Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Boris Karamatić

# Prepoznavanje prometnih znakov z uporabo globokih konvolucijskih nevronskih mrež

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana, 2016

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Detekcija ter razpoznavanje prometnih znakov postajata vse bolj aktualni kompetenci namenskih sistemov računalniškega vida, bodisi tistih v samovozečih avtomobilih ali v sistemih za vodenje evidence prometne signalizacije. Naloga takšnih sistemov je detektirati in razpoznati prometne znake na slikah oz. v video sekvencah. V diplomski nalogi razvijte tak sistem, ki bo na sliki najprej poiskal vse lokacije, na katerih se z veliko verjetnostjo nahajajo prometni znaki, nato pa bo te lokacije preveril ter razpoznal prometne znake oz. jih uvrstil v primerno kategorijo. Za razpoznavanje uporabite globoke konvolucijske nevronske mreže. Zasnujte primerno arhitekturo mreže ter za učenje uporabite dovolj veliko število učnih primerov. Dobljeni klasifikator evalvirajte na standardnih bazah slik prometnih znakov.

*Zahvaljujem se mentorju izr. prof. dr. Danijelu Skočaju za vodstvo ter pomoč pri izdelavi diplomskega dela.*

*Zahvaljujem se tudi svoji družini, še posebej staršema, za večno podporo in spodbudo.*

# Kazalo

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **CNN** | convolutional neural network | konvolucijska nevronska mreža |
| **SVM** | support vector machine | metoda podpornih vektorjev |
| **ROI** | region of interest | regija zanimivosti |
| **ACF** | aggregate channel features | značilnice iz agregiranih kanalov |
| **GTSRB** | German Traffic Sign Recognition Base | Nemška baza za prepoznavanje prometnih znakov |
| **GTSDB** | German Traffic Sign Detection Base | Nemška baza za detekcijo prometnih znakov |
| **BTSC** | Belgium Traffic Sign Dataset for Classification | Belgijska baza za prepoznavanje prometnih znakov |

# Povzetek

**Naslov:** Prepoznavanje prometnih znakov z uporabo globokih konvolucijskih nevronskih mrež

**Avtor:** Boris Karamatić

Problema detekcije in prepoznavanja prometnih znakov postajata pomemben problem pri razvoju samovozečih vozil ter naprednih sistemov za asistenco vozniku. V diplomski nalogi bomo razvili sistem za detekcijo in prepoznavanje prometnih znakov. Za problem detekcije bomo uporabili značilnice iz agregiranih kanalov, za problem prepoznavanja pa globoko konvolucijsko nevronsko mrežo. Opisali bomo kako konvolucijske nevronske mreže delujejo, kako so zgrajene ter razložili pomen posameznih nivojev. Razložili bomo pristop, katerega smo uporabili pri razvoju konvolucijske nevronske mreže. Končne rezultate detekcije in klasifikacije bomo evalvirali na uveljavljenih bazah prometnih znakov.

**Ključne besede:** konvolucijska nevronska mreža, prometni znaki, detekcija, prepoznavanje, klasifikacija.

# Abstract

**Title:** Traffic sign recognition with deep convolutional neural networks

**Author:** Boris Karamatić

The problem of detection and recognition of traffic signs is becoming an important problem when it comes to the development of self driving cars and advanced driver assistance systems. In this thesis we will develop a system for detection and recognition of traffic signs. For the problem of detection we will use aggregate channel features and for the problem of recognition we will use a deep convolutional neural network. We will describe how convolutional neural networks work, how they are constructed and we will explain the use of every layer. We will describe the steps we took to develop our convolutional neural network. We will evaluate the results of detection and classification on established traffic sign datasets.

**Keywords:** convolutional neural network, traffic signs, detection, recognition, classification.

# Poglavje 1

# Uvod

## 1.1 Motivacija

Detekcija in prepoznavanje prometnih znakov predstavljata v današnjem času pomemben problem in so temelj pri razvoju samovozečih vozil ter naprednih sistemov za asistenco voznika. Najpogostejši razlogi, zakaj vozniki ne opazijo prometnih znakov, ležijo v voznikovi nepazljivosti, ali pa prisotnosti motenj, ki voznikovo pozornost odvrnejo od gledanja prometnih znakov. Avtomatizacija detekcije in klasifikacije prometnih znakov (slika 1.1) bi pripomogla k zmanjšanju števila prometnih nesreč [10].

Detekcija in prepoznavanje prometnih znakov veljata za dobro razumljiv problem, saj so barva, oblika in višina prometnih znakov znani in več ali manj konstantni. Tudi postavitev slednjih je taka, da so vozniku lahko vidni. Kljub temu pa je njihova avtomatska detekcija problematična. Razlog za to leži v tem, da je zorni kot gledanja na znake v vsakdanjem življenju različen. Prav tako se tudi njihova barva spreminja odvisno od osvetlive (sonce, senca), vremena (sonce, dež, sneg, megla) in časa (dan, noč, jutro, večer). Znaki so lahko tudi delno zakriti (drevesa, veje). Na njih so lahko prisotni grafiti, nalepke, fizične poškodbe, odbleski sonca, sprane barve in še mnoge druge spremenljivke [23, 29]. Nekaj primerov lahko vidimo na sliki 1.2.

Slika 1.1: Primer pravilno detektiranih in prepoznanih prometnih znakov v videu posnetim med vožnjo z avtomobilom.

## 1.2 Cilji diplomske naloge

Cilj diplomske naloge je razviti sistem, ki bo sposoben detekcije in klasifikacije prometnih znakov v videu, posnetim med vožnjo z avtomobilom. Znake bo na slikah morali najprej detektirati, nato jih iz slike ekstraktati in jih na koncu še prepoznati. Da bomo to dosegli, bomo potrebovali metodo, ki bo sposobna detekcije prometnih znakov ter metodo, ki bo sposobna klasifikacije slednjih. Za ta namen bomo naučili globoko konvolucijsko nevronsko mrežo prepoznavanja prometnih znakov.

## 1.3 Sorodna dela

Problem detekcije prometnih znakov v inteligentnih vozilih je bil popularen že od sredine 1990-tih let naprej. Skozi čas so bile za rešitev tega problema

Slika 1.2: Primeri problematičnih prometnih znakov. V prvem stolpcu so znaki, ki imajo na sebi nalepljene nalepke, v drugem stolpcu vidimo odbleske sonce, ki otežijo prepoznavanje znakov. V tretjem stolpcu je prisoten drugačen zorni kot gledanja na znake. V četrtem stolpcu so znaki zajeti pri slabih osvetlitvenih pogojih. V zadnjem stolpcu sta dva primera delno zakritih znakov.

uporabljene mnoge različne metode [29].

Prometni znaki so točno določenih barv in zato so bile prve metode za njihovo detekcijo in klasifikacijo bazirane na barvah. Za ločitev prometnih znakov od ozadja so uporabljale pragove [5, 18, 19]. Nekatere metode so namesto barvnega prostora RGB uporabljale barvni prostor HSI [12] in dosegale dobre rezultate. Največja slabost pristopov, ki temeljijo na barvah prometnih znakov je, da je zaradi spremenljive osvetljivosti prometnih znakov, težko določiti pravilno vrednost praga [29].

Popularne so bile tudi metode, ki so temeljile na oblikah prometnih znakov. Nekatere so temeljile na iskanju robov prometnih znakov [15], nekatere so uporabljale Houghovo transformacijo [7, 8]. Za detekcijo okroglih prometnih znakov so se razvile metode, ki so uporabljale glajenje ter Laplaceov filter [1]. Ker so pri detekciji prometnih znakov uporabne tako metode, ki temeljijo na barvah, kot tudi metode, ki temeljijo na oblikah, so se razvile metode, ki so uporabljale oboje hkrati [18]. V zadnjih letih zaradi hitrosti in zanesljivosti postajajo vedno bolj uporabljene značilnice iz agregiranih

kanalov (ang. aggregate channel features, v nadaljevanju ACF) [30].

Po uspešni detekciji prometnega znaka, ga je potrebno še klasificirati. Klasični načini reševanja problema klasifikacije prometnih znakov so bili z uporabo Bayesovega klasifikatorja [17], boostinga [21], klasifikacijskih dreves [31] in metode podpornih vektorjev [9]. Vse te metode imajo skupno to, da uporabljajo ročno zasnovane značilnice. Problem pa je, da ročno zasnovati slednje vzame ogromno časa in prav tako ne moremo vedeti v naprej katere značilnice bodo za dani problem dejansko uporabne [13].

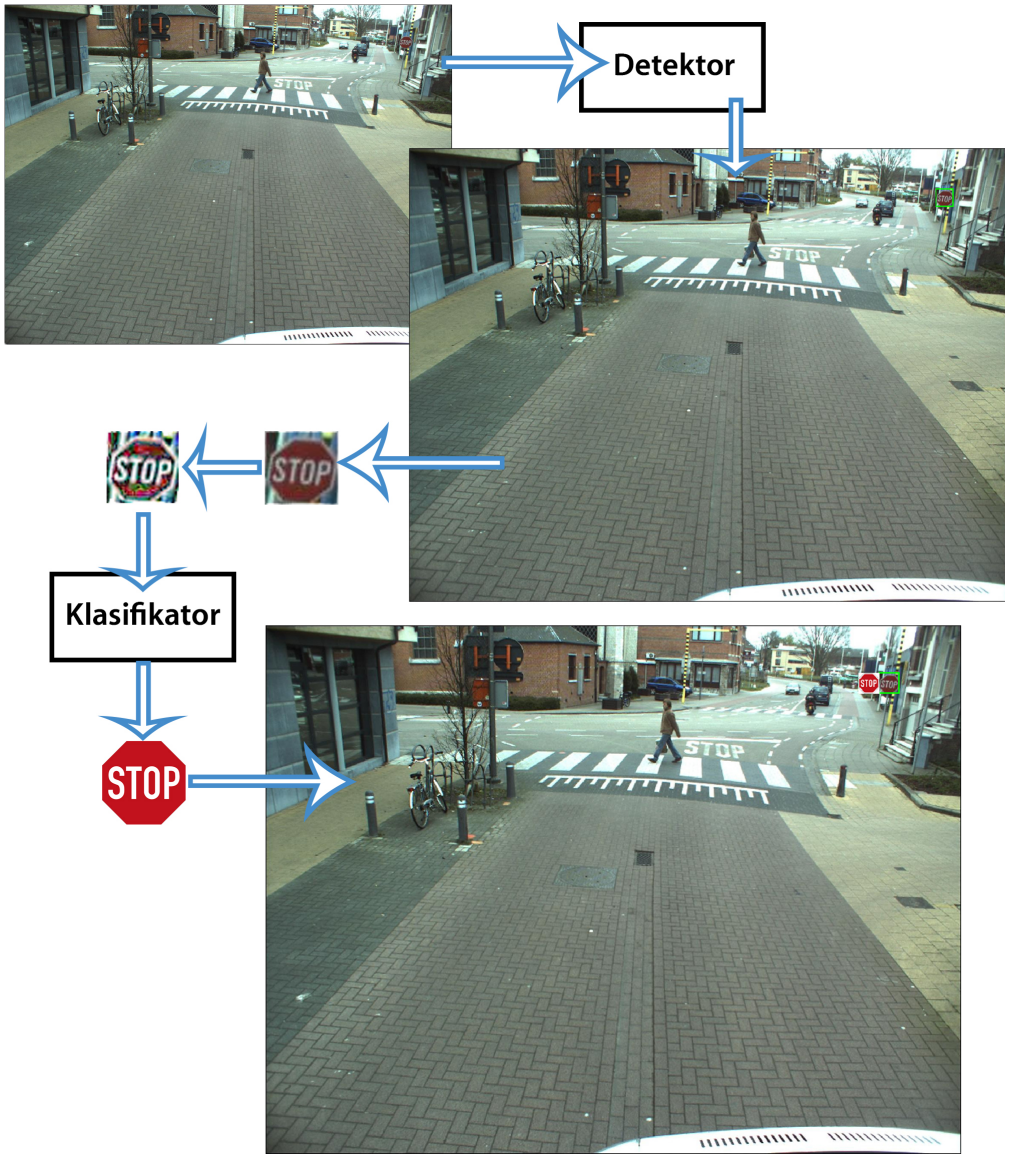
Zaradi tega postajajo v zadnjih letih vedno bolj popularne konvolucijske nevronske mreže (CNN) [3, 4, 10, 13, 23, 27, 29, 32]. Njihova prednost je v tem, da je njihov vhod kar slika in ne ročno zasnovana značilnica. Konvolucijske nevronske mreže so sposobne avtomatske ekstrakcije značilnic in njihove klasifikacije. CNN se obojega nauči v fazi nadzorovanega učenja. Naučeno je ponavadi za to specifično nalogo zelo robustno [13].

## 1.4 Metodologija

Problem prepoznavanja prometnih znakov v videih je dvostopenjski problem. Deli se na problem detekcije in problem prepoznavanja prometnih znakov. Vhodno sliko najprej pošljemo v detektor znakov, kateri nam vrne lokacije prometnih znakov. Znake je nato potrebno izrezati iz vhodne slike, jih obdelati in na koncu še poslati v klasifikator. Ta nam pove kateri znaki se nahajajo na sliki. Kako poteka ta proces, si lahko ogledamo na sliki 1.3. Slike v videih so dandanes velike in posledično je možnih lokacij, kje bi se prometni znaki lahko nahajali ogromno. Potrebovali bomo neko metodo, ki bo sposobna hitrega in zanesljivega iskanja prometnih znakov. Za ta namen bomo uporabili značilnice iz agregiranih kanalov. Ko bomo prometne znake našli, jih bomo morali nato še prepoznati. Za ta namen bomo razvili globoko konvolucijsko nevronsko mrežo, katero bomo naučili klasifikacije prometnih znakov. V diplomski nalogi se bomo bolj posvetili drugemu delu, to je razpoznavanju znakov z uporabo globoke konvolucijske nevronske mreže.

## 1.5 Zgradba diplomskega dela

V 2. poglavju se bomo najprej posvetili teoretični razlagi konvolucijskih nevronskih mrež. Opisali bomo njihove nivoje ter razložili kako poteka njihovo učenje. V 3. poglavju se bomo posvetili implementaciji. Razložili bomo, kako smo se lotili detekcije prometnih znakov in razvoja konvolucijske nevronske mreže. V 4. poglavju se bomo posvetili eksperimentom. Razložili bomo postopek pridobivanja in obdelave podatkov. Tu bomo tudi pokazali končno zgradbo konvolucijske nevronske mreže. Na koncu bomo predstavili še rezultate detekcije in klasifikacije. V 5. poglavju bomo podali kratek zaključek.




Slika 1.3: Vhodno sliko pošljemo v detektor prometnih znakov, kateri nam vrne lokacije znakov. Te nato izrežemo iz slike, jih obdelamo ter pošljemo v klasifikator. Ta nam pove, kateri znak se nahaja na sliki.

# Poglavje 2

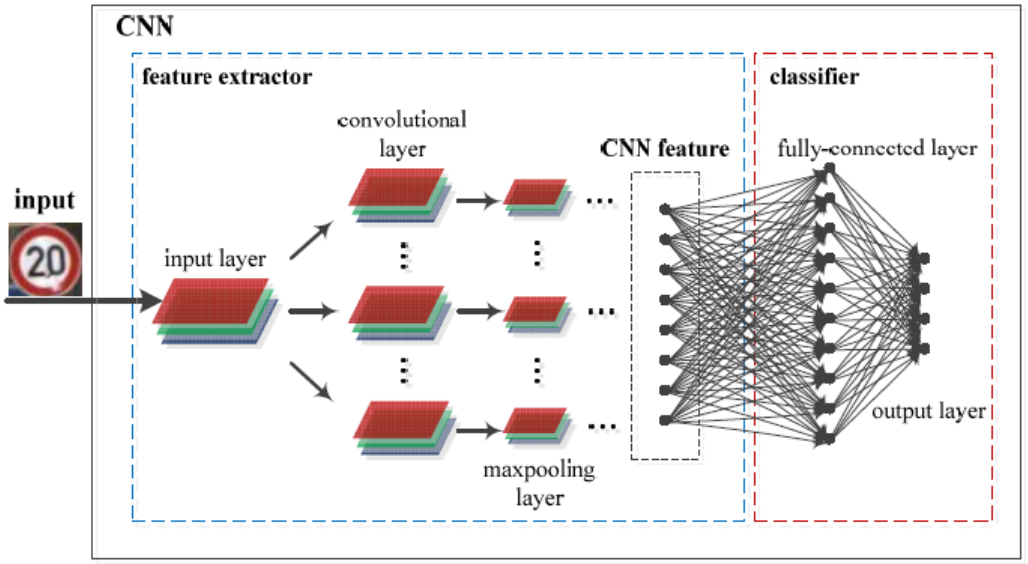
# Konvolucijske nevronske mreže

Za začetek razbijmo celotno ime globokih konvolucijskih nevronskih mrež in poglejmo kaj jih sestavlja.

Ime nevronske mreže se je razvilo iz dejstva, da delovanje in struktura slednjih spominja na povezovanje nevronov v možganih. Čisto osnovna zgradba nevronskih mrež je izmenjavanje konvolucijskih nivojev in nivojev podvzorčenja. Ta struktura spominja na izmenjavanje preprostih in kompleksnih celic primarne vidne skorje v možganih [3]. Ime konvolucijske nevronske mreže izhaja iz dejstva, da so prisotni konvolucijski nivoji. Globoke pa se jim reče zato, ker nista prisotna samo en konvolucijski nivo in en nivo podvzorčenja, temveč jih je več.

Načeloma lahko rečemo, da so CNN-ji sestavljeni iz dveh glavnih delov. Iz ekstraktorja značilk ter iz klasifikatorja (slika 2.1). Ekstraktor značilk vsebuje izmenjujoče se konvolucijske nivoje ter nivoje Max-pooling. Vhod v vsak nivo je izhod iz prejšnjega. Kot rezultat, to ustvari hierarhično zgrajen ekstraktor značilk, ki vhodno sliko mapira v vektor značilk. Ta vektor značilk je nato klasificiran v drugem delu, to je v polno povezanih nivojih, ki sestavljajo tipično usmerjeno nevronsko mrežo [32].

Posamezen CNN je sestavljen iz več nivojev, pri tem, da je vsak nivo sestavljen iz več map ter parametrov. Prisotni parametri so filter, njegova velikost, velikost obrobe ter korak preskakovanja. Filtri z drsečim oknom




Slika 2.1: Arhitektura konvolucijske nevronske mreže. Ekstraktor značilk je obrobljen z modro barvo, klasifikator pa je obrobljen z rdečo barvo [32].

drsijo v smeri x in y. Ko filtri drsijo, se ponavadi ne premikajo po en slikovni element, temveč se premikajo po korakih, določenimi s korakom preskakovanja [13].

V nadaljevanju bomo predstavili vsak nivo posebej.

## 2.1 Konvolucijski nivo

Konvolucijski nivo je določen s številom map, njihovo velikostjo, velikostjo obrobe, velikostjo filtra in korakom preskakovanja. Vsak nivo ima M map enake velikosti ($M_x$, $M_y$) [3]. Vsak konvolucijski nivo izvede nad temi mapami 2D konvolucijo s filtrom velikosti ($K_x$, $K_y$) [4]. Filter mora biti med premikanjem po vhodni sliki stalno popolnoma znotraj slike. Koraka preskakovanja $S_x$ in $S_y$ določata po koliko slikovnih enot filter preskoči v smereh $x$ in $y$ med zaporednimi konvolucijami. Velikost izhodne mape je definirana z enačbo (2.1), kjer indeks $n$ nakazuje nivo. V našem primeru sta višina in širina vhodnih map enaki, posledično se enačba poenostavi na (2.2). Vsaka

mapa $M^n$ v nivoju $L^n$ je povezana z vsemi $M^{n-1}$ mapami iz nivoja $L^{n-1}$ [3].

$$M_x^n = \frac{M_x^{n-1} - K_x^n + 2 * O_x^n}{S_x^n} + 1; M_y^n = \frac{M_y^{n-1} - K_y^n + 2 * O_y^n}{S_y^n} + 1 \quad (2.1)$$

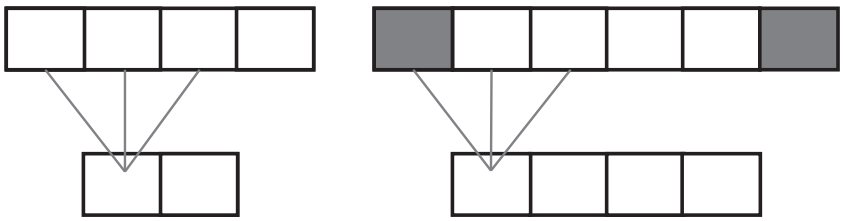
$$M_{x,y}^n = \frac{M_{x,y}^{n-1} - K_{x,y}^n + 2 * O_{x,y}^n}{S_{x,y}^n} + 1 \quad (2.2)$$

Cilj konvolucijskega nivoja je ekstrakcija lokalnih vzorcev. Filtri v CNN so naključno inicializirani in se bodo samostojno naučili postati detektorji robov, barv in specifičnih vzorcev. Filtri znotraj enega nivoja so vedno enake velikosti. Velikost filtrov pa se lahko razlikuje med različnimi nivoji. Filter bo z drsenjem po osi $x$ in osi $y$ konvoluiral vhodno mapo. Število filtrov v nivoju predstavlja število kanalov izhodne mape tega nivoja [13].

Korak preskakovanja nam pove po koliko slikovnih enot premaknemo filter med obdelavo vhodne mape. Pri konvolucijskih nivojih ga najpogosteje puščamo na vrednosti 1, saj želimo, da filter med obdelavo slike ne spušča slikovnih enot. Kot pomemben gradnik bo nastopil pri naslednjem nivoju, to je nivoju Max-pooling.

Drugi osnovni gradnik konvolucijskega nivoja je obroba. Obroba nam pove, koliko dodatnih slikovnih enot okoli mape se ustvari. Njihova vrednost je 0, omogočajo pa natančnejšo obdelavo map. Pri obdelavi slike v konvolucijskem nivoju si želimo, da je izhodna mapa enake velikosti kot vhodna. Dodatek obrobe povzroči večjo izhodno mapo in s tem zmanjšamo ali pa celo preprečimo izgubo dimenzij v konvolucijskem nivoju. Na sliki 2.2 vidimo, da dodatek obrobe omogoči natančnejšo obdelavo mape, saj filtri lahko natančneje obdelajo slikovne vrednosti, ki se nahajajo na vogalih, prav tako pa pri izhodni mapi ohranimo velikost vhodne mape.

Med vsemi nivoji so nivoji Max-pooling zadolženi za podvzorčenje. V konvolucijskih nivojih ne želimo, da bi prihajalo do podvzorčenja, saj bi bilo podvzorčenje prehitro. Zato moramo v konvolucijskih nivojih nastaviti korak preskakovanja na 1, velikost jedra na liho število in dodati obrobo okoli vhodne mape. Pri tem je velikost obrobe enaka $\lfloor velikostFiltra/2 \rfloor$ in



Slika 2.2: Slika prikazuje učinek dodatka obrobe. Zgornji del slike predstavlja vhodno mapo, spodnji del pa izhodno. Na levi sliki vidimo obdelavo vhodne mape s filtrom velikosti 3, kar povzroči izhod velikosti 2. Izhod ima manjšo velikost od vhoda, kar si ne želimo. Na desni sliki vidimo dodatek obrobe velikosti 1 na obeh straneh mape. To povzroči izhod velikosti enake vhodu.

vse vrednosti obrobe so nič [13]. Z enačbo (2.3) lahko vidimo, da izhodna mapa resnično ohrani velikost vhodne mape.

$$\begin{aligned} M^n_{x,y} &= \frac{M^{n-1}_{x,y} - K^n_{x,y} + 2 * O^n_{x,y}}{S^n_{x,y}} + 1 \\ &= \frac{M^{n-1}_{x,y} - K^n_{x,y} + 2 * \lfloor K^n_{x,y}/2 \rfloor}{1} + 1 \\ &= M^{n-1}_{x,y} - (K^n_{x,y} + 2 * \lfloor K^n_{x,y}/2 \rfloor + 1) \\ &= M^{n-1}_{x,y} \end{aligned} \tag{2.3}$$

## 2.2 Nivo Max-pooling

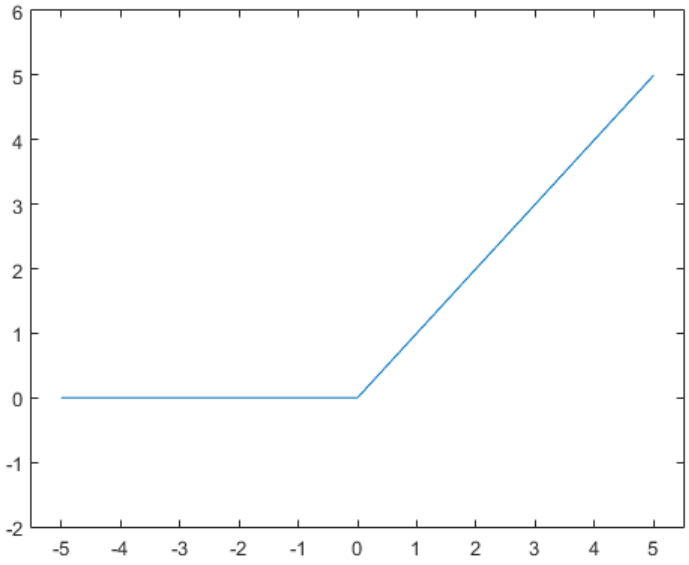
Naloga konvolucijskega nivoja je ekstrakcija lokalnih značilk. Pri izvedbi klasifikacije, je natančna lokacija značilk škodljiva za zmogljivost. Razlog leži v tem, da bodo različne učne instance, z enakim razredom, imele različne točne lokacije značilk. Uporabna metoda za reševanje tega problema je podvzorčenje značilk, s čim zmanjšamo resolucijo mape značilk [13]. Pri CNN-jih tej operaciji rečemo združevanje. Najpogostejši metodi združevanja sta Average-pooling ter Max-pooling. V [22] so pokazali, da je najučinkovitejša metoda združevanja Max-pooling, in to metodo bomo uporabili tudi mi.

Naloga nivoja Max-pooling je torej zmanjšati velikost mape in s tem zmanjšati lokacijsko odvisnost značilk [22]. Na operacijo Max-pooling lahko gledamo tudi kot na operacijo s filtrom. Njeni osnovni gradniki so velikost filtra ($K_x$, $K_y$) ter korak preskakovanja. Na določeni lokaciji operira filter samo na enem kanalu vhodne mape in sicer tako, da izbere največji element na lokaciji, ki jo filter prekriva. Ta element postane element izhodne mape. Filter se nato premakne za vrednost koraka preskakovanja in tam se znova poišče največja vrednost. To se ponavlja, dokler ni obdelana celotna vhodna mapa. Torej operacija Max-pooling podvzorči vsak kanal vhodne mape ter shranjuje lokalne maksimalne vrednosti, medtem ko se majhne vrednosti izgubijo. Nivo Max-pooling ustvari pozicijsko neodvisnost na večjih lokalnih regijah [3] in zmanjša izhodno mapo. Velikost izhodne mape lahko izračunamo z enačbo (2.2). Število kanalov izhodne mape pri tem ostane enako številu kanalov vhodne mape [13]. Izhod iz nivoja Max-pooling so tako najmočnejše aktivacije na vhodni mapi [3, 4].

## 2.3 Nivo ReLu

Konvolucijski nivo se obnaša kot linearen filter. Razlog, da potrebujemo aktivacijsko funkcijo, je vpeljevanje nelinearnosti v nevronsko mrežo. Nelinearnost pomeni, da izhod ne more biti poustvarjen z linearno kombinacijo vhodov. Brez vpeljave nelinearnosti bi se nevronska mreža, ne glede na njeno globino, obnašala kot enoslojni perceptron. V našem primeru smo za nelinearno transformacijo uporabili rectified linear unit (ReLu) (enačba (2.4)), čigar obliko lahko vidimo na sliki 2.3. Glede na [13], da funkcija ReLu enake rezultate kot tradicionalne nelinearne funkcije, kot sta na primer sigmoid ter tanh, vendar pa je učenje z ReLu šestkrat hitrejše.

$$f(x) = max(0, x) \tag{2.4}$$




Slika 2.3: Oblika funkcije ReLu.

## 2.4 Nivo Full-Link

Skupino konvolucijskega nivoja, nivoja Max-pooling ter nivoja ReLu imenujemo en blok. CNN je ponavadi sestavljen iz več blokov. Po prehodu skozi več blokov, so vhodne slike spremenjene v veliko število nizko resolucijskih map značilk. Te značilke se nato združijo v en dolg vektor. Ta vektor igra isto vlogo, kot ročno zasnovane značilke, in je poslan v skriti nivo nevronske mreže. Ta vektor značilk je polno povezan s skritim nivojem. Po nivoju Full-link pride še zadnjič nivo ReLu [13].

## 2.5 Nivo Soft-Max

Nivo Soft-max je zadnji nivo CNN-ja in predstavlja njegov klasifikacijski nivo. S pravilnim izbiranjem velikosti jedra konvolucijskih filtrov, velikosti obrobe, velikosti Max-pooling filtrov ter koraka preskakovanja, lahko dosežemo, da je izhodna mapa zadnjega konvolucijskega nivoja podvzorčena na velikost enega

samega slikovnega elementa na mapo. Izhod iz zadnjega konvolucijskega nivoja je tako poslan v nivo Soft-max, ki je zadnji polno povezani nivo z enim izhodom na razred [3, 4]. Pri čemer vsaka vrednost vektorja predstavlja verjetnost, da slika pripada temu razredu. Posledično je tudi vsota vektorja enaka 1 [13]. Kateremu razredu pripada vhodna slika, se odločimo glede na najvišjo vrednost v vektorju.

## 2.6 Potek učenja nevronske mreže

V procesu nadzorovanega učenja želimo prilagoditi nevronsko mrežo tako, da bodo njeni izhodi ($\hat{Y}$) bili čim bližje našim ciljnim izhodom ($Y$) naše učne množice, ki vsebuje $T$ različnih slik. Cilj je prilagoditi parametre nevronske mreže tako, da bo dala dobre rezultate tudi nad slikami izven učne množice [28].

Če želimo npr. klasificirati neko sliko $t$, bomo želeli, da nam nevronska mreža zanjo vrne izhod $Y(t)$. $Y(t)$ je tako naš ciljni izhod, dejanski izhod iz mreže pa bo $\hat{Y}(t)$, kateri bo lahko različen od našega ciljnega izhoda $Y(t)$. Da rešimo ta problem nadzorovanega učenja moramo določiti neko topologijo za našo nevronsko mrežo, ki dobi vhod $X(t)$ in da izhod $\hat{Y}(t)$, ki je aproksimacija $Y(t)$. Relacija med vhodi in izhodi mora biti odvisna od uteži $W$, katere moramo biti sposobni prilagajati. Določiti moramo učno pravilo, ki bo popravljalo uteži $W$ in s tem aproksimiralo dejanske izhode $\hat{Y}(t)$ proti našim ciljnim izhodom $Y(t)$. Učno pravilo, ki je uporabljeno se imenuje vzvratno razširjanje (ang. backpropagation) [28]. V metodi vzvratnega razširjanja izberemo takšne uteži $W_{ij}$, ki minimizirajo kvadratično napako nad učno množico:

$$E = \sum_{t=1}^{T} E(t) = \sum_{t=1}^{T} \sum_{i=1}^{n} (1/2)(\hat{Y}_i(t) - Y_i(t))^2 \tag{2.5}$$

Metodo vzvratnega razširjanja začnemo z naključno določenimi utežmi $W$. Nato izračunamo izhode $\hat{Y}(t)$ in napako $E(t)$ za dano množico uteži.

Nato izračunamo odvode napake $E$ glede na vse uteži $W$. Če bi povečevanje neke uteži povzročilo večjo napako, popravimo to utež tako, da ji zmanjšamo vrednost. Če pa bi povečevanje neke uteži napako zmanjšalo, bi tej uteži vrednost dvignili. Pri tem *stopnja učenja* določa hitrost spreminjanja uteži. Ko popravimo vse uteži v nevronski mreži, začnemo ta proces znova od začetka. To ponavljamo dokler se napaka ne dovolj zniža. Unikatnost vzvratnega razširjanja leži v metodi računanja odvodov za vse uteži ob samo enem prehodu skozi sistem. Več o tem si lahko preberemo v [28].

Učenje nevronske mreže je razdeljeno na več epohov, pri čemer je vsak epoh prehod preko celotne učne množice, razdeljene na učne instance. Vsaka učna instanca je sestavljena iz $n$ slik. Vrstni red učnih instanc je v vsakem epohu naključno določen. Med učenjem so naše slike razdeljene v dve množici. V učno in v validacijsko. Na učni množici se uči, validacijsko množico pa potrebujemo, da lahko ocenimo kako uspešno poteka učenje ter, da se prepričamo, da ne prihaja na učni množici do premočnega prileganja podatkov (ang. overfitting). Prehod skozi validacijsko množico se zgodi na koncu vsakega epoha. Pri prehodu skozi validacijsko množico, mreža ne spreminja uteži $W$ in zato je tudi ta prehod trikrat hitrejši. Mi smo kot najboljšo nevronsko mrežo izbrali tisto, ki je nad testno množico slik dala najboljše rezultate. Ker je bil naš cilj doseči dobre rezultate nad testno množico, smo za validacijsko množico uporabili kar testno množico slik.

# Poglavje 3

# Implementacija

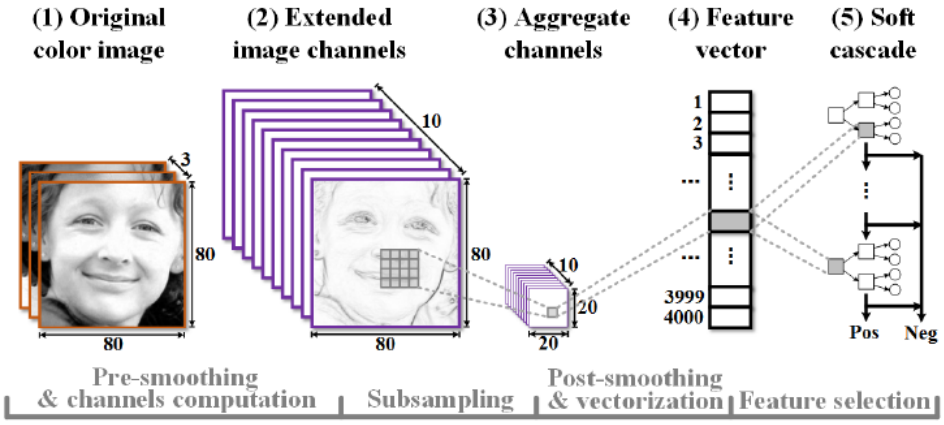
## 3.1 Detekcija

Za problem detekcije prometnih znakov smo uporabili detektor ACF [25] implementiran s knjižnico Piotr's Computer Vision [6].

Osnovna struktura značilnic iz agregiranih kanalov je kanal. Kanali imajo dolgo zgodovino in so obstajali že od razvoja digitalne fotografije naprej. Najbolj osnovni tipi kanalov so barvni kanali slik. Med temi sta najbolj znana barvni kanal RGB ter črno-beli barvni kanal. Poleg barvnih kanalov pa obstaja še mnogo drugih vrst kanalov, ki so bili razviti, da kodirajo različne tipe informacij za bolj težavne probleme. Na splošno lahko definiramo kanale kot mape originalne slike, čigar slikovne enote so bile preračunane iz pripadajočih območij originalne slike. Različni kanali so lahko izračunani z linearno ali nelinearno transformacijo originalne slike. Da bi omogočili detekcijo z drsečim oknom, so transformacije translacijsko invariantne [30].

Značilnice iz agregiranih kanalov delujejo tako, da vzamejo vhodno barvno sliko in izračunajo nove kanale (na primer barvne, gradientne, lokalne histograme). Vse kanale nato podvzorčijo za nek vnaprej določen faktor. Vsi slikovni elementi v podvzorčenih kanalih se nato združijo v vektor značilk. Ta vektor je nato poslan v klasifikator soft cascade. Delovanje značilnic iz agregiranih kanalov si lahko ogledamo na sliki 3.1. V primerjavi s tipično

arhitekturo Viola Jones [14], je prednost značilnic iz agregiranih kanalov v tem, da operirajo na več kanalih hkrati in tako omogočajo bolj bogato predstavitveno kapaciteto. Prav tako so značilke ekstraktane direktno iz slikovnih elementov podvzorčenih kanalov in ne izračunane kot pravokotne vsote na različnih lokacijah v sliki ter pri različnih velikostih slike. To omogoči veliko večje hitrosti delovanja [30].




Slika 3.1: Zgradba detektorja ACF [30].

## 3.2 Razvoj lastne nevronske mreže

### 3.2.1 Prvi poskusi

Razvoj konvolucijske nevronske mreže smo začeli postopoma. Za začetek smo vzeli že obstoječo nevronsko mrežo, imagenet-vgg-verydeep-16, ki je bila eden izmed najuspešnejših modelov na Imagenet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014) [20]. Cilj ILSVRC2014 je bil klasificirati med 1000 različnimi razredi. Posledično ima imagenet-vgg-verydeep-16 za izhod 1000 dimenzionalni vektor, na osnovi katerega se odloči, kateremu razredu pripada dana slika. Če je največji odziv na 20. mestu vektorja, to pomeni, da slika pripada 20. razredu po vrsti. Ta izhodni vektor smo mi uporabili tako, da smo skozi nevronsko mrežo pognali učno množico in na osnovi izhodnega

vektorja naučili metodo podpornih vektorjev (svm) in metodo k najbližjih sosedov (knn). S tem pristopom smo dosegli klasifikacijsko točnost nad testnim razredom enako 35.85% za metodo k najbližjih sosedov ter 48.43% za metodo podpornih vektorjev eden proti vsem. Ker je bila mreža namenjena klasificiranju zelo različnih objektov je posledično znake klasificirala zelo podobno in zato je tudi rezultat s knn-jem ter svm-jem bil slab. Čeprav je bil rezultat neuspešen, pa je vseeno pripomogel k razumevanju delovanja nevronskih mrež.

Po tem smo se postopoma lotili izdelovanja svoje nevronske mreže.

### 3.2.2 Izbira arhitekture

V tem razdelku bomo predstavili, kako smo prišli do končne arhitekture naših dveh konvolucijskih nevronskih mrež. Delali smo s knjižnico MatConvNet [26]. Teorija učenja pravi, da če ima arhitektura konvolucijske nevronske mreže preveliko kapaciteto, bo med učenjem prišlo do premočnega prileganja podatkov nad učno množico in posledično do slabih rezultatov nad testno množico. Če ima model premajhno kapaciteto, pa bo prišlo do preslabega prileganja podatkov nad učno množico in posledično do slabih rezultatov tako nad testno, kot nad učno množico. Potrebno je torej izbrati takšno arhitekturo, ki bo imela kapaciteto prave velikosti [13].

Metoda, ki smo jo izbrali za določitev arhitekture nevronske mreže deluje tako, da najprej dodamo en blok, naučimo mrežo in pogledamo kakšne rezultate smo dobili. Nato dodamo naslednji blok, ponovno naučimo mrežo in ponovno ocenimo dobljene rezultate. Tako dodajamo bloke in ocenjujemo rezultate. Ko konvolucijska nevronska mreža postaja globlja in globlja, se rezultati sprva hitro izboljšujejo, nato pa pričnejo konvergirati. Ko dobimo dovolj dobre rezultate, prenehamo z dodajanjem novih blokov, saj so globlji modeli počasnejši [13]. Pri izbiri zgradbe konvolucijske nevronske mreže smo uporabili požrešno metodo. Ko smo določili želeno arhitekturo mreže, smo nato začeli spreminjati parametre CNN-ja enega za drugim ter jih na ta način optimizirati. Podoben pristop k določanju zgradbe nevronske mreže lahko

vidimo tudi v [13].

Naučili smo dve globoki konvolucijski nevronski mreži. Eno smo naučili na nemški bazi za prepoznavanje prometnih znakov (GTSRB) [24] in eno na belgijski bazi za prepoznavanje prometnih znakov (BTSC) [16]. Obe bazi bomo podrobneje predstavili v poglavju 4.1. Učenje smo najprej začeli na bazi GTSRB ter razvili 15 konvolucijskih nevronskih mrež in izbrali najboljšo. Za učenje nevronske mreže na bazi BTSC smo za začetek vzeli arhitekturo, ki je dala najboljše rezultate nad bazo GTSRB (tabela 3.1). Tako smo za učenje na bazi BTSC morali naučiti samo 4 nove konvolucijske nevronske mreže, preden smo se lahko odločili za najboljšo.

### 3.2.3 Določanje velikosti filtrov

Eno izmed vprašanj, ki se pojavlja, ko gradimo novo konvolucijsko nevronsko mrežo je, kako določiti velikost konvolucijskih in Max-pooling filtrov? Kako vemo kako velika bo izhodna mapa ob izboru filtra neke določene velikosti?

Velikost mape, ki jo dobimo na izhodu konvolucijskega nivoja ali nivoja Max-pooling izračunamo z enačbama (3.1) in (3.2).

$$sirina_{izhod} = \frac{sirina_{vhod} - velikostFiltra + 2 * obroba}{preskok} + 1 \tag{3.1}$$

$$visina_{izhod} = \frac{visina_{vhod} - velikostFiltra + 2 * obroba}{preskok} + 1 \tag{3.2}$$

V našem primeru sta višina in širina vhodne slike enaki, kar pomeni, da lahko, za izračun velikosti izhodne mape, uporabimo poenostavljeno enačbo (3.3), kjer *velikost* predstavlja dolžino ene stranice, t.j. širino oziroma višino mape.

$$velikost_{izhod} = \frac{velikost_{vhod} - velikostFiltra + 2 * obroba}{preskok} + 1 \tag{3.3}$$

V enačbi (3.3) se $velikost_{vhod}$ nanaša na velikost mape, ki jo dobimo iz prejšnjega bloka. Ob prvi konvoluciji je to vhodna, v našem primeru barvna, slika velikost 48x48x3. Nato pa se nanaša na mapo, ki je izhod iz prejšnjega

konvolucijskega nivoja oziroma nivoja Max-pooling. $velikostFiltra$ se nanaša na velikost filtra, s katerim izvajamo konvolucijo ali operacijo Max-pooling. V primeru filtra velikosti $5x5$, bi bila $velikostFiltra$ enaka 5. *obroba* se nanaša na število dodatnih slikovnih enot, z vrednostjo nič, na vsaki strani mape. *preskok* pa pomeni po koliko slikovnih enot na enkrat se ob vsakem premiku filtra premaknemo. V primeru konvolucijskega nivoja je ta preskok enak 1 [13].

## 3.3 Končna arhitektura nevronske mreže

Kot smo omenili v podpoglavju 3.2.2 smo se učenja konvolucijske nevronske mreže lotili postopoma in sicer z dodajanjem novih blokov, dokler nismo dobili želenih rezultatov. V podpoglavju 2.1 smo omenili, da ne želimo, da v konvolucijskem nivoju prihaja do izgube dimenzij, ter, da prihajanje do tega lahko preprečimo z liho velikostjo filtrov, ter dodatkom obrobe velikosti $\lfloor velikostFiltra/2 \rfloor$. Eden izmed razvitih CNN-jev na bazi GTSRB je bil razvit z upoštevanjem teh nasvetov in njegova arhitektura je bila bazirana na arhitekturi določeni v [13]. Dal je dobre končne rezultate in sicer 97.70% klasifikacijsko točnost na testni množici baze GTSRB. Ker ima liho velikost filtrov, smo velikost slik tudi skalirali na liho velikost in sicer na 47 x 47. Na koncu pa smo najboljše rezultate dobili na CNN-ju, ki je imel sodo velikost filtrov.

Naš končni najboljši CNN ima 14 nivojev. Podrobno zgradbo najbolje naučenega CNN-ja na bazi GTSRB lahko vidimo v tabeli 3.1. Na sliki 3.2 lahko vidimo graf, kako je potekalo učenje. Z modro črto je predstavljena učna množica, z rdečo črto pa validacijska. S $top1err$ je predstavljeno kolikokrat je bil prvi predlog za detektirani znak napačen, s $top5err$ pa kolikokrat je bilo prvih 5 predlogov napačnih. Na osi $y$ se nahaja napaka, na osi $x$ pa epoh učenja. V primeru baze GTSRB smo imeli 39209 učnih slik in vsaka učna instanca je bila velikosti 100 slik. To pomeni, da smo v enem epohu imeli 393 učnih instanc. Samih epohov smo imeli 45.

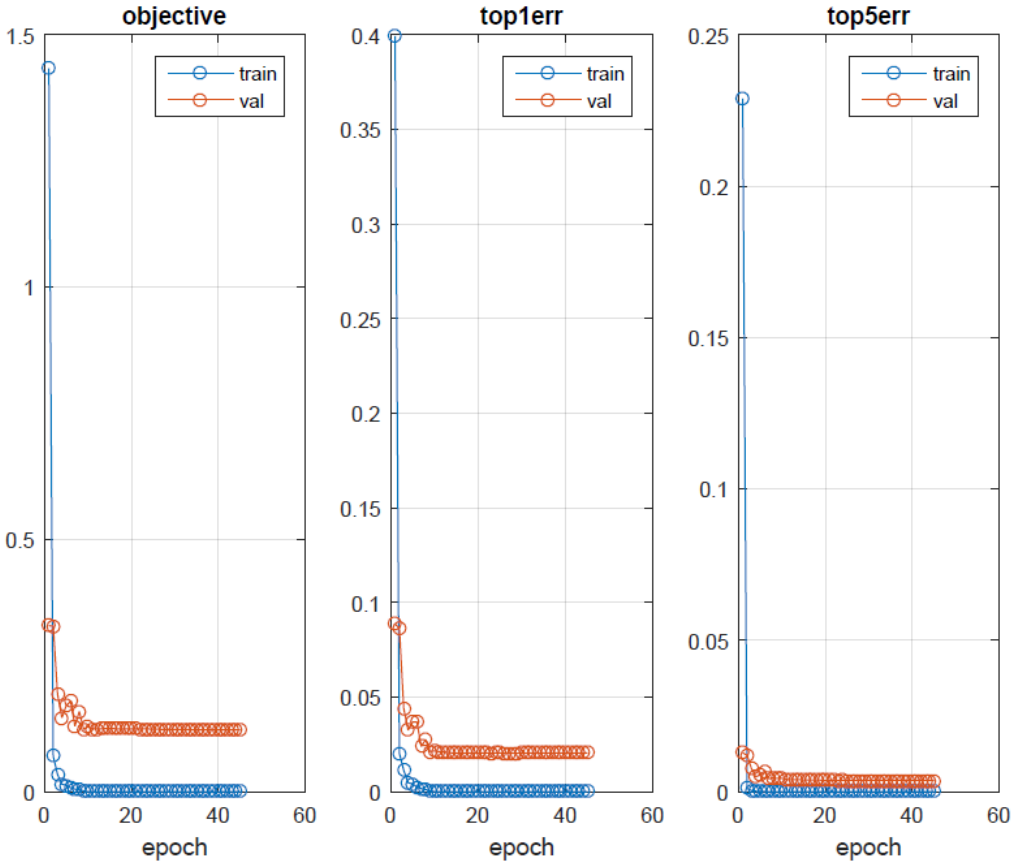
| Končna zgradba CNN | | | | | | |
|---|---|---|---|---|---|---|
| | Tip nivoja | izh mapa | št kanal | filter | fak presk | obroba |
| 1 | Vhodna slika | 48 x 48 | 3 | | | |
| 2 | Konvolucijska | 47 x 47 | 32 | 6 x 6 | 1 | 2 |
| 3 | Max-pooling | 23 x 23 | 32 | 3 x 3 | 2 | |
| 4 | ReLu | 23 x 23 | 32 | | | |
| 5 | Konvolucijski | 24 x 24 | 64 | 4 x 4 | 1 | 2 |
| 6 | Max-pooling | 8 x 8 | 64 | 3 x 3 | 3 | |
| 7 | ReLu | 8 x 8 | 64 | | | |
| 8 | Konvolucijski | 9 x 9 | 64 | 4 x 4 | 1 | 2 |
| 9 | Max-pooling | 4 x 4 | 64 | 3 x 3 | 2 | |
| 10 | ReLu | 4 x 4 | 64 | | | |
| 11 | Full-link | 1 x 1 | 64 | 4 x 4 | 1 | |
| 12 | ReLu | 1 x 1 | 64 | | | |
| 13 | Full-link | 1 x 1 | 43 | 1 x 1 | 1 | |
| 14 | Soft-max | 1 x 1 | 44 | | | |

Tabela 3.1: Končna zgradba najbolje naučenega CNN-ja na bazi GTSRB.

Podrobno zgradbo najbolje naučenega CNN-ja na bazi BTSC lahko vidimo v tabeli 3.2.

| Končna zgradba CNN | | | | | | |
|---|---|---|---|---|---|---|
| | Tip nivoja | izh mapa | št kanal | filter | fak presk | obroba |
| 1 | Vhodna slika | 48 x 48 | 3 | | | |
| 2 | Konvolucijski | 47 x 47 | 48 | 6 x 6 | 1 | 2 |
| 3 | Max-pooling | 23 x 23 | 48 | 3 x 3 | 2 | |
| 4 | ReLu | 23 x 23 | 48 | | | |
| 5 | Konvolucijski | 24 x 24 | 64 | 4 x 4 | 1 | 2 |
| 6 | Max-pooling | 8 x 8 | 64 | 3 x 3 | 3 | |
| 7 | ReLu | 8 x 8 | 64 | | | |
| 8 | Konvolucijski | 9 x 9 | 128 | 4 x 4 | 1 | 2 |
| 9 | Max-pooling | 4 x 4 | 128 | 3 x 3 | 2 | |
| 10 | ReLu | 4 x 4 | 128 | | | |
| 11 | Full-link | 1 x 1 | 128 | 4 x 4 | 1 | |
| 12 | ReLu | 1 x 1 | 128 | | | |
| 13 | Full-link | 1 x 1 | 62 | 1 x 1 | 1 | |
| 14 | Soft-max | 1 x 1 | 63 | | | |

Tabela 3.2: Končna zgradba najbolje naučenega CNN-ja na bazi BTSC.




Slika 3.2: Najbolje naučena konvolucijska nevronska mreža na bazi GTSRB.

# Poglavje 4

# Eksperimenti

## 4.1 Pridobivanje podatkov

Konvolucijsko nevronsko mrežo smo učili prepoznavanja prometnih znakov na German traffic sign recognition base (GTSRB) [24] (slika 4.1) ter na Belgium Traffic Sign Dataset for Classification (BTSC) [16] (slika 4.2).

Baza GTSRB je sestavljena iz 51839 slik, ki so razdeljene na 39209 učnih in 12630 testnih primerov. V bazi se nahaja 43 različnih razredov prometnih znakov. Baza BTSC pa je sestavljena iz 7134 slik, od tega je 4591 učnih ter 2543 testnih. Belgijska baza ima 62 različnih razredov znakov. Kar imata obe bazi skupno je to, da so slike anotirane, in sicer za vsako obstaja podatek:

- **Filename**: ime slike.
- **Width**: širina slike.
- **Height**: višina slike.
- **ROI.x1**: koordinata x zgornjega levega oglišča očrtanega pravokotnika prometnega znaka.
- **ROI.y1**: koordinata y zgornjega levega oglišča očrtanega pravokotnika prometnega znaka.

- **ROI.x2**: koordinata x spodnjega desnega oglišča očrtanega pravokotnika prometnega znaka.
- **ROI.y2**: koordinata y spodnjega desnega oglišča očrtanega pravokotnika prometnega znaka.
- **ClassId**: identifikacijska številka razreda.


Slika 4.1: Vsi razredi baze GTSRB.

## 4.2 Obdelava podatkov

Slike prometnih znakov so bile različnih velikosti ter zajete v različnih osvetlitvenih pogojih (slika 4.3). Posledično jih je bilo pred uporabo potrebno obdelati [4]. Prvi poskus je bil normalizacija slik z odštevanjem povprečne vrednosti vseh slik v učni množici. To je metoda, ki je bila uporabljena pri


Slika 4.2: Vsi razredi baze BTSC.

mreži imagenet-vgg-verydeep-16. Prednost tega pristopa je bila enostavnost in velika hitrost, a rezultati so bili slabi. Po eksperimentiranju, smo prišli do najboljših rezultatov s sledečim postopkom obdelave slik.

Vsaka slika prometnega znaka iz baze GTSRB in BTSC je odrezana tako, da pusti okoli znaka še nekoliko ozadja, in s tem omogoči obdelavo z uporabo metod, ki uporabljajo detekcijo na osnovi robov. Mi smo vse slike izrezali z uporabo koordinat ROI, in tako dobili samo prometne znake. Vse slike smo nato skalirali na enako velikost, in sicer na 48 x 48 slikovnih enot. Tudi slike, ki so imele očrtane pravokotnike (ang. bounding box) pravokotne oblike, so tako postale kvadratne oblike. Med slikami so bile prisotne velike kontrastne razlike in zato smo slike morali normalizirati. To smo dosegli z operacijami *adapthisteq*, *histeq* in *imadjust*, [4] katere smo izvedli na vsakem barvnem kanalu slike. *Adapthisteq* ojača kontraste v slikah in deluje tako, da operira na majhnih regijah, in ne na celi sliki na enkrat, kot to počne *histeq*. *Histeq*

Slika 4.3: Nekaj težkih primerov iz baze GTSRB.

nato ojača kontraste z izenačevanjem histograma. *Imadjust* pa popravi vrednosti intenzivnosti v sliki. Pri tem tudi poveča kontraste v sliki. Tako smo ustvarili učno množico, na osnovi katere bomo učili nevronsko mrežo. Na sliki 4.4 lahko vidimo rezultate obdelave slik s prej omenjenimi koraki.

## 4.3 Povečevanje učne množice

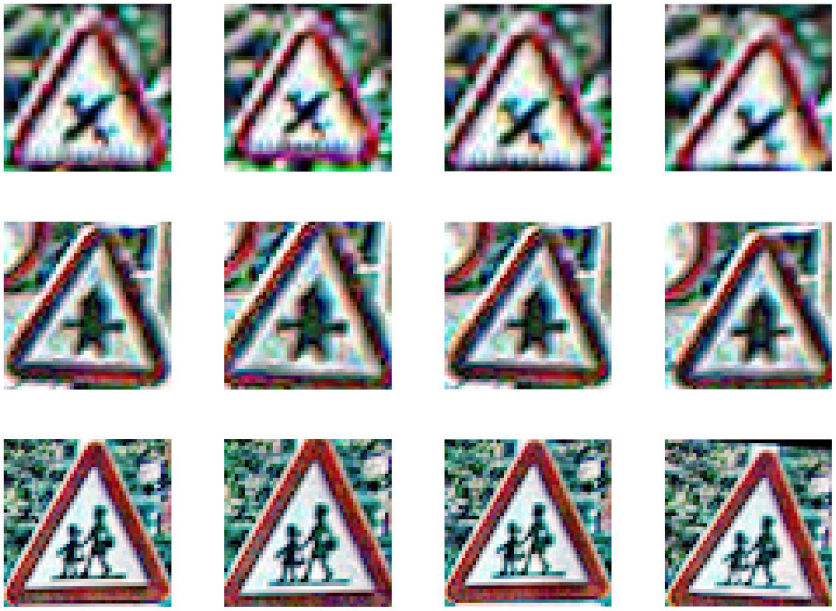
Eden izmed načinov kako izboljšati učenje konvolucijske nevronske mreže je z uporabo večje učne množice. Če imamo omejeno število učnih primerov, in ne moremo dobiti novih, jih lahko umetno ustvarimo tudi sami [13, 23]. Pri bazi GTSRB to ni bilo potrebno, saj je imela dovolj učnih primerov. Baza BTSC pa je imela manj učnih primerov ter več razredov. Da bi dosegli boljše učenje smo bazo povečali z uporabo prostorske perturbacije podatkov (ang. data jittering).

To smo storili tako, da smo vsako učno sliko naključno zamaknili v rangu $[-2, 2]$ slikovnih enot, jo skalirali v rangu $[0.9, 1.1]$ originalne velikosti slike ter rotirali v rangu $[-5, 5]$ stopinj. Tako smo iz 4591 učnih primerov prišli na 18364. Rezultate prostorske perturbacije podatkov lahko vidimo na sliki 4.5.

Slika 4.4: Primer obdelave petih slik. V prvi vrstici je originalna slika. V drugi vrstici je izrezana slika prometnega znaka z uporabo ROI koordinat. V tretji vrstici je slika skalirana na 48 x 48 slikovnih enot ter odbelana z adapthist. V četrti vrstici vidimo sliko po obdelavi s histeq. V peti vrstici je slika po obdelavi z imadjust. Rezultat je slika, katero pošljemo v nevronsko mrežo.

## 4.4 Rezultati detekcije znakov

Za detekcijo prometnih znakov smo naučili detektor ACF. Učili smo na German Traffic Signs Detection Base (GTSDB) [11] katera vsebuje 900 slik posnetih med vožnjo z avtomobilom. Prometni znaki so prisotni na večini slik, a ne na vseh. Na sliki 4.6 vidimo primer slike iz baze GTSDB. Bazo GTSDB smo razdelili na 600 slik učne množice ter na 300 slik testne množice. Prometni znaki so si med seboj različni po obliki in barvi. Posledično je bolje naučiti več detektorjev, po enega za vsako skupino prometnih znakov. Tako



Slika 4.5: Primer treh slik, na katerih smo izvedli prostorsko perturbacijo podatkov. V prvem stolpcu so originalne slike, v drugem so skalirane, v tretjem zamaknjene, v četrtem pa rotirane.

smo na koncu po zgledu [25] naučili sledeče detektorje ACF:

- detektor za vse znake,
- detektor za znake za nevarnost,
- detektor za znake za prepovedi,
- detektor za znake za obveznosti,
- detektor za znake za konec omejitev,
- detektor za znak za prednostno cesto,
- detektor za znak za križišče s prednostno cesto,
- detektor za znak za ustavi (STOP).

Slika 4.6: Primer slike iz baze GTSDB.

Detektorje ACF smo učili ter testirali na bazi GTSDB. Rezultati so vidni v tabeli 4.1. Odstotek prekritosti predstavlja odstotek prekrivanja očrtanega pravokotnika dobljenega z značilnicami iz agregiranih kanalov s tistim, na anotiranih lokacijah (ang. ground truth).

## 4.5 Rezultati klasifikacije znakov

Naš najbolje naučeni CNN na bazi GTSRB (tabela 3.1) je na testni množici baze GTSRB dosegel klasifikacijsko točnost enako 97.96%. Na testni množici baze GTSDB pa klasifikacijsko točnost 100.00%. Na sliki 4.7 lahko vidimo nekaj primerov napačne klasifikacije znakov iz baze GTSRB. Razlogi, zakaj jih nevronska mreža ni uspela klasificirati, so bili slabi osvetlitveni pogoji, nizka resolucija, prisotnost odbleskov svetlobe, zabrisanost in delna zakritost.

Najbolje naučeni CNN na bazi BTSC (tabela 3.2) je na testni množici baze BTSC dosegel klasifikacijsko točnost 96.50%, z dodatkom prostorske perturbacije podatkov v fazi učenja pa 97.99%. Rezultate klasifikacije lahko vidimo v tabeli 4.2.

| Rezultati detekcije znakov | | |
|---|---|---|
| Odstotek prekritosti | Detektor vseh znakov | Vsi detektorji skupaj |
| Večja od 0% | 90.03% | 91.69% |
| Večja od 50% | 89.20% | 91.41% |
| Večja od 60% | 84.49% | 88.09% |
| Večja od 70% | 74.79% | 82.83% |
| Večja od 80% | 45.71% | 55.96% |
| Večja od 90% | 8.86% | 11.91% |

Tabela 4.1: Odstotek detektiranih znakov z uporabo detektorja za vse znake ter z uporabo vseh naučenih detektorjev hkrati.

| Rezultati klasifikacije znakov | | |
|---|---|---|
| Učna baza | Testna baza | Klasifikacijska točnost |
| GTSRB | GTSRB | 97.96% |
| GTSRB | GTSDB | 100.00% |
| BTSC | BTSC | 97.99% |

Tabela 4.2: Rezultati klasifikacije prometnih znakov.

Slika 4.7: Nekaj primerov napačne klasifikacije nad bazo GTSRB.

## 4.6 Rezultati detekcije in klasifikacije znakov

Na testni množici baze GTSDB, ki kot že omenjeno, vsebuje 300 slik, 361 prometnih znakov in 43 različnih razredov, je naš CNN na anotiranih lokacijah prometnih znakov dosegel klasifikacijsko točnost 100%. V nadaljevanju bodo predstavljeni rezultati detekcije ter klasifikacije detektiranih znakov nad bazo GTSDB. Na sliki 4.8 lahko vidimo primer pravilne detekcije in klasifikacije na sličici, iz videa, iz belgijske baze za detekcijo prometnih znakov.

Za detektor vseh znakov, lahko v tabeli 4.3, vidimo poleg rezultatov detekcije, pri različnih odstotkih prekritostih očrtanega pravokotnika dobljenega z značilnicami iz agregiranih kanalov s tistim iz anotiranih lokacij, še kolikšna je bila klasifikacijska točnost samo detektiranih znakov, pri vsakem odstotku prekritosti, ter kolikšno klasifikacijsko točnost to predstavlja glede na vse znake, tudi tiste, katere nismo detektirali. Pri tem se je detektor v 5% detekcij zmotil ter detektiral znak tam, kjer ga v resnici ni bilo. V tabeli 4.4 lahko vidimo rezultate pri uporabi vseh naučenih detektorjev hkrati.

Vidimo lahko, da se klasifikator dobro obnese že samo ob 40% pokritosti

| Detektor vseh znakov | | | |
|---|---|---|---|
| Odstotek prekritosti detekcije ACF ter anotirane lokacije | Odstotek zaznanih znakov | Odstotek pravilno klasificiranih zaznanih znakov | Odstotek pravilno klasificiranih vseh znakov |
| Večja od 40% | 90.03% | 98.77% | 88.92% |
| Večja od 50% | 89.20% | 98.76% | 88.09% |
| Večja od 60% | 84.49% | 99.02% | 83.66% |
| Večja od 70% | 74.79% | 98.89% | 73.96% |
| Večja od 80% | 45.71% | 99.39% | 45.43% |
| Večja od 90% | 8.86% | 100.00% | 8.86% |

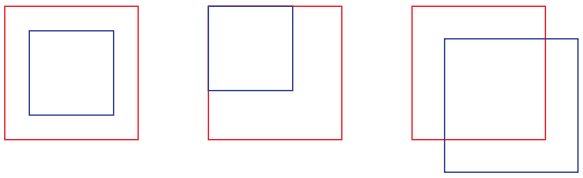
Tabela 4.3: Rezultati detekcije in klasifikacije prometnih znakov z uporabo detektorja vseh znakov.

| Vsi detektorji hkrati | | | |
|---|---|---|---|
| Odstotek prekritosti detekcije ACF ter anotirane lokacije | Odstotek zaznanih znakov | Odstotek pravilno klasificiranih zaznanih znakov | Odstotek pravilno klasificiranih vseh znakov |
| Večja od 40% | 91.69% | 98.49% | 90.30% |
| Večja od 50% | 91.41% | 98.49% | 90.03% |
| Večja od 60% | 88.09% | 98.48% | 86.75% |
| Večja od 70% | 82.83% | 98.33% | 81.45% |
| Večja od 80% | 55.96% | 99.50% | 55.68% |
| Večja od 90% | 11.91% | 100.00% | 11.91% |

Tabela 4.4: Rezultati detekcije in klasifikacije prometnih znakov z uporabo vseh detektorjev hkrati.

Slika 4.8: Primer pravilne detekcije in klasifikacije sličice, iz videa, iz belgijske baze za detekcijo prometnih znakov.

detekcije. Na sliki 4.9 si lahko ogledamo tri primere 40% pokritosti.



Slika 4.9: Trije primeri kaj pomeni 40% pokritost.

V tabeli 4.5 vidimo kolikšen odstotek detekcij je zapadel v določeni rang prekritosti, ter kolikšen je bil odstotek klasifikacijske točnosti v tem rangu.

V videu bi imeli vsako sekundo 24 sličic in med vožnjo bi se znak med približevanjem večal. Prav tako pa tudi odbleski od sonca ne bi bili prisotni na vseh sličicah. V videu bi tako verjetno zaznali večji odstotek znakov,

| | Detektor vseh znakov | | Vsi detektorji hkrati | |
|---|---|---|---|---|
| Odstotek prekritosti detekcije ACF ter anotirane lokacije | Odstotek detekcije | Odstotek klasifikacije | Odstotek detekcije | Odstotek klasifikacije |
| 0% | 9.97% | 0% | 8.31% | 0% |
| 1% - 39.99% | 0% | | 0% | |
| 40% - 49.99% | 0.83% | 100.00% | 0.28% | 100.00% |
| 50% - 59.99% | 4.71% | 94.12% | 3.32% | 100.00% |
| 60% - 69.99% | 9.70% | 100.00% | 5.26% | 100.00% |
| 70% - 79.99% | 29.09% | 98.10% | 26.87% | 95.88% |
| 80% - 89.99% | 36.84% | 99.25% | 44.04% | 99.37% |
| 90% - 100% | 8.86% | 100.00% | 11.91% | 100.00% |

Tabela 4.5: Rangi prekrivanja ter koliko detekcij smo imeli v danem rangu. Prisoten je tudi odstotek pravilne klasifikacije v danem rangu.

čeprav jih mogoče ne bi zaznali prav na vsaki sličici.

Ko smo testirali hitrosti našega sistema za detekcijo ter klasifikacijo prometnih znakov, pri uporabi detektorja za vse znake, na testni množici GTSDB, smo za obdelavo 300 slik, velikosti 1360 x 800 slikovnih enot, potrebovali 59.36 sekund. To pomeni, da je naš sistem deloval s hitrostjo 6.4 sličic na sekundo. Med testiranjem na videu iz belgijske baze za detekcijo prometnih znakov, kjer so sličice velikosti 1628 x 1236 slikovnih enot, pa je deloval s hitrostjo 3.76 sličic na sekundo. Vidimo torej, da velikost slik močno vpliva na hitrost delovanja sistema. Najenostavnejši način, kako bi dosegli obdelavo videa v realnem času bi bil, da bi znake iskali samo na vsaki četrti sličici videa. Toda s tem bi zavrgli veliko število sličic in posledično bi nevarnost, da kakšen prometni znak spregledamo še dodatno povečali. Drugi način bi bil, da bi uporabili manjše velikosti vhodnih slik. Kot smo videli, velikost slik močno vpliva na hitrost sistema. Z uporabo manjših vhodnih slik bi pospešili delovanje detektorja in s tem celotnega sistema. Kje je tista meja, ko bi bile slike premajhne in bi to začelo vplivati na delovanje klasifikatorja,

bi bilo potrebno dodatno raziskati.

# Poglavje 5

# Zaključek

Problem detekcije in prepoznavanja prometnih znakov je v zadnjih letih zelo popularen na področju računalniškega vida. Razlog za to je, da ima ta problem veliko aplikativnost v vsakdanjem življenju. Problem je prisoten recimo pri razvoju samovozečih vozil in pri razvoju naprednih sistemov za asistenco vozniku. Eden izmed pristopov k reševanju tega problema, je ravno z uporabo konvolucijskih nevronskih mrež. Slednje so v zadnjih letih na področju klasifikacije prometnih znakov presegle tradicionalne klasifikacijske metode in pričele redno zmagovati na tekmovanjih iz prepoznavanja predmetov.

Z uporabo konvolucijske nevronske mreže smo tudi mi dosegli zelo dobre klasifikacijske rezultate in to že pri detekcijah, ki so se z anotiranimi lokacijami prekrivale le za 40%. V tem se tudi vidi njihova moč. Kar je še posebej zanimivo je to, kako se učijo. Spreminja se parametre, dodaja nivoje in znova in znova uči, dokler ne dobimo dobrih rezultatov. Mi smo na bazi GTSRB dosegli klasifikacijsko točnost 97.96%, na bazi BTSC pa 97.99%. Oba rezultata bi se z dodatnimi poskusi verjetno dalo izboljšati še za kakšen odstotek, a ker je učenje nevronskih mrež zamudno, bi nam to vzelo precej časa. Slabše rezultate smo dosegli pri detekciji prometnih znakov, saj smo na testni množici GTSDB detektirali le 91.69% vseh prometnih znakov. Po združitvi detektorja ACF ter klasifikatorja CNN smo tako detektirali in klasificirali 90.30% vseh prisotnih prometnih znakov v testni množici baze

GTSDB. Če bi ta odstotek želeli dvigniti, bi se morali posvetiti predvsem izboljšavi detekcije prometnih znakov. V [33] so problem detekcije znakov rešili z uporabo polno konvolucijske mreže, ki je konvolucijska nevronska mreža, katero so prilagodili za reševanje problema detekcije prometnih znakov.

Naš sistem je deloval s hitrostjo od 3.76 do 6.4 sličic na sekundo. Za uporabo v avtomobilih tako ni bil dovolj hiter. Tudi če bi deloval hitreje, še vedno ne bi bil dovolj zanesljiv, saj je spregledal veliko število prometnih znakov. Če bi želeli razviti sistem, katerega bi lahko implementirali v avtomobilih, bi morali izboljšati detektor. Prav tako bi morali celoten sistem še dodatno testirati na več primerih, da se prepričamo, da ne prihaja do premočnega prileganja podatkov. Tudi če bi zaznali 99% vseh znakov, se še vedno pojavlja vprašanje, ali bi to bil dovolj dober rezultat. Za uporabo pri samovozečih vozilih najverjetneje ne, saj bi spregled znaka *Ustavi* lahko povzročil prometno nesrečo. Za uporabo v sistemu za asistenco voznika bi bil morda dovolj dober, saj bi tudi voznik gledal prometne znake in bi mu sistem služil samo v pomoč. A kaj, če bi se voznik na sistem začel preveč zanašati ter nehal sam iskati prometne znake? Bi spregled znaka *Ustavi* tudi tu povzročil prometno nesrečo? Na to temo je bilo napisanih veliko člankov. Eden izmed njih [2] obravna sistem Opel Eye, ki je na tržišču že od leta 2009 naprej. Čeprav je sistem povzročil, da je voznik, zaradi pogledovanja na sistem, večkrat odmaknil pogled s cestišča, so vseeno zaključili, da je sistem pripomogel k večjemu upoštevanju prometnih predpisov. Sistemi za napredno asistenco voznika so torej področje, na katerem lahko pričakujemo še veliko raziskav in izboljšav v naslednjih letih.

# Literatura


[1] Yuji Aoyagi and Toshiyuki Asakura. A study on traffic sign recognition in scene image using genetic algorithms and neural networks. In *Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on*, volume 3, pages 1838–1843. IEEE, 1996.

[2] Lars Holm Christiansen, Nikolaj Yde Frederiksen, Alex Ranch, and Mikael B Skov. Investigating the effects of an advance warning in-vehicle system on behavior and attention in controlled driving. In *Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 121–128. ACM, 2011.

[3] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921. IEEE, 2011.

[4] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[5] Arturo De La Escalera, Luis E Moreno, Miguel Angel Salichs, and José María Armingol. Road traffic sign detection and classification. *IEEE transactions on industrial electronics*, 44(6):848–859, 1997.

[6] Piotr Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). `https://github.com/pdollar/toolbox`.

[7] Miguel Ángel García-Garrido, Miguel Ángel Sotelo, and Ernesto Martín-Gorostiza. Fast road sign detection using hough transform for assisted driving of road vehicles. In *International Conference on Computer Aided Systems Theory*, pages 543–548. Springer, 2005.

[8] Miguel Angel Garcia-Garrido, Miguel Angel Sotelo, and Ernesto Martin-Gorostiza. Fast traffic sign detection and recognition under changing lighting conditions. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 811–816. IEEE, 2006.

[9] Jack Greenhalgh and Majid Mirmehdi. Real-time detection and recognition of road traffic signs. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1498–1506, 2012.

[10] Mrinal Haloi. Traffic sign classification using deep inception based convolutional networks. *arXiv preprint arXiv:1511.02992*, 2015.

[11] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks (submitted)*, 2013.

[12] Ubong Lydia Jau, Chee Siong Teh, and Giap Weng Ng. A comparison of rgb and hsi color segmentation in real-time video images: A preliminary study on road sign detection. In *2008 International Symposium on Information Technology*, volume 4, pages 1–6. IEEE, 2008.

[13] Junqi Jin, Kun Fu, and Changshui Zhang. Traffic sign recognition with hinge loss trained convolutional neural networks. *Intelligent Transportation Systems, IEEE Transactions on*, 15(5):1991–2000, 2014.

[14] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3:14, 2003.

[15] Gareth Loy and Nick Barnes. Fast shape-based road sign detection for a driver assistance system. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 70–75. IEEE, 2004.

[16] Markus Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool. Traffic sign recognition—how far are we from the solution? In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[17] Mirko Meuter, Christian Nunn, Steffen Michael Gormer, Stefan Muller-Schneiders, and Anton Kummert. A decision fusion and reasoning module for a traffic sign recognition system. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1126–1134, 2011.

[18] Jun Miura, Tsuyoshi Kanda, Shusaku Nakatani, and Yoshiaki Shirai. An active vision system for on-line traffic sign recognition. *IEICE TRANSACTIONS on Information and Systems*, 85(11):1784–1792, 2002.

[19] W Ritter. Traffic sign recognition in color image sequences. In *Intelligent Vehicles' 92 Symposium., Proceedings of the*, pages 12–17. IEEE, 1992.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[21] Andrzej Ruta, Yongmin Li, and Xiaohui Liu. Robust class similarity measure for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):846–855, 2010.

[22] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In

*International Conference on Artificial Neural Networks*, pages 92–101. Springer, 2010.

[23] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2809–2813. IEEE, 2011.

[24] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[25] Domen Tabernik, Rok Mandeljc, and Danijel Skočaj. Quality of region proposals in traffic sign detection and recognition, 2015.

[26] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[27] Vedran Vukotic, Josip Krapac, and Siniša Šegvic. Convolutional neural networks for croatian traffic signs recognition.

[28] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[29] Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, and Xiaolin Hu. Traffic sign detection based on convolutional neural networks. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE, 2013.

[30] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Aggregate channel features for multi-view face detection. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–8. IEEE, 2014.

[31] Fatin Zaklouta and Bogdan Stanciulescu. Real-time traffic-sign recognition using tree classifiers. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1507–1514, 2012.

[32] Yujun Zeng, Xin Xu, Yuqiang Fang, and Kun Zhao. Traffic sign recognition using deep convolutional networks and extreme learning machine. In *Intelligence Science and Big Data Engineering. Image and Video Data Engineering*, pages 272–280. Springer, 2015.

[33] Yingying Zhu, Chengquan Zhang, Duoyou Zhou, Xinggang Wang, Xiang Bai, and Wenyu Liu. Traffic sign detection and recognition using fully convolutional network guided proposals. *Neurocomputing*, 2016.