

2.4.3. Adam (engl. *Adaptive Moment Estimation*)

Adam kombinira AdaGrad i RMSProp optimizacijske postupke. Empirijski rezultati pokazuju da Adam u mnogim slučajeva postiže bolje rezultate od stohastičkog gradijentnog spusta [8]. Kao i RMSProp, Adam zasebno skalira stope učenja za različite parametre. Razlika je što pri samom ažuriranju parametara ne koristi obične gradijente već eksponencijalno umanjujući prosjek prethodnih gradijenata kao kod gradijentnog spusta sa zaletom.

3. Konvolucijske neuronske mreže

Duboki konvolucijski neuronski modeli postali su *de facto* standard u području računalnog vida. Uobičajene unaprijedne potpuno povezane neuronske mreže nisu povoljne za obradu slika jer ne uzimaju u obzir da se isti objekt može pojaviti na više pozicija u različitim slikama, već promatraju svaki ulazni piksel pojedinačno. Takav bi model trebao učiti odvojeno svaku moguću poziciju pojave predmeta na slici, što nije izvedivo. Prednost konvolucijskih modela je što izlazi ovise o lokalnom susjedstvu piksela ulaza. Također, potpuno povezane mreže bi za RGB slike velikih dimenzija imale previše parametara što bi znatno otežalo treniranje mreže. Konvolucijski modeli taj problem rješavaju dijeljenjem parametara, odnosno korištenjem istih parametara na različitim mjestima u slici. Manji broj parametara također omogućuje izgradnju mreža s mnogo slojeva bez da treniranje postane presporo. Većina konvolucijskih modela su ekvivarijantni na pomake [9]. To znači da će translacija ulaza u model u pravilu translatirati mapiranja značajki modela za jednak iznos, što je svakako povoljno svojstvo u radu sa slikama. Postoji nekoliko osnovnih gradivnih blokova u konvolucijskim neuronskim mrežama koji su prikazani u nastavku.

3.1. Konvolucijski sloj

Konvolucijski sloj se sastoji od skupa **jezgri** (engl. *kernels, filters*) odnosno višedimenzionalnih tenzora. Tipično se sastoje od tri dimenzije: broj kanala, visina, širina. Prostorne dimenzije su obično jednake i malog iznosa (3×3, 5×5, 7×7). Elementi matrice su parametri modela koji mijenjaju svoju vrijednost napredovanjem optimizacijskog algoritma. Njihova zadaća je da "kližu" po slici i izvlače razne značajke (engl. *feature extraction*). Ako pričamo o arhitekturama s više konvolucijskih slojeva, raniji slojevi će najčešće izvlačiti neke jednostavne značajke poput horizontalnih ili vertikalnih rubova. U dubljim slojevima možemo očekivati da će model "hvatati" složenije oblike poput kontura automobila ili ljudskog lica. Način na koji jezgre izvlače značajke je koristeći operaciju konvolucije. U strojnom učenju se često pod konvo-

lucija zapravo misli unakrsna korelacija. Premda su dvije operacije jako slične, nisu ekvivalentne. Kod unakrsne korelacije elementi jezgre i slike na istim pozicijama se množe, te se svi takvi umnošci zbrajaju (kao skalarni umnožak nad matricama).

$$S(i,j) = (K \circledast I)(i,j) = \sum_{m=m_{min}}^{m=m_{max}} \sum_{n=n_{min}}^{n=n_{max}} K(m,n) \cdot I(i+m, j+n) \tag{3.1}$$

Ako jezgru zarotiramo za 180° u smjeru kazaljke na satu te zatim nad slikom i jezgrom provedemo unakrsnu korelaciju, takva operacija naziva se konvolucija. U daljnjem tekstu pod konvolucija zapravo će se misliti na unakrsnu korelaciju definiranu izrazom 3.1.

Svaka jezgra sloja obavlja konvoluciju neovisno o ostalim jezgrama te stvara vlastitu mapu značajki (engl. *activation map, channel*). Svakom elementu mape značajki dodaje se još prag te se provlači kroz prijenosnu funkciju (kao kod običnih unaprijednih potpuno povezanih mreža). Sve mape značajki nekog sloja se slažu zajedno te tvore vrijednosti ulaza za sljedeći sloj. Na taj način konvolucijski slojevi dobivaju još jednu dimenziju koja se naziva dubina, a označava broj mapa značajki u tom sloju. Stoga je često lakše ne zamišljati konvolucijski sloj kao dvodimenzonalnu matricu brojeva, nego kao volumen s dvije prostorne dimenzije (visina $\times$ širina) i dubinom.

Korak (engl. *stride*) je broj koji određuje za koliko se piksela pomiče jezgra nakon svake konvolucije. Što je korak veći, to će prostorne dimenzije izlaza biti manje. Održavanje iste prostorne dimenzije je moguće ako proširimo ulaznu matricu tako da se gornji lijevi element jezgre može naći na bilo kojem indeksu ulazne matrice po kojoj "kliže". Obično se ti nepostojeći elementi nastali proširivanjem popunjavaju nulama (engl. *zero padding*). Primjer diskretne dvodimenzionalne konvolucije prikazan je na slici 3.1.

$$\begin{pmatrix} 3 & 2 & 1 & 4 \\ 2 & 5 & 2 & 3 \\ 1 & 1 & 4 & 2 \\ 0 & 5 & 3 & 3 \end{pmatrix} \circledast \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 \\ 6 & -3 \end{pmatrix}$$

Slika 3.1: Primjer konvolucije ulazne matrice 4×4 s jezgrom 3×3 i korakom 1, bez nadopunjavanja. Rezultat operacije (mapa značajki) je matrica dimenzija 2×2. Formalno, radi se o operaciji unakrsne korelacije.

3.2. Sloj sažimanja

Za razliku od konvolucijskog, sloj sažimanja (engl. *pooling layer*) ne sadrži nikakve parametre modela. Ovaj sloj se sastoji od jedne jezgre malih dimenzija, najčešće 2×2, koja ne sadrži vrijednosti već obavlja određenu operaciju sažimanja. Uobičajene operacije su: sažimanje maksimalnom vrijednošću, srednjom vrijednošću, L2 normom. Zadaća sažimanja je grupiranje prostorno bliskih značajki na ulazu u jednu vrijednost izlaza. Operacije se u daljnjim slojevima provode nad sažetim značajkama umjesto nad preciznim pozicijama značajki. To čini model robusnijim na varijacije pozicija na kojima se značajke pojavljuju na slici. Pored toga, sažimanje smanjuje prostornu rezoluciju reprezentacije i tako povećava receptivno polje kasnijih konvolucija. Primjer sažimanja je dan na slici 3.2.

$$maxpool_{(2\times2)} \begin{pmatrix} 3 & 2 & 4 & 0 \\ 5 & 1 & 3 & 1 \\ 3 & 2 & 3 & 2 \\ 4 & 4 & 5 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix}$$

Slika 3.2: Primjer sažimanja maksimalnom vrijednošću ulazne matrice 4×4 s jezgrom 2×2 i korakom 2, bez nadopunjavanja. Rezultat operacije je sažeta mapa značajki dimenzija 2×2.

Sažimanje se provodi nad svakom ulaznom mapom značajki zasebno. Najčešće se koristi jezgra dimenzija 2×2 s korakom 2, što će za učinak imati smanjenje prostornih dimenzija mape značajki za faktor 2. U zadnjim slojevima konvolucijskog modela često se koristi globalno sažimanje. Ono radi slično kao i lokalno sažimanje, ali je razlika u tome što se ne gledaju isječci već cijela mapa značajki koja se sažima u jednu vrijednost. Na taj se način najčešće obavlja tranzicija s konvolucijskih na potpuno povezane slojeve na kraju mreže.

3.3. Regularizacija

U poglavlju o nadziranom učenju uveli smo pojam regularizacije kao skup metoda koje smanjuju kapacitet modela da bi se izbjegla prenaučenost. Duboke konvolucijske mreže opsežno koriste razne regularizacije u svojim arhitekturama.

3.3.1. Isključivanje neurona (engl. *dropout*)

Isključivanje neurona se najčešće primjenjuje na potpuno povezane slojeve koji se nalaze na izlazu konvolucijske neuronske mreže. U svakoj iteraciji učenja, svaki neuron u sloju se privremeno isključuje s vjerojatnošću $1 - p$ ili zadržava s vjerojatnošću p. Učinak isključivanja neurona je dodavanje šuma u postupak učenja. Time se želi umanjiti pojava situacija u kojima neuroni u dubljim slojevima ispravljaju greške ranijih slojeva, čime postanu preovisni o ranijim slojevima i zbog toga loše generaliziraju na neviđenim podacima[15]. Pokazalo se da ova regularizacija ne samo čini model robusnijim za nove podatke, već znatno ubrzava učenje.

3.3.2. L_1 i L_2 regularizacija

Jedan od pokazatelja prenaučenosti mogu biti veliki iznosi komponenata gradijenta težina. Intuitivno, ne želimo da se model jako mijenja za male promjene ulaza. L_1 i L_2 regularizacije zbog toga dodaju izraz ukupnoj funkciji gubitka koji penalizira velike norme gradijenata. L_1 uzima u zbroj apsolutnih vrijednosti svih komponenata gradijenta, a L_2 zbroj njihovih kvadrata. Ukupna funkcija gubitka s L_2 regularizacijom se računa prema izrazu

$$\mathcal{J}(\boldsymbol{W}^1, ..., \boldsymbol{W}^l, \boldsymbol{b}^1, ..., \boldsymbol{b}^l) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^{L} ||\boldsymbol{W}^l||^2, \tag{3.2}$$

a za L1 slično:

$$\mathcal{J}(\boldsymbol{W}^1, ..., \boldsymbol{W}^l, \boldsymbol{b}^1, ..., \boldsymbol{b}^l) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^{L} |\boldsymbol{W}^l|. \tag{3.3}$$

λ je hiperparametar koji određuje u kojoj mjeri želimo smanjiti iznose težina i time pojednostavniti model. U slučaju $\lambda = 0$, utjecaj regularizacije nestaje.

Ovakav tip regularizacije će pomoći ako model ustanovi da neka od značajki ne doprinosi donošenju predikcije izlaza jer će dodatno kažnjavati njen nepotrebno velik iznos. Ovaj učinak je vidljiviji kod L_1 regularizacije nego kod L_2 jer se iznos derivacije kvadratne funkcije smanjuje kako se bližimo 0, a derivacija apsolutne vrijednosti je konstantna. Posljedica toga je "razrjeđivanje" vektora težina, odnosno dosta komponenti vektora će postati 0. Time se potencijalno mogu uštedjeti računalni resursi. Međutim, L_2 regularizacija se znatno više koristi u praksi. U literaturi je još poznata pod imenom propadanje težina (engl. *weight decay*).

3.3.3. Umjetno uvećanje skupa podataka

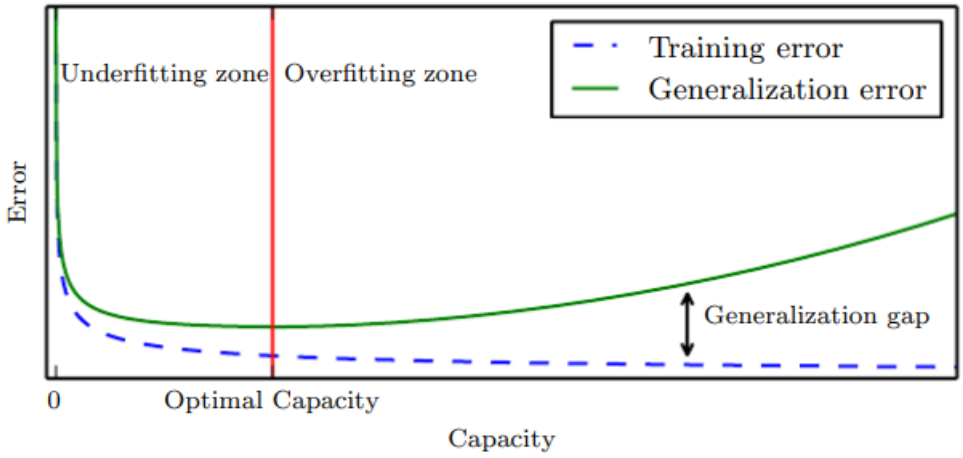
Jedan od provjerenih načina za smanjenje prenaučenosti je učenje na više podataka. Da bi model mogao dobro generalizirati, bitno je da podaci budu reprezentativni, odnosno da dobro pokrivaju prostor mogućih ulaza. Problem je što nije uvijek lako doći do novih podataka.

Uvećanje podataka možemo provoditi primjenom transformacija koje mogu "umjetno" generirati nove podatke. U praksi odabiremo transformacije koje ne mijenjaju semantiku same slike. Na primjer, ako sliku mačke zrcalimo horizontalno ili joj blago promijenimo kontrast, na slici je i dalje mačka. Neće sve augmentacije biti korisne za svaki problem, već s njima treba eksperimentirati. Neki primjeri augmentacija su: uzimanje isječaka slike (engl. *crop*), nadopunjavanje ruba slike, rotacije slike, glađenje (engl. *blur*), zrcaljenje i drugi.

3.3.4. Rano zaustavljanje (engl. *Early stopping*)

Do prenaučenosti može doći ako pustimo model da trenira predugo [16]. Postavlja se pitanje kako odrediti trenutak u kojem treniranje počinje štetiti generalizaciji modela.

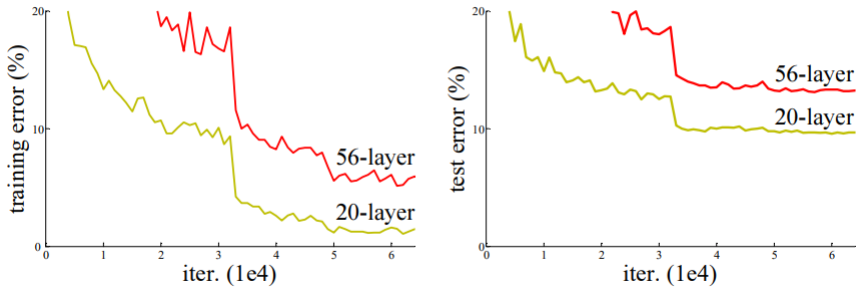
Rano zaustavljanje je jednostavno rješenje kod kojeg se prestaje s učenjem u trenutku kada se pogreška na skupu za evaluaciju krene pogoršavati. U najjednostavnijem obliku, prestaje se s učenjem u prvoj epohi koja postiže veću pogrešku na skupu za evaluaciju od prethodne. Međutim, opadanje funkcije pogreške na skupu za evaluaciju je rijetko monotono te najčešće sadrži puno malih skokova i padova zbog prisutnosti šuma [11]. Zbog toga se često koristi neki drugi uvjet zaustavljanja, poput zaustavljanja nakon nekoliko uzastopnih epoha s većom ili jednakom pogreškom, gledanja prosječne ili apsolutne promjene pogreške kroz nekoliko epoha i drugi. Umjesto pogreške na skupu za evaluaciju može se koristiti i neka druga metrika.



Slika 3.3: Slika prikazuje krivulje funkcije pogreške na skupu za učenje (plava boja) i skupu za evaluaciju (zelena boja). Crvenom crtom označen je trenutak u kojem se evaluacijska pogreška prestaje smanjivati. Daljnje učenje povećava evaluacijsku pogrešku i dovodi do prenaučenosti modela [6].

4. Duboki konvolucijski modeli s rezidualnim vezama (*ResNet*)

Što je reprezentacija modela dublja u slojevima, to je veći prostor parametara koji se može prilagoditi i naučiti za dani problem. Teorija nalaže da bi stoga bilo poželjno graditi modele proizvoljne dubine. U praksi se pokazalo da nakon određene dubine performanse modela počinju degradirati. Jedan od razloga je zloglasni problem umirućeg ili eksplodirajućeg gradijenta. Taj problem je potpuno riješen normaliziranjem ulaznih podataka te normaliziranjem izlaza skrivenih slojeva. Međutim, jako duboki modeli su se i dalje pokazali problematični. Neočekivano se pokazalo da glavni razlog degradacije nije prenaučenost, jer se točnost krene pogoršavati i na *skupu za učenje*.



Slika 4.1: Primjer degradacije modela s puno slojeva pokazan u [7] na skupu CIFAR-10. Korišteni su "klasični" duboki konvolucijski modeli bez rezidualnih veza. Mreža s 56 slojeva ima veću pogrešku na skupu za učenje kao i na skupu za testiranje.

Duboke konvolucijske mreže s rezidualnim vezama (ResNet) su obitelj arhitektura koje rješavaju navedeni fenomen uvođenjem rezidualnih veza ("prečica") u standardne konvolucijske modele [7]. Nastale su od strane Microsoftovog istraživačkog tima 2015. godine, kada su koristeći niz ovakvih mreža osvojili 1. mjesto na prestižnom natjecanju ILSVRC koje se sastoji od klasificiranja slika iz skupa ImageNet-1k. Od tada pa sve do danas, konvolucijske mreže s rezidualnim vezama predstavljaju stanje tehnike na

raznim klasifikacijskim zadacima.

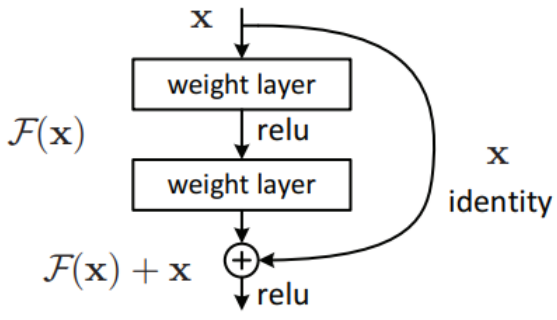
Specifičnost ovih modela je korištenje rezidualnih jedinica umjesto klasičnih slijedova konvolucijskih slojeva. Svaki **rezidualni blok** sastoji se od dvije **konvolucijske jedinice**. Klasična konvolucijska jedinica sastoji se od dva konvolucijska sloja, pri čemu se pobuda drugog sloja zbraja s ulazom prvog sloja (prije primjene prijenosne funkcije). Dakle, izlazna aktivacija rezidualne jedinice se računa prema

$$\boldsymbol{a}^{(l+2)} = \varphi(\boldsymbol{z}^{(l+2)} + \boldsymbol{a}^{(l)}) \tag{4.1}$$

$$\boldsymbol{z}^{(l+2)} = \boldsymbol{W}^{(l+2)}\boldsymbol{a}^{(l+1)} + \boldsymbol{b}^{(l+2)} \tag{4.2}$$

U prikazanoj jednadžbi natpis označava indeks sloja, $\boldsymbol{a}$ aktivaciju, $\boldsymbol{z}$ pobudu sloja prije primjene prijenosne funkcije, a φ prijenosnu funkciju.

Teorija je pokazala da je uvijek moguće izgraditi dublji model koji neće biti *lošiji* od plićeg. Možemo za primjer uzeti mreže sa slike 4.1. U tom bismo primjeru mogli duboki model s 56 slojeva zamijeniti modelom kojem su prvih 20 slojeva ekvivalentni plićem modelu, a preostalih 36 ostvaruju funkciju identiteta $\mathcal{F}(x) = x$, odnosno ulaze preslikavaju u identične izlaze. Postojanje takvog konstrukcijskog rješenja indicira da bi sloj u dubokom modelu u slučaju potrebe uvijek mogao prilagoditi svoje težine tako da ulaz sloja nepromijenjen dolazi do izlaza. Međutim, pokazalo se da većina dosadašnjih modela teško uče funkciju identiteta zbog složenosti optimizacijskog kriterija. S druge strane, glavna prednost rezidualnih jedinica je što **lagano uče funkciju identiteta**, ukaže li se za to potreba. Rezidualne veze zaglađuju funkciju gubitka i poboljšavaju tok gradijenata do ranih slojeva [10], čime se umanjuje problem umirućih gradijenata. Ilustracija jedne rezidualne jedinice je prikazana na slici 4.2.



Slika 4.2: Primjer rezidualne jedinice prikazan u radu [7]. Ulaz prvog sloja se zbraja s izlazom drugog sloja. Tako dobiveni zbroj prolazi kroz ReLU čime se dobiva konačna aktivacija.

4.1. Opis arhitekture ResNet-18

U nastavku se pobliže opisuje duboka arhitektura s rezidualnim jedinicama koja se sastoji od 18 slojeva:

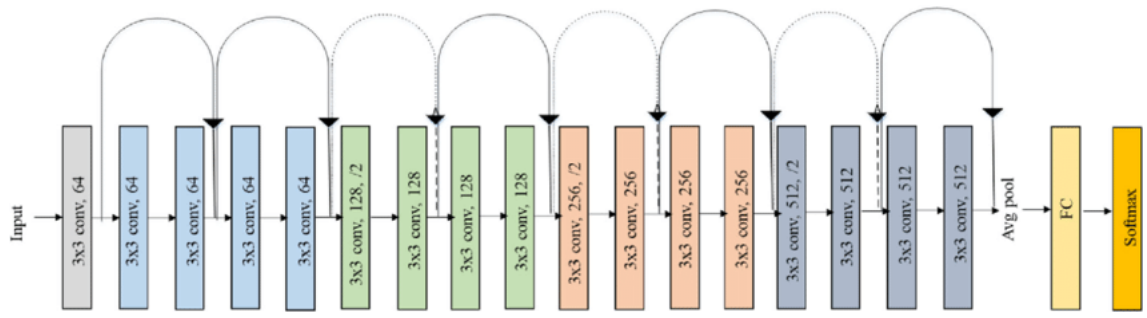
1. 2D konvolucija dubine 64 s jezgrom dimenzija 7×7, korakom 2 (u slučaju manjih slika često se stavlja korak 1) i nadopunjavanjem nulama 3×3, Normalizacija, ReLU
2. Sažimanje maksimalnom vrijednošću s jezgrom dimenzija 3×3 i korakom 2
3. Slijedni sloj koji se sastoji od 2 osnovne rezidualne jedinice
   (a) Prva rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **64** i korakom 1
   (b) Druga rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **64** i korakom 1
4. Slijedni sloj koji se sastoji od 2 osnovne rezidualne jedinice
   (a) Prva rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **128** i koracima 2 i 1, jedna 1×1 konvolucija s dubinom 128 i korakom 2 koja se provodi nad ulazom prve konvolucije (engl. *downsampling*)
   (b) Druga rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **128** i korakom 1
5. Slijedni sloj koji se sastoji od 2 osnovne rezidualne jedinice
   (a) Prva rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **256** i koracima 2 i 1, jedna 1×1 konvolucija s dubinom 256 i korakom 2 koja se provodi nad ulazom prve konvolucije (engl. *downsampling*)
   (b) Druga rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **256** i korakom 1
6. Slijedni sloj koji se sastoji od 2 osnovne rezidualne jedinice
   (a) Prva rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **512** i koracima 2 i 1, jedna 1×1 konvolucija s dubinom 512 i korakom 2 koja se provodi nad ulazom prve konvolucije (engl. *downsampling*)

(b) Druga rezidualna jedinica: dvije uzastopne 3×3 konvolucije s dubinom **512** i korakom 1

7. Globalno sažimanje srednjom vrijednošću

8. Potpuno povezani sloj (512×1000)

Nakon svake konvolucije se provodi normalizacija nad grupom (engl. *batch normalization*). U svakoj osnovnoj rezidualnoj jedinici izlaz prve konvolucije prolazi kroz prijenosnu funkciju zglobnicu (ReLU). Ulazi u prvu konvoluciju svake rezidualne jedinice se zbrajaju s izlazom druge konvolucije nakon čega se nad tim zbrojem primjenjuje zglobnica. Ulazi u rezidualne jedinice s dubinama 128, 256 i 512 prije zbrajanja prolaze kroz 1×1 konvoluciju s korakom 2 koja će im smanjiti prostorne dimenzije za pola (engl. *downsampling*) te udvostručiti broj mapa značajki. To se radi da bi se dimenzije ulaza u prvu konvoluciju poklopile s dimenzijama izlaza iz druge konvolucije i tako uspješno provelo zbrajanje. Takva intervencija nije potrebna u prvom rezidualnom bloku (s dubinom 64) jer se tamo u svakoj konvoluciji koristi korak 1. U svim 3×3 konvolucijama rezidualnih jedinica koristi se nadopunjavanje nulama.



Slika 4.3: Originalna ResNet-18 arhitektura [13].

5. Programska izvedba

Cijela programska implementacija napravljena je u programskom jeziku *Python* koristeći razvojni okvir *PyTorch*. Treniranje i testiranje modela provedeno je na grafičkoj kartici *Nvidia GeForce GTX 1650* koristeći platformu za paralelno izvršavanje naredbi na grafičkim procesnim jedinicama CUDA.

5.1. Razvojni okvir PyTorch

PyTorch je razvojni okvir otvorenog koda koji se koristi za strojno učenje. Bazira se na biblioteci *Torch* [5]. Nastao je 2016. godine pod vodstvom Facebooka (današnja META) kao reimplementacija spomenute biblioteke koja je bila napisana u programskom jeziku *Lua*. Dvije su glavne pogodnosti koje *PyTorch* nudi naspram obične *Python* implementacije algoritama strojnog učenja:

- optimizirane kalkulacije nad vektoriziranim podacima s podrškom za njihovo obavljanje na grafičkim procesnim jedinicama (GPU),
- automatska diferencijacija modela izraženih u *Pythonu*.

Tenzor (iz programskog paketa `torch.Tensor`) je osnovni objekt u *PyTorchu* i služi za pohranu višedimenzionalnih vektora i matrica. Strukturom i operacijama je vrlo sličan *poljima* iz popularne biblioteke *NumPy* s tim da tenzori dodatno imaju podršku za izvršavanje računskih operacija na Nvidijinim grafičkim karticama koje su podržane platformom CUDA te automatsko diferenciranje.

Automatska diferencijacija je iznimno koristan alat koji pomaže trenirati duboke modele bazirane na umjetnim neuronskim mrežama. Pri pozivu metode `backward` nad tenzorom koji predstavlja izračunati gubitak obavlja se unatražni obilazak prethodno izgrađenog računskog grafa te računaju gradijenti za svaki parametar mreže koristeći pravilo ulančavanja.

5.2. Implementacija

Cilj ovog rada bio je odabrati pouzdanu konvolucijsku arhitekturu i istrenirati je tako da s velikom pouzdanošću može klasificirati slike s kapljicama i bez kapljica.

Korištena je gotova implementacija arhitekture ResNet-18 (opisana u potpoglavlju 4.1.) iz paketa `torchvision.models`. Model je inicijaliziran prednaučenim težinama sa skupa ImageNet. Potpuno povezani sloj prednaučenog modela s 1000 izlaznih značajki zamijenjen je potpuno povezanim slojem s 2 izlazne značajke. Svi hiperparametri modela mogu se konfigurirati kroz ulazne argumente modula `main.py`.

Za učitavanje podataka korišten je razred `DataLoader` iz paketa `torch.utils.data`. On nudi sučelje za iteriranje po podacima. Pritom još pruža podršku za učitavanje mini-grupa, slučajno miješanje podataka te učitavanje pomoću više paralelnih procesa. Za učitavanje slika s diska bilo je još potrebno napraviti vlastitu implementaciju razreda `RaindropImageDataset` koji nasljeđuje razred `Dataset` iz `torch.utils.data`.

Korištena funkcija gubitka je unakrsna entropija iz razreda `torch.nn.CrossEntropyLoss`. Optimizatori su Adam i Stohastički gradijentni spust sa zaletom iz paketa `torch.optim`. Za postepeno smanjivanje stope učenja korišten je raspoređivač `lr_scheduler.StepLR` iz istog paketa.

Pretprocesiranje slike obavlja se prilikom učitavanja slike u metodi `__getitem__` razreda `RaindropImageDataset`. Sve korištene transformacije preuzeli smo iz paketa `torchvision.transforms`. Više transformacija se spaja u slijed pomoću objekta `transforms.Compose`.

Za vizualizaciju slika i grafova korištene su vanjske biblioteke `matplotlib` i `seaborn`. Ispis metrika je djelomično implementiran ručno, a za neke se koriste funkcije iz modula `sklearn.metrics`.

Glavna petlja nalazi se u modulu `main.py` u sklopu funkcije `train_and_eval`. Funkcija prolazi kroz zadani broj epoha te u svakoj obavlja fazu treniranja i fazu evaluacije. Osim augmentacije ulaznih podataka, kao još jedna metoda regularizacije implementirano je rano zaustavljanje opisano u potpoglavlju 3.4. Tijekom učenja prati se koji model postiže najbolju točnost na skupu za evaluaciju te se spremaju njegove težine (engl. *checkpointing*). Na kraju učenja težine najboljeg modela se spremaju u direktorij `models` u obliku datoteke s ekstenzijom `.pth`. Nakon što se odabere najbolji model, učitavaju se njegove težine iz `.pth` datoteke u objektnu reprezentaciju te provodi faza testiranja.

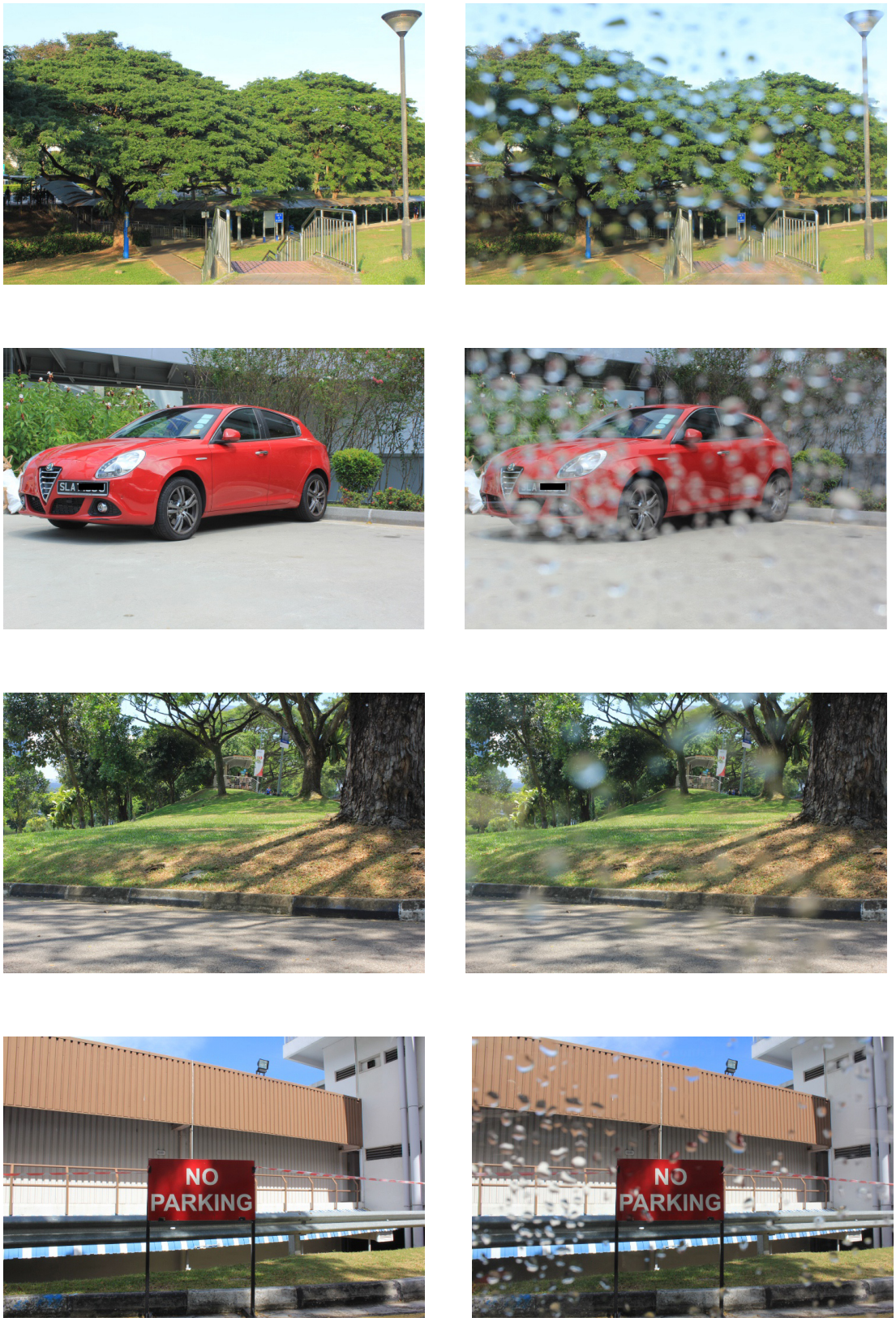
6. Skup podataka

Podaci korišteni u ovom radu dolaze iz rada [12] koji se bavi uklanjanjem kapljica kiše sa slika te restauracijom pozadine. Skup se sastoji od 1168 parova slika. Svaki par prikazuje pozadinu bez kapljica na zaštitnom staklu te istu pozadinu s kapljicama. Cijeli skup je preuzet s *GitHub* repozitorija [14] prethodno spomenutog rada. Slike su na početku bile podijeljenje u tri direktorija: *train*, *test_a*, *test_b*. Slike s kapljicama bile su smještene u poddirektorij *data*, a slike bez kapljica u poddirektorij *gt*. Najprije smo izravnali strukturu direktorija, odnosno sve slike s kapljicama i bez kapljica prebacili u zajednički direktorij i tako stvorili direktorije *train*, *val* i *test*. Predodređeni skup za validaciju (direktorij *test_a*) sadržavao je nedovoljan broj slika za kvalitetno provođenje unakrsne validacije, zbog čega smo 74 para slika s najvećim indeksima iz skupa za testiranje te 43 para slika s najvećim indeksima iz skupa za treniranje prebacili u direktorij *val*.

Dakle, za potrebe ovog rada skup je podijeljen na sljedeći način:

- skup za učenje - 818 slika s kišom, 818 slika bez kiše
- skup za evaluaciju - 175 slika s kišom, 175 slika bez kiše
- skup za testiranje - 175 slika s kišom, 175 slika bez kiše.

U nastavku su prikazani primjeri slika korištenih za treniranje:



Slika 6.1: Primjeri slika korištenih za treniranje modela. Slike dolaze u parovima. Svaki par sastoji se od slike bez kapljica i slike s kapljicama.

7. Eksperimenti

7.1. Odabir hiperparametara

Treniranje i evaluacija se provode kroz 30 epoha pri čemu se prekida s učenjem ako prođe 10 epoha bez napretka točnosti na validacijskom skupu (strpljenje modela je 10). Odabrana veličina mini-grupe je 16. Stopa učenja je 0.0005, a svako 7 epoha množi se faktorom 0.1. Parametar zaleta kod stohastičkog gradijentnog spusta je 0.9. Parametri Adama β_1 i β_2 su redom 0.9 i 0.999, a ϵ iznosi 10^{-8} (preporučeni parametri iz [8]).

7.2. Pretprocesiranje

U svakoj epohi svi ulazni podaci prolaze kroz korak pretprocesiranja prije ulaska u mrežu. Slike iz skupova za validaciju i testiranje uvijek koriste istu kompoziciju transformacija, dok su nad slikama za učenje isprobane dvije različite kompozicije.

Transformacije nad skupovima za validaciju i testiranje:

1. Skaliranje dimenzija slike na 480×720
2. Normalizacija

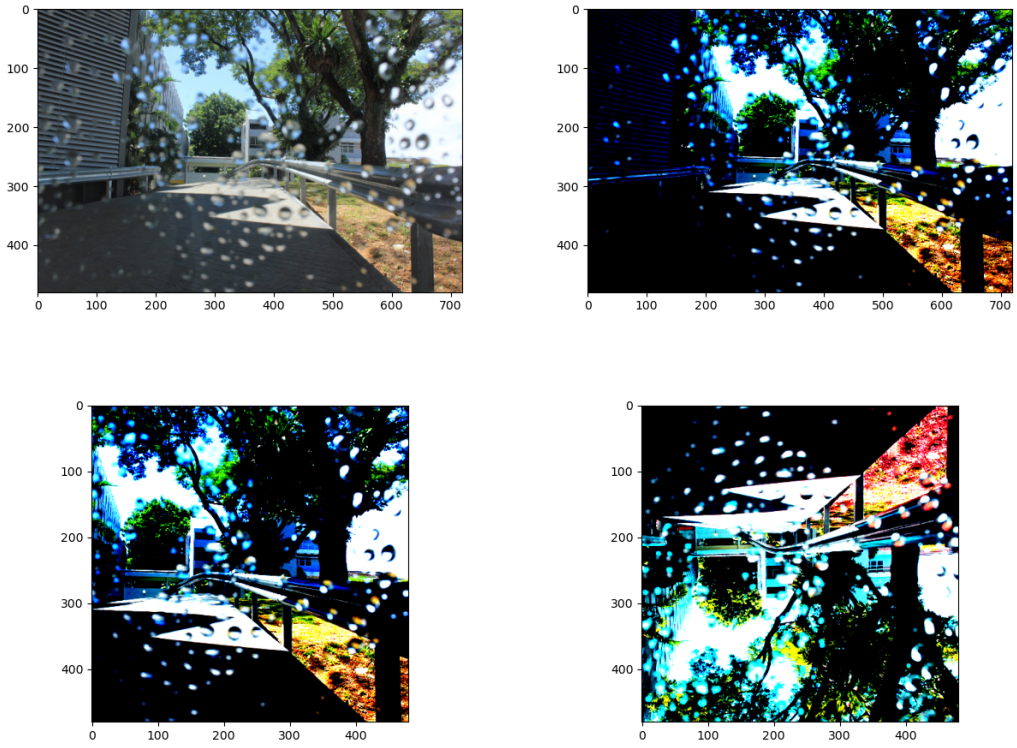
Transformacije nad skupom za učenje (1):

1. Skaliranje dimenzija slike na 480×720
2. Uzimanje isječka slike na nasumično odabranoj poziciji (engl. *Random Crop*)
3. Normalizacija

Transformacije nad skupom za učenje (2):

1. Skaliranje dimenzija slike na 480×720
2. Uzimanje isječka slike na nasumično odabranoj poziciji (engl. *Random Crop*)
3. Rastresanje boje slike (engl. *Color Jitter*)
4. Nasumično horizontalno zrcaljenje slike
5. Nasumično vertikalno zrcaljenje slike
6. Normalizacija

Normalizacija u svim transformacijama obavlja se s obzirom na srednje vrijednosti i standardne devijacije RGB kanala iz skupa slika ImageNet koji se koristio za inicijalizaciju težina modela. Srednje vrijednosti su $(0.485, 0.456, 0.406)$, a standardne devijacije $(0.229, 0.224, 0.225)$. Rastresanje boje mijenja svjetlinu slike nasumičnim faktorom iz intervala $[0.5, 1.5]$ te kontrast, nijansu i saturaciju nasumičnim faktorom iz intervala $[0.7, 1.3]$. Nasumična zrcaljenja obavljaju se s vjerojatnošću 0.5.



Slika 7.1: Primjer korištenja navedenih transformacija. Gore lijevo prikazana je originalna slika, gore desno prva opisana transformacija, dolje lijevo druga, a dolje desno treća.

7.3. Odabir modela

Cilj eksperimenta je pronaći model koji će najbolje ispravno klasificirati dosad neviđene slike u dva razreda: nema kapljica i ima kapljica. Korištena je metoda unakrsne provjere, a kriterij odabira modela jest točnost na validacijskom skupu.

Isprobano je učenje na isječcima slika različitih veličina. Ideja je natjerati model da klasifikacijsku odluku donosi na temelju manjih detalja na slici [2], kao što su kapljice. Ako su dimenzije isječka premale, javlja se rizik da će nasumični isječak biti odabran na "čistom" dijelu slike s kapljicama koji neće imati kapljica u lokalnom susjedstvu. Veličina isječka treba biti dovoljno velika tako da u velikoj većini slučajeva dobro reprezentira cijelu sliku. Zbog toga su isprobane različite dimenzije isječaka: 240×240, 360×360 i 480×480.

Osim mijenjanja veličine isječaka, isprobane su i različite augmentacije nad ulaznim podacima skupa za učenje (opisane u prethodnom potpoglavlju). Prije dodavanja augmentacija primijećeno je da model ima problema s prepoznavanjem kapljica na jako svijetlim pozadinama, stoga je u eksperimentu korišteno rastresanje boje pri čemu je dodatna težina stavljena na rastresanje svjetline slike. Kao dodatna metoda regularizacije, osim rastresanja boje obavlja se i nasumično zrcaljenje slike s obzirom na obje osi. Cilj augmentacija bio je na neki način odvojiti semantiku kapljica od semantike pozadine, odnosno natjerati model da prepoznaje kapljice neovisno o izgledu pozadine i njihovom položaju na slici.

Uspoređeni su i utjecaji različitih optimizatora na točnost raspoznavanja kapljica. Dio eksperimenata proveden je koristeći stohastički gradijentni spust sa zaletom, a dio koristeći optimizator Adam. Rezultati eksperimenata prikazani su u tablici 7.1.

Tablica 7.1: Rezultati eksperimenata. Oznaka "Ne" u četvrtom stupcu označava korištenje samo osnovnih transformacija na slikama skupa za učenje, a oznaka "Da" korištenje proširenih transformacija (augmentacije). Najbolji model na skupu za validaciju dobiven je u osmom treniranju.

Rbr	Veličina isječka	Optimizator	Augmentacije	train acc (%)	val acc (%)
1	240	SGD	Ne	94.86	93.12
2	240	SGD	Da	92.23	93.12
3	240	Adam	Ne	93.33	95.13
4	240	Adam	Da	96.57	96.85
5	360	SGD	Ne	97.31	92.84
6	360	SGD	Da	95.90	95.13
7	360	Adam	Ne	95.78	96.28
8	360	Adam	Da	97.61	97.71
9	480	SGD	Ne	98.17	92.84
10	480	SGD	Da	98.04	93.70
11	480	Adam	Ne	99.69	96.56
12	480	Adam	Da	98.78	97.42

7.4. Testiranje odabranog modela

Premda su modeli iz osmog i dvanaestog treniranja postigli slične rezultate na validacijskom skupu, osmi model je marginalno bolji zbog čega ga odabiremo kao konačni model. Dodatna je prednost što koristi manje isječke slike što ga čini bržim. U nastavku je prikazana uspješnost odabranog modela na skupu za testiranje.

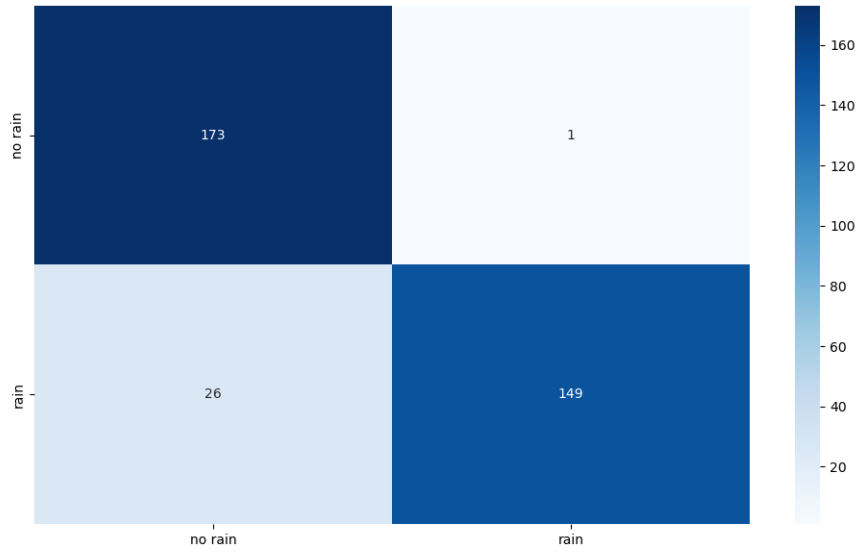
Tablica 7.2: Prikaz često korištenih metrika za odabrani model na skupu za testiranje.

Razred	Preciznost	Odziv	F1 vrijednost
nema kapljica	0.87	0.99	0.93
ima kapljica	0.99	0.85	0.92

Preciznost za prvi razred pokazuje udio slika za koje model ispravno predviđa da nemaju kapljice u odnosu na sve slike za koje predviđa da nemaju kapljice. Odziv pokazuje udio slika za koje model ispravno predviđa da nemaju kapljice u odnosu

na sve slike koje nemaju kapljice. F1 vrijednost je harmonijska sredina preciznosti i odziva. Na analogan se način računaju metrike i za razred "ima kapljica".

Točnost modela na skupu za testiranje iznosi **92.26%**.



Slika 7.2: Matrica konfuzije odabranog modela na skupu za testiranje.

7.5. Rasprava

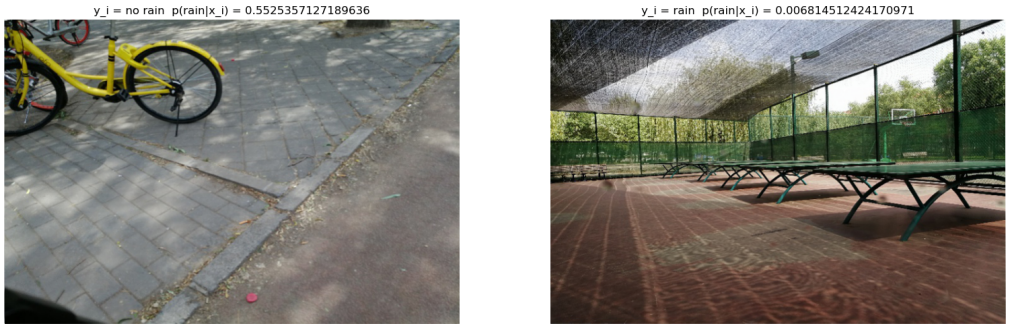
Rezultati prikazani u tablici 7.1 su u dobroj mjeri potvrdili očekivanja i hipoteze iznesene u potpoglavlju 7.3. Isječci dimenzija 240×240 pokazali su se premalim i uvodili šum odabirom nedovoljno reprezentativnih područja slike. Iako su isječci dimenzija 480×480 pokazali najbolju točnost na skupu za učenje, isječci dimenzija 360×360 su bolje generalizirali.

Adam se pokazao kao bolji izbor za optimizator od stohastičkog gradijentnog spusta sa zamahom. Iako postoje radovi [17] koji nastoje objasniti zašto SGD bolje generalizira od Adama, u ovom radu se pokazalo suprotno. SGD je u jako malo epoha pronalazio minimum funkcije gubitka na skupu za učenje, ali se loše snalazio s novim podacima. Kod Adama su obje krivulje učenja imale konstantni trend pada koji je u konačnosti doveo do dobre generalizacije. Točan razlog za to mogao bi se podrobnije istražiti u nekom drugom radu.

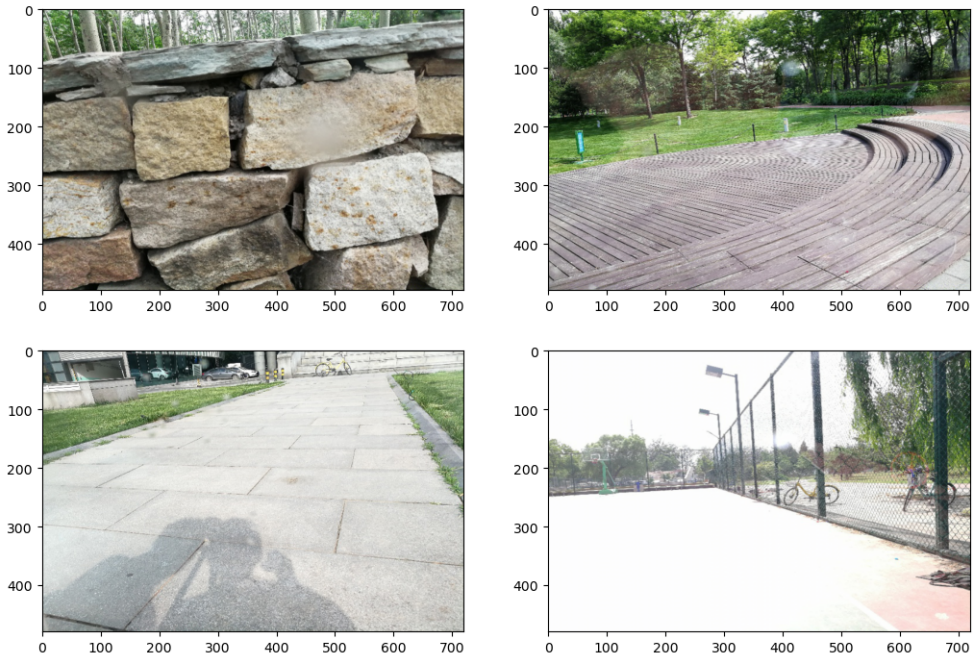
Rastresanje podataka korištenjem dodatnih augmentacija pokazalo se značajno u

smanjivanju prenaučenosti. Uz iste ostale parametre, rastresanje je donijelo uvećanje točnosti na validacijskom skupu za otprilike 1 do 3 postotna boda.

Odabrani model postigao je zadovoljavajuću točnost od 92.26% na skupu za testiranje. U uvodu je spomenuta motivacijska ideja izgradnje modela koji bi se mogao ugraditi u sustav za detekciju kiše u prometu i automatsko paljenje brisača. Metrike pokazane u tablici 7.2 idu u prilog toj ideji. Odziv na slikama bez kapljica iznosi 99%, što znači da ako je vani lijepo vrijeme i ne pada kiša, sustav će to skoro pa sigurno prepoznati. To je važno jer onda imamo garanciju da sustav neće ometati vozača slučajnim paljenjem brisača, što bi potencijalno bilo opasno. Slike na kojima model najviše griješi su one koje sadrže jako malo kiše. To u primjeni nije problem jer tada ni ne želimo da se sustav aktivira. Očekujemo da će sustav periodički provjeravati količinu kapljica na vjetrobranskom staklu te se sam aktivirati ako se na staklu akumulira značajna količina kapljica.



Slika 7.3: Najteže slike za model, odnosno slike na kojima je model najviše griješio. Na lijevoj slici model zbunjuju svijetle točke na tlu koje ga podsjećaju na kapljice iako ih nema na slici. Na desnoj slici ima kapljica, ali ih je malo i slabo su vidljive zbog pozadine.



Slika 7.4: Primjeri ostalih slika s kojima se model mučio odnosno krivo ih klasificirao. Radi se o slikama s jako malo kapljica kiše koje se najčešće ne vide zbog izrazito svijetle pozadine.

8. Zaključak

Ovaj rad se bavio primjenom računalnog vida za rješavanje problema detekcije kapljica kiše na zaštitnom staklu. Na početku je dan pregled osnovnih komponenata algoritma strojnog učenja. Objašnjen je princip rada konvolucijskih neuronskih mreža na kojima se zasnivaju skoro sve moderne arhitekture u računalnom vidu. Pokazani su modeli zasnovani na rezidualnim vezama koji omogućuju izgradnju jako dubokih mreža. Detaljno je opisan ResNet-18 koji se koristio u eksperimentalnom dijelu rada.

Iskorištene su razne augmentacije nad podacima kako bi se proširio skup za učenje i spriječila prenaučenost. Isprobana su dva različita optimizatora i tri različite veličine isječaka ulaznih slika da bi se pronašao optimalan model za dani skup podataka.

Komentirana je potencijalna primjenjivost naučenog modela u kompleksnijim sustavima. Premda ne obavlja klasifikaciju sa stopostotnom točnošću, primijećeno je da su slike na kojima model zakazuje izazovne i za ljudsko oko. Uz dodatno istraživanje i proširivanje skupa podataka, primjena modela u sustavu za detekciju kapljica kiše realna je mogućnost.

Budući rad mogao bi isprobati još neke zanimljive konvolucijske arhitekture za rješavanje ovog problema, poput arhitekture BagNet [4] u kojoj model uzima u obzir sitne detalje na slici u donošenju predikcije. Također, interesantno bi bilo istražiti neke naprednije metode perturbacije slika i na taj način pojačati utjecaj slabo vidljivih kapljica na klasifikacijsku odluku modela.

LITERATURA

[1] Ethem Alpaydin. *Introduction to Machine Learning - The MIT Press*. 2009.

[2] Ivana Barišić. Klasifikacija kapljica na staklu. Diplomski projekt, 2022.

[3] Jan Šnajder Bojana Dalbelo Bašić, Marko Čupić. *Umjetne neuronske mreže, materijali za kolegij Uvod u umjetnu inteligenciju, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*. 2022.

[4] Wieland Brendel i Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *International Conference on Learning Representations*, 2019. URL `https://openreview.net/pdf?id=SkfMWhAqYQ`.

[5] Ronan Collobert, Samy Bengio, i Johnny Marithoz. Torch: A modular machine learning software library, 2002.

[6] Ian J. Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

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

[8] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization (ICLR). 2014.

[9] Karel Lenc i Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence, 2014. URL `https://arxiv.org/abs/1411.5908`.

[10] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, i Tom Goldstein. Visualizing the loss landscape of neural nets. U S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, i R. Garnett, urednici, *Advances in Neural Information Processing Systems*, svezak 31. Curran Associates,

Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf.

[11] Lutz Prechelt. *Early Stopping - But When?*, stranice 55–69. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49430-0. doi: 10.1007/3-540-49430-8_3. URL https://doi.org/10.1007/3-540-49430-8_3.

[12] Rui Qian, Robby T. Tan, Wenhan Yang, Jiajun Su, i Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. U *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[13] Farheen Ramzan, Muhammad Usman Ghani Khan, Asim Rehmat, Sajid Iqbal, Tanzila Saba, Amjad Rehman, i Zahid Mehmood. A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fMRI and residual neural networks. *Journal of Medical Systems*, 44(2), Prosinac 2019. doi: 10.1007/s10916-019-1475-2. URL https://doi.org/10.1007/s10916-019-1475-2.

[14] rui1996 GitHub Repository. Deraindrop. https://github.com/rui1996/DeRaindrop, 2018.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, i Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, i Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016. URL https://arxiv.org/abs/1611.03530.

[17] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Hoi, i Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning (NeurIPS), 2020. URL https://arxiv.org/abs/2010.05627.

[18] Jan Šnajder. *Skripta za kolegij Strojno učenje, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*. 2022.

Detekcija kapljica kiše na zaštitnom staklu između scene i kamere

Sažetak

Mogućnost detekcije kapljica kiše važna je u mnogim modernim automatiziranim sustavima. Ovaj rad doskače tom problemu korištenjem suvremenih tehnika iz područja računalnog vida. Cilj je bio napraviti klasifikator koji će za danu sliku utvrditi nalaze li se na njoj kapljice ili ne. U radu su objašnjeni osnovni principi strojnog učenja i konvolucijskih neuronskih mreža. Opisan je korišteni skup podataka te programska implementacija u sklopu razvojnog okvira *PyTorch*. Odabran je model ResNet-18 prednaučen na skupu ImageNet koji se već pokazao pouzdan na mnogim klasifikacijskim zadacima. Isprobane su razne kombinacije hiperparametara i pronađen je najbolji model metodom unakrsne validacije. Ostvarena je točnost od 97.71% na skupu za validaciju i 92.26% na skupu za testiranje. Na kraju je napravljena diskusija postignutih rezultata i potencijalne primjene modela u stvarnim sustavima.

Ključne riječi: duboko učenje, računalni vid, klasifikacija slika, detekcija kapljica, konvolucijske neuronske mreže, duboki modeli s rezidualnim vezama, ResNet-18

Detection of raindrops on the protective glass between the scene and the camera

Abstract

The ability to detect raindrops is important in many modern automated systems. This paper addresses this problem by utilizing modern techniques in the field of computer vision. The goal was to create a classifier that can determine for a given image whether it contains raindrops or not. The paper explains the basic principles behind machine learning and convolutional neural networks. The dataset used and the software implementation written using *PyTorch* are described. A ResNet-18 model initialized with weights from the ImageNet dataset was selected, which has already proven reliable on many classification tasks. Various combinations of hyperparameters were tested and the best model was found using the cross-validation method. An accuracy of 97.71 % on the validation set and 92.26 % on the test set was achieved. Finally, a discussion of the results achieved and the potential application of the model in real systems was made.

Keywords: deep learning, computer vision, image classification, raindrop detection, convolutional neural networks, deep residual networks, ResNet-18