Mladi za napredek Maribora 2017

34. srečanje

KLASIFIKACIJA VIDEO OBJEKTOV S POSTOPKOM GLOBOKEGA UČENJA

Raziskovalna naloga: Računalništvo

Avtor: NEJC FIRBAS, PATRICK KAČIČ

Mentor: ZDRAVKO KAČIČ

Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR

Maribor, 9.1.2017

KAZALO

KAZALO SLIK

ZAHVALA

Najprej bi se rada zahvalila najinemu mentorju, ki nama je z njegovim znanjem in izkušnjami pomagal skozi celoten proces in brez njegove pomoči ne bi naredila tako uspešne naloge. Zahvaliti se morava tudi dvema zaposlenima na FERI-ju, ki sta nama pomagala in svetovala pri izvedbi eksperimenta.

POVZETEK

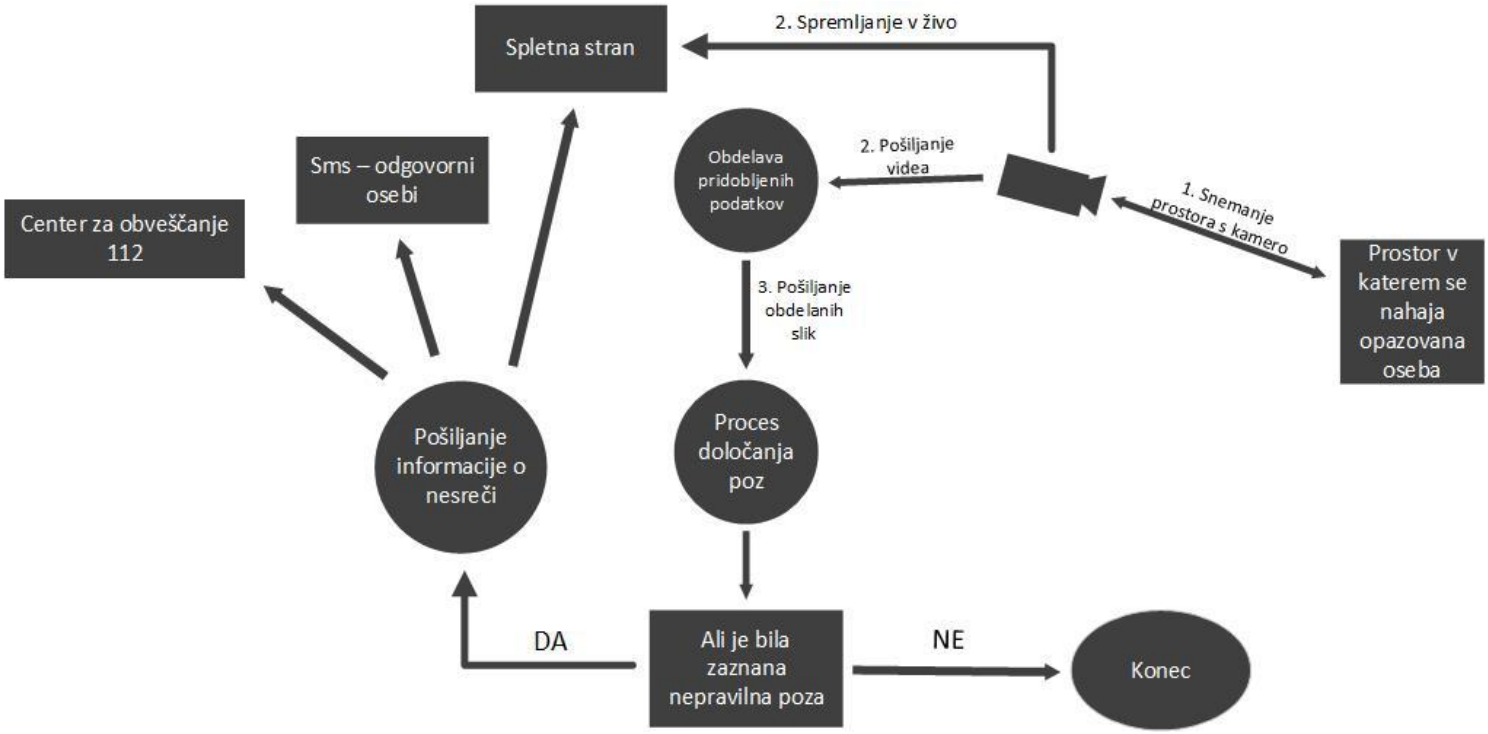
Ideja raziskovalne naloge je analiza video vsebin z uporabo globokega učenja. Globoko učenje je področje umetne inteligence, ki obravnava globoke večslojne nevronske mreže, ki vsebujejo veliko skritih nivojev nevronov med vhodnimi in izhodnimi nivoji. V našem primeru bomo nevronske mreže uporabili za prepoznavanje oziroma klasifikacijo objektov v izbranem video gradivu. Primer praktične rabe takšnega procesiranja je v podpornih življenjskih okoljih, kjer lahko s takšnim sistemom spremljamo aktivnosti starejših oseb in prepoznavamo/klasificiramo njihov trenutni položaj. Z učenjem nevronskih mrež lahko sistem prepozna, ali je oseba v normalnem položaju ali pa je v ogroženem (npr. padec osebe). Nevronske mreže učimo tako, da jim posredujemo veliko število podatkov, v tem primeru slik položajev oseb. Tako bi lahko s takšnim sistemom spremljali aktivnosti starejših oseb in v primeru padca ali kakšnih drugih zanje nevarnih razmer, samodejno priklicali ustrezno pomoč.

1. UVOD

V zadnjem času se vse pogosteje pojavljajo tehnološko podprta okolja, ki jih poznamo pod imeni pametna stanovanja, pametne hiše, podporna življenjska okolja ipd. V takšnih okoljih želimo z vgrajenimi tehnološkimi rešitvami pomagati ljudem pri njihovih vsakodnevnih opravilih. V raziskovalni nalogi smo se osredotočili na problem ugotavljanja nevarnih dogodkov, kot je na primer nenaden padec starejših oseb, ki živijo same. Takšnih oseb je zmeraj več, saj si starejše osebe želijo čim daljšega aktivnega samostojnega življenja. V primeru nepredvidenih nezgod, kot je na primer padec, pri katerem se oseba hudo poškoduje, na primer zlomi kolk, ne more poklicati na pomoč, kar lahko tudi pomeni, da je v tem primeru lahko ogroženo tudi njeno življenje. V nalogi se zato osredotočamo na problem spremljanja aktivnosti starejših ljudi, ki živijo sami in v zvezi s tem zaznavo nevarnih dogodkov. Tehnološka rešitev zahteva namestitev video kamer po prostorih stanovanja. Video kamere pošiljajo video tok centralnemu strežniku, ki sprejete video tokove posameznih kamer procesira in analizira prisotnost osebe v posameznem prostoru oziroma v katerem položaju se oseba trenutno nahaja. Če bi bila oseba v nenaravnem položaju, kar pomeni, da bi oseba na primer padla in ležala na tleh, bi sistem to zaznal in sprožil telefonski klic na pomoč na ustrezno telefonsko številko oziroma seznam telefonskih številk po prioritetnem vrstnem redu, dokler ne bi vzpostavil uspešne telefonske zveze in posredoval sporočilo o izrednem dogodku.

2. OPIS SISTEMA

Sistem, ki s pomočjo kamere spremlja aktivnosti starejših oseb je zgrajen iz 4 glavnih modulov (slika 1): **pridobivanje videa, obdelava pridobljenih podatkov**, **proces določanja poz** in **pošiljanje informacij o nesreči**. Sistem mora biti čim manj opazen v življenjskem okolju osebe, tako da ne moti normalnega življenja osebe in njenih vsakodnevnih aktivnosti. Pri tem imamo v mislih predvsem kamere, ki bi bile nameščene v sobah, saj je drugi del sistema – sistem procesiranja (obdelava pridobljenega gradiva in klasifikator) - običajno nameščen na oddaljeni lokaciji, ki zagotavlja ustrezno zmogljiv računalniški sistem (računalniški oblak). V stanovanju mora biti nameščen zato le še sistem, ki ustrezno združi video tokove iz posameznih video kamer in jih posreduje sistemu procesiranja.



Slika 1: Sistem spremljanja aktivnosti starejših oseb in alarmiranja ob zaznanih nevarnih dogodkih

2.1 Pridobivanje videa

Sistem procesiranja preko kamer, ki so nameščene v prostorih in preko vmesnika (ki služi za pošiljanje videa preko omrežja) pridobi video podatke. Ko sistem prejme video signal, ga obdela tako, da video »razbije« na posamezne digitalne slike in jih shrani v pomnilnik.

2.2 Proces določanja poz

Proces določanja poz je glavni proces, ki uporablja različne algoritme za prepoznavanje slik (podrobnosti v 5. poglavju). Ta proces namreč določa oz. preverja, ali slika vsebuje kritičen dogodek ali ne. Če proces prepozna, da je oseba na sliki v normalnem položaju jo prezre in izbriše, če pa je oseba na sliki v položaju, ki je prepoznan kot kritičen dogodek, sproži proces za **pošiljanje informacij o nesreči**.

2.3 Proces pošiljanja informacij o nesreči

V primeru zaznave kritičnega dogodka sistem predvideva, da je oseba v nevarnosti oz. se ji je nekaj zgodilo. Proces najprej pošlje sms sporočilo pristojni osebi in zraven priloži še sliko, na kateri je bil zaznan kritični dogodek. Tako lahko naslovljena pristojna oseba potrdi pravilnost zaznave kritičnega dogodka, saj bi se lahko tudi zgodilo, da opazovana oseba čisti tla ali pa je v takšnem nerodnem položaju, da sistem misli, da se ji je kaj zgodilo. Tako bi lahko zmanjšali možnosti posredovanja ustreznih reševalnih služb ob napačnih alarmih. Če pa pristojna oseba iz posredovane informacije ugotovi, da je opazovana oseba res padla oz. je ogroženo njeno zdravje, lahko sam pokliče center za obveščanje. Pristojna oseba mora ob prejemu sms sporočila nanj odgovoriti s preprostim povratnim sms sporočilom (tudi posamezna črka, številka ali znak), da sistem ve, da je pristojna oseba posredovano sms sporočilo uspešno sprejela in prebrala. Če sistem za posredovano sms sporočilo po določenem času ne prejme potrditvenega sms sporočila od pristojne osebe, sam pošlje ustrezne informacije centru za obveščanje.

Ko sistem pošlje sms sporočilo, istočasno objavi informacijo na ustrezni spletni strani, kjer lahko pristojna oseba preveri, kako se je zgodil kritičen dogodek, saj spletna stran omogoča prikaz žive slike.

3. GLOBOKO UČENJE – DEEP LEARNING

Pred približno 60 leti je luč sveta uzrla znanstvena panoga, ki danes sodi med dve najbolje financirani panogi na svetu. Govorimo o razmahu umetne inteligence. Če so v dvajsetem stoletju moč mišic nadomestili stroji, se nam obeta v 21. stoletju nekaj podobnega, ko nas bo umetna inteligenca vse bolj prevzemala tudi umsko zahtevnejša dela. Sprva nam bo

pomagala pri bolj enostavnih in monotonih opravilih, kasneje pa bo verjetno pričela dohitevati tudi umske sposobnosti ljudi. Ko nas bo dohitevala, bo lahko med drugim začela tudi izpopolnjevati samo sebe in nas bo zato kmalu lahko verjetno tudi presegla. To bo eden od največjih prelomov v razvoju človeštva, ko poti nazaj ne bo več. Futurologi so to prelomno točko zgodovinskega razvoja poimenovali singularnost.

Današnja umetna inteligenca pozna različne metode, ki imajo sposobnost, da iz kopice učnih primerov izluščijo bistvo oziroma splošno veljavna pravila. Veljavnost teh pravil lahko nato preverimo s testnimi primeri, ki se nekoliko razlikujejo od učnih. Sposobnost iskanja pravil oziroma znanja z eno besedo imenujemo učenje. In če se naprava uči samostojno, brez naše pomoči, lahko kmalu njeno znanje prekaša tudi naše, ljudsko znanje. Znanih je že kar nekaj takih primerov (svetovni prvak v šahu in igri Go, avto brez voznika, zmagovalec v znamenitem TV kvizu,...).

Ene najbolj znanih metod, ki se lahko učijo, so nastale po vzoru možganov in se imenujejo nevronske mreže. Njihov razvoj poteka že več kot pol stoletja in je v tem času doživljal številne vzpone in padce. Zadnjo stopnjo v njihovi evoluciji imenujemo globoke nevronske mreže (deep neural networks, DNN). Pridevnik "globoke" nam sporoča, da so te nevronske mreže nekoliko bolj obsežne od svojih prednic, kar jim omogoča boljše učenje in posledično tudi globlje znanje. V zadnjem obdobju znanstveniki kar tekmujejo v objavljanju svojih dosežkov, kjer z učenjem globokih nevronskih mrež dosegajo in presegajo sposobnosti ljudskih možganov. Začelo se je s prepoznavanjem fotografij, sledilo je razumevanje govora in igranje računalniških iger.

3.1 Umetne nevronske mreže

Umetna nevronska mreža, ki jo velikokrat v praksi imenujemo kar nevronska mreža ali v angleščini artificial neural network, je računski oziroma matematični model, ki temelji na delovanju bioloških nevtronskih mrež. Zgrajena je iz preprostih procesnih elementov, ki so med seboj povezani, sestavljena je tudi iz umetnih nevronov (angl. artificial neuron), ki imajo utežene povezave in te povezave določajo, kako se nevronska mreža obnaša. Osnovna ideja delovanja te mreže je podobna, kot bi jo naj uporabljali človeški možgani, saj ima mreža sposobnost organiziranja posameznih skupin nevtronov, tako da ti opravljajo samo določene naloge, kot so na primer zaznavanje, nadzor gibanja, razpoznavanje vzorcev ipd. Računalnik na primer za rešitev problema, ki ga naši možgani rešijo v desetinki sekunde, pogosto še zmeraj potrebuje veliko več časa , npr. več dni. Značilnosti nevronskega procesiranja so v tem, da upoštevajo nakopičeno znanje, ki ga pridobijo v fazi učenja. Deluje pa tako, da odgovarja na vhodne dogodke na način, ki je najbližji glede na izkušnje, ki je pridobilo v fazi učenja. Kar pa pomeni, da je za procesiranje mrež značilna tolerantnost do napak in posploševanje na vhodu, kar je vsekakor lastnost, zaradi katerih nevronske mreže uvrščamo med inteligentne procesorske sisteme. Začetki nevtronskih mrež nas popeljejo do leta 1943. Matematični model za nevrone oziroma živčne celice sta postavila McCulloch in Pitt. Model

iz leta 1943 še danes uporabljamo. Deluje tako, da predstavlja element, v katerega vodijo utežni vhodi, izhodi elementa pa so odvisni od vrednosti vsote produktov na vhodnih spremenljivkah z ustreznimi utežmi. Če je vrednost vsote nad pragom, potem ima izhod vrednost a, v nasprotnem primeru pa vrednost b oziroma drugo vrednost.

3.2 Umetni nevron

Umetni nevron je model, ki je zasnovan kot približek biološkemu nevronu. Dendrite pri biološkem nevronu nadomestijo vhodi (***Xi***), ki imajo vsak svojo utež (***Wi***). Če obravnavamo več nevronov, potem se uteži povezav nevrona ***k*** zapišejo kot ***Wki***. Izhod nevrona **k**, ki predstavlja vsoto vhodov, pomnoženih z ustreznimi utežmi, imenujemo tudi aktivacija (***Uk***). Aktivacija nevrona je vhod v aktivacijsko funkcijo (activation function), včasih imenovano tudi funkcija stiskanja, ki omeji amplitudo izhoda nevrona. Vrednosti izhoda tipično pripadajo intervalu [0, 1] ali intervalu [-1, 1]. Izhod nevrona ***k*** označimo z ***Yk***. Model nevrona ***k*** vsebuje tudi zunanji parameter, **prag** (0*k*), ki predstavlja nivo signala, pri katerem se sproži nevron. Nevron opišemo z naslednjim parom enačb.

$$u_k = \sum_{j=1}^{S} w_{kj} x_j$$

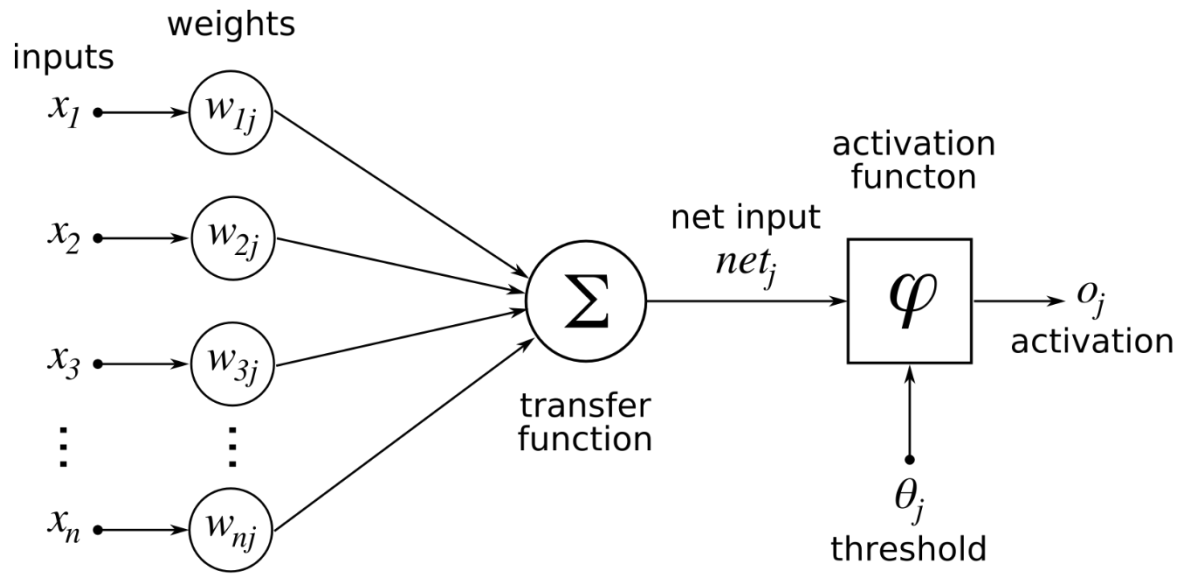
$$y_k = \varphi(u_k - 0_k)\ ,$$

pri tem so ***X1***, ***X2***,..., ***Xs*** vhodni signali, ***Wk1***, ***Wk2***, ..., ***Wks*** sinaptične uteži nevrona, ***Uk*** je aktivacija, ***0k*** je prag, φ je aktivacijska funkcija in ***Yk*** je izhodni signal nevrona. Če definiramo vrednosti ***Wk0*** = ***0k*** in ***X0*** = -1 ter zapišemo ***Vk*** = ***Uk*** - ***0k***, lahko preoblikujemo zgornji enačbi v naslednjo obliko:

$$v_k = \sum_{j=0}^{S} w_{kj} x_j$$

$$y_k = \varphi(v_k)$$

Model umetnega nevrona prikazuje slika 1.1.



Slika 1: Model umetnega nevrona. Povzeto po [7].

3.2.1 Tipi aktivacijskih funkcij

Obstajajo naslednji tipi aktivacijskih funkcij:

1. **Pragovna (stopničasta) funkcija** *(threshold function):*

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

Pri tem je izhod nevrona določen z (izpustimo indeks k):

$$y = \varphi(v) = \varphi(u - 0) \begin{cases} 1, & u \geq 0 \; oziroma \; v \geq 0 \\ 0, & u < 0 \; oziroma \; v < 0 \end{cases}$$

Funkcijo uporabljamo za klasifikacijo linearno ločljivih vzorcev.

2. **Odsekoma linearna funkcija:**

$$\varphi(v) = \begin{cases} 1, v \geq 1/2 \\ v + 1/2\,, -\frac{1}{2} < v < 1/2 \\ 0, v \leq -1/2 \end{cases}$$

3. **Sigmoidna funkcija** je najpogosteje uporabljana oblika aktivacijske funkcije pri nevronskih mrežah. Omenimo tri tipe sigmoidne funkcije:

   a) **Logistična funkcija** (logistic function):

$$\varphi(v) = \sigma(v) = \frac{1}{1 + e^{-av}}$$

   pri tem je *a* parameter naklona (slope parameter). S spreminjanjem tega parametra lahko določimo sigmoidne funkcije z različnimi nakloni.

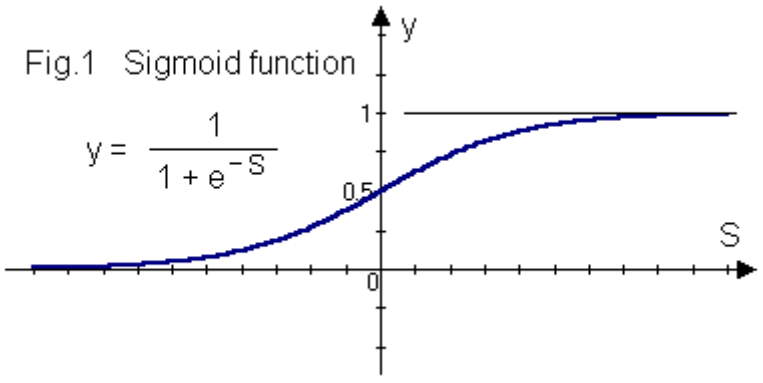
   b) **Funkcija signum:**

$$\varphi(v) = sgn(v) = \begin{cases} 1, v > 0 \\ 0, v = 0 \\ -1, v < 0 \end{cases}$$

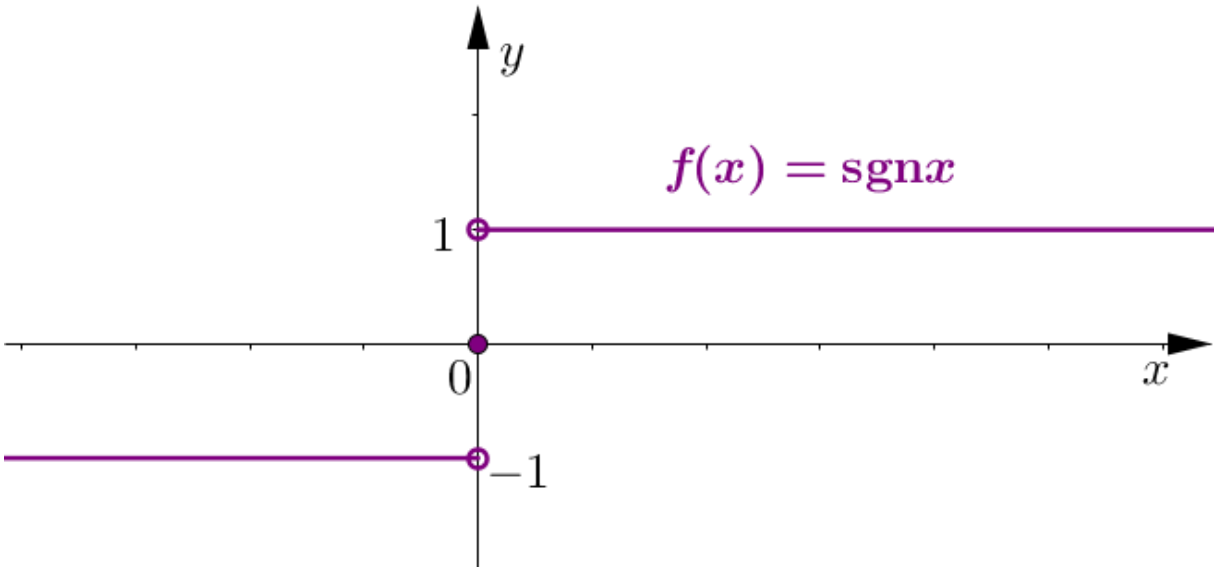
   c) **Funkcija hiperbolični tangens:**

$$\varphi(v) = tanh\frac{v}{2} = \frac{1 - e^{-v}}{1 + e^{-v}}$$

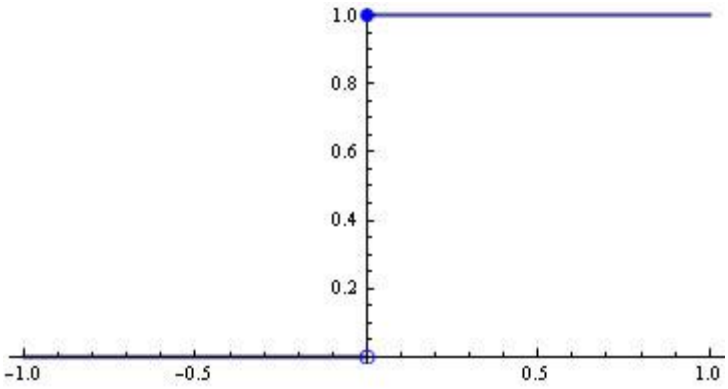
Oblike funkcij so prikazane na naslednjih slikah.



Slika 2:Logistična sigmoidna funkcija. Povzeto po [8].



Slika 3: Signum funkcija. Povzeto po [9].



Slika 4: threshold funkcija[11]

3.3 Proces učenja umetne nevronske mreže

Najpomembnejša lastnost nevronske mreže je zmožnost učenja in izboljševanja svoje učinkovitosti skozi proces učenja. Ta proces določi vrednosti parametrov nevronske mreže tako, da na določeno množico vhodov reagira z določeno množico izhodov. Učenje poteka kot iterativni proces prilagajanja, ta se izvaja na utežeh in pragovih, glede na določeno učno pravilo. Pri nevronskih mrežah učenje definiramo na naslednji način:

»Učenje je proces, pri katerem se prosti parametri nevronske mreže prilagodijo skozi nenehen proces vzpodbude iz okolja, v katerega je mreža vložena. Tip učenja določa način, po katerem se spreminjajo parametri.«

Ta definicija vključuje naslednje zaporedje dogodkov:

1. Nevronsko mrežo vzbuja okolica.
2. Nevronska mreža se spremeni zaradi vzbujanja.
3. Nevronska mreža na nek način odgovori na vzbujanja izokolja zaradi sprememb, ki se zgodijo v njeni notranji zgradbi.

Naj bo **Wkj(n)** vrednost sinaptične uteži **Wkj** v času n. Prilagodimo sinaptično utež **Wkj(n)** z **ΔWkj(n)** tako, da dobimo osveženo vrednost **Wkj(n+1)**:

$$W_{kj}(n+1) = W_{kj}(n) + \Delta W_{kj}(n)$$

Prilagoditev ΔWkj(n) se izračuna kot rezultat vzbujanja iz okolice (dogodek 1). Osvežena vrednost **Wkj(n+1)** predstavlja spremembo v mreži, kot rezultat tega vzbujanja (dogodek 2). Dogodek 3 se izvrši, ko se izračuna odgovor nove mreže, ki deluje z osveženo vrednostjo parametra **Wkj(n+1)**. Opisano množico pravil imenujemo algoritem učenja (learning algorithm). Obstaja več algoritmov za učenje nevronskih mrež, ki se v osnovi razlikujejo po načinu prilagoditve sinaptične uteži **Wkj**.

Ločimo naslednje vrste algoritmov učenja:

- Učenje s popravljanjem napak (error-correction learning),
- Hebbovo učenje (Hebbian learning),
- tekmovalno učenje (competitive learning) in
- Boltzmannovo učenje (Boltzmann learning).

Ob algoritmih učenja poznamo še **paradigme** učenja (learning paradigms), ki se nanašajo na modele okolja, v katerem delujejo nevronske mreže. Ločimo tri osnovne paradigme učenja:

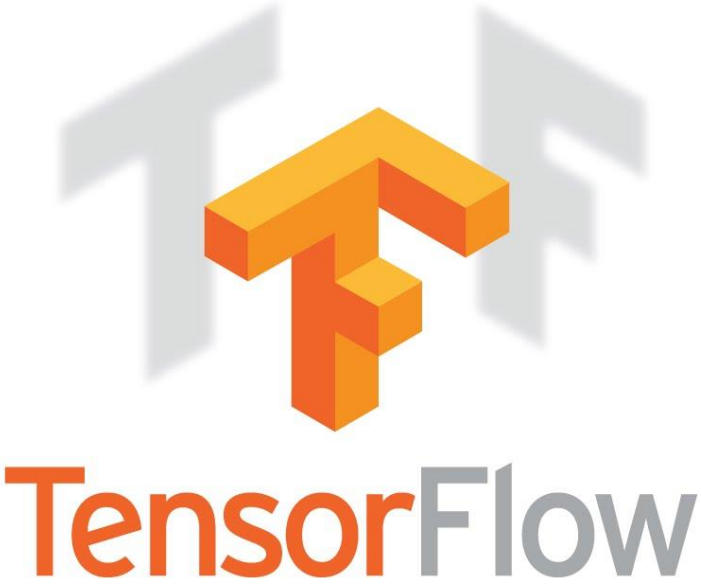
- nadzorovano ali vodeno učenje (supervised learning),
- okrepitveno učenje (reinforcement learning) in
- samoorganizirajoče (self-organizing) ali nenadzorovano učenje (unsupervised learning)

4. PROGRAMSKO ORODJE TENSORFLOW

TensorFlow je odprtokodna knjižnica za numerično računanje z uporab grafov pretoka podatkov. Vozlišča v grafu predstavljajo matematične operacije, ko pa robovi grafa predstavljajo večdimenzionalne podatkovne nize (tensors), ki komunicirajo med seboj. Fleksibilna arhitektura omogoča, da lahko računanje razdelimo na enega ali več procesorskih enot (CPU) ali grafičnih procesorskih enot (GPU-) nameščenih v namiznem računalniku, strežniku ali na mobilni napravi z enojnim programskim vmesnikom. Knjižnico TensorFlow so prvotno izdelali raziskovalci in inženirji, ki delajo v skupini »Google Brain Team« v raziskovalnem oddelku »Google's Machine Intelligence research«, z namenom izvajanja

strojnega učenja in raziskav o globokih nevronskih mrežah, vendar pa je sistem dovolj splošen, da ga lahko *uporabljamo* na različnih področjih.

Knjižnici zajema veliko različnih področji uporabe globokega učenja, pri izvedbi raziskovalne naloge pa smo uporabil področje »image processing«, kjer so na voljo orodja, potrebna za izdelavo zastavljenega eksperimenta.



Slika 5:Knjižnica TensorFlow[11]

5. PROCESIRANJE VIDEA IN SLIKE

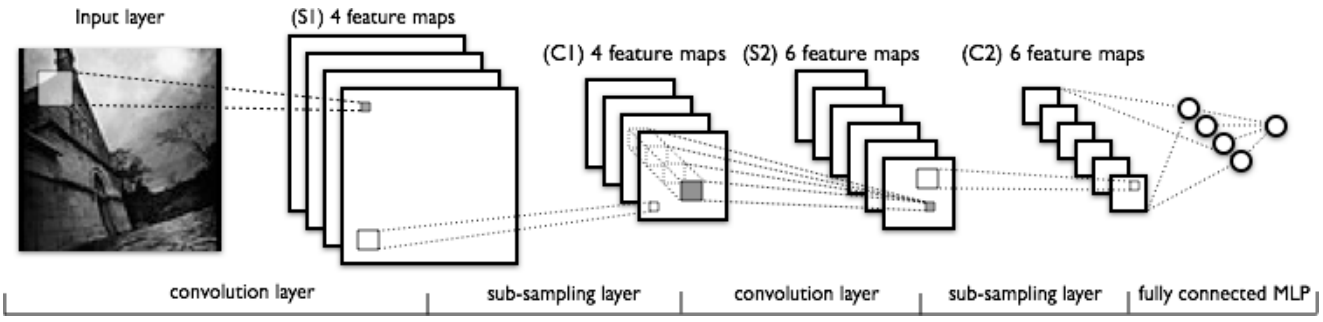
5.1 Digitalna slika in video

Kamera pretvori svetlobo v tabelo številk, ki jih imenujemo piksli oziroma slikovni elementi. Ti so lahko zapisani z barvnim modelom RGB. V tem primeru je posamezen piksel najpogosteje zapisan s 24 biti, ki podajajo deleže rdeče, zelene in modre barve (delež vsake barve je določena z 8 biti). Velikost digitalne slike je določena s številom pikslov, ki jo sestavljajo. Velikost tako podajamo v številu pikslov, ki določajo širino in število pikslov, ki določajo višino slike – na primer 1920X1080 ali 2048X1563 ali kako drugače. Če ti dve števili

pomnožimo, dobimo število pikslov, ki sestavljajo sliko – zelo pogosto je teh nekaj milijonov, zato govorimo o megapikslih. Rečemo lahko, da računalnik vidi sliko kot skupino majhnih kock z barvami. Barve barvnega modela RGB, rdeča, zelena in modra, so osnovne barve, s katerimi lahko določimo v primeru 24 bitne predstavitve več kot 16 milijonov različnih barvnih odtenkov. Digitalno sliko obravnavamo kot dvodimenzionalni signal.

5.2 Procesiranje slike z nevronskimi mrežami

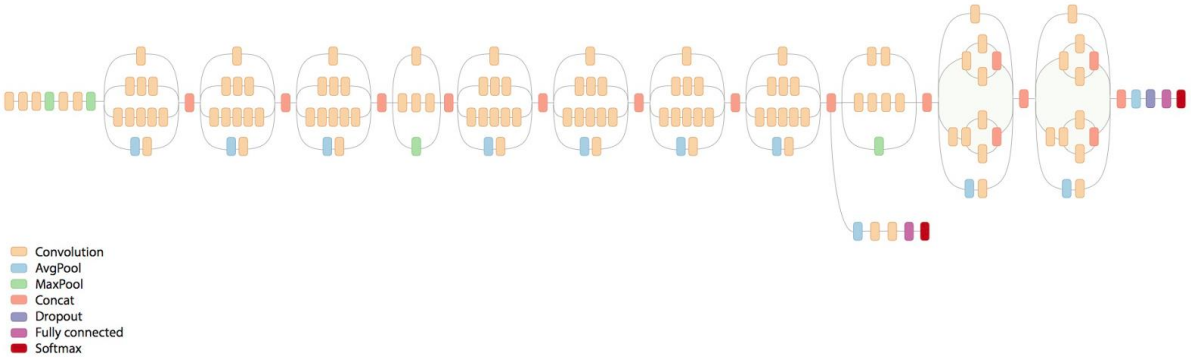
Kako nevronska mreža procesira slike. Nevronsko mrežo moramo najprej naučiti, kako izgleda ciljni objekt. V našem primeru je ciljni objekt človek, zato moramo na vhod podati določeno število slik, ki vključujejo ciljni objekt. Nevronska mreža se potem nauči, kakšne barve so piksli in kako izgledajo ter kateri se držijo skupaj. Iz tega si potem naredi maske za objekte, saj je lahko objekt slikan oziroma sneman iz različnih strani. Ko se na vhodu mreže pojavi slika, jo ta procesira tako, da skuša ujemat definirane maske. V primeru, da se katera ujema, določi, da je objekt tisto kar maska definira – kar posamezna maska definira je bilo določeno v fazi učenja.



Slika 6: Konvolucijska nevronska mreža. Povzeto po [13].

6. EKSPERIMENT

V okviru eksperimenta smo definirali klasifikator, ki naj v slikah prepozna človeka, če se ta v sliki nahaja. Pri izvedbi se nismo lotili izdelave celotne nevronske mreže, temveč smo spremenili zadnji nivo obstoječe nevronske mreže (slika 7). Uporabili smo že ustvarjeno in naučeno nevronsko mrežo, ki jo je izdelalo podjetje Google (Inception), poleg tega smo uporabili programsko orodje TensorFlow, ki je podrobneje opisano v poglavju 3. Pri eksperimentu smo uporabljali prenosni računalnik z operacijskim sistemom Windows 10, na katerem pa nismo mogli uporabiti knjižnice TensorFlow. Zato smo na prenosni računalnik namestili program VirtualBox. VirtualBox je odprtokodni program, ki ustvari in zažene virtualno napravo, na kateri je nameščen poljubni operacijski sistem, kot so na primer Linux, OS X, Windows, Solaris.



Slika 4: GoogleLeNet. Povzeto po [14].

6.1 Kaj bomo zgradili

V eksperimentu bomo uporabljali postopek prenesenega učenja (»Transfer learning«), kar pomeni, da bomo uporabili začetni model, ki je bil že naučen za drug problem. V okviru eksperimenta ga bomo še dodatno učili za podoben problem. Izvedba postopka globokega učenja iz nič je namreč računsko zelo zahtevna in lahko traja dneve. S postopkom prenosnega učenja pa lahko učenje izvedemo v bistveno krajšem času.

Pri izdelavi bomo uporabljali mrežo **Inception v3 network**. Mreža Inception v3 je bila učena za tekmovanje »ImageNet Large Visual Recognition Challenge«, uporabljeni pa so bili podatki iz leta 2012. Vsebuje lahko več kot 1000 različnih razredov objektov, kot na primer Dalmatinci ali pa pomivalni stroji. Uporabili bomo enak model, ki pa ga bomo dodatno učili z namenom, da bo znal prepoznati, ali je na sliki prisoten človek.

6.2 Postavitev delovnega okolja in pridobivanje rezultatov

Najprej smo morali na operacijski sistem Linux v virtualnem okolju namestiti knjižnico TensorFlow. Za uspešno namestitev knjižnice TensorFlow smo morali namestiti programski jezik Python, ki pa je bil na naši napravi že nameščen in sicer različica Python2.7.

Za namestitev knjižnice TensorFlow smo imeli na voljo več načinov inštalacije. Namestili smo jo s pomočjo orodja Docker. To je orodje, ki ustvari okolje, v katerem se izvaja sistem in je povsem izolirano od programov na gostiteljskem računalniku.

Pridobiti smo morali tudi bazo ustreznih slik. Za sprejemljivo klasifikacijo je bilo potrebno imeti okoli 100 slik, ki pa so morale biti iste velikosti in istega tipa in sicer jpg. Naša mapa je vsebovala 131 slik tipa jpg.

6.2.1 Inštalacija Dockerja.

Najprej smo namestili dodatni paket in **curl**. Curl se uporablja v ukaznih vrsticah ali skriptah za prenašanje podatkov preko URLs.

V terminal smo vpisali naslednjo kodo:

```
$ sudo apt-get update

$ sudo apt-get install curl \

linux-image-extra-$(uname -r) \

linux-image-extra-virtual
```

Nato smo se lotili namestitve preko skladišča »**repository**«. Sprva smo morali ustvariti novo skladišče (repositiory).

1. Namestili smo paket, ki omogoča paketnemu orodju APT (APT – Advanced Package Tool je uporabniški vmesnik, ki sodeluje z jadrno knjižnico za vodenje inštalacij in odstranjevanje programov na distribucijah Linuxa), da uporablja skladišče preko HTTPS:

```
$ sudo apt-get install apt-transport-https \ ca-certificates
```

2. Dodati smo morali uradni GPG ključ:

```
$ curl -fsSL https://yum.dockerproject.org/gpg | sudo apt-key add -
```

3. Nazadnje smo naredili skladišče stabilno:

```
$ sudo apt-get install software-properties-common

 $ sudo add-apt-repository \

 "deb https://apt.dockerproject.org/repo/ \

ubuntu-$(lsb_release -cs) \ main"
```

Sledila je namestitev orodja Docker. Z ukazom: $ sudo apt-get -y install docker-**engine** smo Docker namestili na sistem. Ko je bil Docker nameščen, smo zagnali skladišče Docker z binarno sliko knjižnice TensorFlow:

```
$ docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow
```

6.2.2. Pridobivanje slik

Preden smo lahko pričeli z učenjem, smo potrebovali niz slik za učenje mreže o novem razredu, ki ga želimo prepoznavati. Tako da smo naredili mapo **tf_files**, ki je vsebovala še eno mapo, v kateri so se nahajale slike z ljudmi.

TensorFlow Docker sprva ni vseboval podatkov o ljudeh, zato smo slike z njim povezali virtualno:

```
docker run -it -v $HOME/tf_files:/tf_files  gcr.io/tensorflow/tensorflow:latest-devel
```

Nato smo morali pridobiti še potrebna orodja iz strežnika GitHub, ki so potrebna za nadaljevanje:

```
cd /tensorflow \ git pul
```

6.2.3. Ponovno učenje Inceptiona

Učenje modela smo izvedli z naslednjo množico ukazov:

```
python tensorflow/examples/image_retraining/retrain.py \
--bottleneck_dir=/tf_files/bottlenecks \
--how_many_training_steps 500 \
--model_dir=/tf_files/inception \
--output_graph=/tf_files/retrained_graph.pb \
--output_labels=/tf_files/retrained_labels.txt \
--image_dir /tf_files/people
```

Zgornja skripta naloži že učen model Inception v3, odstrani zadnji stari nivo in nauči novega na podatkih, ki smo jih priložili.

Prva faza analizira vse slike na disku in izračuna vrednosti **bottlenecka** za vsako sliko posebej. Bottleneck je neformalni izraz, ki ga uporabljamo za nivo tik pred zadnjim izhodnim nivojem, ki dejansko izvede klasifikacijo.

Vsaka slika je med učenjem uporabljena večkrat. Računanje nivojev za vrednost bottleneck za vsako sliko vzame veliko časa. S predpomnjenjem izhodov nižjih nivojev in njihovim zapisom v pomnilnik omogočimo, da jih ni potrebno zmeraj znova izračunavati.

Ko so vrednosti bottleneck določene, se dejansko učenje zadnjega nivoja lahko začne. Sistem med učenjem izpisuje različne vrednosti, ki podajajo **natančnost učenja**, **natančnost potrjevanja** in **križno entropijo.**

- **Natančnost učenja:** prikazuje odstotek uporabljenih slik v trenutni seriji učenja, ki so bile označene z ustreznim razredom.
- **Natančnost potrjevanja:** je natančnost (odstotek pravilno označenih slik) na naključno izbrani skupini slik iz različnih sklopov.
- **križna entropija:** je funkcija izgube, ki daje vpogled v to, kako dobro učni proces napreduje. (Nižje številke so boljše).

Privzeto skripta zažene 4000 učnih korakov. Vsak korak naključno izbere 10 slik iz učne množice, poišče ustrezne vrednosti bottleneck iz predpomnilnika in jih naloži v zadnji nivo za napoved. Te napovedi se primerjajo z dejanskimi oznakami za posodobitev uteži na zadnjem nivoju preko procesa povratnega napredovanja (back-propagation).

Z nadaljevanjem procesa je iz poročila razvidno, da se natančnost izboljšuje. Ko so vsi koraki izvedeni, skripta še zadnjič testira končno oceno natančnost na izbrani množici slik, ki so ločene od učnih in potrjevalnih slik. Takšen test vrednotenja zagotavlja najboljšo oceno, kako se bo izkazal model pri klasifikacijskih nalogah v realnem življenju.

6.2.4. Uporaba novega modela

Z naslednjo kodo naložimo novo grafično datoteko in napovedujemo z njo:

```
import tensorflow as tf, sys

image_path = sys.argv[1]

image_data = tf.gfile.FastGFile(image_path, 'rb').read()

label_lines = [line.rstrip() for line

                in tf.gfile.GFile("tf_files/retrained_labels.txt")]

with tf.gfile.FastGFile("tf_files/retrained_graph.pb", 'rb') as f:

  graph_def = tf.GraphDef()

  graph_def.ParseFromString(f.read())

  _ = tf.import_graph_def(graph_def, name='')
```

```
with tf.Session() as sess:
   softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')
   predictions = sess.run(softmax_tensor, \
         {'DecodeJpeg/contents:0': image_data})
   top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
   for node_id in top_k:
      human_string = label_lines[node_id]
      score = predictions[0][node_id]
      print('%s (score = %.5f)' % (human_string, score))
```

Pred tem moramo resetirati Docker:

```
docker run -it -v $HOME/tf_files:/tf_files  gcr.io/tensorflow/tensorflow:latest-devel
```

Nato zaženemo datoteko Python, ki smo jo prej ustvarili na želeni sliki:

```
python /tf_files/label_image.py /tf_files/people/slika35.jpg
```

Izpiše se končni rezultat v obliki:

```
people (score = 0.89071)
```

Iz rezultata lahko izberemo, da je naš klasifikator 89,071% prepričan, da je na sliki, ki jo je sistem procesiral, prisoten človek.

7. ZAKLJUČEK

Z izvedbo eksperimenta smo ugotovili, da je definiran sistem nevronskih mrež potrdil možnost ugotavljanja prisotnosti ljudi v slikah. Ugotovili smo tudi, da je glede zmožnosti postopkov globokega učenja mogoče izvesti klasifikator, ki bi znal ločevati med posameznimi pozami človeka v sliki ter tudi ugotoviti, katera poza je kritična za človeka in katera ni. Za izvedbo te funkcionalnosti bi morali ponoviti izveden postopek in uporabiti bazo slik, v katerih bi bili ljudje v položajih, ki predstavljajo kritične dogodke.

8. LITERATURA

[1] Globoko učenje [online]. Wikipedija. 2016. [7.2.2017]. Dostopno na: https://sl.wikipedia.org/wiki/Globoko_u%C4%8Denje

[2] DeepLearning [online]. DeepLearning. 2015. [7.2.2017]. Dostopno na:

http://deeplearning.net/

[3] DeepLearning - An MIT Press book [online]. DeepLearningbook. 2016. [7.2.2017]. Dostopno na: http://www.deeplearningbook.org/

[4] DeepLearning [online]. Neural networks and deeplearning. 2017. [7.2.2017].

Dostopno na: http://neuralnetworksanddeeplearning.com/chap6.html

[5] Nevronske mreže [online]. Wikipedija. 2016. [7.2.2017]. Dostopno na: https://sl.wikipedia.org/wiki/Nevronska_mre%C5%BEa

[6] Nevronske mreže [online]. Robin2. 2007. [7.2.2017]. Dostopno na:

http://robin2.r.uni-mb.si/predmeti/krmilna/predavanja/krtn.html

[7] Slika. Nevronske mreže [online] Dostopno na: https://www.google.pt/search?q=umetni+nevron&espv=2&biw=1707&bih=817&tbm=isch&imgil=vdsMzzC33eFdQM%253A%253BBoyUpMoWdFUtaM%253Bhttp%25253A%25252F%25252Fwww.monitor.si%25252Fclanek%25252Fracunalniki-se-ucijo%25252F171558%25252F&source=iu&pf=m&fir=vdsMzzC33eFdQM%253A%252CBoyUp

MoWdFUtaM%252C_&usg=__42fouJAsRDZhOXzvXjpOrvlia7Y%3D&ved=0ahUKEwjF5ISc8oHSAhXINpoKHQUDBswQyjcIIw&ei=xMmbWMXyOMjt6ASFhpjgDA#imgrc=vdsMzzC33eFdQM:

[8] Slika. sigmoidna funkcija [online] Dostopno na: https://www.google.pt/search?q=umetni+nevron&espv=2&biw=1707&bih=817&tbm=isch&imgil=vdsMzzC33eFdQM%253A%253BBoyUpMoWdFUtaM%253Bhttp%25253A%25252F%25252Fwww.monitor.si%25252Fclanek%25252Fracunalniki-se-ucijo%25252F171558%25252F&source=iu&pf=m&fir=vdsMzzC33eFdQM%253A%252CBoyUpMoWdFUtaM%252C_&usg=__42fouJAsRDZhOXzvXjpOrvlia7Y%3D&ved=0ahUKEwjF5ISc8oHSAhXINpoKHQUDBswQyjcIIw&ei=xMmbWMXyOMjt6ASFhpjgDA#tbm=isch&q=+sigmoidna+funkcija&imgdii=s6KRKZwjBeC6SM:&imgrc=B9M8L1x8TTUrtM:

[9] Slika. Signum funkcija [online] Dostopno na: https://www.google.pt/search?q=umetni+nevron&espv=2&biw=1707&bih=817&tbm=isch&imgil=vdsMzzC33eFdQM%253A%253BBoyUpMoWdFUtaM%253Bhttp%25253A%25252F%25252Fwww.monitor.si%25252Fclanek%25252Fracunalniki-se-ucijo%25252F171558%25252F&source=iu&pf=m&fir=vdsMzzC33eFdQM%253A%252CBoyUpMoWdFUtaM%252C_&usg=__42fouJAsRDZhOXzvXjpOrvlia7Y%3D&ved=0ahUKEwjF5ISc8oHSAhXINpoKHQUDBswQyjcIIw&ei=xMmbWMXyOMjt6ASFhpjgDA#q=signum+funkcija&tbm=isch&tbs=rimg:CaNuy_1DOVa8-Iji1wDjG_1kqEexg6ZRaiZehMpZP9kqrf1b-zO5fcxfYqxIkhBMsPqeYO3UE0CjuIvRKFRtPFZR5DMCoSCbXAOMb-SoR7EWOjf2vazDC_1KhIJGDplFqJl6EwRzqPTqVE3NTAqEgmlk_12Sqt_1VvxGfOJc2QIeXgSoSCbM7l9zF9irEETJESKVIPs1BKhIJiSEEyw-p5g4RiQlcVYiYDzwqEgndQTQKO4i9EhEDYx4fGdyv_1SoSCYVG08VlHkMwEaIJ1tIQtZVC&imgdii=GDplFqJl6Ew2BM:&imgrc=o27L8M5Vrz4I9M:

[10] Slika. threshold funkcija [online] Dostopno na: https://www.google.pt/search?q=umetni+nevron&espv=2&biw=1707&bih=817&tbm=isch&imgil=vdsMzzC33eFdQM%253A%253BBoyUpMoWdFUtaM%253Bhttp%25253A%25252F%25252Fwww.monitor.si%25252Fclanek%25252Fracunalniki-se-ucijo%25252F171558%25252F&source=iu&pf=m&fir=vdsMzzC33eFdQM%253A%252CBoyUpMoWdFUtaM%252C_&usg=__42fouJAsRDZhOXzvXjpOrvlia7Y%3D&ved=0ahUKEwjF5ISc8oHSAhXINpoKHQUDBswQyjcIIw&ei=xMmbWMXyOMjt6ASFhpjgDA#q=threshold+funkcija&tbm=isch&tbs=rimg:CfRjpIh_1zx6sIjgq3Z1-xrnTMUrr78oF4wHborRVowZ99FkcrKchTBUg8x16EmWhCM5Rm4QcE9gsw9O4kEm3mVJuQyoSCSrdnX7GudMxEeNZx_1HUN2ARKhIJSuvvygXjAdsRZXJCJH6i27IqEgmitFWjBn30WREREugT25j8eioSCRyspyFMFSDzEWcmfDoqX7VoKhIJHXoSZaEIzlERfkk6FTj2XrUqEgmbhBwT2CzD0xHYJbQgoClatyoSCbiQSbeZUm5DEQ2iFsxvLrgx&imgrc=CTBwQNujF_O0IM:

[11] Slika. TensorFlow [online] Dostopno na: https://www.google.pt/search?q=odsekoma+funkcija&espv=2&biw=1707&bih=817&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiL9dPp9IHSAhWFIpoKHUcDAi4Q_AUIBigB#tbm=isch&q=tensorflow&imgrc=89zLMX16i2DP9M:

[12] About TensorFlow [online]. TensorFlow. 2016. [7.2.2017]. Dostopno na: https://www.tensorflow.org/

[13] Slika. Convolution neural network[online] Dostopno na:

https://www.google.pt/search?q=Convolution+neural+network&espv=2&biw=1707&bih=817&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjw6_n39YHSAhWDyRoKHUMZCroQ_AUIBigB#imgrc=8rxBNlzyWvi0gM:

[14] Slika. GoogleLeNet [online] Dostopno na: https://www.google.pt/search?q=Convolution+neural+network&espv=2&biw=1707&bih=817&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjw6_n39YHSAhWDyRoKHUMZCroQ_AUIBigB#tbm=isch&q=googlelenet&imgrc=cWlPHI47GfCovM:

[15] TensorFlow for Poets [online]. CodeLab 2015. [7.2.2017]. Dostopno na: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/?utm_campaign=chrome_series_machinelearning_063016&utm_source=gdev&utm_medium=yt-desc#3