UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Identificarea semnelor de circulatie prin retele convulutionale

propusă de

George-Alin Atodiresei

Sesiunea: *iulie, 2019*

Coordonator științific

Lect. Dr. Anca Ignat

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Identificarea semnelor de circulatie prin retele convulutionale

George-Alin Atodiresei

Sesiunea: *iulie, 2019*

Coordonator științific

Lect. Dr. Anca Ignat

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAŢIE privind originalitatea conţinutului lucrării de licenţă

Subsemntatul(a) ……

domiciliul în ……..

născut(ă) la data de ……., identificat prin CNP ……., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de …….. specializarea ……., promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: _____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Identificarea semnelor de circulatie prin retele convolutionale*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent *George-Alin Atodiresei*

_____

(semnătura în original)

Cuprins

Introducere.....6

Contributii 9

1. Glosar 10

2. Setul de date 10

3. Preprocesarea imaginilor 13

4. Construirea retelei convulutionale.....14

5. Antrenarea retelei.....21

6. Clasificarea semnelor de circulatie.....22

7. Concluzii si directii viitoare.....22

8. Bibliografie 24

Introducere

Mai sus avem un semn de circulatie oarecare. Pentru unii dintre noi acesta reprezinta doar un cerc si un numar. Pentru altii, reprezinta limita dintre o amenda usturatoare sau o luna fara permis de conducere. Fie ca suntem soferi, biciclisti, motociclisti sau pietoni, semnele de circulatie sunt peste tot in jurul nostru si incearca sa organizeze haosul care este traficul de pe sosele, sa ofere informatii tuturor celor care le vad si sa impuna restrictiile necesare cand este nevoie.

Odata cu trecerea timpului, omul a tot cautat sa-si delege din atributii catre calculator si roboti, care deveneau si devin din ce in ce mai inteligenti si capabili sa preia din responsabilitatile cotidiene. Astfel au aparut si primii asistenti in trafic, prima oara prin intermediul GPS-ului, care nu recunoaste semnele de circulatie in timp real, ci are zonele in care sunt aplicate salvate odata cu traseul. Insa, dupa ceva timp, odata ce toate masinile au inceput sa aiba camera ce sunt capabile sa filmeze in toate directiile, au inceput sa apara si sisteme inteligente care se ocupa cu recunoasterea semnelor de circulatie, a semafoarelor si mai nou, a benzilor de circulatie, cu tehnologia Lane

Assist. In momentul actual, Tesla este aproape pregatita sa scoata o masina total autonoma pe piata, dar si celelalte marci nu se lasa mai prejos, masina autonoma fiind urmatorul mare punct al evolutiei industriei automotive.

Exemplu de sistem de identificare al semnelor de circulatie:

[1]

Toate aceste progrese se datoreaza computer vision-ului.

Computer vision este un domeniu care in ultimii ani a ajuns sa fie foarte folosit in viata de zi cu zi a oamenilor. Fie ca este vorba de recunoastere faciala pentru deblocarea telefonului, de filtrele folosite pe aplicatiile de social media sau pentru a vedea orice oras din lume in 3D, cu ajutorul Google Maps, acest domeniu capata din ce in ce mai mult teren in viata oamenilor. Computer vision poate fi impartit in mai multe subdomenii: clasificare de imagini, restaurare de imagini, motion tracking etc.

Clasificarea de imagini reprezinta unul dintre campurile care avanseaza cel mai rapid, fiind nevoie de aceasta tehnologie mai ales in industria auto, unde masinile autonome incep sa fie din ce in ce mai sigure, fiind capabile sa se descurce in orice situatie din trafic. Pentru a ajunge in stadiul acesta, ele au trebuit intai sa poata intelege mediul in care isi desfasoara activitatea, de liniile de pe strada pana la semnele de circulatie pe care le intampina. O cautare mai in detaliu arata si modul in care

[1] Ford Focus-Traffic Sign Recognition: https://www.youtube.com/watch?v=kJfa2HsTtlg

aceste probleme au fost rezolvate, mai ales in cazul semnelor de circulatie. Solutia este folosirea metodelor de invatare automata pentru a antrena o retea specializata pentru recunoasterea acestor semne. Pentru antrenament avem nevoie de o baza de date destul de mare cu semne de circulatie si o platforma pe care sa antrenam reteaua respectiva. Setul de date este cea de la GTSRB, care contine 43 de clase, cate una pentru fiecare semn de circulatie si peste 30000 de imagini pe care putem antrena reteaua. Pentru antrenamentul propriu zis, vom folosi retele convulutionale cu backend Tensorflow, pentru a putea face antrenamentul pe laptop-ul personal. Vom urmari astfel, cum se gaseste si cum se foloseste setul de date, cum se construieste reteaua, cum se antreneaza aceasta si cum este folosita pentru cazuri noi.

In final, pentru verificarea corectitudinii solutiei, vom testa reteaua pe poze cu semne de circulatie care nu se regasesc in setul de antrenament si vom vedea daca aceasta il identifica corect.

Contributii

- Construirea unei retele potrivite pentru sarcina urmarita
- Gasirea si utilizarii unei baze de date cu peste 30000 de imagini pentru antrenament
- Realizarea unui mediu unde antrenamentul retelei se poate face in timp util
- Compararea intre diferite medii de executie
- Testarea retelei pe imagini neintalnite de aceasta
- Construirea unor grafice ce reflecta capacitatile retelei

1. Glosar

- Acuratete - probabilitatea modelului de a prezice corect un termen
- Functie de activare – o functie ce transforma un input dintr-un strat printr-o anumita formula in input pentru urmatorul strat
- Batch- numar de elemente folosite intr-o iteratie
- Epoch- reprezinta o trecere prin tot setul de date
- Layer- un set de neuroni unde se proceseaza un set de input sau output-ul altor layere
- Loss – o unitate de masura care indica cat de departe este predictia retelei de rezultatul asteptat
- Neuron- reprezinta un nod din retea, care de obicei ia mai multe valori ca si input si genereaza un output
- One-hot encoding- reprezinta un vector in care unul din elemente este 1 iar restul 0
- Prediction- output-ul unui model atunci cand ii primeste un input
- Underfitting- reprezinta caracteristica unui model care nu are o acuratete mare
- Overfitting- reprezinta caracterstica unui model care devine prea specializat pe setul de data antrenat si nu este pregatit sa lucreze cu date noi[2]

2. Setul de date

In momentul de fata, cel mai mare impediment in fata progresului aplicatiilor bazate pe clasificarea de imagini este lipsa de seturi de date de mari dimensiuni pe care sa se poate face antrenamentul. Cu cat setul de date de antrenament este mai mare, cu atat aplicatia v-a fi mai sigura si de incredere. In cazul de fata, avem nevoie de un set de date care sa contina cat mai multe imagini cu semne de circulatie. Solutia la aceasta problema a fost oferita in cadrul Conferintei Internationale de Retele Neuronale, unde a fost introdus GTSRB (The German Traffic Sign Benchmark) care contine 43 de clase, fiecare reprezentand un semn de circulatie gasit pe strazile din Germania si peste 25000 de imagini pentru antrenament si peste 10000 de imagini pentru validarea modelului.

Exemple de imagini:

[2] Machine Learning Glossary: https://developers.google.com/machine-learning/glossary/

[3]

Figure 1

De asemenea, imaginile sunt destul de diversificate, neexistand o anumita clasa care sa contina prea multe imagini fata de celelalte clase. Acest lucru asigura faptul ca toate semnele de circulatie au sanse aproximativ egale pentru a fi recunoscute dupa antrenament.

[3] GTRSB: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

Distributia claselor este astfel:

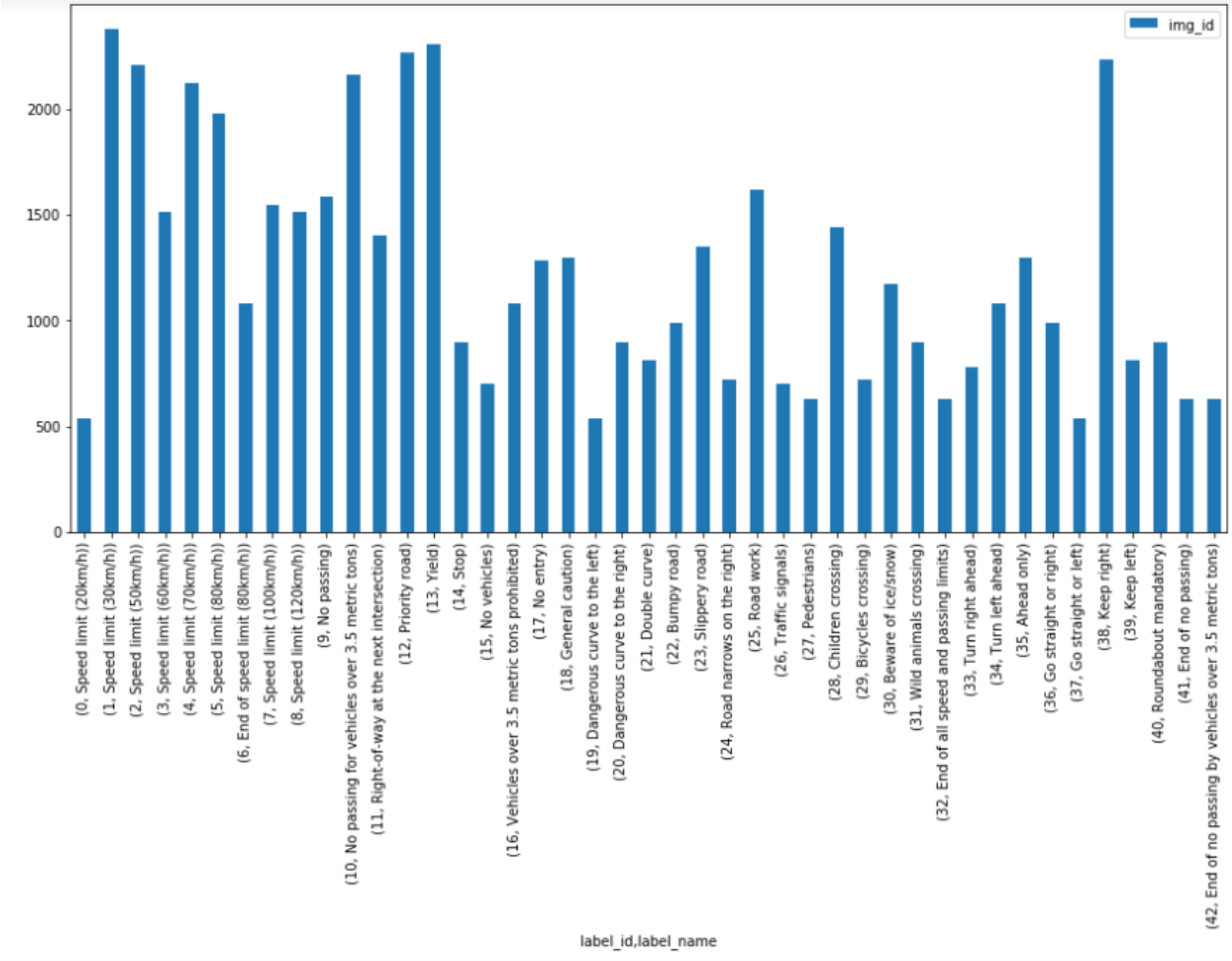


Figure 2[4]

Imaginile au dimensiuni intre 15x15 si 250x250 de pixeli, ceea ce pune dificultati retelei. Pentru a rezolva aceasta problema, am introdus un pas de preprocesare, unde imaginile sunt redimensionate la dimensiunea standard de 48x48 de pixeli.

O alta optiune pentru setul de date ar fi fost BelgiumTS Dataset, care contine mai multe clase de imagini decat GTSRB, 62 mai exact si peste 4000 de imagini de antrenament. Deci, acest set de data contine are doar 10% din dimensiunile celui ales. De asemenea, acest set nu are o

[4] Recognising Traffic Signs With 98% Accuracy Using Deep Learning: https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab

varietate foarte mare in luminozitatea imaginilor, unghiul de vizualizare, claritatea si, in mare, diversitatea pe care o ofera cel german.

Exemple de imagini:

Figure 3

Daca comparam exemplele de imagini din Figura 1 si Figura 3, putem observa diferentele notabile dintre cele doua seturi de date. Cele din Figura 1 captureaza un spectru mai larg de conditii de luminozitate si vizualizare. Imaginile din Figura 3 au, in mare parte, aceeasi luminozitate si aproximativ acelasi unghi, fiind prea rigid pentru nevoile retelei noastre. Dupa cum am spus in Introducere, o retea are nevoie de cat mai multe date pe care sa se antreneze, din care sa obtina informatii noi si sa le valorifice. In cazul nostru, diversitatea parametrilor imaginii reprezinta unul dintre cei mai importanti metrici in alegerea setului de date de lucru.

3. Preprocesarea imaginilor

In acest pas, imaginile sunt preprocesate pentru pregatirea lor de intrare in retea. In cazul nostru, preprocesarea consta in modificarea dimensiunilor fiecarei imagini in dimensiunile

standard de 48x48. In mod normal, incarcarea imaginilor in memorie nu ar trebui sa dureze foarte mult timp. Insa, luand in calcul preprocesarea si volumul mare de date, acest pas dureaza foarte mult timp, aproximativ 10 minute din timpul total de executie.

4. Construirea retelei convulutionale

O retea convulutionala este o retea capabila sa ia o imagine ca date de intrare, sa asigneze importanta diferitelor aspecte din imagini si sa faca diferenta intre ele. Conexiunile dintr-o retea convulutionala este asemanatoare conexiunilor neuronale din creierul uman. [5]

Structura grafica a unei retele convulutionale generale:

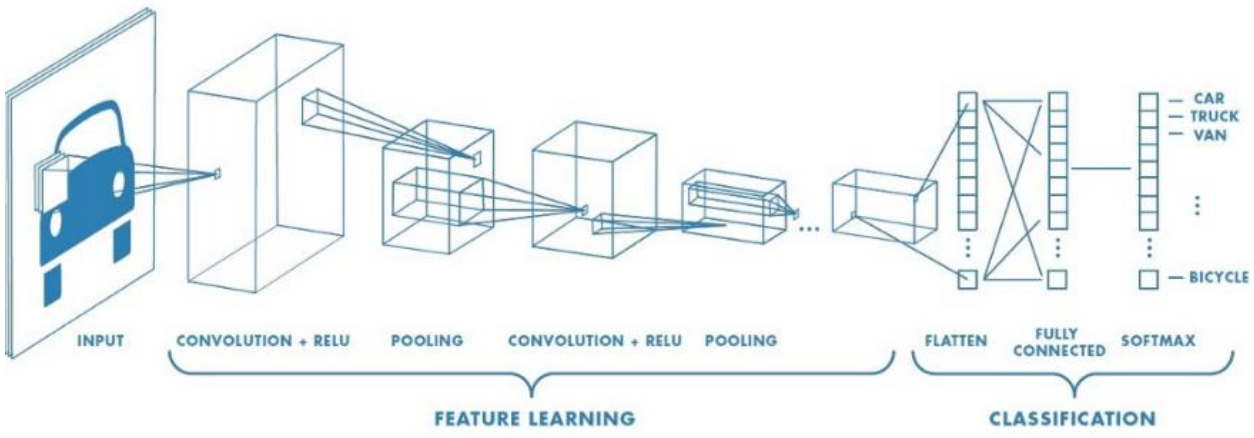


Figure 4 [1]

In Figura 2 sus regasim urmatoarele notiuni:

- Input
- Convolution layer
- ReLU activation
- Pooling
- Flatten
- Fully Connected

[5] A Comprehensive Guide to Convolutional Neural Networks: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

- Softmax

In cazul nostru, structura retelei este urmatoarea:

```
model = Sequential ([Conv2D (16, (3, 3), padding='same',
                        input_shape= (3, 48, 48),
                               activation='relu'),
                        Conv2D (16, (3, 3), activation='relu'),
                        MaxPooling2D (pool_size= (2, 2)),
                        Dropout (0.5),
                        Conv2D (32, (3, 3), padding='same',
                               activation='relu'),
                        Conv2D (32, (3, 3), activation='relu'),
                        MaxPooling2D (pool_size= (2, 2)),
                        Dropout (0.5),
                        Conv2D (64, (3, 3), padding='same',
                               activation='relu'),
                        Conv2D (64, (3, 3), activation='relu'),
                        MaxPooling2D (pool_size= (2, 2)),
                        Dropout (0.5),
                        Flatten (),
                        Dense (512, activation='relu'),
                        Dropout (0.5),
                        Dense (43, activation='softmax')])
```

Observam astfel ca reteaua este una generala, adaptata la datele problemei prezentate.

Pentru a intelege intregul proces de invatare, vom parcurge reteaua de la un capat la altul:

1. Primul strat convolutional (Conv2D) – Primul parametru din metoda, 16, reprezinta batch_size, mai exact, dimensiunea grupurilor pe care reteaua le va folosi pentru antrenament. Dimensiunea acestor grupuri trebuie sa fie in concordanta cu setul de date, ele influentand viteza cu care se va face antrenamentul si validitatea acestuia. La finalul trecerii unui batch prin retea, se recalculeaza greutatile din matricea filtru. De asemenea, primul layer din retea este si locul unde se primeste input-ul, care, aici, este de (3,48,48), unde 3 este numarul de canale de culoare, si 48 reprezinta lungimea si latimea imaginilor pe care

reteaua le va primi ca si input. In acest prim strat, pentru a "invata" imaginea, reteaua are un **filtru**, reprezentata de o matrice, care drept elemente niste greutati, alese aleatoriu pentru prima iteratie. Acest filtru, in cazul nostru de (3,3), parcurge imaginea, care la randul ei este tot o matrice, in cazul nostru de 48x48x3, fiecare element reprezentand un pixel din imaginea primita ca input. Filtrul incepe parcurgerea din coltul stanga sus, si inmulteste valorile elementelor din matricea filtru si valorile elementelor din matricea ce reprezinta pixelii din imagini, dupa care aduna valorile intre ele plus un bias, si obtine un numar, care este stocat. Apoi filtrul este mutat catre dreapta cu o valoare data, in cazul de fata, 1, si se repeta procesul anterior. Cand ajungem in marginea din dreapta, filtrul este mutat in partea din stanga, cu valoarea data mai jos. Astfel, se produce o glisare a filtrului pe toata imaginea data ca si input. La final, pentru cazul nostru, vom avea o matrice de 46x46x3. Aceastra matrice obtinuta dupa parcurgerea imaginii se numeste activation map.[6]

In Figura 5 avem reprezentata operatia de convulutiune:

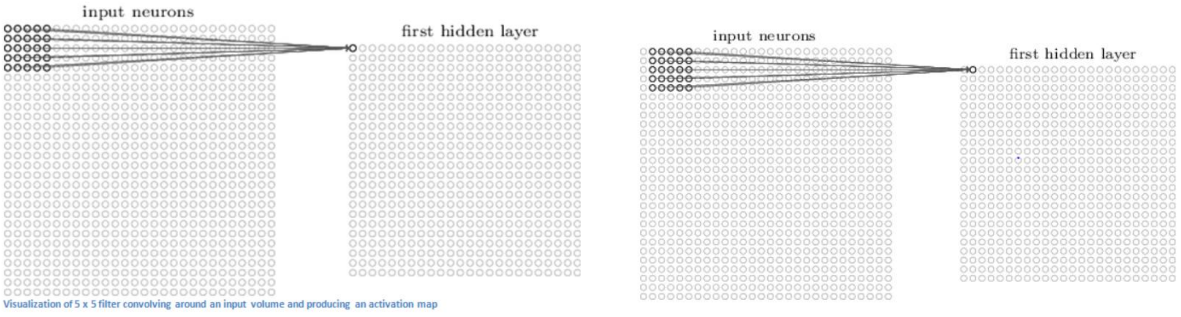


Figure 5 [7]

2. Functia de activare (ReLu activation) – Functia de activare decide daca un neuron din retea trebuie activat sau nu, prin calcularea greutatii acestui neuron. Acesta decide daca informatia primita de neuron este relevanta sau trebuie ignorata. Principalul rol este de a adauga non-linearitate in output-ul unui neuron. O retea neuronala fara o functie de activare este doar un

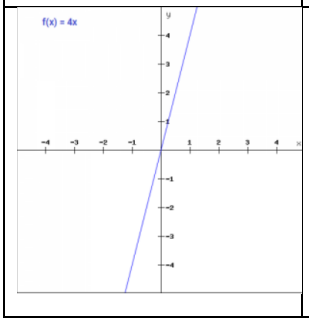
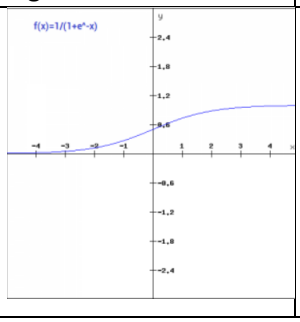
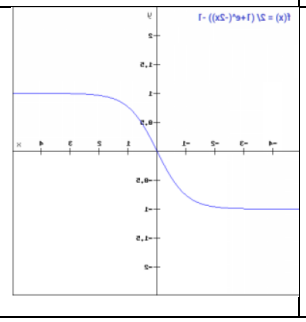
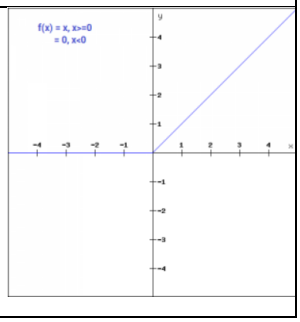
[6] A Beginner's Guide To Understanding Convolutional Neural Networks: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

[7] Deep Learning By Michael Nielsen: http://neuralnetworksanddeeplearning.com/index.html

model de regresie liniar. Prin adaugarea functiei de activare, modelul capata abilitatea de a invata si realiza actiuni mai complexe. Neuronii nu pot invata doar cu functie liniara atasata. Din acest motiv, neuronii au atasate functii non-liniare.

Printre cele mai folosite functii de activare sunt:

- Linear function: $f(x)=ax$
- Sigmoid function: $f(x)=1/(1+e^{-x})$
- Tanh function: $f(x)=\frac{2}{1+e^{-2x}}-1$
- RELU (Rectified linear unit): f(x)=max (0, x)

Linear function	Sigmoid function	Tanh function	RELU function
			

[8]

Functia de activare folosita in cazul nostru este cea RELU, din urmatoarele motive:

- Este o functie non-liniara, dupa cum putea vedea din graphic.
- Calculele necesare nu necesita la fel de multa putere de calcul ca in cazul altor functii.
- Nu activeaza toti neuronii in acelasi timp, facand reteaua eficienta.
- RELU lucreaza cel mai bine ca un aproximator general.

3. Pooling (`MaxPooling2D`) - Acest strat este folosit pentru a reduce dimensiunea input-ului pentru fiecare strat de convolutiune. Stratul de pooling impiedica reteaua sa devina prea specifica (fenomenul de overfitting), sa generalizeze. Astfel, prin aplicarea stratului de pooling, reteaua nu devine prea rigida cu localizarea caracteristicilor, mai ales prin schimbarea orientarii caracteristicii detectate. Spre exemplu, daca la un nivel inferior o caracteristica gasita este o anumita linie, prin pooling, ne asiguram ca acea linie va fi

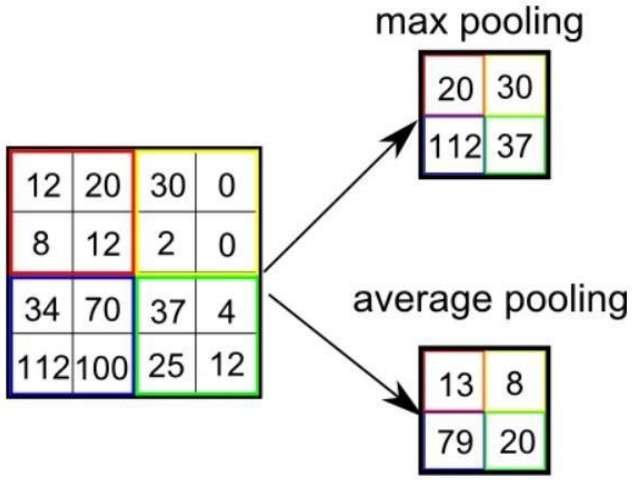
[8] Fundamentals of Deep Learning – Activation Functions and When to Use Them? : https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/

identificata si in nivelele superioare, indiferent de orientarea liniei. De asemenea, ne asiguram ca zona exacta in care se afla caracteristica nu este definitorie.

Pooling-ul poate fi de doua feluri:

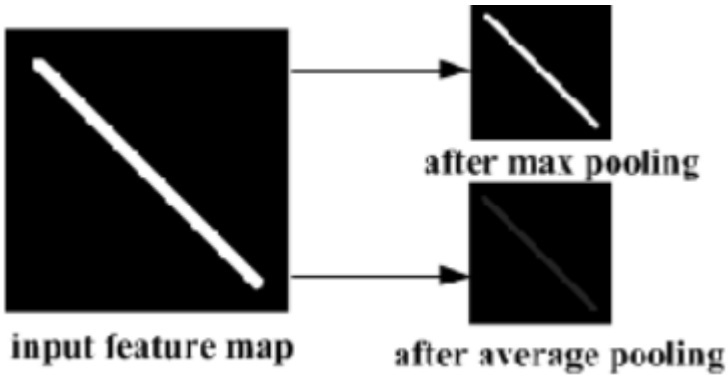
- Max pooling
- Average pooling

Reprezentare pentru pooling:



[9]

Pentru a intelege diferentele dintre cele doua tipuri de pooling, exemplul de mai jos este sugestiv

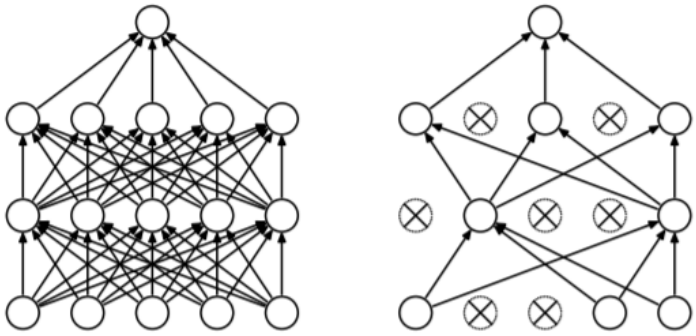


[9] The Ultimate NanoBook to understand Deep Learning based Image Classifier: https://towardsdatascience.com/https-medium-com-rishabh-grg-the-ultimate-nanobook-to-understand-deep-learning-based-image-classifier-33f43fea8327

Din cate putem observa mai sus, max pooling-ul este mai potrivit pentru setul nostru de date, intrucat acest procedeu pastreaza mai efficient caracteristicile ce ne intereseaza dintr-o imagine, cum ar fi marginile si colturile.

4. Dropout – Dropout-ul este o tehnica prin care se previne tendinta retelei de a deveni prea specifica. Tehnica consta in dezactivarea unor neuroni, eliminarea acestora temporar din retea, cu tot cu conexiunile pe care acestia le au, input si output. Prin eliminarea unor neuroni, vom obtine astfel o noua arhitectura de retea. Modul in care se elimina neuronii este dupa un parametru p, care reprezinta probabilitatea unui neuron de a fi eliminat. In cazul de fata, p are valoare 0.5, adica fiecare neuron are probabilitate de 50% de a fi eliminat din retea. Odata eliminate, fiecare neuron care inca este activ trebuie sa invete sa lucreze cu alte unitati alese aleator. Asta ar trebui sa faca ca fiecare strat sa fie mai robust si sa ii ofere abilitatea de a fi independent de unitati din alte straturi pentru a-si corecta greselile. [11]

   Exemplu de retea inainte si dupa aplicarea dropout-ului:



(a) Standard Neural Net (b) After applying dropout.

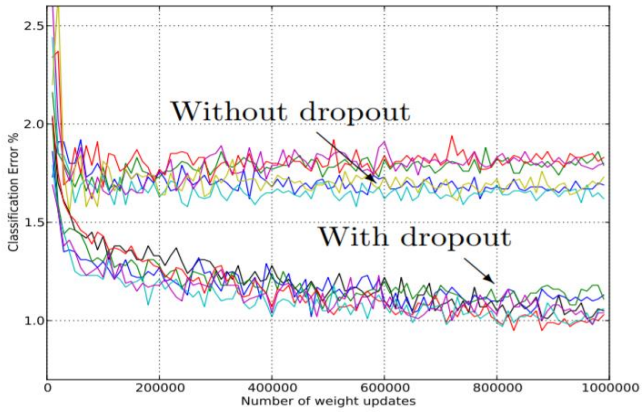
11

Prin folosirea dropout-ului si asigurarea capabilitatii retelei de a evita overfitting-ul, ne putem astepta ca aceasta sa fie mai performanta decat una in care nu s-a folosit.

[10] Sursa: https://www.researchgate.net/figure/Toy-example-illustrating-the-drawbacks-of-max-pooling-and-average-pooling_fig2_300020038

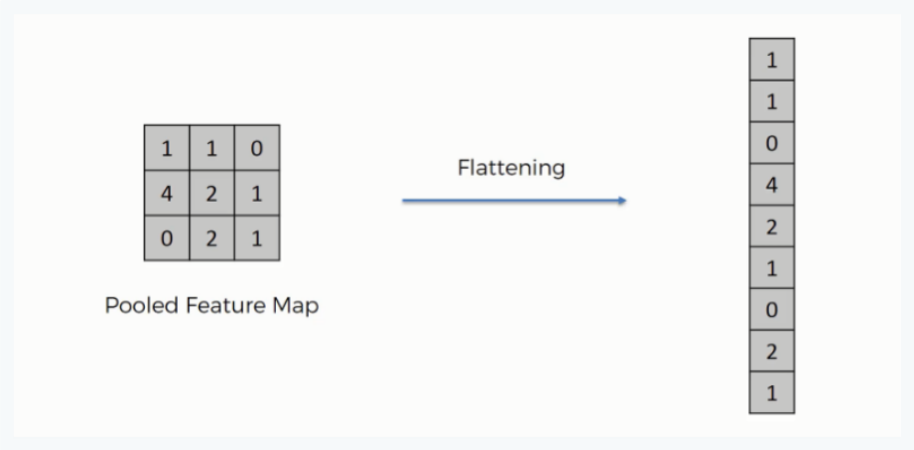
[11] Dropout: A Simple Way to Prevent Neural Networks from Overfitting: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Pentru a vedea efectele dropout-ului asupra performantei modelului, avem urmatorul grafic:



[12]

5. Flatten – Este stratul in care reteaua se pregateste de clasificarea propriu-zisa. Aici, input-ul este transformat din 2D in 1D, rezultand un vector ce v-a fi transmis straturilor fully connected pentru clasificare.



[13]

6. Fully connected – Dense – Este ultimul layer din cadrul retelei, unde se proceseaza tot antrenamentul interior. In acest layer, fiecare neuron din layerul anterior este conectat cu fiecare neuron din layerul curent. Acest layer este cel mai dens, cu cei mai multi parametri. De asemenea, aici este pusa si o functie de activare, in cazul nostru RELU, ca si in cazul

[12] Dropout: A Simple Way to Prevent Neural Networks from Overfitting: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[13] Convolutional Neural Networks (CNN): Step 3 – Flattening: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

straturilor anterioare, iar la final, folosim functia softmax. Functia softmax face clasificarea efectiva. Clasificatorul softmax returneaza probabilitatea fiecarei clase din setul nostru de date de a se regasi in imagine. In cazul nostru, dupa aplicarea clasificatorului, vom avea un vector de 43 de elemente, fiecare element i reprezentand probabilitatea semnului cu id-ul i sa fie cel din imagine. Spre exemplu, daca pe pozitia 10 avem o probabilitate de 75%, atunci intelegem ca reteaua considera, in proportie de 75% ca semnul din imagine este cel cu id-ul 10.

5. Antrenarea retelei

Dupa ce am parcurs construirea retelei, urmeaza sa vedem cum antrenam aceasta retea. Pentru antrenamentul retelei am folosit Tensorflow si Keras, care sunt tool-uri pentru Python specializate pe folosirea retelelor neuronale. Pentru ca lucram cu imagini, am ajuns sa folosim o versiune speciala de Tensorflow, unde, pentru viteza, antrenamentul retelei se face cu puterea de procesare a placii video.

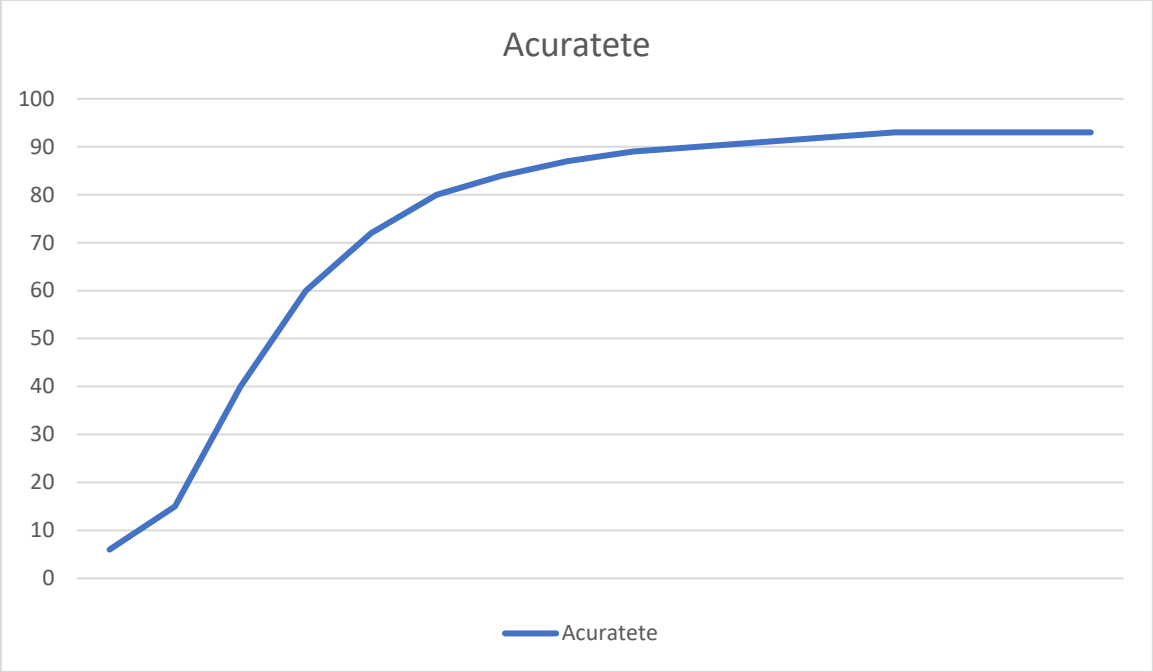
In prima faza, antrenarea retelei se facea cu ajutorul CPU-ului, iar, pentru un ciclu de 10 epochs, rularea programului dura aproximativ 26 de minute, 10 pentru a incarca imaginile si 16 pentru antrenamentul propriu-zis. Dar avand in vedere ca datele noastre sunt imagini, ar fi indicat sa folosim o unitate speciala pentru lucrul cu imagini, si anume, placa video. Tensorflow ofera un posibilitatea de a face acest lucru. Pentru a lucre eficient, am recurs la Anaconda, o platforma pentru Python, unde se pot crea diferite environment-uri, fiecare cu pachetele sale. Astfel, am creat un environment unde procesarea se face pe CPU si unul unde procesarea se face pe GPU.

Pentru a observa diferentele dintre cele doua avem urmatorul tabel:

	CPU	GPU
Timp total de executie	~26 de minute	~ 12 minute
Timp total per epoch	~78 secunde	~6 secunde
Acuratete	87.23%	92.11%

Din cate putem observa din tabelul de mai sus, antrenarea unui epoch dureaza de aproximativ 12 ori mai mult pe CPU decat pe GPU. De aici, observam ca folosirea placii video pentru antrenament este necesara pentru obtinerea unui timp decent de antrenament. De asemenea, acuratatea este mai mare in cazul antrenamentului pe GPU.

De asemenea, observam ca reteaua creste in acuratate cu fiecare epoch, insa, la un moment dat aceasta se plafoneaza. Putem observa evolutia acesteia in timp in urmatorul graphic:



6. Clasificarea semnelor de circulatie

Odata obtinut modelul antrenat, acesta trebuie testat pe semne de circulatie pe care nu le-a vazut vreodata, pentru a-i verifica corectitudinea. Acest lucru se face foarte usor cu ajutorul Keras-ului. Pentru clasificare, incarcam modelul in memorie si-i oferim ca si input o imagine cu un semn de circulatie. De asemenea, imaginea este preprocesata inainte de a fi bagata in retea, pentru a se asemana cu imaginile pe care modelul le-a vazut.

7. Concluzii si directii viitoare

Dupa implementarea acestei solutii, am dedus ca aceasta noua arie are foarte multe de oferit si foarte mult spatiu de 22ntegrat. Pentru ca ne-am axat pe circulatie si vehicule, consider ca solutia poate fi imbunatatita, astfel incat sa poata juca un rol de asistent de drum pentru orice om. Pe viitor, o recunoastere a semafoarelor ar trebui sa fie urmatorul pas. Dupa aceea, recunoasterea pietonilor, a marcajelor stradale si chiar a altor masini ar fi niste pasi evidenti in continuarea aplicatiei. Desi, multe din noile masini vin cu asemenea sisteme, consider ca solutia oferita in lucrarea prezentata este pentru cei dintre noi care inca se afla la volanul unor masini, la iesirea din fabrica, aveau drept

cel mai performant sistem sistemul audio MP3. Astfel, o implementare care sa poata fi integrata in masinile mai vechi ar putea feri oamenii de niste accidente produse din neatentie.

8. Bibliografie

1. Ford Focus-Traffic Sign Recognition:https://www.youtube.com/watch?v=kJfa2HsTtlg

2. GTRSB: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

3. Recognising Traffic Signs With 98% Accuracy Using Deep Learning: https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab

4. A Comprehensive Guide to Convolutional Neural Networks: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

5. A Beginner's Guide To Understanding Convolutional Neural Networks: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

6. Deep Learning By Michael qNielsen: http://neuralnetworksanddeeplearning.com/index.html

7. Fundamentals of Deep Learning – Activation Functions and When to Use Them? : https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/

8. The Ultimate NanoBook to understand Deep Learning based Image Classifier: https://towardsdatascience.com/https-medium-com-rishabh-grg-the-ultimate-nanobook-to-understand-deep-learning-based-image-classifier-33f43fea8327

9. Sursa: https://www.researchgate.net/figure/Toy-example-illustrating-the-drawbacks-of-max-pooling-and-average-pooling_fig2_300020038

10. Dropout: A Simple Way to Prevent Neural Networks from Overfitting: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

11. Convolutional Neural Networks (CNN): Step 3 – Flattening: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening