SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1550

Klasifikacija slika dubokim modelima i Fisherovim vektorima

Ivana Viduka

Zagreb, srpanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 15. ožujka 2017.

DIPLOMSKI ZADATAK br. 1550

Pristupnik: **Ivana Viduka (0036473466)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Klasifikacija slika dubokim modelima i Fisherovim vektorima**

Opis zadatka:

Klasifikacija slika je važna komponenta razumijevanja prirodnih scena. Taj problem možemo rješavati dubokim konvolucijskim modelima u kojima se slikovna informacija postupno transformira iz slikovne u simboličku domenu. Parametri konvolucijskih slojeva tipično se uče na slikovnoj kolekciji ImageNet. Međutim, postoje različiti načini prijenosa znanja u novu domenu, i to: i) učenjem plitkog klasifikatora nad agregiranim konvolucijskim značajkama te ii) ugođavanjem svih parametara dubokog modela. Tema ovog rada je istražiti prednosti i nedostatke obaju pristupa za detekciju objekata u prometnim scenama.

U okviru rada, potrebno je proučiti i ukratko opisati postojeće pristupe klasifikaciji slika utemeljene na dubokom učenju. Implementirati agregaciju konvolucijskih značajki usrednjavanjem originalnih značajki odnosno njihovih reprezentacija u Fisherovom prostoru. Uhodati postupke učenja i validiranja hiperparametara. Primijeniti naučene modele na slikama prometnih scena. Prikazati i ocijeniti ostvarene rezultate. Analizirati i interpretirati pogrešne predikcije modela. Predložiti pravce budućeg razvoja.

Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 29. lipnja 2017.

Mentor:

Izv. prof. dr. sc. Siniša Šegvić

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem se svojoj obitelji i prijateljima za pruženu potporu i pomoć tijekom čitavog školovanja. Zahvaljujem se i svom mentoru izv. prof. dr. sc. Siniši Šegviću i Ivanu Kreši na ukazanom strpljenju i brojnim savjetima koji su mi omogućili izradu ovog diplomskog rada.

SADRŽAJ

1. Uvod

Klasifikacija slika je jedan od temeljnih problema računalnog vida (engl. *computer vision*). Zadatak klasifikacijskih modela je za sliku na ulazu odrediti pripadnost jednom ili više razreda iz unaprijed zadanog skupa razreda. Kod nadgledanog učenja modela za klasifikaciju slika se učenje provodi uz pomoć skupa slika s pripadajućim oznakama razreda. Iako bi takvo učenje za čovjeka bilo lagan zadatak, računalu je otežano zbog mogućih varijacija u veličini i vidljivosti objekta na slici, kuta gledanja, osvijetljenja i razlika između primjeraka istog razreda. Uspješnost klasifikacijskog modela ovisi o kvaliteti značajki koje se dobivaju iz slika, a na kojima se temelji određivanje vjerojatnosti pripadnosti razredima.

U ovom radu će biti opisani koncepti i implementacije dvaju klasifikacijskih modela. U poglavlju 2 je predstavljen model dubokog učenja - konvolucijska neuronska mreža (engl. *convolutional neural network*, CNN). To je model koji daje *state-of-the-art* rezultate u klasifikaciji slika, a radi tako da nelinearnim transformacijama u svojim skrivenim slojevima razlaže sliku na konvolucijske značajke koje u izlaznom sloju klasificira. Poglavlje 3 opisuje klasifikaciju Fisherovim vektorima. To je model koji kombinira generativni i diskriminativni pristup klasifikaciji. Fisherovi vektori daju informaciju o odstupanju značajki slika od njihovog vlastitog generativnog modela. Kako bi se odredila pripadnost razredima koristi se linearni klasifikator nad Fisherovim vektorima.

U sklopu ovog rada implementirana je i naučena duboka konvolucijska mreža temeljena na VGG16 [14] arhitekturi. Fisherovi vektori su generirani za konvolucijske značajke dobivene uz pomoć javne parametrizacije VGG19 mreže [15], a metode korištene za klasifikaciju su logistička regresija i metoda potpornih vektora (SVM, engl. *Support Vector Machines*). Detalji implementacije se nalaze u poglavlju 5. Svi eksperimenti, čiji su rezultati prikazani u poglavlju 6, su provedeni na skupovima slika s hrvatskih prometnica - Zebre [19] i Znakovi [18]. Klasifikacijski zadatak je bio određivanje postoji li na slici zebra odnosno trokutasti prometni znak.

2. Duboki klasifikacijski model

Struktura dubokih modela koji se koriste za rješavanje složenih problema iz područja umjetne inteligencije može se promatrati kao "produbljenje" već postojećih koncepata rješavanja problema. Stare metode su obuhvaćale odvojeno učenje značajki i njihovu plitku klasifikaciju. Konstantni tehnološki napredak i veća dostupnost moćnijih računalnih resursa posljednjih godina potakli su razvoj složenijih algoritama. Duboko učenje obuhvaća algoritme koji u više uzastopnih slojeva provode nelinearne transformacija podataka velike dimenzionalnosti. Modeli dubokog učenja dijele se na diskriminativne i generativne i postižu *state-of-the-art* rezultate u područjima računalnog vida, obrade prirodnog jezika, automatskog razumijevanja govora, prepoznavanja zvuka i bioinformatike.

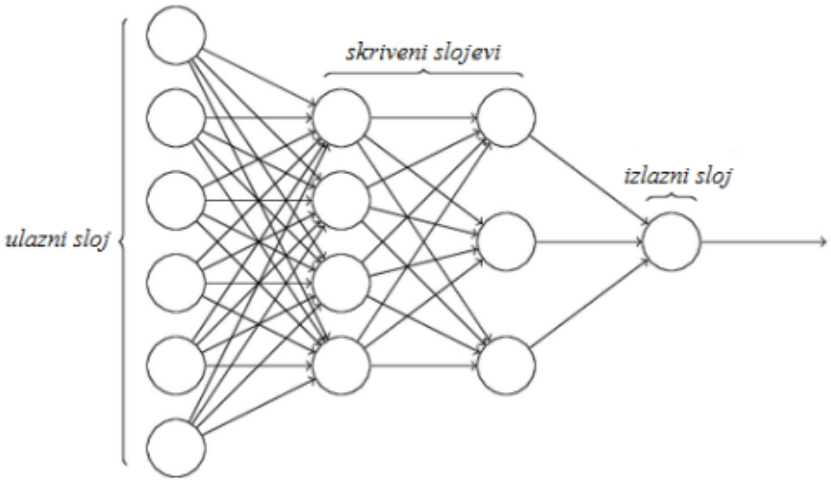
2.1. Konvolucijska neuronska mreža

Umjetne neuronske mreže (engl. *artifical neural networks*) su diskriminativni model strojnog učenja široko korišten za regresiju i klasifikaciju, a svoje početke vuče još od ideje umjetnih neurona i perceptrona iz 1950-ih. Klasična arhitektura neuronske mreže prikazana je na slici 2.1. U slučaju dodavanja više skrivenih slojeva riječ je o dubokoj neuronskoj mreži. U nastavku će biti razmotren osnovni princip rada neuronske mreže.

Na neurone ulaznog sloja, čiji broj odgovara dimenzionalnosti vektora $\mathbf{x}$, dovodi se vektor podatka $\mathbf{x}$. Neuroni dvaju susjednih slojeva i i j su potpuno povezani i na svojim vezama nose težine w_{ij}. Svaki neuron sadrži i aktivacijsku funkciju koja preslikava sklarni umnožak podatka i težina na izlaz neurona. Preslikavanje aktivacijske funkcije je nelinearno. Još jedan parametar mreže je b (engl. *bias*), koji predstavlja pomak i proširuje područje djelovanja neurona. Zbog pojednostavljenja se pomak promatra kao težina w_0 koja je uvijek povezana s ulaznom vrijednošću 1. Tada se izlaz neurona

može promatrati kao:

$$\begin{aligned} y_i &= \sum_j f(w_{ij}x_j) \\ &= f(\mathbf{w}^T\mathbf{x}) \end{aligned} \tag{2.1}$$



Slika 2.1: Arhitektura neuronske mreže.

Kako svaki neuron nekog sloja sadrži vektor težina koje ga povezuju s neuronima prethodnog sloja, sve težine je moguće zapisati u matricu težina **W**. Također, N ulaznih podataka dimenzije D može se zapisati u matricu **X** dimenzija $N \times D$. Sve operacije potrebne za simulaciju rada neuronske mreže time postaju matrične operacije.

Svaki neuron na izlazu koristi aktivacijsku funkciju. Početno korištena step funkcija ubrzo je zamijenjena sigmoidalnom funkcijom:

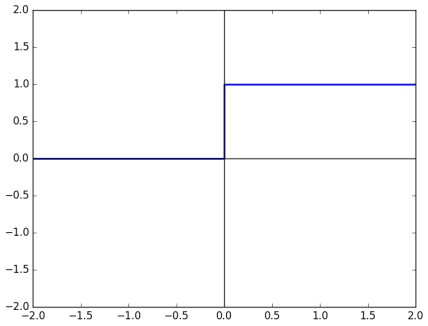
$$sigmoida(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

To je nelinarna, monotono rastuća funkcija koja izgleda poput zaglađene step funkcije, što se može vidjeti na slikama 2.2a i 2.2b. Prednosti sigmoidalne funkcije su to što male promjene na ulazu uzrokuju male promjene na izlazu i ograničavanje izlaza na konačnu vrijednost. Svojstvo derivabilnosti koju ima sigmoidalna funkcija potrebno je kod učenja neuronske mreže širenjem greške unatrag (engl. *backpropagation*). Hiperbolni tangens je još jedna sigmoidalna funkcija često korištena kao aktivacijska. Glavni nedostatak sigmoidalnih funkcija je pojava fenomena nestajućeg gradijenta (engl. *vanishing gradient*). Vrijednosti gradijenata sigmoidalnih funkcija ograničene su na intervale $\langle\, 0, \frac{1}{4}]$ za sigmoidu i $\langle\, 0, 1]$ za tangens hiperbolni. U procesu učenja neuronske

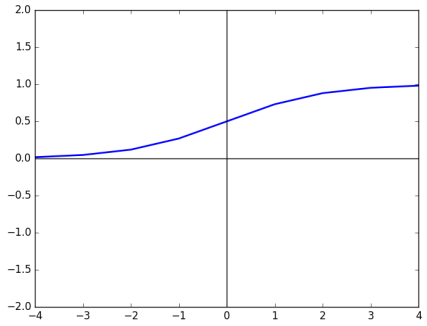
mreže je potrebno računati gradijente funkcije gubitka po svim parametrima. Zbog pravila ulančavanja gradijenata, više dijelova izraza za gradijente funkcije gubitka po parametrima prvih slojeva su gradijenti sigmoidalne funkcije. Zbog međusobnog množenja malih brojeva se nakon nekoliko iteracija učenja događa da se vrijednosti gradijenata u početnim slojevima naglo približavaju nuli. Posljedica toga je otežano učenje parametara u tom dijelu mreže. Zglobnica (engl. *Rectified Linear Unit*, ReLU) je funkcija koja uspješno izbjegava taj problem:

$$f(x) = max(0, x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases} \tag{2.3}$$

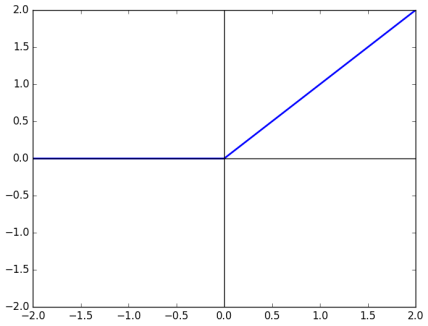
Linearni oblik ove funkcije ubrzava konvergenciju parametara mreže. Zglobnica je najčešće korištena aktivacijska funkcija u dubokim modelima. Opasnost od trajne deaktivacije neurona rješavaju varijacije funkcije poput propuštajuće zglobnice (engl. *Leaky ReLU*) kod koje vrijednost funkcije za $x < 0$ nije 0 već neka mala vrijednost veća od 0. [16]



(a) Step funkcija



(b) Sigmoidalna funkcija

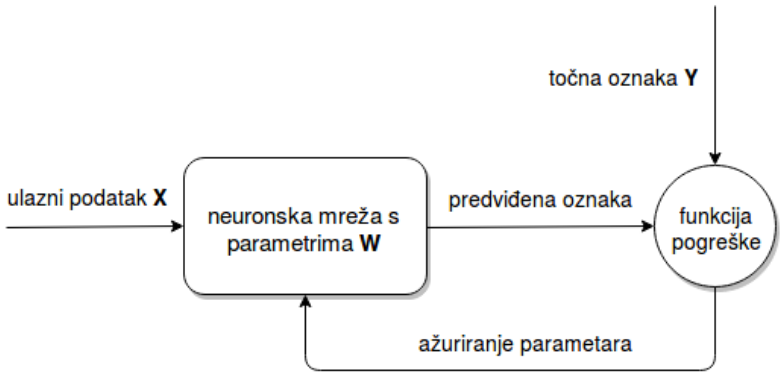


(c) Funkcija ReLU

Slika 2.2: Primjeri aktivacijskih funkcija.

Parametri neuronske mreže koji se mijenjaju u postupku učenja su **W** i **b**, a učenje može biti nadgledano ili nenadgledano. Razlika je u tome što kod nadgledanog učenja

postoji poznat izlaz $\mathbf{y}_i$ za svaki ulaz $\mathbf{x}_i$. Slikovni prikaz se nalazi na slici 2.3. Razlika između željenog i dobivenog izlaza kažnjava se uz pomoć dobro definirane funkcije gubitka. Cilj je minimizirati funkciju gubitka što se postiže promjenama parametara u smjeru negativnog gradijenta funkcije gubitka.



Slika 2.3: Model nadgledanog učenja neuronske mreže.

Prednost dubokih neuronskih mreža u odnosu na one sa samo jednim skrivenim slojem je mogućnost hijerarhijskog razlaganja složenijih problema. Nedostatak potpuno povezanih dubokih neuronskih mreža u rješavanju problema računalnog vida je slaba sposobnost generalizacije. Metode regularizacije se koriste kako bi se nadoknadio taj nedostatak i bolje iskoristile prednosti postojećeg modela. Potpuno povezani model se može regularizirati smanjenjem kapaciteta na način da se sve ne-lokalne veze postave na 0 i dijeljenjem težina. Tako se dolazi do konvolucijskog modela koji pokazuje najbolje rezultate kod problema računalnog vida. U radu sa slikama koje su podaci velike dimenzionalnosti je smanjenje broja parametara dodatna prednost koja ubrzava izračune u postupku učenja.

2.1.1. Slojevi konvolucijske mreže

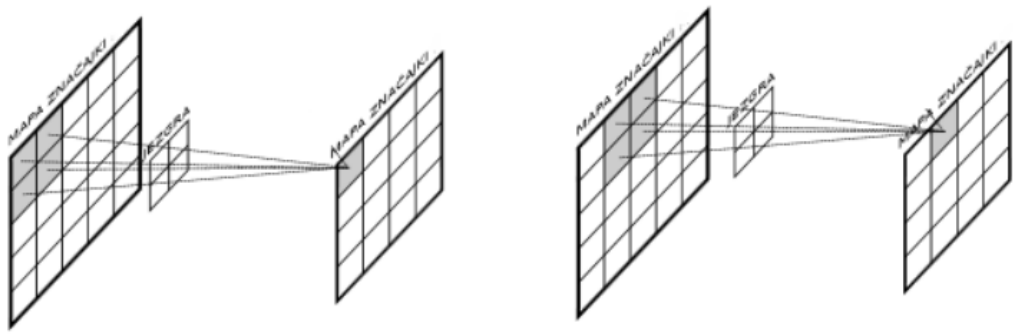
Konvolucijski sloj

Temeljna operacija konvolucijskog sloja je konvolucija ulazne mape značajki i odgovarajućih težina. Na samom ulazu u mrežu ulazna mapa značajki je matrica s intenzitetima piksela slike. Ukoliko je riječ o RGB slici postoje 3 mape. Jezgra konvolucijskog sloja konvoluira sa svim ulaznim mapama na jednak način i na izlaz daje nove mape značajki. Broj izlaznih mapa je jednak broju jezgara, a njihova veličina ovisi o veličinama ulaznih mapa i jezgre. Ako trodimenzionalni tenzor ulaza označimo s $\mathbf{V}$, element

i-te ulazne mape, na poziciji (j, k) je v_{ijk}. Tenzor jezgre **K** ima 4 dimenzije jer svaka konvolucijska jezgra mora biti povezana sa svim ulaznim mapama. Element na poziciji (k, l) koji povezuje ulaznu mapu j i izlaznu mapu i je k_{ijkl}. Tada je izlaz konvolucije pohranjen u tenzor izlaza **Z** definiran na sljedeći način:

$$z_{ijk} = \sum_{l,m,n} v_{l,j+m,k+n} k_{i,l,m,n} \tag{2.4}$$

Pojednostavljeni prikaz konvolucije jezgre i mape značajki u 2 dimenzije je prikazan na slici 2.4.



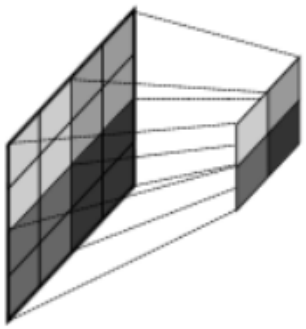
Slika 2.4: Prikaz konvolucije u konvolucijskom sloju. [17]

Ako prozor veličine jezgre prolazi ulaznom mapom s korakom većim od 1, izračun se ubrzava. Za razliku od potpuno povezanog sloja, neuroni konvolucijskog sloja su povezani samo s dijelom ulaznih informacija. Time se ostvaruje korelacija između prostorno bliskih piksela. Receptivno polje neurona (engl. *receptive field*) odgovara veličini konvolucijske jezgre. Ako je struktura mreže takva da izlazne mape značajki nisu povezane sa svim ulaznim mapama značajki, riječ je o rijetkoj povezanosti konvolucijskih slojeva. [17]

Sloj sažimanja

Sloj sažimanja (engl. *pooling layer*) slijedi nakon jednog sloja ili bloka konvolucijskih slojeva. Sažimanjem se smanjuje veličina mapa značajki s izlaza konvolucijskog sloja tako da se pritom zadržavaju bitne informacije. Najviše korišten tip sažimanja je sažimanje maksimumom (engl. *max pooling*). To je postupak kod kojeg se uzimaju pravokutna područja iz mape značajki s izlaza konvolucijskog sloja i pronalaze maksimalne vrijednosti. U rezultantnoj mapi značajki čitavo to pravokutno područje je zamijenjeno tom jednom vrijednošću. Slikovni prikaz ovakvog postupka nalazi se

na slici 2.5. Tipična veličina odabranog područja je 2×2 što smanjuje svaku mapu značajki 4 puta i nosi značajno ubrzanje postupka učenja konvolucijske mreže. Osim ubrzanja, sažimanje donosi i veću otpornost na malene razlike između piksela dvije slike - slična, ali ne i jednaka područja mogu imati jednaku maksimalnu vrijednost. Koriste se još i sažimanje usrednjavanjem, kod kojeg se umjesto maksimalne uzima srednja vrijednost, i sažimanje metrikom L_p.



Slika 2.5: Pojednostavljeni prikaz postupka sažimanja. [17]

Sažimanje metrikom L_p može postići bolje rezultate od sažimanja maksimumom uz optimiranje parametra p. Postupak se provodi tako da se za svako promatrano područje M izračuna M' prema jednadžbi 2.5 u kojoj G označava Gaussov filter. [17]

$$M' = \left(\sum_i \sum_j M_{ij}^p G_{ij} \right)^{1/p} \tag{2.5}$$

Izlazni sloj

Nakon nekoliko uzastopnih blokova konvolucije i sažimanja, a prije izlaza iz mreže, dobivene mape značajki se ponovno transformiraju. U starijim arhitekturama kao što su AlexNet [9] i VGG [14] se mape značajki izravnate u vektor puštaju na dva potpuno povezana sloja. Novije arhitekture DenseNet [6] i ResNet [4] nakon posljednje konvolucije provode globalno sažimanje koje za svaku mapu značajki izračuna jedan skalar koji označava maksimalnu ili srednju vrijednost svih vrijednosti te mape. U oba opisana slučaja dobiveni vektori idu na potpuno povezani sloj koji ima onoliko neurona koliko je razreda klasifikacije. Na izlazu konvolucijske mreže se nalazi funkcija softmax koja je općenitiji oblik sigmoidalne funkcije, a definirana je na sljedeći način:

$$softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.6}$$

To je funkcija koja na ulaz prima izlaz skrivenog sloja, vektor **y** duljine C, u kojem svaka od vrijednosti označava s kolikom sigurnošću slika s ulaza u konvolucijsku mrežu pripada jednom od C razreda. Funkcija softmax računa distribuciju vjerojatnosti pripadnosti uzorka preko svih mogućih razreda - dobiveni konačni izlaz sadržavat će pozitivne brojeve, a zbroj vjerojatnosti pripadanja za sve razrede bit će 1.

2.1.2. Učenje konvolucijske mreže

Nakon postavljanja arhitekture konvolucijske mreže koja je prihvatljiva za rješavanje postojećeg problema potrebno je pronaći parametre koji će osiguravati ispravne rezultate klasifikacije. Matrice težina **W** u skrivenim slojevima inicijaliziraju se na slučajne vrijednosti, a zatim podešavaju postupkom učenja. Inicijalizacija vrijednostima iz distribucija kao što su Gaussova ili uniformna osigurava gubljenje simetričnosti među neuronima koja bi uzrokovala jednak smjer učenja neurona povezanih s istim ulaznim podacima. Da bi učenje modela bilo moguće potrebno je definirati funkciju gubitka. U mrežama koje na izlaz daju distribuciju vjerojatnosti pripadnosti podatka po razredima prikladan odabir gubitka je gubitak unakrsne entropije (engl. *cross entropy loss*). Unakrsna entropija je mjera udaljenosti između dvije distribucije, a u ovom slučaju su to prava distribucija i ona dobivena na izlazu mreže. Gubitak za jedan podatak se može definirati na sljedeći način:

$$L = -\sum_{i=1}^{C} y_i \log(softmax_i(\mathbf{x})) \tag{2.7}$$

C je broj razreda, **y** ovdje predstavlja vektor točne distribucije preko svih razreda, a $softmax_i(\mathbf{x})$ je i-ta komponenta softmax funkcije izlaza mreže za uzorak predstavljen vektorom **x**. Prava distribucija je takva da je komponenta vektora **y** koja odgovara ispravnom razredu 1, a ostale su 0. Zato u izrazu 2.7 preostaje jedino negativni logaritam one komponente izlaza iz funkcije softmax koja označava dobivenu vjerojatnost ispravnog razreda. Cilj je minimizirati vrijednost funkcije gubitka uz pomoć uzoraka za učenje. Time se očekuje smanjiti očekivanu pogrešku klasifikacije za bilo koji novi uzorak izvan skupa za učenje. Ukupna funkcija gubitka konvolucijske mreže je srednja vrijednost gubitaka za svaki od uzoraka iz skupa za učenje. Metode učenja računaju gradijente funkcije gubitka po svim parametrima kako bi se odredio smjer pomaka vrijednosti parametara. Zbog skupoće izračuna gradijenta uz prolazak kroz čitav skup uzoraka za učenje primjenjuje se učenje na podskupovima uzoraka koji se nazivaju mini grupe (engl. *minibatch*). Uzorci za mini grupe se odabiru slučajnim odabirom kako bi se dobila nepristrana procjena gradijenta. Postupci koji ažuriraju vrijednost

parametara u odnosu na gradijent za samo jedan uzorak nazivaju se stohastički. Stohastičko i učenje na mini grupama korištenjem procjene gradijenta umjesto pravog gradijenta skupa za učenje unose regularizacijski efekt. U ovom slučaju to znači da pomažu u sprječavanju upadanja optimizacijskog postupka u neki od lokalnih minimuma.

Algoritmi učenja modela

Temeljni optimizacijski algoritam proširenjem kojeg su nastali i mnogi drugi algoritmi je stohastički gradijentni spust (engl. *Stohastic Gradient Descent*, SGD). Glavni koraci algoritma su opisani u pseudokodu koji slijedi.

Algorithm 1 Stohastički gradijentni spust, SGD

Precondition: zadana stopa učenja ϵ; početno inicijalizirani parametri Θ

1: **while** nije zadovoljen kriterij zaustavljanja **do**
2: uzorkovati mini grupu veličine m iz skupa za učenje $\{x^i ... x^m\}$ s pripadnim oznakama razreda y^i
3: izračunati procjenu gradijenta: $\hat{g} \leftarrow \frac{1}{m} \frac{\partial}{\partial \Theta} \sum_i L(f(x^i, \theta), y^i)$
4: ažurirati parametre: $\Theta \leftarrow \Theta - \epsilon \hat{g}$
5: **end while**

Glavni kriteriji za zaustavljanje algoritma koji mogu biti uzeti u obzir su: konvergencija gradijenta, smanjivanje pogreške ispod zadanog praga, dostizanje unaprijed zadanog najvećeg broja iteracija. Podešavanje stope učenja bitan je korak prema uspješnom nalaženju optimuma ciljne funkcije L. Ako je stopa premalena postupak će sporo doći do konvergencije, a u slučaju prevelike stope učenja ϵ moguće je preskakanje željenog minimuma. Podešavanje stope učenja posebno je bitna kod korištenja mini grupa zbog šuma u procjeni gradijenta. Smanjivanje stope učenja kroz iteracije može biti dobro rješenje. Umjesto unaprijed određenog koraka promjene parametra ϵ, postoji čitav spektar algoritama s adaptivnim pristupima tom problemu. Jedan od poznatijih je AdaGrad kod kojeg se za različite parametre koriste i različite stope učenja. One se adaptiraju skaliranjem s inverzom kvadratnog korijena zbroja kvadriranih vrijednosti pripadne parcijalne derivacije funkcije gubitka kroz povijest. Dodavanje momenta dodatno ubrzava učenje klasifikacijskog modela. Riječ je o akumuliranju eksponencijalnog pomičnog prosjeka prethodnih gradijenata i nastavku pomicanja u tom smjeru. To je moguće uvođenjem nove varijable v koja predstavlja spomenuti prosjek gradijenata i inicijalizirana je na 0 i parametra α s tipičnim vrijednostima iz

intervala $[0.5, 0.99]$. U stohastičkom gradijentnom spustu s korištenjem momenta se parametri Θ ažuriraju na sljedeći način:

$$v \leftarrow \alpha v - \epsilon \frac{1}{m} \frac{\partial}{\partial \Theta} \sum_i L(f(x^i, \theta), y^i) \tag{2.8}$$

$$\Theta \leftarrow \Theta + v \tag{2.9}$$

Jedan od novijih i najbolje prihvaćenih algoritama učenja je Adam (engl. *Adaptive Moments*) [8]. To je metoda koja dobro radi sa širokim skupom parametara i obuhvaća prednosti rada s momentima i adaptivnom stopom učenja. Za razliku od AdaGrad-a ne smanjuje gradijente kontinuirano već dodaje eksponencijalno gušenje starijih vrijednosti što za rezultat daje bolju prilagodbu korekcije stope učenja ϵ na ponašanje gradijenta. U nastavku su navedeni glavni koraci algoritma Adam.

Algorithm 2 Adam (Adaptive Moments)

Precondition: zadana stopa učenja ϵ (preporuka 0.001), faktori eksponencijalnog smanjivanja ρ_1 i ρ_2 (preporuka 0.9 i 0.999), malena konstanta δ za numeričku stabilizaciju (preporuka 10^{-8}); početno inicijalizirani parametri Θ

1: inicijalizirati varijable 1. i 2. momenta: s = 0, r = 0
2: inicijalizirati vremenski korak: t = 0
3: **while** nije zadovoljen kriterij zaustavljanja **do**
4: uzorkovati mini grupu veličine m iz skupa za učenje $\{x^i ... x^m\}$ s pripadnim oznakama razreda y^i
5: izračunati procjenu gradijenta: $\hat{g} \leftarrow \frac{1}{m} \frac{\partial}{\partial \Theta} \sum_i L(f(x^i, \theta), y^i)$
6: t $\leftarrow$ t + 1
7: ažurirati pristranu procjenu 1. momenta: $s \leftarrow \rho_1 s + (1 - \rho_1)\hat{g}$
8: ažurirati pristranu procjenu 2. momenta: $r \leftarrow \rho_2 r + (1 - \rho_2)\hat{g} \odot \hat{g}$
9: ispravak pristranosti 1. momenta: $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$
10: ispravak pristranosti 2. momenta: $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$
11: ažurirati parametre: $\Theta \leftarrow \Theta - \epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$
12: **end while**

Postupci regularizacije

Regularizacija je dodavanje novog člana funkciji gubitka modela kako bi se sprječila prenaučenost mreže (engl. *overfitting*). Smanjivanjem originalne greške na skupu za učenje mreža se može previše prilagoditi viđenom skupu podataka i na novim podacima pokazati slabu moć generalizacije. Često je korištena regularizacija funkcije

pogreške uz pomoć L_p normi kako bi se sprječio potencijalni kontinuirani rast težina. Regularizirana funkcija pogreške L definirana je na sljedeći način:

$$L' = L + \lambda \left\| \mathbf{w} \right\|_p \tag{2.10}$$

gdje je $\mathbf{w}$ vektor težina, a λ faktor regularizacije. Složeniji modeli imaju veće magnitude težina, a dodavanjem regularizacijskog člana funkciji pogreške i minimizacijom L' smanjuju se i L i težine. Regularizacija L_2 normom ($\left\| \mathbf{w} \right\|_2 = \sqrt{\sum_j w_j^2}$) je najčešće korištena tehnika regularizacije kod koje doprinos regularizacije ovisi o vrijednostima težina. Kod regularizacije L_1 normom ($\left\| \mathbf{w} \right\|_1 = \sum_j |w_j|$) doprinos regularizacije ovisi samo o predznaku $\mathbf{w}$ i ova regularizacija vodi na modele za koje su neke vrijednosti parametara 0. Takve modele zovemo rijetkima (engl. *sparse model*).

Slučajno izostavljanje neurona (engl. *dropout*) je postupak koji simulira istovremeno treniranje više različitih klasifikatora i usrednjavanje njihovih rezultata. Iz toga proizlazi regularizacijski efekt jer bi svaka pojedina mreža, iako sama možda i prenaučena, donijela drugačiji doprinos konačnom rezultatu. U svakoj iteraciji unaprijednog prolaza kroz mrežu s uzorcima jedne mini grupe i unatražnog u kojem se računaju gradijenti funkcije pogreške se iz rada mreže isključuju slučajno odabrani neuroni skrivenih slojeva.

Normalizacija nad grupom

Normalizacija podataka je postupak koji može pomoći mreži kod učenja, tj. optimizacije funkcije gubitka. Provođenje normalizacije svih podataka i nakon svake iteracije učenja bilo bi računski skupo pa se kao dobro rješenje koristi metoda normalizacije nad grupom (engl. *batch normalization*) [7]. Normalizacija nad grupom je postupak normalizacije izlaza svakog neurona na temelju izlaza tog neurona za sve uzorke trenutne mini grupe. Nakon što je dobivena matrica $\mathbf{H}$ odziva svih neurona za jednu mini grupu, potrebno je izračunati srednju vrijednost i standardno odstupanje vrijednosti iz te matrice. Nova matrica odziva koja ide dalje kroz mrežu sadrži odzive normalizirane po stupcima:

$$\hat{h}_{kl} = \frac{h_{kl} - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} \tag{2.11}$$

Normalizirani odzivi matrice $\hat{\mathbf{H}}$ mogu se podvrgnuti dodatnoj afinoj transformaciji skaliranjem i dodavanjem fiksnog pomaka zbog vraćanja ekspresivnosti mreže koja je smanjena normalizacijom. Parametri afine transformacije su različiti za svaki neuron i uče se gradijentnim spustom. Normalizacija nad grupom se provodi prije propuštanja

izlaza kroz nelinearnost zglobnice. Kod normalizacije nad grupom koja se provodi u konvolucijskom sloju ideja ostaje ista, uz promjene koje slijede karakterističnost same konvolucije. Promjena se uvodi na način da se svi elementi izlazne mape značajki normaliziraju jednako. Svaki element izlazne mape se normalizira u odnosu na sve druge elemente iste mape značajki za sve uzorke trenutne mini grupe. Parametri afine transformacije se uče za svaku mapu značajki umjesto za svaki neuron.

2.2. Arhitektura klasifikacijskog modela

Za rješavanje klasifikacijskog problema u ovom radu, korištena je konvolucijska mreža arhitekture inspirirane mrežom VGG sa 16 slojeva parametara. To je mreža koju je razvio Visual Geometry Group (VGG) sa Svečilišta Oxford [14], a koja se pokazala vrlo uspješnom u klasifikaciji slika iz javne baze ImageNet. Osim arhitekture, javno su dostupne i težine koje je mreža naučila gledajući milijun slika s ImageNet-a. Inicijalizacijom vlastite mreže s tim težinama se postupak učenja znatno ubrzava. Glavna razlika između originalne VGG16 arhitekture i ovdje korištene implementacije je ispuštanje jednog potpuno povezanog sloja i smanjenje postojećih. Umjesto dva potpuno povezana sloja veličine 4096 i posljednjeg veličine 1000, koristi se jedan sloj s 2048 neurona i posljednji sa samo 2 jer je riječ o binarnoj klasifikaciji. Sve konvolucijske jezgre su veličine 3×3, a broj izlaznih mapa kroz slojeve varira od 64-512. Konvolucijski slojevi slagani su u blokovima po 2 ili 3 sloja. Tri uzastopna opisana konvolucijska sloja efektivno imaju receptivno polje veličine 7×7, ali umjesto jedne imaju trostruku nelinearnost od strane aktivacijskih funkcija. To doprinosi diskriminativnosti konačne funkcije odluke, a u isto vrijeme je broj parametara kod 3 konvolucijska sloja s jezgrama 3×3 značajno manji nego kod 1 s jezgrom 7×7. Između kovolucijskih blokova korišteno je sažimanje maksimalnom vrijednošću uz prozor veličine 2×2, a na izlazu se nalazi funkcija softmax. Detaljna arhitektura konvolucijske mreže se nalazi u tablici 2.1.

RGB slika na ulazu
conv3-64
conv3-64
maxpool
conv3-128
conv3-128
maxpool
conv3-256
conv3-256
conv3-256
maxpool
conv3-512
conv3-512
conv3-512
maxpool
conv3-512
conv3-512
conv3-512
maxpool
FC-2048
FC-2
softmax

Tablica 2.1: Arhitektura konvolucijske mreže.

3. Fisherovi vektori

Prije razvitka dubokih modela za klasifikaciju slika je najčešće korišten model zbirke (vizualnih) riječi (engl. *Bag of Words*, BoW). Nakon grupiranja opisnika slika (npr. SIFT opisnika [10]) dobiveni centroidi postaju vizualne riječi. Za svaki postojeći opisnik slike se odredi koja mu je vizualna riječ najsličnija. Tada se slika može predstaviti histogramom učestalosti pojavljivanja svake od vizualnih riječi. Klasifikacija se temelji na usporedbi histograma umjesto puno većih skupova opisnika slika. Može se primjetiti kako model zbirke riječi ne sadrži informaciju o udaljenostima opisnika od najbliže vizualne riječi. Predstavljanje slika Fisherovim vektorima ispravlja neke od nedostataka zbirke riječi. Riječ je o pristupu temeljenom na Fisherovoj jezgri koji uzima u obzir generativni model podataka. Fisherov vektor neke slike iz skupa podataka prikazuje odstupanje opisnika slike od generativnog modela svih opisnika tog skupa. Još neke od prednosti Fisherovih vektora u odnosu na zbirku riječi su bolji rezultati klasifikacije jednostavnim linearnim klasifikatorima i to što je za izračun Fisherovih vektora potreban manji skup opisnika. [13] U odjeljku 3.2 će biti detaljnije opisani Fisherovi vektori. Za bolje razumijevanje je potrebno objasniti koncepte jezgrenih funkcija i Fisherove jezgre.

3.1. Jezgrene funkcije

Algoritmi strojnog učenja za prepoznavanje i analizu uzoraka za glavni zadatak imaju pronalaženje i proučavanje veza među podacima u skupovima točaka, vektora, slika, tekstualnih dokumenata i sl. Dobivene informacije mogu se koristiti za grupiranje, rangiranje, koreliranje ili klasifikaciju podataka. Postoji podskup tih algoritama koji se ističu zbog korištenja jezgrenih funkcija. Algoritmi pomoću jezgrenih funkcija preslikavaju podatke u prostor veće dimenzionalnosti u kojem će podaci biti bolje strukturirani i lakše razdvojivi. Općenito, takvu funkciju preslikavanja možemo zapisati kao $f(x) : \mathbb{R}^m \mapsto \mathbb{R}^n$, gdje je $m < n$.

Uzimajući u obzir χ prostor značajki, jezgrenu funkciju možemo zapisati na sljedeći način:

$$\kappa : \chi \times \chi \mapsto \mathbb{R} \tag{3.1}$$

Jezgrena funkcija se koristi kao mjera sličnosti između podataka $\mathbf{x}$ i $\mathbf{x}'$ iz skupa χ i tada najčešće zadovoljava svojstva simetričnosti: $\forall \mathbf{x}, \mathbf{x}' \in \chi, \kappa(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}', \mathbf{x})$ i pozitivnosti $\forall \mathbf{x}, \mathbf{x}' \in \chi, \kappa(\mathbf{x}, \mathbf{x}') \geq 0$.
Za $X = \{x_1, ..., x_n\}$ koji označava set od n uzoraka iz χ može se definirati matrica $\mathbf{K}(X, \kappa)$ ili skraćeno $\mathbf{K}$ tako da je $\mathbf{K}_{ij} = \kappa(x_i, x_j)$. Tako definirana matrica $\mathbf{K}$ naziva se Gramovom matricom. Ako je matrica $\mathbf{K}$ pozitivno definitna za $\forall X \subseteq \chi$, jezgrena funkcija κ se naziva Mercerova ili pozitivno definitna jezgra.
Simetrična matrica $\mathbf{M}$ je pozitivno definitna ukoliko za bilo koji netrivijalni vektor $\mathbf{x} \neq 0$ vrijedi:

$$\mathbf{x^T M x} > \mathbf{0} \tag{3.2}$$

Prema Mercerovom teoremu za pozitivno definitnu matricu $\mathbf{K}$ možemo izračunati dekompoziciju:

$$\mathbf{K} = \mathbf{U}^T \Lambda \mathbf{U} \tag{3.3}$$

gdje je Λ dijagonalna matrica koja sadrži n svojstvenih vrijednosti matrice $\mathbf{K}$. Definiranjem funkcije $\phi(x_i) = \sqrt{\Lambda}\mathbf{U}_{:,i}$ dekompozicija $\mathbf{K}_{ij}$ se može zapisati kao:

$$\mathbf{K}_{ij} = \phi(x_i)^T \phi(x_j) \tag{3.4}$$

Dakle, za svaku Mercerovu jezgru postoji funkcija ϕ koja preslikava $\mathbf{x}$ i $\mathbf{x}'$, elemente našeg skupa X, u prostor značajki za koji ne postoji ograničenje dimenzionalnosti. To se može opisati pomoću skalarnog umnoška funkcije ϕ primjenjene na vektore $\mathbf{x}$ i $\mathbf{x}'$:

$$\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}') \tag{3.5}$$

Snaga metoda koje koriste jezgrene funkcije je preslikavanje podataka u prostor značajki veće dimenzionalnosti u kojem je te iste podatke moguće odvojiti jednostavnim linearnim klasifikatorom. Postoje jezgrene funkcije koje imaju tu prednost da se za preslikavanje nekog vektora $\mathbf{x}$ u drugi prostor značajki ne treba računati bazna funkcija $\phi(\mathbf{x})$. Dovoljno je dobiti mjeru sličnosti između $\mathbf{x}$ i $\mathbf{x}'$ izračunom matrice $\mathbf{K}$. Time operacije ostaju u prostoru veličine $n \times n$. Taj se postupak naziva jezgrenim trikom (engl. *kernel trick*).

Neke od najčešće korištenih jezgara su linearna i Gaussova. Uz $\phi(\mathbf{x}) = \mathbf{x}$ linearna jezgra je definirana kao skalarni umnožak vektora $\mathbf{x}$ i $\mathbf{x}'$ koji je definiran u jednadžbi

3.5. U ovom slučaju se preslikavanje radi u prostor jednake dimenzionalnosti što je karakteristično za slučajeve kad je dimenzionalnost podataka vrlo visoka, a podaci razdvojivi.
Gaussova jezgra, poznata i kao RBF (engl. *radial basis function*) definira se na sljedeći način:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Sigma (\mathbf{x} - \mathbf{x}')) \tag{3.6}$$

gdje je Σ matrica kovarijance značajki. Gaussova jezgra preslikava podatke u prostor beskonačne dimenzionalnosti, što se postiže korištenjem jezgrenog trika. [5] [11]

U sljedećem odjeljku će biti detaljnije predstavljena Fisherova jezgra, čija je ideja prikaz udaljenosti u prostoru izglednosti između različitih objekata za neki naučeni generativni model.

3.1.1. Fisherova jezgra

Metode za klasifikaciju uzoraka mogu se podjeliti prema tome imaju li diskriminativni ili generativni pristup. Kod diskriminativnog pristupa naglasak je izravno na klasifikaciji podataka, bez modeliranja distribucije vjerojatnosti podataka kao što je slučaj kod generativnog pristupa. Ta razlika teoretsku prednost kod postizanja velike točnosti u klasifikaciji daje diskriminativnim metodama. Neke od prednosti generativnih metoda su mogućnost za rad s neoznačenim podacima, generiranje novih podataka i pronalazak bitnih značajki u podacima.

Fisherova jezgra kombinira prednosti diskriminativnog i generativnog pristupa.
$X = \{x_t.t = 1...T\}$ označava set od T uzoraka iz χ. Onda je u_λ funkcija gustoće vjerojatnosti koja modelira generativni proces elemenata iz χ, a $\lambda = \{\lambda_i, i = 1...M\}$ je skup od M parametara funkcije u_λ. U statistici se koristi izraz za gradijent logaritma izglednosti podataka u odnosu na model:

$$G_\lambda^X = \frac{\partial}{\partial \lambda} \log u_\lambda(X) \tag{3.7}$$

Gradijent iz jednadžbe 3.7 naziva se Fisherova mjera i opisuje doprinos svakog parametra na generativni proces i kako bi se isti trebali promjeniti da bolje opisuju X. Dimenzionalnost G_λ^X ovisi isključivo o broju parametara M u λ. [13]

Prema jednadžbi 3.8 očekivanje i ujedno prvi moment Fisherove mjere je 0.

$$\begin{aligned} E\left[\frac{\partial}{\partial\lambda}\log u_\lambda(X) \mid \lambda\right] &= \int \frac{\frac{\partial}{\partial\lambda}u_\lambda(x)}{u_\lambda(x)}u_\lambda(x)dx \\ &= \frac{\partial}{\partial\lambda}\int u_\lambda(x)dx \\ &= \frac{\partial}{\partial\lambda}1 = 0 \end{aligned} \tag{3.8}$$

Fisherova informacija odgovara varijanci i ujedno drugom momentu Fisherove mjere:

$$F_\lambda = E\left[\left(\frac{\partial}{\partial\lambda}\log u_\lambda(X)\right)^2 \mid \lambda\right] = \int\left(\frac{\partial}{\partial\lambda}\log u_\lambda(x)\right)^2 u_\lambda(x)dx \tag{3.9}$$

Fisherova informacija se može promatrati i kao mjera za količinu informacije koju slučajna varijabla X nosi o parametru λ svoje funkcije gustoće vjerojatnosti u_λ. Ukoliko u_λ ima M parametara, tako da je $\lambda = \{\lambda_i, i = 1...M\}$, Fisherova informacija se zapisuje u matričnom obliku:

$$F_\lambda = E_{x\sim u_\lambda}\left[(G_\lambda^X)(G_\lambda^X)^T\right] \tag{3.10}$$

Fisherova informacijska matrica ima dimenzije $M \times M$. To je pozitivna semidefinitna simetrična matrica. Ako je pozitivno definitna može biti promatrana kao Riemmanova metrika. [12] To je metrika definirana na Riemannovoj mnogostrukosti (engl. *manifold*). Predstavljanje gradijentnog prostora parametara modela uz pomoć Riemannove mnogostrukosti omogućava usporedbu podataka u tom prostoru. Skup distribucija $\mathcal{U} = \{u_\lambda, \lambda \in \Lambda\}$ promatran kao Riemannova mnogostrukost označava se s M_Λ. Mnogostrukost u n dimenzija je topološki prostor kod kojeg svaka točka ima susjedstvo homeomorfno Euklidskom prostoru dimenzije n. Riemannova mnogostrukost ima definiran Riemmanov metrički tenzor u svakoj točki p. Metrički tenzor je funkcija koja u točki p uzima tangencijalne vektore i daje skalarni umnožak g_p. Time omogućava definiranje i izračun duljina zaobljenosti na mnogostrukosti, kuteva, površina, zakrivljenosti i gradijenata funkcija.

Fisherova informacija se može promatrati kao zakrivljenost grafa logaritma izglednosti. Gledajući zakrivljenosti maksimuma logaritamske izglednosti za neki parametar λ funkcije gustoće vjerojatnosti u_λ i uzorak X zaključuje se sljedeće: ako $u_\lambda(X)$ ima oštre ekstreme onda X pruža puno informacija o parametru λ, a ako je $u_\lambda(X)$ raširena funkcija, sam X ne govori puno o λ. Zato vrijedi promatrati varijancu funkcije $u_\lambda(X)$ nazvanu Fisherovom informacijom.

Time dolazimo i do same Fisherove jezgre (engl. *Fisher kernel*) koja je mjera sličnosti između uzoraka X i Y:

$$K_{FK}(X, Y) = (G_\lambda^X)^T F_\lambda^{-1} (G_\lambda^Y) \tag{3.11}$$

Zbog računske složenosti Fisherove jezgre, ona se u postupcima računanja koristi u aproksimiranom obliku. [13] Iz oblika Fisherove jezgre dobivenog aproksimacijom Fisherove informacijske matrice dijagonalnom matricom može se izlučiti izraz za Fisherov vektor. To je vektor koji se dobiva sažimanjem lokalnih značajki slike i koji će biti detaljnije opisan u nastavku.

3.2. Fisherov vektor

Za Fisherovu informacijsku matricu F_λ i njezin inverz vrijedi da su pozitivne semidefinitne matrice. Zato se F_λ^{-1} može faktorizirati Choleskyjevom dekompozicijom:

$$F_\lambda^{-1} = L_\lambda^T L_\lambda \tag{3.12}$$

gdje je L donjetrokutasta matrica. Koristeći to svojstvo, Fisherova jezgra se može zapisati i u obliku skalarnog umnoška:

$$K_{FK}(X, Y) = (\mathcal{G}_\lambda^X)^T (\mathcal{G}_\lambda^Y) \tag{3.13}$$

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X = L_\lambda \frac{\partial}{\partial \lambda} \log u_\lambda(X) \tag{3.14}$$

Normalizirani vektor gradijenta iz jednadžbe 3.14 naziva se Fisherovim vektorom uzorka X. Dimenzionalnost mu je jednaka dimenzionalnosti gradijentnog vektora G_λ^X. Korištenje Fisherovih vektora u linearnom klasifikatoru ekvivalentno je korištenju Fisherove jezgre u nekoj nelinearnoj metodi. Velike prednosti korištenja linearnog klasifikatora su brža evaluacija i jednostavnost.

Kao funkcija gustoće vjerojatnosti u_λ koristi se model Gaussove mješavine (GMM, engl. *Gaussian mixture model*). GMM može s proizvoljnom preciznošću aproksimirati bilo koju kontinuiranu razdiobu. Parametri GMM-a s K komponenata su $\lambda = \{w_k, \mu_k, \Sigma_k, k = 1...K\}$. Pritom je w_k težinski faktor mješavine, μ_k vektor srednje vrijednosti, a Σ_k kovarijacijska matrica za k-tu Gaussovu razdiobu. Uz u_k kao oznaku k-te Gaussove razdiobe u_λ postaje:

$$u_\lambda(x) = \sum_{k=1}^{K} w_k u_k(x) \tag{3.15}$$

$$u_k(x) = \frac{1}{(2\pi)^{D/2} \left|\Sigma_k\right|^{1/2}} \exp\left\{-\frac{1}{2}\left(x-\mu_k\right)^T \Sigma_k^{-1}\left(x-\mu_k\right)\right\} \tag{3.16}$$

Da bi u_λ bila valjana distribucija, mora vrijediti sljedeće:

$$\forall_k : w_k \geq 0, \sum_{k=1}^{K} w_k = 1 \tag{3.17}$$

Uobičajena pretpostavka je da je kovarijacijska matrica dijagonalna matrica i tada se dijagonala Σ_k, tj. vektor varijanci označava s σ_k^2. U postupku učenja parametara GMM-a koristi se algoritam maksimizacije očekivanja (EM, engl. *Expectation-Maximization*) kako bi se optimirao kriterij maksimalne izglednosti (ML, engl. *Maximum Likelihood*). Uzorci za treniranje koji se pritom koriste mogu biti vektori značajki slika iz nekog skupa za učenje. Ti vektori značajki su često lokalni opisnici dobiveni npr. SIFT algoritmom (engl. *Scale Invariant Feature Transform*). To je algoritam Davida Lowea iz 1999. godine [10] za pronalazak i opis istaknutih lokalnih značajki u slikama. Druga mogućnost je korištenje konvolucijskih značajki ekstrahiranih s izlaza nekog od slojeva konvolucijske neuronske mreže. Detaljnije o tom pristupu bit će napisano u sljedećem odjeljku. Općenito, skup T vektora značajki koje opisuju neku sliku možemo označiti s $X = \{x_t, t = 1...T\}$.

Kako bi se osigurala zadovoljenost jednadžbe 3.17 koristi se reparametrizacija težinskih faktora w_k uz uvođenje α_k i definiranje sljedećeg preslikavanja:

$$w_k = \frac{e^{\alpha_k}}{\sum_{j=1}^{K} e^{\alpha_j}} \tag{3.18}$$

Gradijenti izglednosti nekog vektora značajki x_t u odnosu na parametre $\lambda = \{w_k, \mu_k, \Sigma_k, k = 1...K\}$ GMM modela definirani su na sljedeći način:

$$\frac{\partial}{\partial \alpha_k} \log u_\lambda(x_t) = \gamma_t(k) - w_k \tag{3.19}$$

$$\frac{\partial}{\partial \mu_k} \log u_\lambda(x_t) = \gamma_t(k)\left(\frac{x_t - \mu_k}{\sigma_k^2}\right) \tag{3.20}$$

$$\frac{\partial}{\partial \sigma_k} \log u_\lambda(x_t) = \gamma_t(k)\left[\frac{\left(x_t - \mu_k\right)^2}{\sigma_k^3} - \frac{1}{\sigma_k}\right] \tag{3.21}$$

Posteriorna vjerojatnost ili odgovornost (engl. *responsibility*) $\gamma_t(k)$ daje informaciju o tome koliko je k-ta komponenta GMM-a pridonijela generiranju vektora podataka x_t:

$$\gamma_t(k) = \frac{w_k u_k\left(x_t\right)}{\sum_{j=1}^{K} w_j u_j\left(x_t\right)} \tag{3.22}$$

Zbog velike dimenzionalnosti vektora x_t i eksponencijale u jednadžbi 3.16 vrijednosti izglednosti za k-tu komponentu GMM-a $u_k(x_t)$ mogu biti jako malene. Da bi se postigla stabilnost računskih postupaka u praksi se izglednost u_k računa u logaritamskoj domeni:

$$\log u_k(x) = -\frac{1}{2}\sum_{d=1}^{D}\left[\log(2\pi) + \log\left(\sigma_{kd}^2\right) + \frac{(x_d - \mu_{kd})^2}{\sigma_{kd}^2}\right] \tag{3.23}$$

Pritom d označava jednu od D dimenzija vektora x_t. Da bi se izračunao logaritam izglednosti definirane jednadžbom 3.15 inkrementalno se računa $\log u_\lambda(x) = \log\left(w_1 u_1(x) + \sum_{k=2}^{K} w_k u_k(x)\right)$.

Na putu do izračuna Fisherovog vektora kao u jednadžbi 3.14 za skup vektora značajki neke slike nedostaje još L_λ. Za matricu L_λ vrijedi:

$$L_\lambda = \sqrt{F_\lambda^{-1}} \tag{3.24}$$

Pod pretpostavkom dijagonalnosti matrice F_λ, normalizacijom vektora gradijenata dobivaju se sljedeći normalizirani gradijenti koji su komponente Fisherovog vektora:

$$\mathcal{G}_{\alpha_k}^X = \frac{1}{\sqrt{w_k}}\sum_{t=1}^{T}(\gamma_t(k) - w_k) \tag{3.25}$$

$$\mathcal{G}_{\mu k}^X = \frac{1}{\sqrt{w_k}}\sum_{t=1}^{T}\gamma_t(k)\left(\frac{x_t - \mu_k}{\sigma_k}\right) \tag{3.26}$$

$$\mathcal{G}_{\sigma_k}^X = \frac{1}{\sqrt{w_k}}\sum_{t=1}^{T}\gamma_t(k)\frac{1}{\sqrt{2}}\left[\frac{(x_t - \mu_k)^2}{\sigma_k} - 1\right] \tag{3.27}$$

Konačan oblik Fisherovog vektora dobiva se konkatenacijom D-dimenzionalnih vektora $\mathcal{G}_{\mu k}^X$ i $\mathcal{G}_{\sigma_k}^X$ i skalara $\mathcal{G}_{\alpha_k}^X$ za svih K komponenata GMM-a. Zato je dimenzija Fisherovog vektora $(2D+1)K$. [13]

3.2.1. Normalizacija Fisherovih vektora

Postupci normalizacije poboljšavaju rezultate klasifikacijskih algoritama koji koriste Fisherove vektore.

U slučajevima kad su veličine skupa značajki neke slike X različite, Fisherov vektor se normalizira veličinom uzorka T:

$$\mathcal{G}_\lambda^X \leftarrow \frac{1}{T}\mathcal{G}_\lambda^X \tag{3.28}$$

Time se poništava utjecaj veličine skupa X.

Preslikavanje vektora značajki slike u prostor Fisherovih vektora radi se pod pretpostavkom međusobne neovisnosti vektora značajki što u stvarnosti nije slučaj. Normalizacija potenciranjem (engl. *power normalization*) umanjuje negativne efekte takve pretpostavke.
Postupak normalizacije potenciranjem nad vektorom $\mathbf{z}$ opisan je sljedećom jednadžbom:

$$\mathbf{z} \leftarrow sign\left(\mathbf{z}\right)\left|\mathbf{z}\right|^\rho, 0 < \rho < 1 \tag{3.29}$$

Taj se postupak radi po svakoj dimenziji Fisherovog vektora $\mathcal{G}_\lambda^X$. Parametar ρ postavlja se na $\frac{1}{2}$. Povećanjem broja komponenata GMM-a, smanjuje se udio vrijednosti u Fisherovom vektoru koji nose neku informaciju. Takva struktura problematična je kod izračuna skalarnih umnožaka. Normalizacija potenciranjem uspijeva povećati broj vrijednosti u Fisherovom vektoru koje su bitno različite od 0.

Uobičajena je pojava da različite slike imaju različit udio pozadinske informacije zbog čega će i njihovi Fisherovi vektori biti različitiji nego što se sami sadržaji slika razlikuju. Kako bi se poništio negativan utjecaj te pojave koristi se L_2 normalizacija, iako bi se sličan efekt postigao dijeljenjem Fisherovog vektora bilo kojom L_p normom. [13]
L_2 normalizacija nad vektorom $\mathbf{z}$ koji ima K komponenata opisana je sljedećom jednadžbom:

$$\mathbf{z} \leftarrow \frac{\mathbf{z}}{\sqrt{\sum_{k=1}^{K}\left|z_k\right|^2}} \tag{3.30}$$

3.3. Arhitektura klasifikacijskog modela

Kao što je spomenuto u odjeljku 3.1, doprinos jezgrenih funkcija u proceduri klasifikacije slika je u tome što omogućuju preslikavanje vektora značajki koji opisuju sliku u prostor različite dimenzionalnosti. U tom novom prostoru dobivaju se značajke koje drukčije opisuju sliku i kada te informacije dobro ističu specifičnosti pojedine slike u odnosu na ostale iz skupa, čitav skup postaje lakše odvojiv. To znači i da postaje moguće efikasno treniranje nekog jednostavnijeg klasifikatora. Linearni klasifikatori

koji su često korišteni u takvoj proceduri klasifikacije su logistička regresija i linearni SVM (engl. *Support Vector Machine*).

Logistička regresija je diskriminativni algoritam koji definira linearne granice između klasa. Afina redukcija podataka se spljošćuje na interval $[0, 1]$ i time kao rezultat daje vjerojatnosnu razdiobu podataka po mogućim razredima. U binarnom slučaju mogući razredi su $C = \{0, 1\}$. Podatak na ulazu u model je vektor $\mathbf{x}$ dimenzija $1 \times D$, a parametri modela su parametar $\mathbf{w}$ dimenzija $D \times 1$ i skalar b. Afina redukcija definirana je kao $\mathbf{wx} + b$. Vjerojatnost pripadanja podatka $\mathbf{x}$ razredima c_0 i c_1, označena s Y, definirana je uz pomoć sigmoidalne funkcije:

$$P(Y = c_1|\mathbf{x}) = \sigma(\mathbf{wx} + b) \tag{3.31}$$

$$P(Y = c_0|\mathbf{x}) = 1 - P(c_1|\mathbf{x}) \tag{3.32}$$

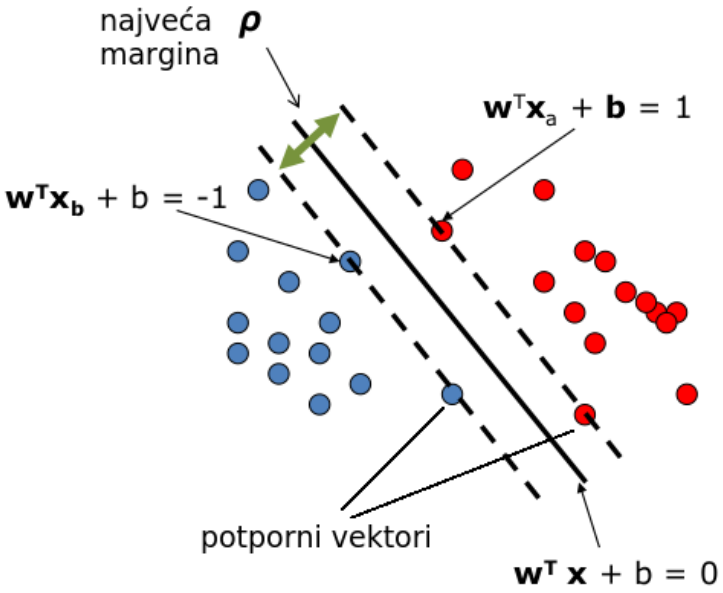
Na početku postupka učenja parametri modela $\mathbf{w}$ i b su slučajno inicijalizirani, a cilj učenja je pronaći vrijednosti $\mathbf{w}$ i b koji maksimiziraju vjerojatnost pripadanja podatka $\mathbf{x}$ ispravnom razredu. Proširivanjem problema na čitav skup N ulaznih podataka predstavljenih matricom $\mathbf{X}$ dimenzija $N \times D$, cilj učenja postaje optimirati matricu parametara $\mathbf{W}$ dimenzija $D \times C$ i vektor $\mathbf{b}$ dimenzija $N \times C$. Uz $(\mathbf{x}_i, y_i)$ koji predstavlja par vektora podatka i pripadajuće ispravne klase, moguće je definirati funkciju gubitka L koja kažnjava krivo klasificirane podatke. Kao i kod učenja neuronske mreže, koje je opisano u odjeljku 2.1.2, i ovdje gubitak odgovara unakrsnoj entropiji. Uz korišteni zapis, funkcija gubitka se definira na sljedeći način:

$$L(\mathbf{W}, \mathbf{b}|\mathbf{x}_i, y_i) = -\sum_{i=1}^{N} \log P(Y = y_i|\mathbf{x}_i) \tag{3.33}$$

Izraz opisuje ukupni gubitak jer zbroj iterira po svim ulaznim primjerima. Zbog konveksnosti funkcije gubitka L je moguće optimiranje parametara metodama poput gradijentnog spusta bez opasnosti od upadanja u lokalni optimum. Optimiranje se izvodi iterativno, deriviranjem funkcije gubitka po parametrima $\mathbf{W}$ i $\mathbf{b}$ i pomicanjem u smjeru negativnih gradijenata. S težinom problema povećava se i broj iteracija potrebnih za konvergenciju.

Klasifikator raširene upotrebe koji u svojoj linearnoj verziji daje rezultate bliske logističkoj regresiji je metoda potpornih vektora (SVM, engl. *Support Vector Machines*). SVM je, za razliku od logističke regresije, deterministička metoda. Cilj je pronalazak hiper-ravnine koja ima najveću marginu razdvajanja razreda. Margina (engl. *decision boundary*) se definira kao udaljenost između točaka iz skupa podataka koje su najbliže

plohi razdvajanja. Funkcija odluke definirana je preko potpornih vektora (SV, engl. *support vectors*), onih točaka iz skupa podataka za učenje koje se nalaze najbliže plohi razdvajanja.



Slika 3.1: Metoda potpornih vektora. [2]

Hiper-ravnina razdvajanja je $\mathbf{w}$, uobičajene oznake razreda su -1 i 1, a klasifikator je određen funkcijom $sign(\mathbf{w}\mathbf{x}_i + b)$. Kao što se vidi na slici 3.1, margina ρ je širina razdvajanja između potpornih vektora suprotnih klasa, a udaljenost između plohe i točke x je:

$$r = \frac{\mathbf{w}\mathbf{x}_i + b}{\|\mathbf{w}\|} \tag{3.34}$$

Uz pretpostavku da je $r \geq 1$, vrijede sljedeća ograničenja na parovima primjera za učenje $(\mathbf{x}_i, y_i)$:

$$y_i = 1 \rightarrow \mathbf{w}\mathbf{x}_i + b \geq 1 \tag{3.35}$$

$$y_i = -1 \rightarrow \mathbf{w}\mathbf{x}_i + b \leq -1 \tag{3.36}$$

Uz r definiran kao u jednadžbi 3.34, margina je:

$$\rho = \frac{2}{\|\mathbf{w}\|} \tag{3.37}$$

Time su formulirani parametri optimizacijskog problema koji se može preoblikovati u minimizaciju $\|\mathbf{w}\|$. To je kvadratni minimizacijski problem uz ograničenja u kojem se traže takvi $\mathbf{w}$ i b takvi da $\|\mathbf{w}\|$ odnosno $\mathbf{ww}$ imaju minimalnu vrijednost. Način

rješavanja tog optimizacijskog problema je prevođenje u dualni problem i korištenje Lagrangeovih multiplikatora. [2]
Postoji i nelinearni SVM koji izravno koristi prednosti jezgrenih funkcija opisanih u 3.1 tako da preslikava linearno neodvojive podatke u prikladniji prostor značajki.

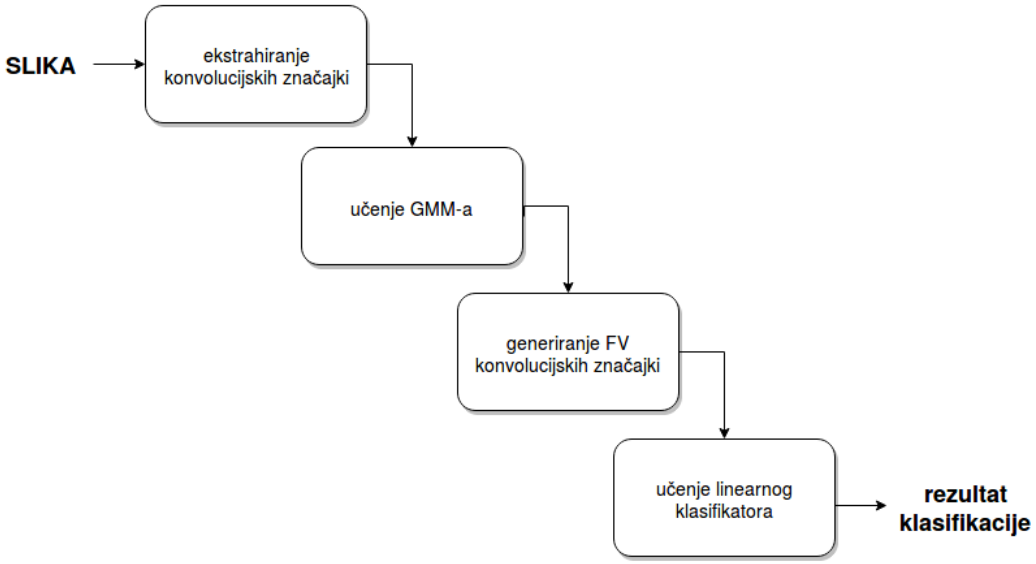
Prvi korak u arhitekturi klasifikacijskog modela uz pomoć Fisherovih vektora značajki je dobivanje prikladnih značajki iz skupa slika koji se koristi za učenje i testiranje. Za ovaj rad su za to korištene konvolucijske značajke. Riječ je o značajkama koje su ekstrahirane uz pomoć javne parametrizacije VGG19 mreže. Uspješan prelazak iz slikovne u simboličku domenu uz pomoć konvolucijske neuronske mreže koja pri svom treniranju nije vidjela taj skup slika moguć je zbog učenja iste uz pomoć jako velikog broja različitih slika iz ImageNet baze. Reprezentacija slike konvolucijskim značajkama je većih dimenzija, ali samo djelomično popunjena s vrijednostima različitima od 0. Te preostale vrijednosti ističu dijelove slike bitne za klasifikaciju.

Sljedeći korak je učenje GMM-a. Za to se koriste konvolucijske značajke slika iz skupova za učenje i testiranje. Slučajnim odabirom uzima se $\approx 2 \times 10^6$ značajki i uz pomoć algoritma maksimizacije očekivanja (engl. *Expectation Maximization*, EM) uče parametri GMM-a sa 128 komponenata. Svi se parametri zapisuju i spremaju u matričnom obliku.

Uz pomoć naučenog GMM-a se generiraju Fisherovi vektori konvolucijskih značajki. Svaki Fisherov vektor sadrži gradijente izglednosti značajki koje pripadaju određenoj slici. Ti su gradijenti sažeti i normalizirani L_2 i normalizacijom potenciranjem. U odnosu na sirove konvolucijske značajke, Fisherov vektor dodatno naglašava dijelove slike koji ga razlikuju od svih ostalih.

Posljednji dio procedure prije dobivanja rezultata klasifikacije je učenje prikladnog linearnog klasifikatora. Zbog manje složenosti tog postupka i u cilju dobivanja najboljeg mogućeg rezultata ovdje su trenirana 2 slična modela - logistička regresija i linearni SVM.

Slika 3.2 daje kratki pregled svih dijelova arhitekture modela za klasifikaciju slika koji koristi konvolucijske značajke, Fisherove vektore i plitki linearni klasifikator.



Slika 3.2: Arhitektura klasifikacijskog modela.

4. Korišteni skupovi podataka

U svim eksperimentima su korištena dva skupa podataka - Zebre i Znakovi. To su dva skupa slika za učenje i testiranje slična po tome što prikazuju scene s prometnica, a glavna razlika među njima je veličina samog predmeta interesa klasifikacije. Prosječna veličina zebre na slici je puno veća nego veličina prometnog znaka.

4.1. Zebre

Slike iz ovog skupa podataka dobivene su iz geo-referenciranih videa snimljenih za E-roads projekt u hrvatskim gradovima Karlovcu i Sisku. Slike su prikupljene i označene za istraživanje automatskog mapiranja cestovnog okruženja sa slabo nadziranim lokalizacijskim modelom u suradnji FER-a i francuskog instituta INRIA. [19] Za ovaj rad koriste se samo oznake koje govore postoji li u slici zebra ili ne. Slike su skalirane na veličinu 1280×720 piksela. U tablici 4.1 je prikazana veličina skupa i udio pozitiva i negativa. Pozitivi prikazuju zebre s različitih udaljenosti i različitih stupnjeva istrošenosti, a negativi često sadrže objekte uzoraka sličnih zebri. Slike mogu sadržavati više od jedne zebre, a veličine objekta interesa variraju od manje od 1% ukupne veličine slike (30% objekata) pa do više od 7% (22% objekata).

	Veličina	**Broj pozitiva**
Skup za učenje	1299 slika	572 slika
Skup za testiranje	1028 slika	495 slika

Tablica 4.1: Skup slika Zebre.

Na slici 4.1 možemo vidjeti primjer negativa s objektima sličnima zebri, a na slici 4.2 pozitive s različitim brojem i izgledom zebri.

Slika 4.1: Primjer negativa u skupu Zebre.

(a) Jedan objekt

(b) Dva objekta

Slika 4.2: Primjeri pozitiva u skupu Zebre.

4.2. Znakovi

Puni naziv ovog skupa podataka koji je nastao u sklopu projekta MULTICLOD (*Multi-class Object Detection*) je "The traffic sign dataset TS2010A". [18] Riječ je o projektu

koji se bavi detekcijom i lokaliziranjem različitih objekata u slici, a u svrhu korištenja u prometu i transportu. Slike su snimljene na hrvatskim lokalnim cestama i veličine su 720×576 piksela. Slike na kojima se nalaze prometni znakovi mogu sadržavati jedan ili više prometnih znakova. Ukupan broj različitih znakova (prema Bečkoj konvenciji o cestovnom prometu) je 87, a broj različitih oblika znakova je 10. Za eksperimente izvođene za ovaj rad, od interesa su samo znakovi trokutastog oblika. To su znakovi upozorenja na cestama i postoje 33 različita tipa trokutastih znakova. U tablici 4.2 je prikazana veličina skupa i udio pozitiva i negativa. Među pozitivima su većinski slike s jednim trokutastim znakovima i dio s 2. Najveći izazov kod klasificiranja ovog skupa slika je to što su prometni znakovi često veličine manje od 30×30 piksela, što čini samo 0.2% slike.

	Veličina	**Broj pozitiva**
Skup za učenje	1705 slika	542 slika
Skup za testiranje	1591 slika	530 slika

Tablica 4.2: Skup slika Znakovi.

Na slici 4.3 možemo vidjeti primjer negativa, a na slici 4.4 pozitive s različitim veličinama i brojem prometnih znakova koji su od interesa.

Slika 4.3: Primjer negativa u skupu Znakovi.

(a) Jedan trokutasti znak

(b) Dva trokutasta znaka

Slika 4.4: Primjeri pozitiva u skupu Znakovi.

5. Detalji implementacije

5.1. Korišteni alati

Izvedba klasifikacijskih modela opisanih u odjeljcima 2.2 i 3.3 napravljena je u programskom jeziku Python. Python je minimalistički programski jezik koji korisniku dopušta puno veću usredotočenost na problem koji rješava nego na detalje same izvedbe. To je moguće jer je riječ o jeziku visoke razine. Zato je jedan od glavnih nedostataka Pythona sporost izvođenja. Korištenje biblioteka napisanih u programskim jezicima kao što su C ili C++ značajno ubrzava izvođenje koda. Postoji veliki broj takvih biblioteka koje su dobro prilagođene za korištenje unutar Python koda. Slijedi popis korištenih biblioteka:

NumPy - Biblioteka široke upotrebe napisana u C-u, korištena za velik broj različitih operacija s vektorima i matricama.

SciPy - Biblioteka koja obuhvaća brojne module za optimizaciju, linearnu algebru, integraciju i procesiranje signala i slika.

scikit-learn - Biblioteka s algoritmima strojnog učenja, temeljena na bibliotekama NumPy i SciPy. Ovdje su uz pomoć scikit-learna ostvareni SVM i logistička regresija.

scikit-image - Obuhvaća algoritme za geometrijske transformacije, segmentaciju, analizu i filtriranje slika i sl. Biblioteka je temeljena na bibliotekama NumPy i SciPy. Ovdje je korištena za jednostavan rad sa slikama iz skupova Zebre i Znakovi.

matplotlib - Biblioteka koja se koristi za iscrtavanje različitih grafova.

TensorFlow - Biblioteka javno objavljena 2015. godine od strane Google Brain Teama. Koristi se za strojno učenje, a temelji se na prikazu matematičkih operacija i tenzora podataka kao čvorova grafa odnosno veza između njih. Prilagođena je izvršavanju na raspodijeljenim i GPU arihitekturama. Prednost TensorFlow-a je to što olakšava izgradnju dubokih modela kao što su konvolucijske mreže. Korištenje TensorFlow-a

u programu uvijek obuhvaća odvojene faze oblikovanja računskog grafa i zadavanja računskih upita. Računski graf sadržava čvorove iza kojih stoje parametri (`tf.Variable`), ulazni podaci (`tf.placeholder`) i operacije koje definiraju model, funkciju gubitka i postupak učenja. Računski upiti odabranog gradijentnog postupka izvršavaju se automatski, ali to je moguće tek nakon inicijalizacije izvedbenog konteksta koji je pohranjen u objektu tipa `tf.Session`. Ova biblioteka je napisana uz pomoć visoko optimiranog C++ koda i u širokoj je primjeni u brojnim znanstvenim i komercijalnim projektima.

Yael - Biblioteka francuskog instituta Inria [3] s razvijenim sučeljima za C, Python i Matlab. Obuhvaća implementacije velikog broja računski zahtjevnih funkcija korištenih u algoritmima grupiranja i traženja najbližih susjeda koji rade sa slikama. Učenje GMM-a i rad s Fisherovim vektorima je ostvaren uz pomoć modula ynumpy biblioteke Yael.

SPAMS - Institut Inria osmislio je SPAMS (engl. *SPArse Modeling Software*) [1] kao skup optimizacijskih alata koji sadrži algoritme za rješavanje problema strojnog učenja i obrade signala uz tehnike rijetkih regularizacija. Temelji se na programskom jeziku C++. Ovdje je SPAMS korišten za implementaciju logističke regresije uz $L_{2,1}$ regularizaciju.

MatConvNet - Skup alata koji omogućuju jednostavan i efikasan rad s konvolucijskim neuronskim mrežama u MATLAB-u. MatConvNet [15] sadrži i brojne već naučene konvolucijske mreže za klasifikaciju, segmentaciju, prepoznavanje lica ili teksta. Svaki od tih modela je javno dostupan, a ovdje je korištena mreža VGG19 za ekstrakciju konvolucijskih značajki.

5.2. Implementacija konvolucijske mreže

Konvolucijska mreža arhitekture opisane u odjeljku 2.2 ostvarena je uz pomoć biblioteke TensorFlow. Slojevi mreže uz željenu normalizaciju i regularizaciju slagani su uz pomoć `contrib` paketa. U svim slojevima je korištena L_2 regularizacija. Normalizacija nije korištena. Slijedi isječak koda koji prikazuje inicijalizaciju početnih konvolucijskih slojeva:

```
import tensorflow as tf
import tensorflow.contrib.layers as layers
...
```

```
with tf.contrib.framework.arg_scope([layers.convolution2d],
   kernel_size=3, stride=1, padding='SAME', rate=1,
   activation_fn=tf.nn.relu, normalizer_fn=None,
   weights_initializer=None,
   weights_regularizer=layers.l2_regularizer(weight_decay)):
       net = layers.convolution2d(inputs, 64, scope='conv1_1')
       net = layers.convolution2d(net, 64, scope='conv1_2')
       net = layers.max_pool2d(net, 2, 2, scope='pool1')
```

Ovo je isječak iz modula **vgg.py** koji sadrži funkcije za definiranje arhitekture konvolucijske mreže, funkcije gubitka i način inicijalizacije parametara mreže uz pomoć binarnih datoteka s već naučenim težinama. Argument koji određuje način nadopune kod konvolucije (engl. *padding*) postavljen je tako da se vrši nadopuna nulama na rubovima što omogućava neograničenu dubinu mreže. Nakon posljednjeg sloja sažimanja, a prije potpuno povezanih slojeva, mapa značajki se sažima uz pomoć `reduce_mean` funkcije. Time se mapa konvolucijskih značajki zamjenjuje izračunatim prosjekom svih značajki. Za svaku sliku mini grupe koja se trenutno obrađuje dobiva se jedna konvolucijska značajka veličine 512. Funkcijom `flatten` se sve značajke izravnavaju u jedan vektor.

Iz modula **train.py** se pokreće učenje mreže i provodi evaluacija mreže nakon svake epohe. Ovdje se inicijalizira konvolucijski model i unutar izvedbenog konteksta pokreće optimizacija odabranim algoritmom učenja. U svim eksperimentima je korišten Adam. Zbog veće preglednosti, neki od parametara pohranjeni su u konfiguracijskoj datoteci definiranjem zastavica.

Bilo je potrebno prilagoditi slike iz skupova podataka za prolazak kroz konvolucijsku mrežu. Moduli **ts2010_dataset.py** i **zebre_dataset.py** sadrže sve potrebne funkcije za učitavanje mini grupa i pripadnih oznaka. Zbog memorijskih ograničenja su slike iz skupa Zebre korištene smanjene na 50% originalne veličine - 640×360 piksela. Nije bilo potrebe za smanjivanjem slika iz skupa Znakovi. Bitan implementacijski detalj kod učitavanja slika je zamjena kanala R i B u RGB slikama. Razlog tome je to što je učenje originalne VGG mreže čije se težine koriste za inicijalizaciju parametara modela iz ovog eksperimenta provedeno na BGR slikama.

5.3. Implementacija klasifikacije Fisherovim vektorima

Kao što je navedeno u odjeljku 3.3, za eksperimente s Fisherovim vektorima su korištene konvolucijske značajke dobivene iz slika uz pomoć javne parametrizacije VGG19 mreže iz MatConvNet-a. Riječ je modelu naučenom na slikama s ImageNet-a koji je dostupan u obliku MAT datoteke. Postupak se pokreće iz modula **vgg19_extract.py**. Korištena je SciPy funkcija za učitavanje binarne MATLAB datoteke u Python kod. Uz pomoć već naučenih parametara konvolucijskih slojeva mreže VGG19 inicijaliziran je TensorFlow model. Željene slike su stavljane na ulaz mreže, a konvolucijske značajke su dobivene ekstrakcijom izlaza posljednjeg konvolucijskog bloka mreže. Razlika VGG19 mreže i VGG16 koja je opisana u odjeljku 2.2 je dodatak po jednog konvolucijskog sloja na kraj svakog od posljednja 3 konvolucijska bloka.

Uz pomoć dobivenih konvolucijskih značajki se unutar modula **fv_gmm_learn.py** pokreće proces učenja GMM-a. Za svaku sliku su generirane značajke veličine 512, a njihov broj po slici je *n = visina* $\times$ *širina slike*. Korištene su slike pune rezolucije za oba skupa. Da bi bilo moguće naučiti GMM s $K = 128$ komponenata potreban je veliki broj značajki. Slučajnim odabirom je za svaku sliku iz skupova za učenje i testiranje odabrano 600 značajki kako bi se ukupno dobilo približno 2×10^6 konvolucijskih značajki. Pozivom funkcije `gmm_learn` iz biblioteke Yael pokreće se iterativni postupak estimiranja parametara GMM modela na temelju predanih značajki. Za to se u pozadini koristi algoritam maksimizacije očekivanja. Pretpostavka početnih parametara komponenata Gaussove mješavine dobivena je uz pomoć k-means grupiranja. Za svaku značajku se računa vjerojatnost pripadnosti svakoj od komponenata, a zatim se ažuriraju parametri komponenata kako bi se povećala izglednost značajki s postojećom pretpostavkom grupiranja. Postupak se ponavlja do konvergencije, kada za svaku od K komponenata dobivamo težinski faktor w, vektor srednje vrijednosti μ i kovarijacijsku matricu Σ. Pozivanjem Yael-ove funkcije `fisher` računaju se gradijenti izglednosti konvolucijskih značajki u odnosu na μ i Σ. Fisherov vektor za svaku sliku dobiva se sažimanjem vektora gradijenata izračunatih za sve značajke koje pripadaju toj slici. Dimenzija Fisherovog vektora je 131 072, tj. $2D \times K$.

Nakon normalizacije Fisherovih vektora potenciranjem i L_2 normom provodi se klasifikacija vektora implementacijama logističke regresije i SVM-a iz bibilioteke scikit-learn. Učenje klasifikatora provedeno je za različite tipove regularizacije. Dok scikit-learn podržava L_1 i L_2 regularizaciju, za učenje klasifikatora uz regularizaciju $L_{2,1}$ bilo je potrebno iskoristiti modul `fistaFlat` biblioteke SPAMS. $L_{2,1}$ regularizacija je kombinacija L_2 regularizacije unutar svake komponente i L_1 regularizacije nad svim

komponentama. Cijeli postupak klasifikacije Fisherovih vektora pokreće se iz modula **fv_classification.py** i **spams_fv_classification.py**.

6. Rezultati

Eksperimenti su izvođeni na računalu s Intel Xeon E5-2620 2.00 GHz jezgrom i grafičkom karticom NVIDIA GTX 980 i računalu s Intel Xeon E3-1200 4.00 GHz jezgrom i grafičkim karticama NVIDIA GTX 1070 i NVIDIA GTX 970. Kvaliteta klasifikatora je mjerena pomoću točnosti (engl. *accuracy*) i prosječne preciznosti (engl. *Average Precision*, AP). Točnost je omjer broja točno klasificiranih primjera i ukupnog broja primjera. AP je jednak površini ispod krivulje preciznosti i odziva (engl. *precision-recall curve*). AP binarne klasifikacije se računa na sljedeći način:

$$AP = \frac{\sum_i preciznost_i \mathbf{Y}_{ri}}{\sum_i \mathbf{Y}_{ri}} \tag{6.1}$$

Sume iteriraju po svim ulaznim primjerima. $\mathbf{Y}_{ri}$ je lista točnih razreda za sve primjere sortirana prema vjerojatnostima pripadanja podataka $\mathbf{x}_i$ razredu koji smo odredili kao pozitivan. Te su vjerojatnosti dobivene na izlazu klasifikatora. $preciznost_i$ je preciznost klasifikacije u slučaju kada su samo podaci s indeksom jednakim ili većim od *i* pridruženi pozitivnom razredu.

6.1. Klasifikacija dubokim modelom

Učenje konvolucijske mreže je provedeno u 15 epoha. Zbog različitih veličina slika u tim skupovima podataka, veličina mini grupe za skup Zebre je 3, a za skup Znakovi je 4 slike. Početna stopa učenja ϵ je 10^{-4} za Zebre i 5×10^{-5} za Znakove. Svakih 5 epoha se stopa učenja smanjuje za 50%. Eksponencijalno opadanje stope učenja je ostvareno uz korištenje TensorFlow funkcije `tf.train.exponential_decay`.

Rezultati klasifikacije za ovakav duboki model navedeni su u tablicama koje slijede. Slike iz skupa Zebre koje se krivo klasificiraju često sadrže cestovne prizore s uzorcima koji su vrlo slični zebrama. Slika 4.1 prikazuje tipičan primjer takve situacije. Osim toga, krivo se klasificiraju slučajevi u kojima je zebra skoro u potpunosti izbrisana ili

	Točnost	AP
Skup za učenje	100%	100%
Skup za testiranje	96.48%	99%

Tablica 6.1: Rezultati klasifikacije za skup slika Zebre.

	Točnost	AP
Skup za učenje	100%	100%
Skup za testiranje	98.99%	100%

Tablica 6.2: Rezultati klasifikacije za skup slika Znakovi.

predaleko od promatrača. Navedene situacije možemo okarakterizirati kao teške za klasifikator i one objašnjavaju zašto se maleni postotak slika ne klasificira točno.

Kada je riječ o slikama iz skupa Znakovi, one se generalno mogu ocijeniti kao teži problem za klasifikator zbog veličine objekta interesa. Na mnogim je slikama i ljudskom oku teško na prvi pogled uočiti trokutasti prometni znak. Upravo takvih primjera je najviše među krivo klasificiranim slikama.

Alternativni pristup opisanoj klasifikaciji je zamjena potpuno povezanih slojeva s plitkim klasifikatorom. Nakon 14 epoha učenja se ekstrahiraju konvolucijske značajke iz posljednjeg konvolucijskog sloja. Za klasifikaciju tih značajki upotrebljena je scikit-learn implementacija logističke regresije. Rezultati klasifikacije na skupovima za testiranje su u tablici koja slijedi.

	Točnost	AP
Zebre	96.21%	99.41%
Znakovi	95.16%	95.71%

Tablica 6.3: Rezultati klasifikacije s naučenim konvolucijskim značajkama i plitkim klasifikatorom.

Za skup Zebre je kvaliteta rezultata približno jednaka, ali pogoršanje rezultata za skup Znakovi daje prednost korištenju potpuno povezanih slojeva.

6.2. Klasifikacija Fisherovim vektorima

Detalji postupka klasifikacije slika Fisherovim vektorima opisani su u odjeljku 5.3, a rezultati klasifikacije su u tablicama 6.4 i 6.5. Tablice sadrže i prosječnu gustoću

modela po komponentama GMM-a (engl. *Average Component Density*) za svaki istrenirani klasifikator. Unatoč očekivanju sličnih rezultata klasifikacije korištenjem logističke regresije i SVM-a, eksperimenti su provedeni uz pomoć oba klasifikatora kako bi se dobili najbolji mogući rezultati. Također se može vidjeti utjecaj izbora tipa regularizatora i normalizacije Fisherovih vektora prije učenja klasifikatora.

		Logistička regresija			**SVM**		
Normal.	**Regul.**	**ACD**	**Točnost**	**AP**	**ACD**	**Točnost**	**AP**
–	L_1	50%	89.19%	95.79%	50%	88.82%	94.78%
–	L_2	100%	90,39%	97.10%	100%	91.04%	97.02%
–	$L_{2,1}$	100%	90.57%	97.11%	–	–	–
+	L_1	64.06%	**93.07%**	**98.19%**	42.19%	91.04%	97.16%
+	L_2	100%	92.88%	98.16%	100%	92.51%	98.14%
+	$L_{2,1}$	95.31%	92.70%	98.09%	–	–	–

Tablica 6.4: Rezultati klasifikacije Fisherovim vektorima za skup Zebre.

		Logistička regresija			**SVM**		
Normal.	**Regul.**	**ACD**	**Točnost**	**AP**	**ACD**	**Točnost**	**AP**
–	L_1	50%	80.58%	76.91%	50%	79.45%	73.66%
–	L_2	100%	77.12%	72.36%	100%	76.93%	71.72%
–	$L_{2,1}$	100%	77.18%	72.58%	–	–	–
+	L_1	32.81%	84.98%	86.54%	18.75%	84.04%	86.73%
+	L_2	100%	79.95%	80.97%	100%	80.46%	82.01%
+	$L_{2,1}$	17.97%	**85.98%**	**87.65%**	–	–	–

Tablica 6.5: Rezultati klasifikacije Fisherovim vektorima za skup Znakovi.

Prisutne su određene razlike među rezultatima dobivenima različitim klasifikatorima. Uz korištenje zadanog parametra regularizacije $C = 1$, SVM je generalno pokazao veću uspješnost od logističke regresije kod klasifikacije s normaliziranim Fisherovim vektorima. Pretragom prostora parametra C pronađene su vrijednosti za koje logistička regresija dostiže ili čak premašuje rezultate postignute SVM-om. U eksperimentima klasifikacije logističkom regresijom za normalizirane Fisherove vektore iz skupa Zebre najboljima su se pokazali parametri $C = 3000$ uz L_1 regularizaciju i $C = 300$ uz L_2 regularizaciju. Za isti skup, najbolji rezultati klasifikacije SVM-om uz L_1 regularizaciju su postignuti za $C = 100$. Parametar regularizacije u klasifikaciji normaliziranih Fi-

sherovih vektora iz skupa Znakovi logističkom regresijom je $C = 10$. U svim ostalim eksperimentima s L_1 i L_2 regularizacijom je korišten $C = 1$. Validiranjem parametra $lambda$ logističke regresije uz $L_{2,1}$ regularizaciju u nekoliko vrijednosti između 1×10^{-4} i 1×10^{-3} odabrana je $lambda = 1 \times 10^{-4}$. Iznimka je slučaj klasifikacije normaliziranih Fisherovih vektora iz skupa Znakovi za koji je $lambda = 5 \times 10^{-4}$. Izbor regularizacije je također utjecao na rezultate. L_1 i $L_{2,1}$ regularizacija su se pokazale dobrim izborom za problem kakav imamo u skupu Znakovi. Korištenje te dvije vrste regularizacije vodi na rijetki model - samo dio težina koje se uče ostaje različit od 0. Time najveći utjecaj na predviđanje klasifikacijskog modela imaju mali, posebno istaknuti dijelovi slike. Najbolji rezultati na skupu Znakovi su postignuti korištenjem $L_{2,1}$ regularizacije - postignut je bolji ACD uz sličnu točnost i AP kao i uz L_1 regularizaciju. Izbor regularizacije nije imao velik utjecaj na rezultate klasifikacije na skupu Zebre, a nešto bolji rezultat od ostalih postignut je logističkom regresijom uz L_1 regularizaciju.

Već samim pogledom na prosječnu gustoću modela koja se može pronaći u tablicama 6.4 i 6.5 možemo vidjeti utjecaj regularizacije na vrijednosti koeficijenata klasifikacijskog modela. Kako bi se vizualno prikazao utjecaj gustoće modela na klasifikaciju, prikazuje se odziv klasifikatora na Fisherove vektore slike. Za nekoliko primjera pozitiva iz skupova Zebre i Znakovi su izračunati Fisherovi vektori za svaku konvolucijsku značajku slike. Odzivi naučenih koeficijenata logističke regresije za te Fisherove vektore su naduzorkovani kako bi se njihov prikaz mogao usporediti s originalnom slikom. Na slikama 6.1 i 6.2 se mogu vidjeti originalne slike i odzivi logističke regresije uz korištenje L_1 i L_2 regularizacije.

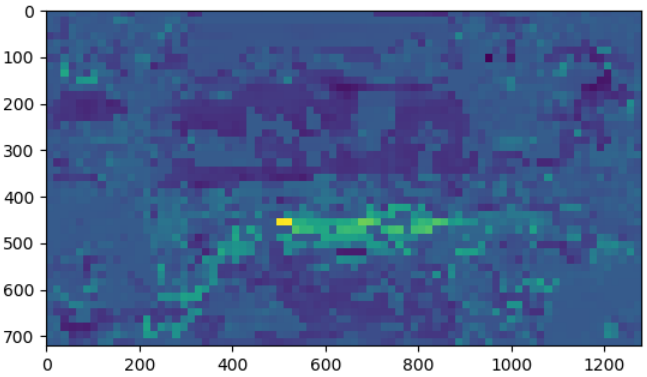
Proveden je još jedan eksperiment kako bi se naglasio povoljan utjecaj korištenja Fisherovih vektora značajki u klasifikaciji slika. Plitki klasifikatori su naučeni na samim konvolucijskim značajkama slika dobivenima uz pomoć javne parametrizacije VGG19 mreže. Prikazani su samo najbolji rezultati - za skup Zebre su dobiveni SVM-om uz L_2 regularizaciju, a za skup Znakovi logističkom regresijom uz L_1 regularizaciju.

	Točnost	**AP**
Zebre	89.00%	95.84%
Znakovi	84.54%	84.00%

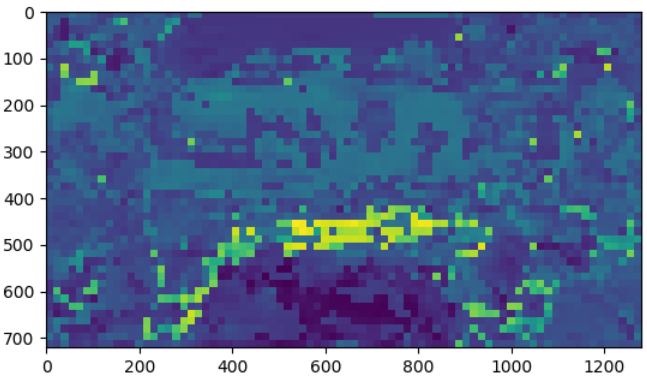
Tablica 6.6: Rezultati klasifikacije s konvolucijskim značajkama i plitkim klasifikatorom.

Za oba skupa slika je korištenjem Fisherovih vektora postignuto poboljšanje rezultata klasifikacije.

(a) Originalna slika



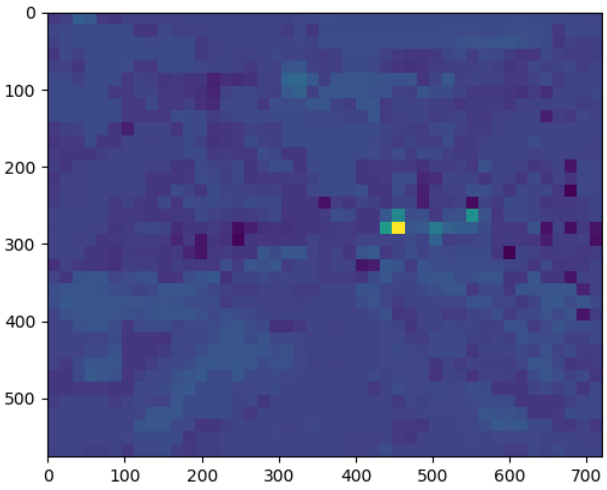
(b) Odziv logističke regresije uz L_1 regularizaciju



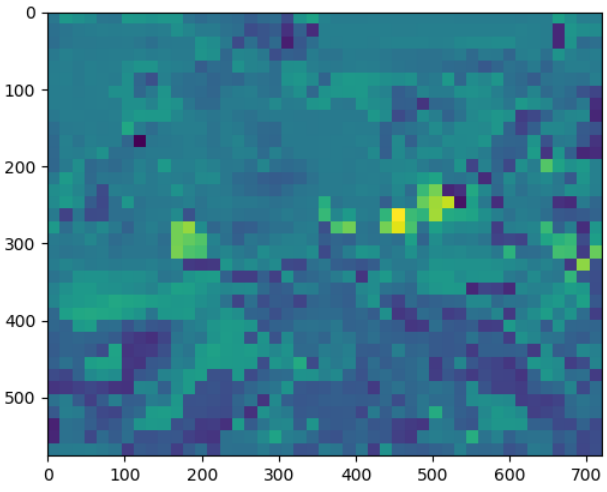
(c) Odziv logističke regresije uz L_2 regularizaciju

Slika 6.1: Prikaz utjecaja gustoće modela na klasifikaciju na primjeru iz skupa Zebre.

(a) Originalna slika



(b) Odziv logističke regresije uz L_1 regularizaciju



(c) Odziv logističke regresije uz L_2 regularizaciju

Slika 6.2: Prikaz utjecaja gustoće modela na klasifikaciju na primjeru iz skupa Znakovi.

7. Zaključak

U ovom radu su predstavljena dva različita pristupa klasifikaciji slika. Prvi je klasifikacija dubokim modelom, metoda u kojoj se klasifikacija provodi s kraja na kraj (engl. *end-to-end*). Učenjem konvolucijske neuronske mreže se izdvajaju značajke slika stavljenih na ulaz, a u konačnici se nad njima provodi klasifikacija. Drugi pristup radi s konvolucijskim značajkama dobivenima uz pomoć javne parametrizacije VGG19 mreže. Klasifikacija tih značajki Fisherovim vektorima uključuje učenje parametara generativnog modela značajki, Gaussove mješavine (GMM), i generiranje Fisherovih vektora. Za klasifikaciju je potrebno naučiti plitki linearni klasifikator. U sklopu ovog rada su provedeni eksperimenti i uspoređeni rezultati klasifikacije dubokim modelom i Fisherovim vektorima. Svi su eksperimenti provedeni na dva skupa slika - u skupu Zebre objekti interesa su zebre, a u skupu Znakovi trokutasti prometni znakovi. Značajno veću uspješnost je pokazao *end-to-end* pristup. Arhitektura naučene mreže temeljena je na VGG16 arhitekturi, s razlikom smanjenja broja potpuno povezanih slojeva i dodatnog sažimanja mapa značajki prije prvog potpuno povezanog sloja. Ishod tih promjena je pozitivan jer su dobivene vrlo visoka točnost i prosječna preciznost klasifikacije konvolucijskom mrežom. Korištenje Fisherovih vektora značajki donosi poboljšanje u odnosu na klasifikaciju samih značajki slika. Primjećen je pozitivan utjecaj normalizacije Fisherovih vektora na klasifikacijske rezultate, kao i primjene L_1 regularizacije na probleme kao što je pronalazak malenih prometnih znakova u slici. Klasifikacija konvolucijskih značajki nad kojima je proveden fine-tuning učenjem konvolucijske mreže nije u potpunosti usporediva s klasifikacijom sirovih značajki i Fisherovih vektora koji nisu naučeni na slikama koje se klasificiraju. Takvu specijaliziranost konvolucijske mreže možemo smatrati nedostatkom tog klasifikacijskog modela. Budući pravac razvoja bi mogao uključivati proširenje postojećeg rješenja na razvoj klasifikatora koji imaju moć prepoznavanja i drugih elemenata cestovne scene osim zebri i trokutastih znakova. To mogu biti druge vrste prometnih znakova ili različite vrste prometnih traka. Zbog pokazane visoke kvalitete rezultata, kao metoda klasifikacije bila bi korištena konvolucijska neuronska mreža.

LITERATURA

[1] SPArse Modeling Software, verzija 2.6, 2017. http://spams-devel.gforge.inria.fr/.

[2] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] Matthijs Douze i Hervé Jégou. The Yael Library. *Proceedings of the 22Nd ACM International Conference on Multimedia*, stranice 687–690, 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] Thomas Hofmann, Bernhard Schölkopf, i Alexander J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

[6] Gao Huang, Zhuang Liu, i Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[7] Sergey Ioffe i Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. U *ICML*, svezak 37 od *JMLR Workshop and Conference Proceedings*, stranice 448–456. JMLR.org, 2015.

[8] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, i Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. U *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012.

[10] David G. Lowe. Object Recognition from Scale-Invariant Features. *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, stranice 1150–1157, 1999.

[11] Dino Pačandi. Klasificiranje slika Fisherovim vektorima izgrađenim nad slučajnim distribucijskim šumama značajki slike. Diplomski rad, Fakultet Elektrotehnike i Računarstva, Zagreb. 2015.

[12] Mikhail Prokopenko, Joseph T. Lizier, Oliver Obst, i X. Rosalind Wang. Relating Fisher information to order parameters. *Physical Review E*, 84, 2011.

[13] Jorge Sanchez, Florent Perronnin, Thomas Mensink, i Jakob Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

[14] Karen Simonyan i Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2014.

[15] A. Vedaldi i K. Lenc. Matconvnet: Convolutional neural networks for matlab. U *ACM International Conference on Multimedia*, 2015.

[16] Ivana Viduka. Duboke neuronske mreže za semantičku segmentaciju slika. Diplomski seminar, Fakultet Elektrotehnike i Računarstva, Zagreb. 2016.

[17] Vedran Vukotić. Raspoznavanje objekata dubokim neuronskim mrežama. Diplomski rad, Fakultet Elektrotehnike i Računarstva, Zagreb. 2014.

[18] Valentina Zadrija, Josip Krapac, Jakob Verbeek, i Siniša Šegvić. Patch-level Spatial Layout for Classification and Weakly Supervised Localization. GCPR, German Conference on Pattern Recognition, stranice 492–503, 2015.

[19] Valentina Zadrija, Josip Krapac, Jakob Verbeek, i Siniša Šegvić. Towards Automatic Road Environment Mapping with Sparse Weakly Supervised Localization Models. 2016.

Klasifikacija slika dubokim modelima i Fisherovim vektorima

Sažetak

Za ovaj rad su opisane i implementirane dvije metode klasifikacije slika. Prvu metodu čini duboki model temeljen na arhitekturi VGG16 konvolucijske mreže čiji su parametri inicijalizirani težinama naučenima na slikama ImageNet baze. Drugi pristup uključuje konvolucijske značajke dobivene uz pomoć javne parametrizacije VGG19 mreže iz kojih se generiraju Fisherovi vektori, a zatim se provodi klasifikacija pllitkim klasifikatorom. Usporedbom rezultata klasifikacije dobivenih učenjem i testiranjem na dva skupa slika koji prikazuju scene s hrvatskih prometnica - Zebre i Znakovi, zaključuje se da *end-to-end* pristup učenju klasifikatora kakav je prisutan kod konvolucijske mreže donosi puno bolje rezultate.

Ključne riječi: klasifikacija slika, konvolucijske neuronske mreže, jezgrene funkcije, Fisherovi vektori, linearni klasifikatori

Image classification with deep models and Fisher vectors

Abstract

Two different image classification methods are described and implemented for this work. The first method is a deep model based on the architecture of VGG16 convolutional network, having parameters initialized with weights learned on images from ImageNet database. The second approach includes Fisher vectors generated from convolutional features that are extracted with the help of the publicly available pretrained VGG19 network. Shallow classifier is then used for the classification of Fisher vectors. Comparing the classification results obtained for images from two datasets containing traffic scenes of Croatian roads, we draw a conclusion that the *end-to-end* approach to the image classification, like the one used in our deep model, shows significantly better results.

Keywords: image classification, convolutional neural network, kernel functions, Fisher vectors, linear classifiers