Klasifikacija objekata detektiranih ispred vozila pomoću kamere na prednjoj strani vozila

Kulić, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

KLASIFIKACIJA OBJEKATA DETEKTIRANIH ISPRED VOZILA POMOĆU KAMERE NA PREDNJOJ STRANI VOZILA

Diplomski rad

Filip Kulić

Osijek, 2018.

SADRŽAJ

1. UVOD

Kako bi se osigurala autonomija automobila, tj. kako bi se uspješno napravio automobil koji samostalno vozi (engl. *self-driving car*), računala koja upravljaju automobilom trebaju raspolagati informacijom što se nalazi ispred njih. Kako bi čovjek mogao odlučiti kako se ponašati u različitim situacijama, mora znati što se nalazi ispred njega, odnosno mora moći detektirati objekte te ih svrstati u jednu od kategorija. „Algoritam“ koji radi klasifikaciju kod ljudi nalazi se u mozgu, a ulaz u „algoritam“ jedan je od „senzora“ (ili više njih). Taj „senzor“ najčešće je vid. Detekcija i klasifikacija objekata osnova su „rada“ čovjeka bez kojih je nezamislivo kako bi čovjek donosio odluke o tome što činiti i kako se ponašati u pojedinim situacijama. Isto se može reći i za vozila.

Kako bi sustav za upravljanje automobilom (ADAS, engl. *Advanced driver-assistance system*) mogao donositi odluke u pojedinim situacijama, treba raspolagati s informacijama što se nalazi u okolini vozila, ponajviše s prednje strane vozila tijekom normalne vožnje. Te informacije dobivaju se uz pomoć algoritama za detekciju objekata i njihovom klasifikacijom. Izlazi iz algoritama mogu odlučivati o radu automobila: kočenje, zaobilaženje i slično. Iz ovoga postaje očito da je preciznost algoritama za detekciju i klasifikaciju kao i izvođenje u stvarnom vremenu osnova za praktičnu uporabu ovakvih algoritama. Ulaz u ADAS-u predstavljaju informacije sa elektroničkih senzora, a to su najčešće kamere, radari i lidari. Slika s kamera s prednje strane vozila predstavlja ulaz u algoritam za detekciju čiji izlaz predstavlja ulaz za algoritam za klasifikaciju. Ovi algoritmi najčešće su spojeni u jedan algoritam detekcije koji obavlja i lokalizaciju i klasifikaciju objekta na slici. Izlaz algoritma za klasifikaciju je vrijednost koja jedinstveno označava pojedinu klasu objekta, poput pješaka, automobila, prometnog znaka i slično.

Problemi klasifikacije slika danas se uglavnom rješavaju korištenjem neuronskih mreža, odnosno konvolucijskih neuronskih mreža, a taj se pristup koristi i u ovom diplomskom radu. Iako je bitno postići što bolju preciznost klasifikacije, to nije glavni cilj ovog diplomskog rada jer na algoritmima s visokom preciznošću rade veliki timovi stručnjaka te je takve rezultate samostalno vrlo teško postići. Glavni cilj ovog diplomskog rada je upoznati se s metodama klasifikacije pogodnim za primjenu u području automobilske industrije, odnosno s učenjem konvolucijskih neuronskih mreža i načinom rada istih te mogućnostima implementacije takvih algoritama na ugradbenu platformu i primjenu u praksi.

U drugom poglavlju ovog diplomskog rada predstavljene su metode klasifikacije, razlika između detekcije i klasifikacije objekata, način rada konvolucijskih neuronskih mreža i pojedinih slojeva

konvolucijskih neuronskih mreža te su na kraju poglavlja predstavljena i postojeća rješenja koja rješavaju problem klasifikacije objekata. U trećem poglavlju prikazan je kompletni proces izrade vlastitog rješenja za klasifikaciju objekata na računalu, a to uključuje prikupljanje podataka za učenje mreže, odabir arhitekture i učenje modela mreže te na kraju i evaluaciju naučenog modela, što uključuje prikaz rezultata vlastitog rješenja i usporedba s postojećim uspješnim rješenjem. Na kraju rada dan je zaključak rada.

2. METODE KLASIFIKACIJE

Naziv ovog diplomskog rada je klasifikacija objekata detektiranih ispred vozila pomoću kamere na prednjoj strani vozila, ali što je to objekt i na koji način je detektiran? Kao što je već spomenuto u uvodu, ulaz u algoritam za klasifikaciju nije cjelokupna slika s kamere. Ulaz u algoritam za klasifikaciju je objekt, odnosno granice objekta i cjelokupna slika ili samo odrezan dio slike na kojemu se objekt nalazi. Na slici 2.1. prikazan je tok algoritma za detekciju i algoritma za klasifikaciju, iako su u praksi ovi algoritmi najčešće spojeni u jedan algoritam detekcije koji obavlja i detekciju i klasifikaciju objekata.

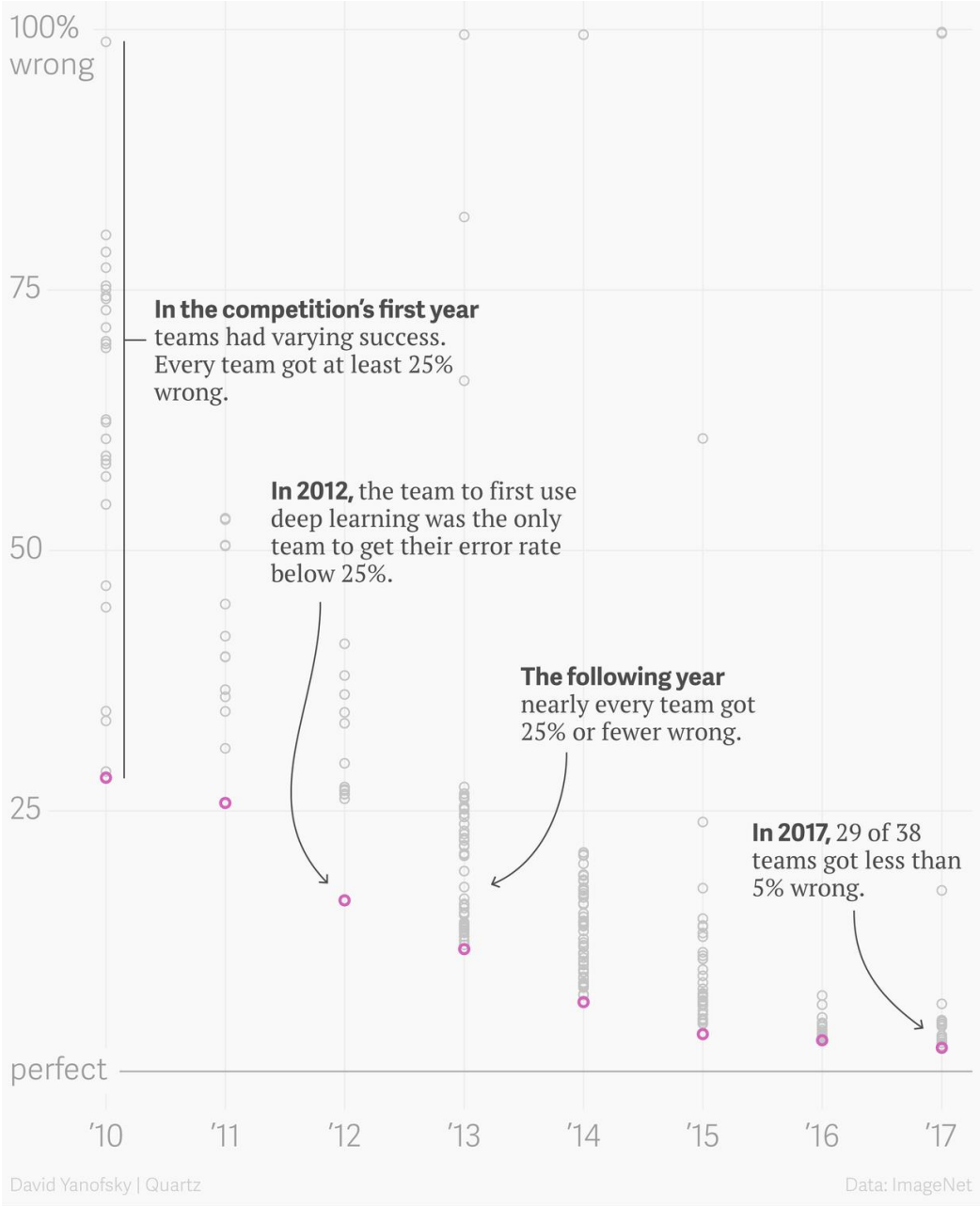
Slika 2.1. Tok algoritama detekcije i klasifikacije

Dakle, detektirani objekt zapravo je izrezana slika koja je ulaz u algoritam za klasifikaciju. Algoritam za klasifikaciju će zatim odrediti klasu pojedinog objekta, npr. pješak, pas, biciklist. Ovaj rad bavi se klasifikacijom objekata detektiranih ispred vozila što znači da se pretpostavlja da su objekti već detektirani na cjelokupnoj, većoj (složenijoj) slici, tj. rad se ne bavi algoritmima detekcije.

Kada se priča o klasifikaciji slika mora se spomenuti ImageNet natjecanje u klasifikaciji slika. 2012. godine prvi su puta upotrijebljene metode dubokog učenja (engl. d*eep learning*) u klasifikaciji slika i u potpunosti promijenile područje računalnog vida (engl. *computer vision*). To najbolje pokazuju i riječi članka economist.com portala: „Rezultati ImageNet natjecanja pokazali

su što duboko učenje može postići. Odjednom su ljudi počeli obraćati pozornost, ne samo unutar AI zajednice, već cijela tehnološka industrija.“ [1].

Duboko učenje omogućilo je preciznost veću od 75% već 2012. godine kada je prvi put upotrijebljeno u ImageNet natjecanju te je već sljedeće godine većina prijavljenih timova uspjela ponoviti prošlogodišnji uspjeh. 2017. godine većina prijavljenih timova postigla je preciznost veću od 95%, odnosno metode klasifikacije postale su preciznije od čovjeka i time definitivno potvrdila vrijednost dubokog učenja u klasifikaciji slika što se i vidi na slici 2.2.



Slika 2.2. Rezultati ImageNet natjecanja [2]

Dakle, klasifikacija slika u današnje vrijeme pretežno se radi uz pomoć dubokog učenja te se ovom metodom baviti i ovaj rad.

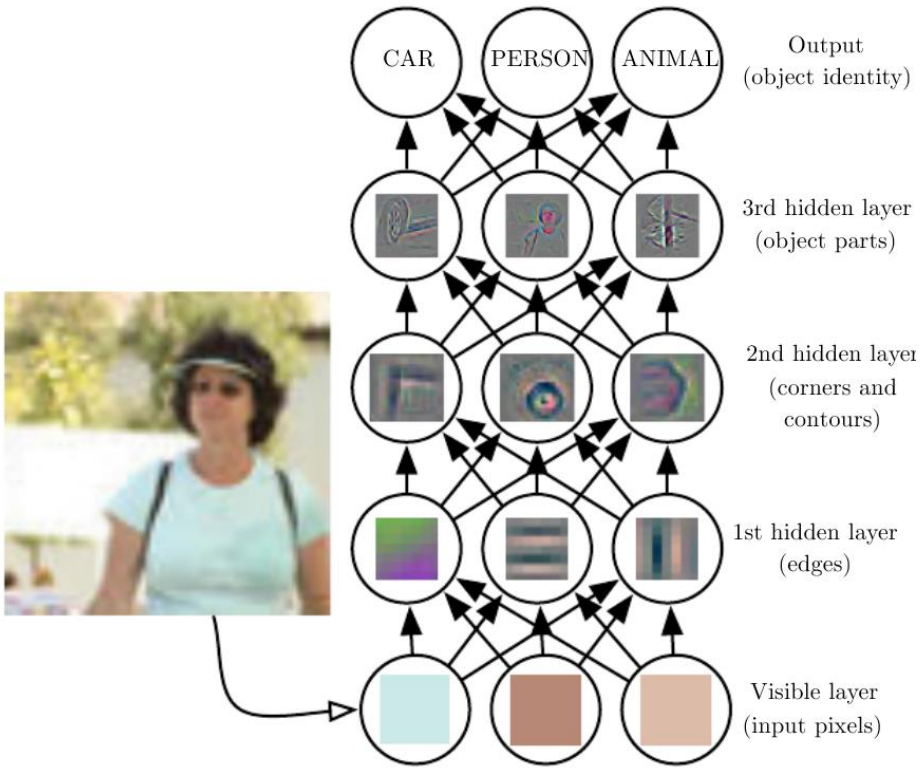
2.1. Uvod u duboko učenje

Duboko učenje područje je strojnog učenja i umjetne inteligencije koje podrazumijeva korištenje višeslojnih neuronskih mreža kako bi se riješili problemi kao što su detekcija i klasifikacija objekata, prepoznavanje govora, itd. Duboko učenje razlikuje se od tradicionalnog strojnog učenja po tome što nije potrebno uvođenje dodatnih pravila ili poznavanja ljudske domene, odnosno ručnog izvlačenja značajki, kako bi se omogućio rad s podacima kao što su slike, video, tekst i slično. Fleksibilna arhitektura dubokih mreža, tj. višeslojnih neuronskih mreža, omogućuje učenje iz neobrađenih podataka i može povećati točnost predikcije kada se pruži više podataka.

Duboko učenje osnova je mnogih nedavnih otkrića u području umjetne inteligencije kao što su autonomna vozila (engl. *self driving car*), inteligentni glasovni asistenti (engl. *voice assistants*) na pametnim telefonima i mnoge druge stvari.

Duboko učenje koristi već spomenute višeslojne neuronske mreže, odnosno neuronske mreže koje imaju više skrivenih slojeva (engl. *hidden layers*). Ta dubina dubokih neuronskih mreža, odnosno višeslojnost skrivenih slojeva, omogućuje računalu da nauči karakteristične značajke (engl. *features*) pojedinih podataka i na taj način omogući primjerice, efikasnu klasifikaciju slika.

Na slici 2.3. prikazan je način na koji bi duboka neuronska mreža mogla naučiti prepoznavati značajke iz podataka, odnosno slike. Prvi sloj je ulazni sloj, odnosno slika. Drugi sloj, odnosno prvi sakriveni sloj bio bi zadužen za detekciju rubova, drugi skriveni sloj za detekciju kutova i zakrivljenosti, a treći skriveni sloj za prepoznavanje dijelova objekta kojeg klasificira. Posljednji sloj je izlazni sloj koji govori kolika je vjerojatnost pojedine klase.



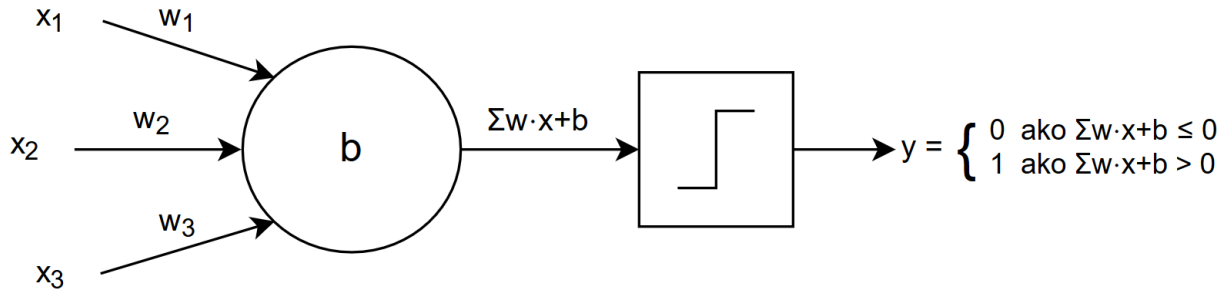
Slika 2.3. Primjer naučene duboke neuronske mreže [3]

Neuronske mreže mogu imati više vrsta slojeva koji imaju odgovarajuće funkcije. Za razumijevanje principa rada pojedine mreže potrebno je razumjeti funkcionalnosti koje njeni slojevi obavljaju. Neki od najčešće korištenih slojeva su potpuno povezani sloj (engl. *fully connected layer*), konvolucijski sloj (engl. *convolutional layer*) i sloj sažimanja (engl. *pooling layer*). Također, iako su aktivacijske funkcije dio potpuno povezanih i konvolucijskih slojeva, aktivacijske funkcije često se prikazuju kao aktivacijski slojevi zbog preglednosti i lakšeg kodiranja.

Skup slojeva i način na koji su povezani nazivamo arhitekturom neuronske mreže. Svaka neuronska mreža počinje sa ulaznim, a završava sa izlaznim slojem koji je u slučaju klasifikacije *softmax* sloj. U ovom radu razmatraju se unaprijedne neuronske mreže (engl. *feedforward neural networks*) u kojima se propagacija signala odvija od ulaza prema izlazu, bez povratnih veza, kao što je vidljivo na slici 2.3.

Neuronske mreže nazivaju se neuronske jer je osnovni gradbeni element neuron. Postoje različite vrste neurona, a podjela vrsta neurona bazira se na aktivacijskim funkcijama. Aktivacijske funkcije zapravo su standardne funkcije koje računaju izlaz koji je ovisan o ulazu.

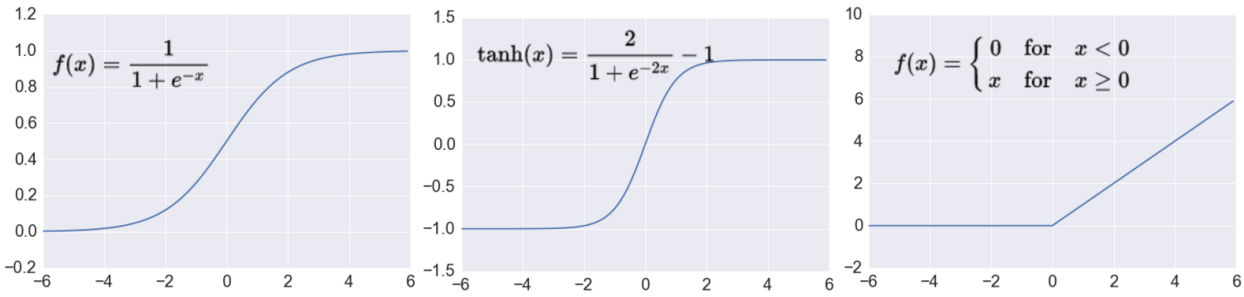
Najjednostavniji tip neurona je perceptron, neuron kojem je izlaz 0 ili 1. Svaki neuron, uključujući i perceptron, sastoji se od jednog ili više ulaza, pripadajućom težinom za svaki od ulaza i pomaka (engl. *bias*). Perceptron je neuron koji se aktivira step aktivacijskom funkcijom koja će dati izlaz 1, tj. neuron će okinuti, ukoliko je ulaz veći od 0. Izlaz iz step aktivacijske funkcije biti će 0, tj. neuron neće okinuti, ukoliko je ulaz manji ili jednak 0. Ulaz u aktivacijsku funkciju zbroj je umnoška ulaza (engl. *inputs*) i pojedinih težina (engl. *weights*) i pomaka (engl. *bias*), a zbog često velikog broja ulaza, ove operacije najčešće se prikazuju kao matrične operacije. Na slici 2.4. prikazan je neuron sa step aktivacijskom funkcijom, koji se još naziva i perceptron.



Slika 2.4. Neuron sa step aktivacijskom funkcijom (perceptron)

Neuron prikazan na slici 2.4. neuron je s tri ulaza, gdje je svaki ulaz predstavljen s oznakom x_i. Uz svaki ulaz veže se i pripadajuća varijabla težine označena s oznakom w_i, tako da neuron ima i tri težine i jednu varijablu pomaka b. Step aktivacijska funkcija, odnosno perceptron, može dati samo izlaz 0 ili 1 što nije previše korisno jer mala promjena ulaza u step funkciju može drastično promijeniti izlaz, a velika promjena ne mora. Primjerice, ukoliko je ulaz u step funkciju 1, izlaz iz step funkcije bit će 1. Relativno mala promjena na ulazu, kada ulaz postaje -1, na izlazu će dati 0, dok će velika promjena kada je ulaz, primjerice, 5000 i dalje dati izlaz 1. Upravo zbog ovog, step funkcija u neuronskim mrežama nije previše korisna, ali je vrlo korisna što se tiče razumijevanja neurona i načina na koji rade.

Kako bi se riješio problem binarnog izlaza 0 ili 1, tj. da se ugradi nekakva dinamika promjene u model neurona, potrebno je koristiti druge aktivacijske funkcije. Neke od njih prikazane su na slici 2.5.



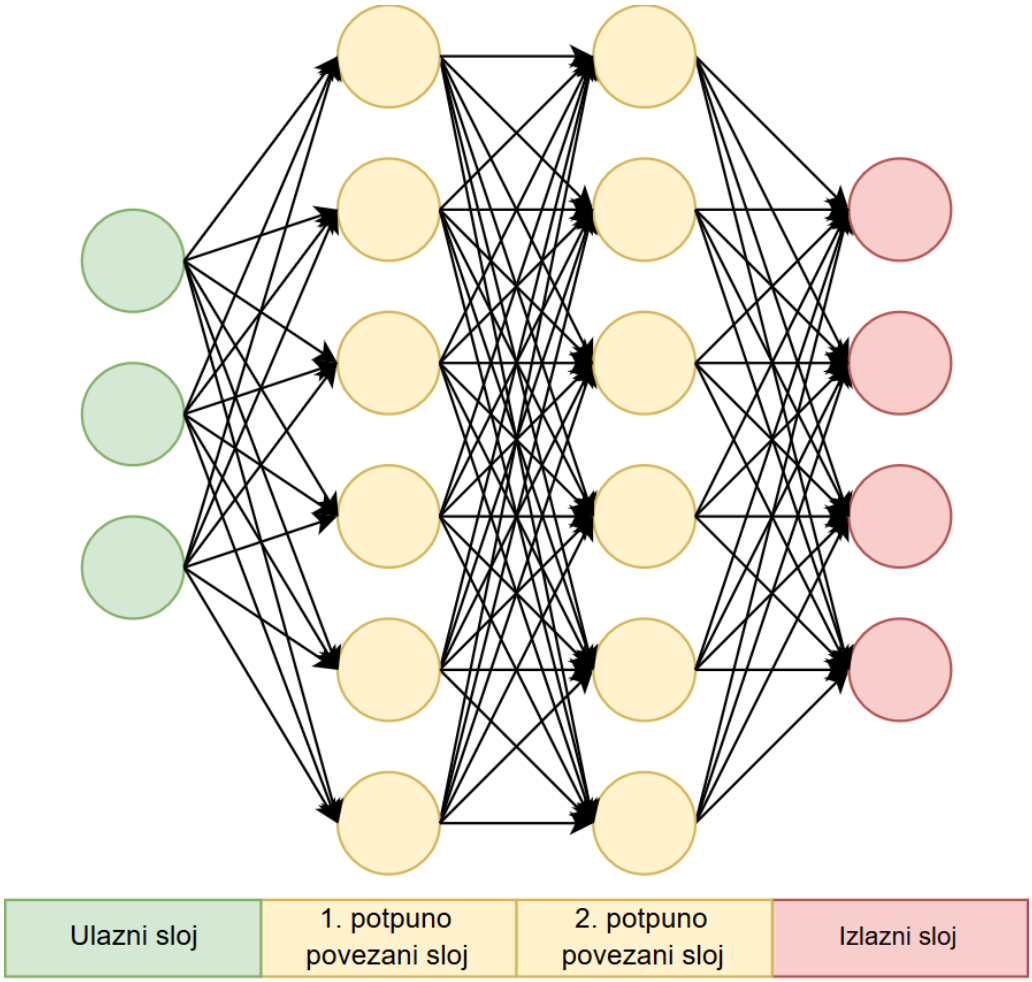
a) Logistička funkcija b) TanH c) ReLU

Slika 2.5. Primjer aktivacijskih funkcija kod neuronskih mreža [4]

Logistička funkcija (engl. *sigmoid*) aktivacijska je funkcija koja se koristi kada se izlaz predviđa kao vjerojatnost, tj. kada vrijednost izlaza treba biti između 0 i 1. TanH funkcija slična je logističkoj funkciji, ali se vrijednost izlaza uvijek nalazi između -1 i 1. TanH kao funkcija često se koristi kao aktivacijska funkcija u binarnoj klasifikaciji. ReLU aktivacijska funkcija za svaku negativnu vrijednosti na ulazu vraća 0 kao izlaz. Ukoliko ulaz u aktivacijsku funkciju nije negativna vrijednost, ReLU će ulaznu vrijednost proslijediti kao izlaz [5].

2.1.1. Potpuno povezani sloj

Jedan od najčešće korištenih slojeva koji se nalazi u gotovo svakoj arhitekturi neuronske mreže jest potpuno povezani sloj (engl. *fully connected layer*) koji se često naziva i gusti sloj (engl. *dense layer*). Potpuno povezani sloj je sloj kod kojeg je svaki neuron povezan sa svim neuronima iz susjednih slojeva. Kod arhitektura koje se koriste za klasifikaciju slika, potpuno povezani sloj najčešće se koristi kao poveznica između konvolucijskih slojeva i *softmax* izlaznog sloja. Ulazi i izlazi iz potpuno povezanog sloja najčešće se prikazuju kao vektori. Na slici 2.6. prikazana je mreža sa 2 skrivena sloja gdje su oba sloja potpuno povezani slojevi. Također, kod mreže na slici 2.6., ulazni i izlazni sloj također su potpuno povezani slojevi.



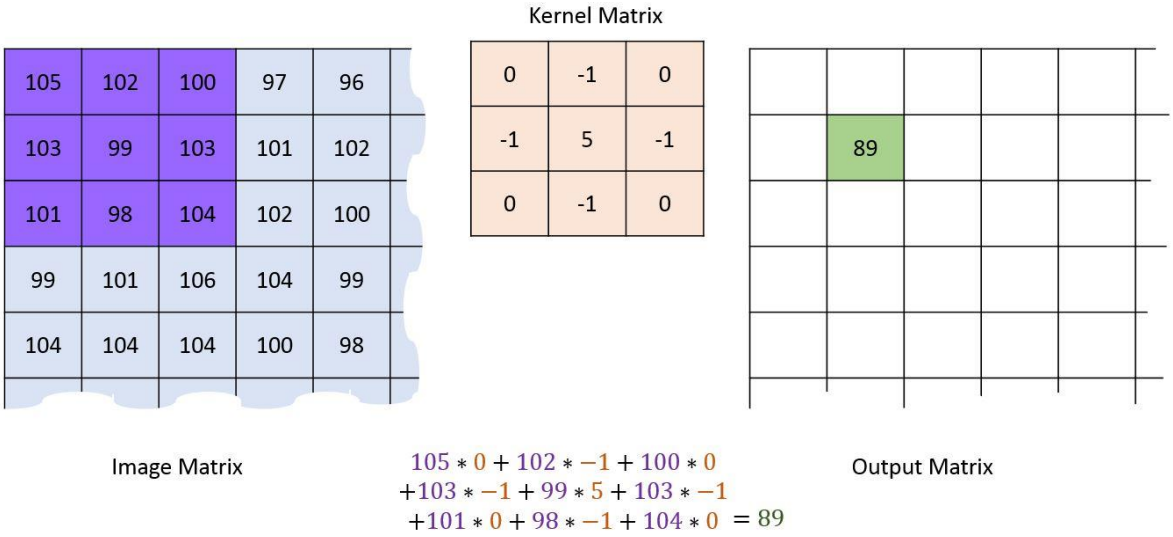
Slika 2.6. Mreža sa 2 skrivena potpuno povezana sloja

2.1.2. Konvolucijski sloj

U današnje vrijeme uz pomoć dubokog učenja, klasifikaciju slika nije moguće zamisliti bez konvolucijskog sloja. Konvolucijski sloj bazira se na, kao što i sam pojam sugerira, konvoluciji: integralu koji opisuje količinu preklapanja jedne funkcije s drugom dok se jedna pomiče preko druge. Primjena konvolucije na sliku naziva se konvolucija slike, odnosno dvodimenzionalna konvolucija.

Slika se, u računarstvu, najčešće predstavlja trodimenzionalnom matricom čiji elementi imaju vrijednosti 0 do 255 (ukoliko se radi o 8-bitnom formatu). Veličina matrice odgovara umnošku visine i širine slike te broju kanala. Ako je slika u nijansama sive boje (engl. *grayscale*), tada je predstavljena jednim kanalom, a ako je slika u boji postoje tri kanala gdje jedan predstavlja crvenu, drugi zelenu, a treći plavu boju (ukoliko se radi o RGB formatu boje). Dvodimenzionalna konvolucija, odnosno konvolucija slike zapravo predstavlja matrični umnožak konvolucijske

jezgre (engl. *kernel*), odnosno filtra, s pojedinim elementima slike, odnosno matricom koja odgovara pojedinom dijelu slike. Konvolucija slike prikazana je na slici 2.7.



Slika 2.7. Konvolucija slike [5]

Pojedina vrijednost konvolucijskog sloja, gdje ulaz ima veću dubinu od jedan, računa se tako da se napravi umnožak filtra i dijela ulazne matrice (kao što je prikazano na slici 2.7) za svaki od ulaznih kanala te se vrijednosti dobivene tom operacijom zbroje pa tek onda upisuju u izlaznu matricu.

Konvolucijski sloj tenzor je s tri dimenzije koje se nazivaju širina konvolucijskog sloja (broj stupaca pojedine matrice, 1. dimenzija), visina konvolucijskog sloja (broj redaka pojedine matrice, 2. dimenzija) i dubina konvolucijskog sloja (koliko ima filtara, 3. dimenzija), a sve tri dimenzije spominju se u kontekstu veličine konvolucijskog sloja. Konvolucijski slojevi u neuronskim mrežama najčešće imaju više filtara, a veličina sloja ovisi o ulaznoj visini i širini matrice, broju i veličini filtra, nadopuni (engl. *padding*) i pomaku (engl. *stride*). Pomak kontrolira kako će se konvolucijski filtar pomicati po tenzoru. Kod dvodimenzionalne konvolucije pomak označava za koliko će se mjesta filtar pomicati vertikalno, a za koliko horizontalno. Nadopuna predstavlja dodavanje dodatnih elemenata matrice na svaki od rubova kako bi svaki element matrice utjecao jednako na izlaz. Bez nadopune konvolucijskog sloja, čak i kada je vrijednosti pomaka jednaka jedan po svim dimenzijama, uvijek bi dimenzije konvolucijskog sloja bile manje od dimenzija veličine ulaznog tenzora. Kod dvodimenzionalne konvolucije to znači da visina i širina konvolucijskog sloja može biti jednaka visini i širini veličini ulaznog tenzora.

Matematički izrazi pomoću kojih se računa veličina konvolucijskog sloja:

$$W_c = \frac{W_{in} - W_f + 2P}{S_w} + 1 \tag{2-1}$$

gdje je:

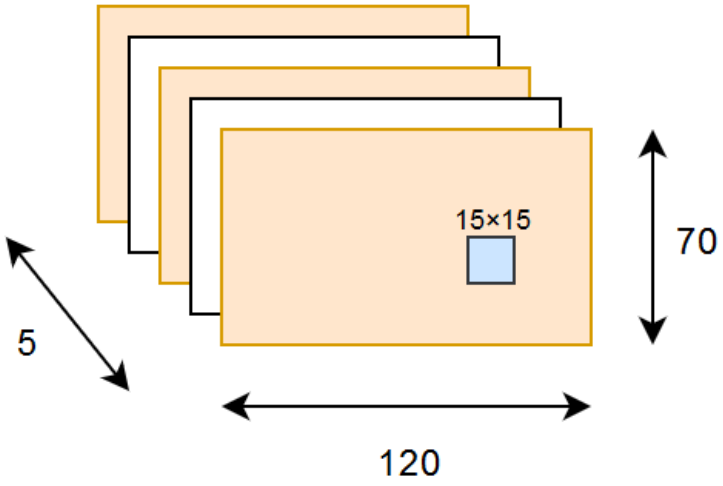
- W_c – širina matrice konvolucijskog sloja
- W_{in} – širina ulazne matrice
- W_f – širina filtra konvolucijskog sloja
- P – nadopuna
- S_w – pomak po širini,

$$H_c = \frac{H_{in} - H_f + 2P}{S_H} + 1 \tag{2-2}$$

gdje je:

- H_c – visina matrice konvolucijskog sloja
- H_{in} – visina ulazne matrice
- H_f – visina filtra konvolucijskog sloja
- P – *nadopuna*
- S_H – pomak po visini.

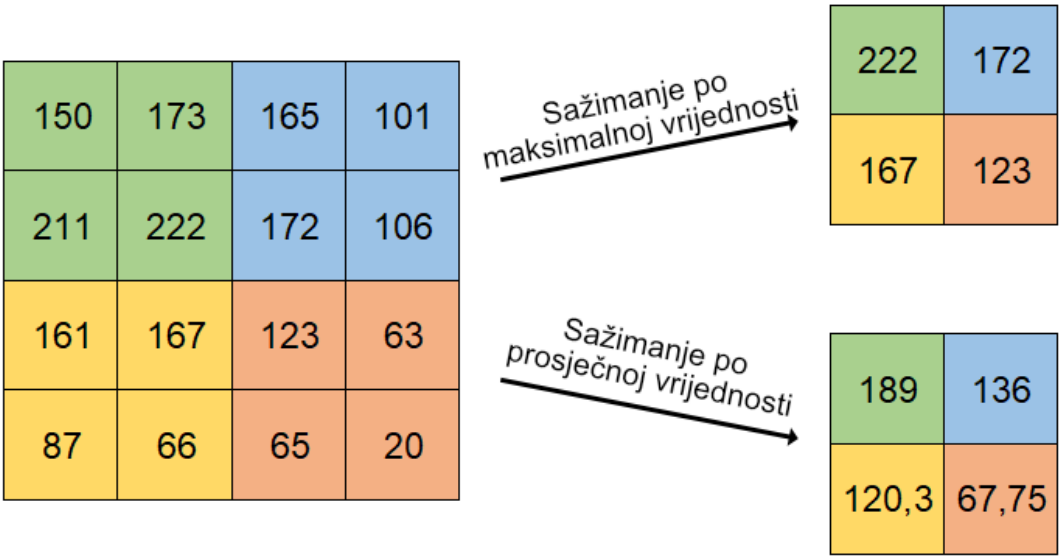
Dakle, konvolucijski sloj u neuronskim mrežama sastoji se ne samo od *n* izlaznih matrica, tj. tenzora, već i od konvolucijskih filtara gdje je jedan filtar pridružen jednoj izlaznoj matrici, a grafički se najčešće prikazuju kao na slici 2.8. gdje je prikazan konvolucijski sloj dimenzija 120×70×5 s konvolucijskim filtrom dimenzija 15×15.



Slika 2.8. Primjer konvolucijskog sloja dimenzija 120×70×5 i veličine filtara 15×15

2.1.3. Sloj sažimanja

Sloj sažimanja (engl. *pooling layer*, *subsampling layer*) jedan je od slojeva u neuronskim mrežama koji se ne uči, tj. ne postoje nikakvi podesivi parametri. Sloj sažimanja obavlja, kao što sama riječ opisuje, sažimanje podataka, a to čini na jedan od dva načina: sažimanje maksimalnom vrijednosti (engl. *max pooling*) ili sažimanje srednjom vrijednosti (engl. *average pooling*, *mean pooling*). Sloj sažimanja uzima predefiniranu veličinu podmatrice i koristi jednu od operacija kako bi sažeo podatke. Ova operacija jasnije je prikazana na slici 2.9. na presjeku jedne dubine (engl. *single depth slice*).



Slika 2.9. Prikaz operacije sažimanja s veličinom podmatrice 2×2 i pomakom 2

2.1.4. *Softmax* sloj

Softmax sloj zapravo je potpuno povezani sloj sa *softmax* aktivacijskom funkcijom. *Softmax* sloj koristi se kao izlazni sloj kod mreže kojoj je namjena višeklasna (engl. *multiclass*) klasifikacija. Broj izlaznih neurona neke mreže koja obavlja višeklasnu klasifikaciju, tj. broj neurona *softmax* sloja, jednak je broju klasa iz skupa podataka na kojemu se mreža uči. *Softmax* sloj kao izlazni sloj neke mreže određuje vjerojatnosti da objekt koji se klasificira pripada pojedinoj klasi, a ukupan zbroj vjerojatnosti uvijek je jednak jedan. Klasa kojoj *softmax* aktivacijska funkcija pridijeli najveću vjerojatnost uzima se kao klasa koju mreža predviđa za neki objekt.

2.2. Učenje mreže

Neuronska mreža sastavljena je od niza slojeva gdje su neki slojevi neuronski slojevi kao što su potpuno povezani sloj koji je objašnjen u odjeljku 2.1.1. i konvolucijski sloj koji je objašnjen u odjeljku 2.1.2. Slojevi kojima su osnovni gradbeni elementi neuroni kao takvi imaju niz parametara (težine i pomaci, filtri konvolucijskog sloja) koje je potrebno podesiti na odgovarajuće vrijednosti kako bi mreža ispravno klasificirala ulazne podatke. Ugađanje ovih parametara naziva se učenje neuronske mreže i ono se provodi na skupu podataka za učenje. Učenje neuronske mreže kod problema klasifikacije odvija se metodom nadziranog učenja, pri čemu se na ulaz mreže dovode ulazni podaci za koje se zna kojoj klasi pripadaju.
Za učenje mreže potrebno je imati metriku gubitka (engl. *loss*), tj. pogreške. Kod neuronskih mreža koje se koriste za klasifikaciju najčešće se gubitak računa kao gubitak unakrsne entropije (engl. *cross entropy loss*) te se za učenje mreže u ovom radu koristi ta metrika [6]. Cilj učenja jest pronaći parametre mreže koji stvaraju najmanji mogući gubitak, a za to je potrebno znati i točan izlaz koji mreža treba dati za pojedini ulazni podatak, tj. sliku u slučaju ovog rada. Skup podataka na kojemu se uči mreža za klasifikaciju mora sadržavati oznake, a dio te oznake uvijek je i klasa kojoj pojedini podaci pripadaju pa zbog toga klasifikacija pripada u nadzirano učenje.
Prilikom učenja unaprijedne mreže, prvi korak predstavlja prolazak prema naprijed (engl. *forward propagate*) pri čemu se ulazni podatak propušta kroz slojeve mreže do izlaza. Zatim se uz pomoć poznatog točnog izlaza pročitanog iz oznake računa gubitak pomoću neke funkcije kao što je gubitak unakrsne entropije. Kako bi se parametri pojedinih neuronskih slojeva mreže mogli naučiti, odnosno prilagoditi tako da daju točan izlaz, potrebno je mrežom proći unatrag (engl. *back propagation*). Prilikom prolaska unatrag provodi se prepodešavanje

parametara mreže kako bi se smanjio gubitak, pri čemu se koristi određena stopa učenja (engl. *learning rate*) i diferencijal gubitka. U ovom se radu pri učenju mreže koristi *Adam* metoda optimizacije [7].

Neuronske mreže najčešće se uče u nekoliko epoha (engl. *epoch*) gdje je pojedina epoha podijeljena u više koraka (engl. *step*) učenja. Jedna epoha predstavlja prolazak kroz cijeli skup podataka. Broj koraka koji je potreban za izvođenje jedne epohe određen je veličinom grupe podatka (engl. *batch size*) koja se istovremeno koristi za učenje mreže. Broj koraka u epohi određuje se kao količnik ukupnog broja podataka u epohi i veličine grupe podataka koji se istovremeno koristi za učenje.

Za učenje mreže, tj. učenje što preciznijeg modela, potrebno je imati što veći i raznovrsniji skup podataka. Ovo nije uvijek moguće pa se zbog toga nerijetko koristi augmentacija podataka. Augmentacija podataka predstavlja sitne promjene u podacima kako bi se stvorilo više podataka za učenje bez potrebe ponovnog prikupljanja i označavanja podataka pa se time postiže i veća raznovrsnost ili poopćenje podataka.

Još jedna tehnika koja se koristi kako bi se povećala preciznost naučenog modela neuronske mreže jest ubacivanje *dropout* slojeva. To su slojevi koji izbacuju zadani udio neurona kako pojedini neuroni ne bi previše sudjelovali u klasifikaciji, tj. smanjuje moguću nerazmjernost utjecaja pojedinih neurona u klasifikaciji do koje može doći prilikom učenja.

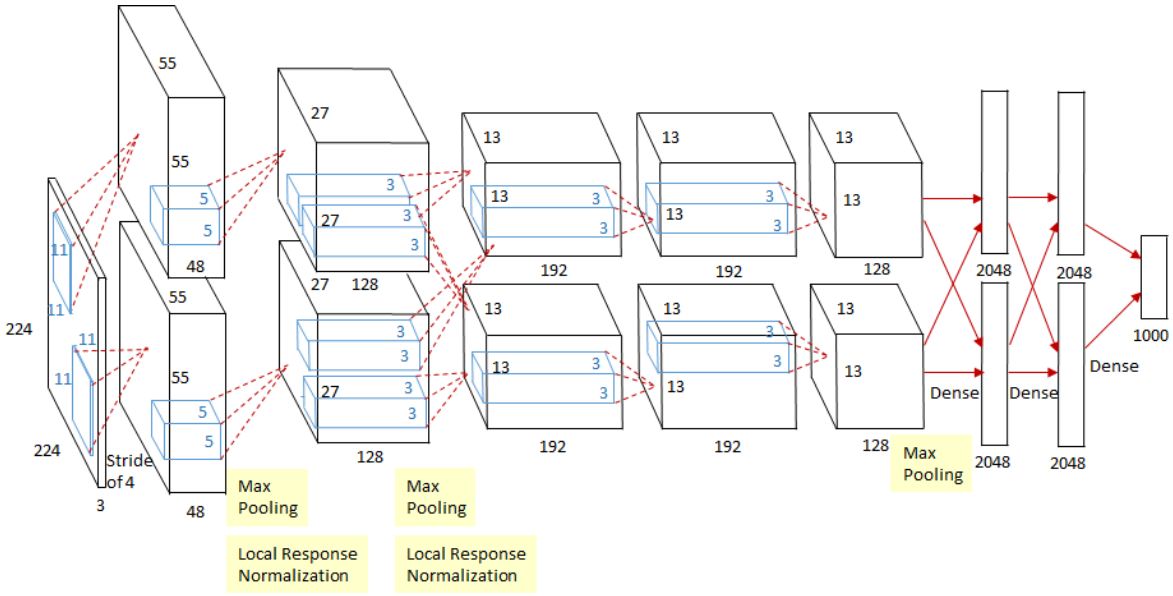
Tehnike ubacivanja *dropout* slojeva i augmentacije slika korištene su u ovom radu, ali se treba naglasiti da se prilikom testiranja ne radi augmentacija slika, a *dropout* slojevi se izbacuju iz konačnog modela neuronske mreže koji se koristi za testiranje.

2.3. Postojeće uspješne arhitekture

Današnja primjena dubokog učenja je neupitna, ali nije oduvijek bilo tako. Duboko učenje dobilo je na važnosti povećanjem broja podataka (npr. označenih slika) i povećanjem računalne moći koja je omogućila podešavanje složenih mreža na velikim skupovima podataka. Jedna od prvih konvolucijskih mreža, koja je pokazala mogućnosti dubokih neuronskih mreža, bila je LeNet5 mreža [8]. LeNet5 napravljena je 1994. godine, ali zbog nedostatka računalne moći duboko učenje nije odmah zaživilo.

Tek 2010. godine Dan Claudiu Ciresan i Jurgen Schmidhuber objavljuju prvu uspješniju GPU implementaciju neuronskih mreža. Ova implementacija bila je implementirana na NVIDIA GTX 280 grafičkoj kartici i mogla je podržati do devet slojeva.

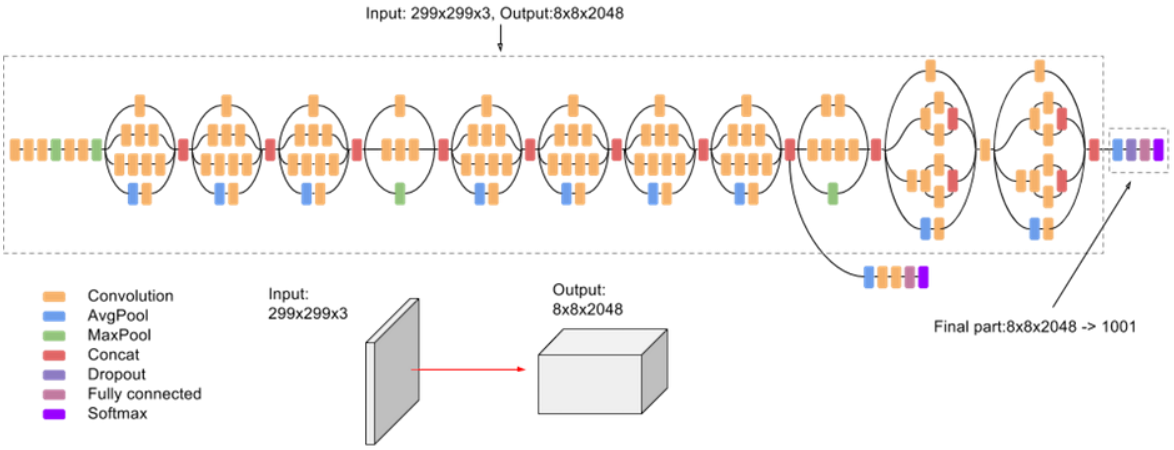
2012. godine kreće prava revolucija kada Alex Krizhevsky objavljuje AlexNet [8], dublju i veću implementaciju LeNet5 mreže, i pobjeđuje na ImageNet natjecanju, što se vidi i na slici 2.2. AlexNet mreža, prikazana na slici 2.10., koristila je ReLU aktivacijske funkcije umjesto TanH kako bi omogućila nelinearnu funkcionalnu ovisnost uz značajno ubrzanje sa stajališta podešavanja parametara mreže i ubrzala rad za šest puta postižući istu preciznost. Također, AlexNet mreža koristila je *dropout* sloj, preklapanje sažimanja po maksimalnoj vrijednosti i NVIDIA GTX 580 grafičku karticu kako bi se smanjilo vrijeme učenja mreže.



Slika 2.10. Arhitektura AlexNet mreže [9]

Trenutno jedne od najčešće korištenih mreža su Googlove GoogLeNet i Inception mreže. Prvu verziju razvio je Christian Szegedy 2014. godine, a glavni cilj mreže bio je poboljšanje performansi, ali najčešće spominjana i korištena mreža je Inception V3 prikazana na slici 2.11. Glavne ideje ove mreže su [10]:

- Maksimiziranje protoka informacija u mreži uravnotežujući dubinu i širinu
- Kada se povećava dubina mreže, povećava se i broj značajki i širina sloja
- Iskorištavanje povećanja širine sloja kako bi se povećao broj kombinacija značajki
- Ukoliko je moguće, koristiti samo 3×3 veličinu konvolucijskog filtra



Slika 2.11. Arhitektura InceptionV3 mreže [11]

Pošto se ovaj rad bavi klasifikacijom objekata s kamere koja se nalazi na prednjoj strani vozila, treba spomenuti i VoNet mrežu. VoNet mreža je mreža kojoj je namjena klasificirati kako je automobil okrenut, odnosno koja strana automobila se vidi: prednja, stražnja, bočna, prednja i bočna te zadnja i bočna. Također, cilj VoNeta je napraviti mal i brz model koji može raditi na hardveru sa ograničenim resursima. Rezultati koje je VoNet uspio postići prikazani su u tablici 2.1. gdje se također vidi i uspješnost AlexNet, GoogLeNet mreža. Testovi su rađeni na CompCars [12] skupu podataka, a za učenje mreža korištena je Nvidia GRID K520 4GB grafička kartica.

Tablica 2.1. Usporedba performansi različitih arhitektura na CompCars skupu podataka [13]**Error! Reference source not found.**

	Preciznost	Brzina inferencije (engl. *inference speed*)	Veličina modela	Broj parametara
AlexNet	97,14%	179,23 ms	227 MB	56.888.709
GoogLeNet	97,32%	106,50 ms	41,3 MB	10.318.655
VoNet	95,45%	57,04 ms	1,6 MB	394.789

Brzina inferencije izuzetno je bitna metrika za ugradbene sustave gdje ne postoji toliko računalne moći. Brzina inferencije, tj. brzina donošenja odluke, kod neuronskih mreža predstavlja vrijeme potrebno da mreža da izlaz za neki ulazni podatak. Iz rezultata se vidi da AlexNet postiže vrlo visoku preciznost, ali i brzina inferencije je vrlo velika sa prosjekom od 179,23 ms te je model vrlo velik s čak 56.888.708 parametara. GoogLeNet postiže najbolju

preciznost, ali je model i dalje dosta velik, a time i brzina inferencije. VoNet, iako ne postiže najbolju preciznost, ima najmanji model sa samo 394.789 parametara te je zbog toga veličina modela samo 1,6 MB, a brzina inferencije iznosi 57,03 ms. Zbog svega ovoga može se zaključiti da za sustave s ograničenim resursima, onakvi kakvi se nalaze u automobilima, najviše odgovara arhitektura s manjim brojem parametara, odnosno VoNet ukoliko se bira između navedenih arhitektura.

3. RAZVOJ VLASTITOG RJEŠENJA

Kako bi se razvilo vlastito rješenje, najprije je potrebno odabrati razvojno okruženje u kojemu će se razviti model. Za razvoj na računalu u ovom radu odabran je Tensorflow *framework* zbog dostupnosti velikog broja materijala za učenje i velikog broja korisnika, a time i dobre podrške. Tensorflow je softverska biblioteka otvorenog koda (engl. *open source*) namijenjena izračunu numeričkih operacija uz visoke performanse. Originalno razvijen od strane Google AI organizacije, Tensorflow pruža jaku podršku strojnom i dubokom učenju. Fleksibilna arhitektura Tensorflowa omogućuje lakoću razvoja na različitim platformama, tj. omogućuje razvoj na centralnim procesorskim jedinicama, grafičkim procesorskim jedinicama i tenzor procesorskim jedinicama [14].

Nadalje, odabran je Keras *framework* koji je sastavni dio Tensorflow *framework* paketa, a iz kojeg je vrlo lako prebaciti model nazad u Tensorflow. Razlog tomu je brži razvoj mreža, pregledniji kod i lakše otkrivanje pogrešaka [15].

3.1. Razvoj modela na računalu

Kao što je već spomenuto, za razvoj modela na računalu odabran je Keras *framework*. Nakon što se razvije zadovoljavajući model u Kerasu, model će se prebaciti u Tensorflow *framework*. Keras je *framework* koji je napisan u Python programskom jeziku te je jasno da će se za razvoj modela u Kerasu koristiti Python, najčešći programski jezik u podatkovnim znanostima (engl. *data science*). Kako bi se razvio funkcionalan model potrebno ga je i naučiti, a za učenje neuronskih mreža potreban je skup podataka.

3.1.1. Prikupljanje i predobrada podataka

Za odabir skupa podataka potrebno je najprije definirati klase objekata za koje se namjerava napraviti klasifikator, odnosno za koje se namjerava naučiti neuronska mreža. Budući da se ovaj rad bavi razvojem rješenja za klasifikaciju objekata s prednje strane vozila, određeno je osam klasa: automobil, autobus, biciklist, kamion, kombi, pješak, semafor i znak. U stvarnom slučaju klase bi trebalo dodatno razdvojiti, odnosno bilo bi potrebno dodatno klasificirati semafore ovisno o svjetlu koje je upaljeno, znakove ovisno o kojemu se znaku radi i slično, te dodati dodatne klase za, primjerice, životinje kao što su krave, divlje svinje, ovce i slično.

Za učenje mreže potreban je skup podataka na kojemu će se neuronska mreža naučiti i skup podataka na kojemu će se testirati naučeni model neuronske mreže. Postoji mnogo javno dostupnih

skupova podataka, tj. skupova podataka koje je moguće preuzeti, koji su zapravo skup slika i oznaka. Postoje skupovi podataka koji se sastoje samo od slika, odnosno objekti su već izrezani te su oni jedino što se nalazi na slici. Međutim, većina skupova podataka sastoji se od slika na kojima se nalazi više objekata. Pozicija i veličina graničnog okvira (engl. *bounding box*) objekta i klasa koja je pridijeljena svakom objektu naziva se oznakom. Kako bi se model mogao dobro naučiti potrebno je praviti neke osnovne uvjete pri izradi skupa podataka kao što su:

- dovoljno velika rezolucija slika (širina od barem 50 piksela), odnosno visoka kvaliteta
- raznolikost slika – kako bi model bio što općenitiji
- dovoljno slika za svaku klasu, što za skup podataka za učenje mreže, što za testiranje

S obzirom da ove uvjete nije bilo lako zadovoljiti s jednim skupom podataka, odabran je pristup kombiniranja skupova podataka kako bi se dobio dovoljan broj slika za svaku klasu te kako bi se dobio dovoljan broj raznovrsnih slika. U tablici 3.1. navedeni su korišteni skupovi podataka te broj preuzetih slika iz istih.

Tablica 3.1. Korišteni skupovi podataka, preuzet broj slika iz pojedinog skupa podataka

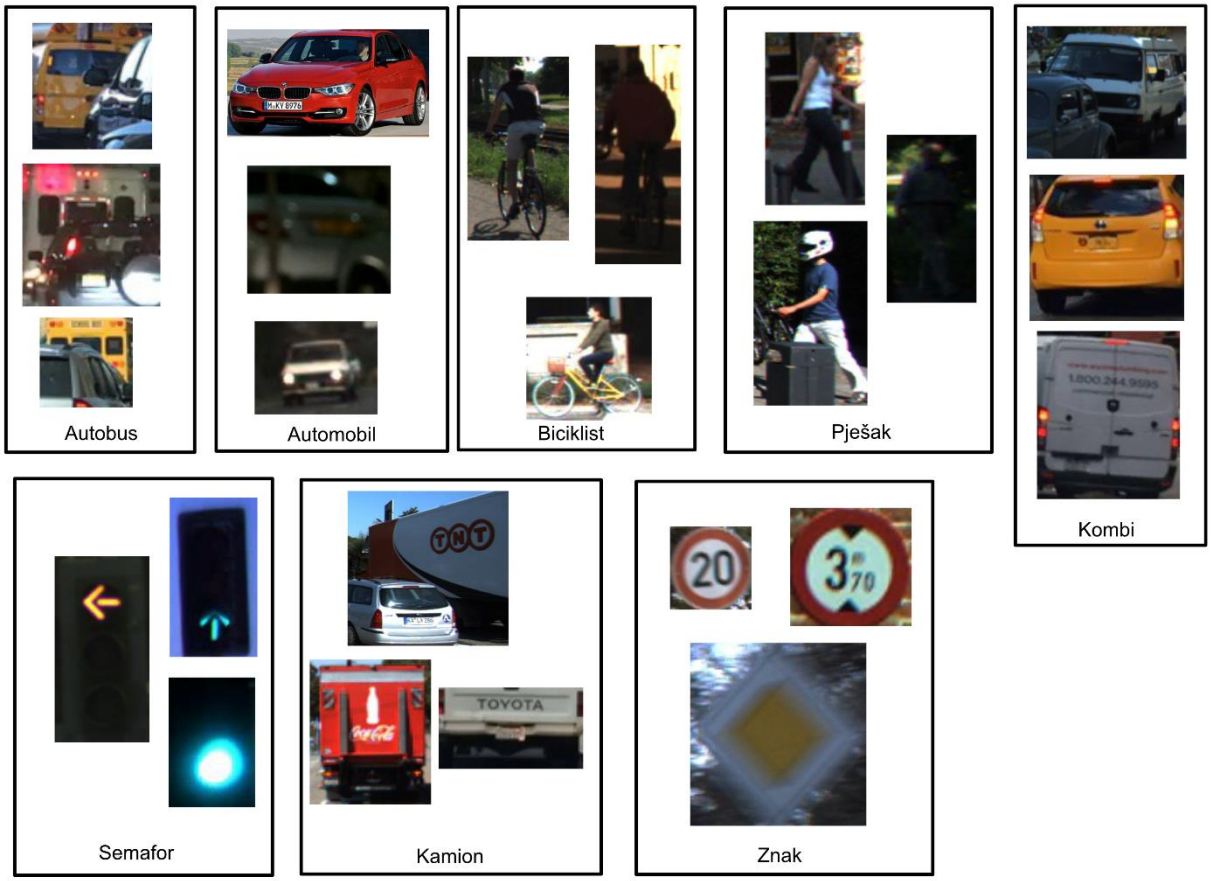
	Automobil	**Autobus**	**Biciklist**	**Kamion**	**Kombi**	**Pješak**	**Semafor**	**Znak**
Belgium Traffic Sign [16]	-	-	-	-	-	-	-	7125
Bosch Small Traffic Lights [17]	-	-	-	-	-	-	42	-
Stanford Cars [18]	16185	-	-	-	-	-	-	-
Tsinghua-Daimler Cyclist [19]	-	-	10002	-	-	-	-	-
The German Traffic Sign [20]	-	-	-	-	-	-	-	2194
The KITTY vision 2012 [21]	20663	-	597	694	2002	1637	-	-
LISA Traffic Light [22]	-	-	-	-	-	-	9984	0
Nexar [23]	70940	2589	-	7806	15513	-	-	-
Udacity Annotated Driving [24]	57219	-	1005	3448	-	4081	3328	-
Pedestrian [25]	-	-	-	-	-	785	-	-
Penn-Fudan [26]	-	-	-	-	-	392	-	-
Ukupno	**165007**	**2589**	**11604**	**11948**	**17515**	**6895**	**13354**	**9319**

U prvom koraku izrade „vlastitog“ skupa podataka napravljena je Python skripta koja kombinira sve skupove podataka u jedan. Ovaj korak je vremenski zahtjevan jer ne postoji jedinstven način na koji su svi skupovi podataka označeni pa je potrebno proučiti na koji način su označeni skupovi podatka, a zatim napisati skriptu koja će izvlačiti, odnosno spremati izrezane dijelove slika koje predstavljaju klase koje su označene graničnim okvirom. U ovom koraku moguće je implementirati neki filtar, odnosno kriterij koji će odrediti ide li slika u kombinirani skup podataka ili ne. Kriterij primijenjen za većinu skupova podataka je minimalna širina graničnog okvira od 50 piksela. Ovaj filtar mogao bi se implementirati još lakše u drugom koraku, ali implementacijom u prvom koraku štedi se na prostoru diska zbog velike količine slika.

Drugi korak predstavlja izdvajanje nekog broja slika za svaku od klasa slučajnim odabirom za potrebe izgradnje skupa podataka za učenje i skupa podataka za testiranje. Na temelju broja dostupnih slika odlučeno je koristiti 3000 slika za svaku klasu što se tiče skupa podataka za učenje, odnosno 2500 slika za klasu autobus. Za testiranje, tj. evaluaciju modela odabrano je 100 slika svake klase, odnosno 89 slika za autobus. Time se dobiva prilično ujednačen (engl. *balanced*) skup podataka za učenje i skup podataka za testiranje.

Broj slika za klasu autobus je takav jer ukupno, nakon kombiniranja skupova podataka, postoji samo 2589 slika za klasu autobus te se time ograničava i koliko slika se može uzeti za ostale klase jer se želi izbalansirati da mreža nauči sve klase klasificirati podjednako dobro. Ukoliko bi se izbacila klasa autobus, sljedeće ograničenje predstavljala bi klasa znak, koja ima 9319 slika nakon kombiniranja skupova podataka. Naravno, postoje i metode povećanja postojećeg skupa podataka kao što je augmentacija slika [27], ali za potrebe ovog rada 3000 (2500 za klasu autobus) slika za učenje mreže za pojedinu klasu sasvim je dovoljno.

Skup podataka je generiran tako da postoje dva direktorija: *train* (učenje) i *test* (evaluacija). Unutar tih direktorija nalazi se po osam direktorija za svaku klasu, a unutar tih direktorija nalaze se slike koje su odabrane slučajnim odabirom iz kombiniranog skupa podataka. Kao što je već spomenuto ranije, svaka klasa u direktoriju za učenje ima 3000 slika (2500 za klasu autobus) te 100 slika u direktoriju za testiranje koji služi za evaluaciju modela. Na slici 3.1. prikazan je primjer slika za pojedinu klasu.



Slika 3.1. Primjer slika iz skupa podataka

Na slici 3.1., gdje je prikazan primjer slika iz skupa podataka, vidi se da pojedine slike mogu pripadati i drugim klasama, a ne samo u one koje su označene, pogotovo unutar klasa autobus, kamion i kombi. Ovo proizlazi iz toga što su slike uzete iz različitih skupova podataka gdje neki skup podataka označava neko vozilo kao automobil, dok bi neki drugi skup podataka označio isto to vozilo kao kamion ili kombi. Također, vidi se da bi neke slike i čovjek vrlo teško klasificirao zbog same kvalitete slike.

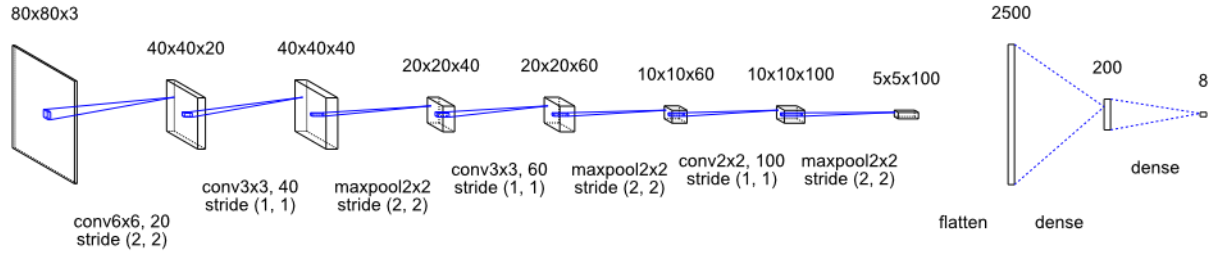
3.1.2. Odabir arhitekture i izrada modela

Kao što je spomenuto ranije, za razvoj modela na računalu koristi se Keras *framework*, a nakon što se razvije zadovoljavajući model isti se prebacuje u Tensorflow.

S obzirom da bi se mreža trebala izvodit na ugradbenom sustavu, bitna stavka dizajna arhitekture jest što manji broj parametara, odnosno što brže vrijeme inferencije, ali bez prevelikog

kompromisa što se tiče preciznosti. To bi značilo da mreža ne treba imati prevelike hardverske zahtjeve, tj. broj slojeva (dubina) mreže i veličina slojeva tih slojeva ne treba biti prevelika.

Odabrana arhitektura za ovaj rad sastoji se od četiri konvolucijska sloja, tri sloja sažimanja po maksimalnoj vrijednosti, dva potpuno povezana sloja od kojih je jedan posljednji sloj koji se aktivira *softmax* aktivacijskom funkcijom. Svi ostali konvolucijski slojevi i jedan potpuno povezani sloj aktiviraju se ReLU aktivacijskom funkcijom, a nakon prvog potpuno povezanog sloja postoji i *dropout* sloj. Ulaz u neuronsku mrežu predstavljaju slike u boji, odnosno slike rezolucije 80×80 s tri kanala (RGB). Arhitektura odabranog modela prikazana je na slici 3.2.



Slika 3.2. Arhitektura odabranog modela

Odabrana arhitektura predstavlja sekvencijalan model, odnosno nema slojeva koji se paralelno izvode. Keras *framework* omogućuje vrlo laku izradu sekvencijalnih modela koristeći *Sequential* konstruktor. Nakon kreiranja objekta *Model* potrebno je samo pozivati *add* metodu *Model* klase kojoj je potrebno predati objekt koji predstavlja pojedini sloj. Konstruktori za kreiranje slojeva su:

- *Conv2D* za dvodimenzionalni konvolucijski sloj,
- *MaxPooling2D* za dvodimenzionalni sloj sažimanja po maksimalnoj vrijednosti,
- *Flatten* za sloj koji izravnava konvolucijski sloj u potpuno povezani (ne kreira dodatni sloj s parametrima, već kreira samo drugačiju prezentaciju sloja),
- *Dropout* za kreiranje *dropout* sloja (koristi se samo kod učenja mreže) te
- *Dense* za kreiranje potpuno povezanog sloja.

Nakon dodavanja svih slojeva potrebno je kompajlirati model koristeći *compile* metodu kojoj kao argumente treba predati funkciju gubitka koja će se koristiti kod učenja mreže, metodu optimizacije koja će se koristiti kod učenja i metriku prema kojoj će se evaluirati uspješnost učenja. U nastavku se nalazi kod kojim se kreira željeni model u Kerasu.

```
model = Sequential()
model.add(Conv2D(20, (6, 6), padding='same', strides=(2, 2),
                 input_shape=(img_width, img_height, 3), activation="relu"))

model.add(Conv2D(40, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(60, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(100, (2, 2), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dropout(0.25))

model.add(Dense(200, activation="relu"))
model.add(Dropout(0.25))

model.add(Dense(num_classes, activation="softmax"))

opt = keras.optimizers.Adam(lr=0.001)

model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

Za učenje mreže korišten je skup podataka koji je napravljen kako je objašnjeno u odjeljku 3.1.1. Keras uz pomoć klase *ImageDataGenerator* omogućuje augmentaciju slika i učitavanje (engl. *feed*) slika iz odabranog direktorija u svrhu učenja mreže. Za svrhu učenja odabranog modela kreiran je *ImageDataGenerator* objekt za učenje. Generatoru je predan argument *rescale* koji označava normalizaciju podataka i dodatni argumenti za augmentaciju slika. Normalizacija podataka, tj. *rescale* argument sa vrijednosti 1/255, predstavlja mapiranje vrijednosti pojedinog elementa matrice slike na broj između 0 i 1. Nakon kreiranja objekata pozvana je metoda *flow_from_directory* koja govori generatoru da slike uzima iz direktorija koji mu je predan kao putanja u argumentu. Kod za izradu generatora nalazi se u nastavku.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    fill_mode="nearest",
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    rotation_range=10)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    seed=104,
    class_mode="categorical")
```

Kako bi se započeo proces učenja na izgrađenom modelu, potrebno je na objektu *Model* pozvati metodu *fit_generator*. Metodi su, kao argumenti, predani generator za učenje, broj epoha i TensorBoard *callback* koji omogućuje nadgledanje modela kako bi se lakše otkrili potencijalni problemi kod učenja mreže. U nastavku je prikazan kod za učenje mreže.

```
1.  tbCallback = TensorBoard(log_dir="./logs/" + model_name,
2.      batch_size=batch_size, write_images=True)
3.
4.  model.fit_generator(
5.      train_generator,
6.      epochs=epochs,
7.      callbacks=[tbCallback])
8.
9.  if not os.path.isdir(save_dir):
10.     os.makedirs(save_dir)
11. model_path = os.path.join(save_dir, model_name)
12. model.save(model_path)
```

U nastavku teksta predložena mreža naziva se KuleNet.

3.2. Učenje i evaluacija modela

Za klasificiranje slika i evaluaciju bilo kojeg klasifikatora jedno od najvažnijih mjerila je preciznost. Preciznost predstavlja omjer točno klasificiranih slika i ukupnog broja slika na kojemu se radio test, a često se prikazuje kao postotak.

$$p = \frac{n_T}{n} \tag{3-1}$$

gdje je:

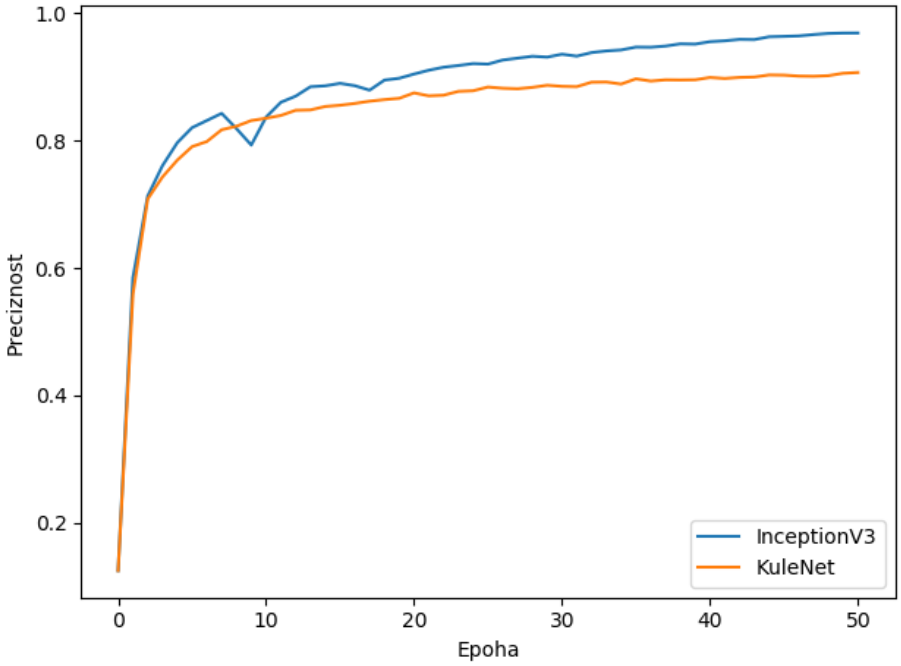
- p – preciznost,
- n_T – broj točno klasificiranih slika,
- n – ukupan broj slika.

Kod evaluacije modela često se koristi matrica zabune (engl. *confusion matrix*). Matrica zabune vizualizacija je koja prikazuje kako je mreža predvidjela klase na nekom skupu podataka. Jedna os u matrici zabune predstavlja točne klase, a druga predviđene. Svaki element u matrici zabune predstavlja broj ili postotak predviđenih klasa. Na glavnoj dijagonali matrice zabune nalaze se točno predviđene klase, dok ostatak matrice zabune predstavlja netočno predviđene klase.

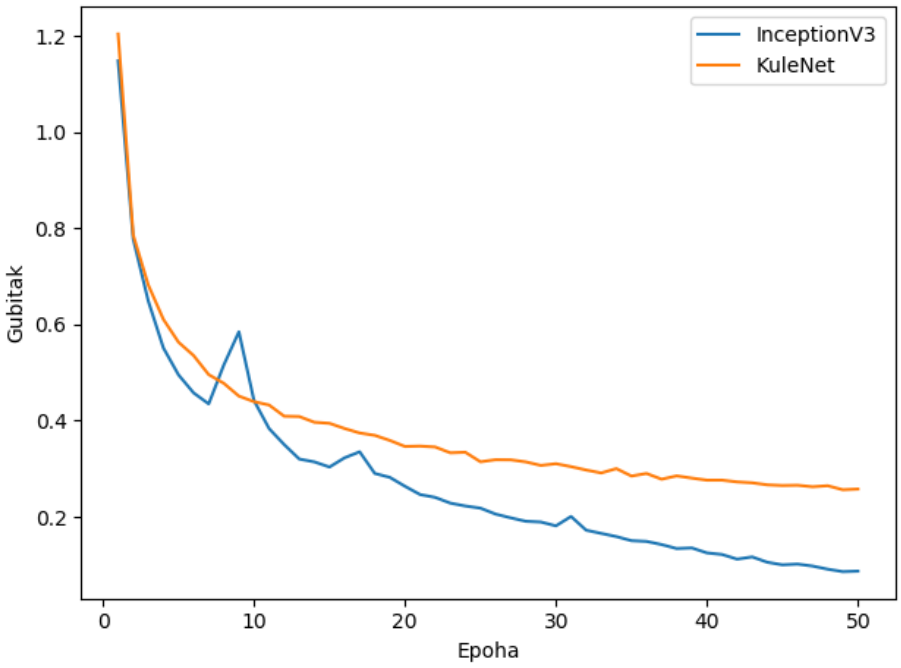
U svrhu usporedbe dobivenih rezultata u ovom radu naučena je i Google Inception V3 mreža kao referentna mreža. Jedina promjena u odnosnu na originalnu Inception V3 mrežu je prilagodba na skup podatka pa je prema tome veličina ulaznih slika 139×139 piksela, a zadnji potpuno povezani

sloj (*softmax* sloj) ima 8 izlaza koliko je i klasa u skupu podataka. Model mreže naučen je u potpunosti ispočetka kao i model KuleNet mreže.

Tijekom procesa učenja praćena je preciznost i gubitak obje mreže na skupu za učenje te su te dvije metrike prikazane i na slikama 3.3. i 3.4.



Slika 3.3. Preciznost mreža tijekom učenja



Slika 3.4. Gubitak mreža tijekom učenja

Nakon 15 epoha učenja KuleNet mreže, postignuta je preciznost od 85,02% i gubitak (gubitak unakrsne entropije) od 0,4064 na skupu podataka za učenje, odnosno 86,19% na testnom skupu podataka. Nakon 50 epoha učenja mreže, postignuta je preciznost od 90,67% i gubitak od 0,2577 na skupu podataka za učenje, odnosno 88,47% na testnom skupu podataka.

Nakon 15 epoha mreža Inception V3 postiže preciznost od 87,32% i gubitak od 0,3513 na skupu podataka za učenje, odnosno 84,79% na testnom skupu podataka. Nakon 50 epoha Inception V3 mreža postiže preciznost od 96,87% i gubitak od 0,087, odnosno 89,73% na testnom skupu podataka.

Za učenje mreže korištena je Nvidia GeForce GTX 1060 6GB grafička kartica. Vrijeme učenja po epohi iznosilo je u prosjeku 37 sekundi za KuleNet mrežu te 108 sekundi za Inception V3 mrežu. Postignuti rezultati, kao i karakteristike izgrađenih mreža, prikazane su u tablici 3.2. Za mjerenje brzine inferencije na grafičkoj kartici (GPU) korištena je već spomenuta Nvidia GeForce GTX 1060 6GB grafička kartica, a za mjerenje brzine inferencije na centralnom procesoru (CPU) korišten je Intel i7-4790 CPU @ 3.60 GHz procesor.

Tablica 3.2. Postignuti rezultati i karakteristike izgrađenih mreža nakon 15 i nakon 50 epoha

	Preciznost		Gubitak (učenje)	Brzina inferencije [ms]		Broj parametara	Veličina modela [MB]
	Učenje	Test		GPU	CPU		
KuleNet [15 epoha]	0,8502	0,8619	0,4064	0,0095	0,0025	556.988	6,43
KuleNet [50 epoha]	0,9067	0,8847	0,2577				
InceptionV3 [15 epoha]	0,8732	0,8479	0,3513	0,1234	0,0235	21.819.176	250,46
InceptionV3 [50 epoha]	0,9687	0,8973	0,0870				

U daljnjem tekstu rezultati se odnose na rezultate dobivene učenjem mreža nakon 50 epoha.

Za današnje pojmove, kada neuronske mreže postižu preciznosti i preko 95%, preciznost od 88,47%, odnosno 89,73% za Inception V3, nije toliko impresivna. Ovo se događa zbog sličnosti u slikama između pojedinih klasa, kao što je i prikazano na slici 3.7. Analizom matrice zabune na podacima za testiranje koja je prikazana na slici 3.5. za KuleNet i analizom matrice zabune na podacima za testiranje za mrežu Inception V3 arhitekture koja je prikazana na slici 3.6. može se

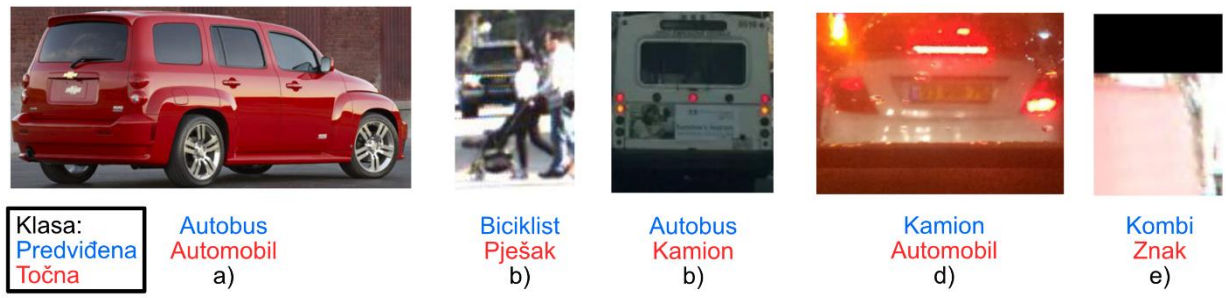
zaključiti da je mreža KuleNet preciznija u klasifikaciji klase autobus, a lošija u klasifikaciji klase automobil u odnosu na Inception V3 mrežu.

Slika 3.5. Matrica zabune KuleNet mreže

Slika 3.6. Matrica zabune InceptionV3 mreže

Također, iz matrica zabune vidi se da mreže u dosta velikom postotku krivo klasificiraju automobil kao znak (25% i 17%) i znak kao automobil (11% i 15%). Iz matrice zabune vidi se i da KuleNet manje griješi kod klasifikacije pješaka, a više kod klasifikacije semafora. Ostale klase podjednako dobro klasificiraju, a pogotovo kod klasa kamion i kombi gdje postižu istu preciznost (100% i 99%).

Analizom slika koje su krivo klasificirane može se uočiti i da su krivo klasificirane slike lošije kvalitete, ili vrlo slične klase (primjer: autobus – automobil). Neke od slika koje su krivo klasificirane prikazane su na slici 3.7.



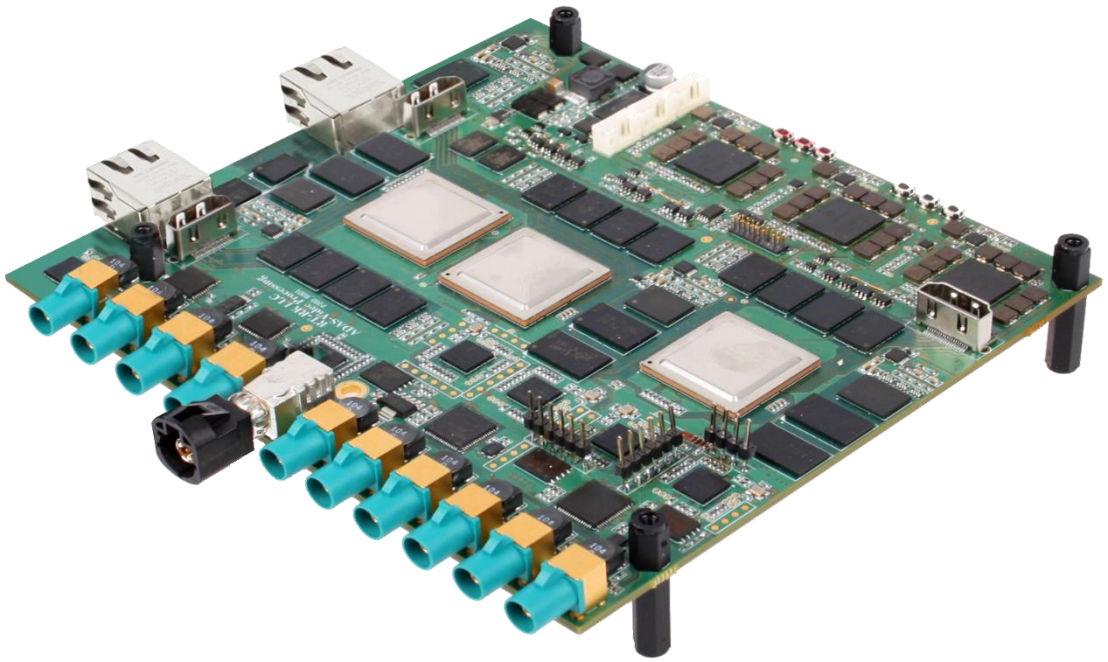
Slika 3.7. Neke od pogrešno klasificiranih slika modela KuleNet

Za ugradbene računalne sustave izuzetno je bitna što manja veličina modela i što kraće trajanje inferencije mreže koja vrši klasifikaciju. Iz tablice 3.2. vidi se da KuleNet s 556.988 parametara ima gotovo 40 puta manje parametara nego Inception V3 mreža postižući neznatno manju preciznost. Manji broj parametara rezultira i puno manjom veličinom modela pa model KuleNet mreže zauzima samo 6,43 MB diskovnog prostora naprema 250,46 MB koliko zauzima model Inception V3 mreže. Brzina, tj. trajanje inferencije također je znatno bolje (kraće) kod KuleNet mreže koja je više od 9 puta brža od Inception V3 mreže kod izvođenja na CPU, tj. oko 13 puta brža kod izvođenja na GPU.

4. IMPLEMENTACIJA RJEŠENJA NA ALPHA PLOČU

Alpha ploča je razvojna ploča namijenjena automobilskoj industriji [28]. Cilj Alpha ploče je ubrzati razvoj *Advanced driver-assistance systems* (ADAS) algoritama te omogućiti pokretanje istih u demonstracijske svrhe. Ciljana namjena jest razvoj algoritama usko vezanih uz autonomnu vožnju i algoritama koji će pomoći vozaču pri vožnji, odnosno olakšati i učiniti vožnju sigurnijom po život čovjeka. Primjeri takvih namjena su: ptičja perspektiva, praćenje stanja vozača, detekcija i klasifikacija objekata, itd.

Glavne komponente Alpha ploče su tri TDA2x SoC-a (*System on Chip*). Jedan TDA2x SoC sastoji se od dvije ARM Cortex A15 jezgre, dvije ARM Cortex M4 jezgre, dva C66x DSP-a (*Digital Signal Processor*) i četiri EVE (*Embedded vision Engine*) jezgre. Također, na ploči su dostupni *Ethernet*, HDMI, DCAN, JTAG, UART portovi te MicroSD utor. Ploča podržava do 10 kamera koje moraju bit kompatibilne s DS90UB964 i DS90UB914A *deserializerima*. Alpha ploča prikazana je na slici 4.1.



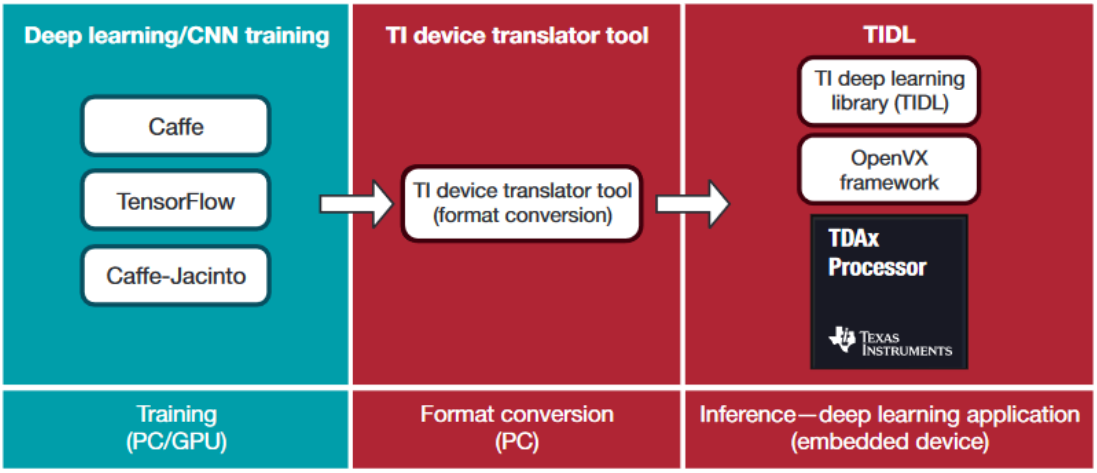
Slika 4.1. Alpha ploča [28]

Postoji više verzija Alpha razvojne ploče, ali sve su relativno slične jer su svakom novom verzijom uvedene samo sitne promjene koje se rješavaju promjenom konfiguracijske datoteke u razvojnom

okruženju. Razvojno okruženje za Alpha razvojnu ploču pruženo je od strane Texas Instrumentsa, a zove se Vision SDK.

Nakon razvoja rješenja na računalu prema poglavlju 3 gdje se razvio model KuleNet mreže, plan je bio prilagoditi mrežu za izvođenje na Alpha razvojnoj ploči. Alpha ploča podržava neuronske mreže kroz *Texas Instruments deep learning library* (TIDL) *framework*. Kako bi se došlo do TIDL modela, koji se može osposobiti na Alpha ploči, potrebno je na računalu naučiti mrežu u Caffe, Tensorflow ili Caffe-Jacinto *frameworku* kako bi alat za prevođenje mogao prevesti naučeni model neuronske mreže u TIDL model. Postupak razvoja TIDL modela iz službenog dokumenta o TIDL *frameworku* prikazan je na slici 4.2.

TIDL *translator tool* dolazi kao dio Vision SDK razvojnog okruženja, kao i dokumentacija za korištenje istog.



Slika 4.2. Postupak razvoja TIDL modela [29]

Iz postupka razvoja TIDL modela može se zaključiti da je za dio razvoja na računalu potrebno odabrati jedan od tri navedena *frameworka*. Kao što je opisano u potpoglavlju 3.1. za razvoj modela na računalu odabran je Keras *framework*. Iako na slici 4.2. Keras nije naveden kao podržan, Keras može koristiti Tensorflow kao *backend framework* zbog čega je moguće Keras modele lako spremiti kao Tensorflow modele. To znači da prvi korak prilagođavanja modela izvođenju na Alpha ploči jest spremanje modela kao Tensorflow model, a nakon toga i korištenjem TIDL *translator tool* alata prevesti model u TIDL model.

Međutim, TIDL *translator tool* nije uspio prevesti izgrađene mreže iz Tensorflowa, već je uvijek javljao grešku tijekom prevođenja. S ovim problemom susretao se dovoljan broj korisnika TIDL

translator toola pa su i postojala moguća rješenja. Razlog za neuspješno prevođenje bio je taj što TIDL *translator tool* još nije bio dovoljno dobro razvijen da uvijek prevede Tensorflow modele bez greške. Zbog same složenosti KuleNet mreže i mreže zasnovane na Inception V3 arhitekturi, ručna implementacija ovih mreža na Alpha ploču iziskivala bi mnogo vremena.

Kako bi se riješio ovaj problem, potrebno je bilo koristiti Caffe-Jacinto *framework*, ali su postojali problemi i prilikom same instalacije *frameworka* tako da ovaj dio rada nije uspio, odnosno implementacija na Alpha ploču bila je bezuspješna.

5. ZAKLJUČAK

Cilj ovog diplomskog rada je napraviti i opisati postupak izrade algoritma koji će raditi klasifikaciju objekata detektiranih ispred vozila pomoću kamere na prednjoj strani vozila. U ovom radu predlaže se klasifikacija koristeći duboke konvolucijske mreže za klasifikaciju slika. U radu je predložena KuleNet mreža koja se sastoji od ulaznog sloja veličine 80×80×3, četiri konvolucijska sloja, tri sloja sažimanja po maksimalnoj vrijednosti, dva potpuno povezana sloja od kojih je jedan (posljednji) izlazni *softmax* sloj. Konvolucijski slojevi koriste 6×6, 3×3 i 2×2 veličine konvolucijskih filtara, a slojevi sažimanja koriste 2×2 filtre. Ukupan broj parametara KuleNet mreže iznosi 556.988 parametara.

Za učenje i testiranje izgrađenih mreža napravljen je poseban skup podataka koji se sastoji od više javno dostupnih skupova podataka. Klase u koje su podijeljene slike su: automobil, autobus, biciklist, kamion, kombi, semafor i znak. Iako kombiniranje različitih skupova podataka poopćuje skup podataka jer dolaze iz različitih izvora, tu također dolazi i do problema. Različiti skupovi podataka različito označavaju objekte, odnosno objekt koji jedan skup podataka označava kao automobil, u drugome može biti označen kao kombi. Oznake, odnosno klase u koje je podijeljen skup podataka nisu dovoljno detaljne da bi se izdvojile karakteristike pojedine klase. Također, skup podataka sadrži mnoštvo slika na kojima se preklapaju objekti, slike su lošije kvalitete ili su objekti tek djelomično uhvaćeni na slici.

Za razvoj modela na računalu korišten je Python programski jezik i Keras *framework*. Model duboke konvolucijske mreže predložen u ovom radu postiže preciznost od 88,47% na skupu podataka za testiranje. Iako preciznost ne prelazi već standardnih 95% koje današnji modeli postižu na različitim problemima klasifikacije objekata, postoje i razlozi tome. Jedan od njih zasigurno je i skup podataka na kojemu je učen model. U usporedbi sa Inception V3 mrežom, predložena mreža ima neznatno lošiju preciznost uz znatno manji broj parametara što je važno sa stajališta brzine inferencije budući da je ideja ovakvu mrežu implementirati na ugradbenu računalnu platformu.

Krajnji cilj ovog rada bio je implementacija predloženog modela na Alpha ploču. Kako bi se model mogao prevesti u TIDL *framework* koji se koristi na Alpha ploči potrebno je bilo najprije prevesti model u Tensorflow model. Prevođenje modela iz Tensorflowa u TIDL nije bilo uspješno. Ovaj problem pokušao se riješiti na način da se mreža nauči u Caffe-Jacitno *frameworku*, a čiji model bi se trebao moći bez problema prevesti u TIDL model, ali ni ovo rješenje nije uspjelo.

LITERATURA

[1] „From not working to neural networking“, *The Economist*, 25-lip-2016. [Na internetu]. Dostupno na: https://www.economist.com/special-report/2016/06/25/from-not-working-to-neural-networking. [Pristupljeno: 27-kol-2018].

[2] D. Gershgorn, „The data that transformed AI research—and possibly the world“, *Quartz*. [Na internetu]. Dostupno na: https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/. [Pristupljeno: 21-ruj-2018].

[3] Ian Goodfellow, Yoshua Bengio, i Aaron Courville, *Deep Learning*. MIT Press.

[4] „A Practical Introduction to Deep Learning with Caffe and Python // Adil Moujahid // Data Analytics and more“. [Na internetu]. Dostupno na: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/. [Pristupljeno: 21-ruj-2018].

[5] „Image Convolution“. [Na internetu]. Dostupno na: http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html. [Pristupljeno: 21-ruj-2018].

[6] Emanuel, „Cross entropy and likelihood: a tender relation that ‚probably‘ works!“, *Go Deep*, 31-sij-2018. [Na internetu]. Dostupno na: http://www.godeep.ml/cross-entropy-likelihood-relation/. [Pristupljeno: 21-ruj-2018].

[7] D. P. Kingma i J. Ba, „Adam: A Method for Stochastic Optimization“, *ArXiv14126980 Cs*, pros. 2014.

[8] „A Brief History Of Neural Network Architectures“, *TOPBOTS*, 09-lip-2017. [Na internetu]. Dostupno na: https://www.topbots.com/a-brief-history-of-neural-network-architectures/. [Pristupljeno: 29-kol-2018].

[9] S. H. Tsang, „Review: AlexNet, CaffeNet — Winner of ILSVRC 2012 (Image Classification)“, *Medium*, 09-kol-2018. [Na internetu]. Dostupno na: https://medium.com/coinmonks/paper-review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160. [Pristupljeno: 21-ruj-2018].

[10] T. Dixit, „Rethinking the Inception Architecture for Computer Vision — Part 1“, *Medium*, 16-srp-2018. [Na internetu]. Dostupno na: https://medium.com/coinmonks/rethinking-the-inception-architecture-for-computer-vision-part-1-2938cc7c7872. [Pristupljeno: 29-kol-2018].

[11] „Advanced Guide to Inception v3 on Cloud TPU | Cloud TPU“, *Google Cloud*. [Na internetu]. Dostupno na: https://cloud.google.com/tpu/docs/inception-v3-advanced. [Pristupljeno: 21-ruj-2018].

[12] „CompCars Dataset“. [Na internetu]. Dostupno na: http://mmlab.ie.cuhk.edu.hk/datasets/comp_cars/index.html. [Pristupljeno: 21-ruj-2018].

[13] R. You i J.-W. Kwon, *VoNet: vehicle orientation classification using convolutional neural network*. 2016.

[14] „TensorFlow“. [Na internetu]. Dostupno na: https://www.tensorflow.org/. [Pristupljeno: 21-ruj-2018].

[15] „Keras Documentation“. [Na internetu]. Dostupno na: https://keras.io/. [Pristupljeno: 21-ruj-2018].

[16] V. A. Prisacariu, R. Timofte, K. Zimmermann, I. Reid, i L. van Gool, „Integrating object detection with 3D tracking towards a better driver assistance system“, u *Twentieth International Conference on Pattern Recognition*, Istanbul, Turkey, 2010, str. 1–4.

[17] K. Behrendt, L. Novak, i R. Botros, „A deep learning approach to traffic lights: Detection, tracking, and classification“, u *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, str. 1370–1377.

[18] J. Krause, M. Stark, J. Deng, i L. Fei-Fei, „3D Object Representations for Fine-Grained Categorization“, u *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[19] X. Li *i ostali*, „A new benchmark for vision-based cyclist detection“, u *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, str. 1028–1033.

[20] J. Stallkamp, M. Schlipsing, J. Salmen, i C. Igel, „Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition“, *Neural Netw.*, izd. 0, str. , 2012.

[21] „Are we ready for autonomous driving? The KITTI vision benchmark suite - IEEE Conference Publication“. [Na internetu]. Dostupno na: https://ieeexplore.ieee.org/document/6248074/. [Pristupljeno: 21-ruj-2018].

[22] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, i M. M. Trivedi, „Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives“, *IEEE Trans. Intell. Transp. Syst.*, sv. 17, izd. 7, str. 1800–1815, srp. 2016.

[23] „Nexar Challenge #2 - Nexar“. [Na internetu]. Dostupno na: https://www.getnexar.com/challenge-2/. [Pristupljeno: 16-ožu-2018].

[24] *The Udacity open source self-driving car project. Contribute to udacity/self-driving-car development by creating an account on GitHub*. Udacity, 2018.

[25] „Pedestrian Dataset“. [Na internetu]. Dostupno na: https://www.kaggle.com/sudipdas/pedestriandataset. [Pristupljeno: 19-tra-2018].

[26] „Pedestrian Detection Database“. [Na internetu]. Dostupno na: https://www.cis.upenn.edu/~jshi/ped_html/. [Pristupljeno: 21-ruj-2018].

[27] B. Raj, „Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2“, *Medium*, 11-tra-2018. .

[28] „RT-RK - Automotive“. [Na internetu]. Dostupno na: http://www.rt-rk.com/services/automotive. [Pristupljeno: 21-ruj-2018].

[29] M. Mathew, K. Desappan, P. K. Swami, S. Nagori, i B. M. Gopinath, „Embedded low-power deep learning with TIDL“, str. 8, 2018.

SAŽETAK

U današnje doba pokušava se postići što veća autonomija vozila, a za to su zaduženi napredni sustavi za pomaganje vozaču (ADAS, engl. *Advanced driver-assistance systems*). Krajnji cilj razvoja jednog takvog sustava jest omogućiti potpunu autonomiju vozila. Kako bi jedan takav sustav mogao donositi odluke, treba raspolagati informacijama koje pružaju uvid u okolinu vozila, tj. objekte koji okružuju automobil. Prilikom normalne vožnje (unaprijed), sustavu je najbitnije raspolagati informacijama o objektima koji se nalaze ispred vozila.

Ovaj rad se bavi klasifikacijom objekata detektiranih s prednje strane vozila gdje ulaz u algoritam klasifikacije predstavljaju slike objekata. Diplomski rad opisuje klasifikaciju objekata koristeći duboke konvolucijske mreže te je kao dio rada opisan postupak izrade jedne takve mreže, od prikupljanja slika koje predstavljaju podatke pa do samog učenja mreže. Također, opisani su i pojedini slojevi od kojih se takva duboka neuronska mreža sastoji. U radu je izvršena evaluacija izrađenih mreža i razmatrani su problemi koji se mogu stvoriti kod krive klasifikacije i zašto do njih dolazi.

Ključne riječi: ADAS, klasifikacija, neuronska mreža, duboko učenje, konvolucija

ABSTRACT

Classification of objects detected in front of the vehicle using camera mounted on front side of the vehicle

Nowadays, we are trying to achieve as much vehicle autonomy as possible by developing Advanced driver-assistance systems (ADAS). The ultimate goal of developing such a system is allowing complete autonomy of the vehicle. In order for such a system to make decisions, it should have information that provides insight into the environment of the vehicle, i.e. the objects surrounding the car. During forward driving, it is most important for the system to have information about the objects in front of the vehicle.

This paper describes the classification method of using deep convolutional neural networks of the objects detected in the front of the vehicle. The input into the classification algorithm is represented by the image of the object. This paper describes the process of making such a network, from collecting and grouping the images that represent the data up to the learning of the network itself. This paper also describes individual layers which comprise such a deep neural network. In this paper, an evaluation of created networks has been carried out and the paper also discusses why and what problems could occur when using such a method for classification.

Keywords: ADAS, classification, neural network, deep learning, convolution

ŽIVOTOPIS

Filip Kulić rođen je 25.01.1995. godine u Osijeku. Nakon završene Osnovne škole Jagode Truhelke, upisuje Elektrotehničku i prometnu školu Osijek, smjer tehničar za računalstvo gdje maturira 2013. godine. Iste godine upisuje sveučilišni preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U rujnu 2016. godine završava preddiplomski studij i upisuje sveučilišni diplomski studij računarstva.

Potpis:

_____