BABES“ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

**Pedestrian Detection**

Raport practică

Bobletec Raul, Sisteme Distribuite in Internet, 254

2020

**Abstract**

Statisticile arată că sute de oameni sunt omorâți în fiecare an în trafic. Căutând un motiv pentru aceste accidente, de obicei vina este lipsa de antenție a șoferului. În ultimii ani a fost un mare interes în dezvoltarea sistemelor de detecție a pietonilor lucru care să poată aduce o reducere a numarului de accidente.

Detectarea pietonilor reprezită unul dintre domeniile bine studiate ale algoritmilor de detecție de obiecte, pe lângă detectarea facială. În consecință multe pespective au fost verificate pe parcursul vieții sale dar soluțiile mai puțin inteligente găsite erau insuficiente pentru varietatea de factori care trebuie luați în considerare.

**Contents**

[**1.** **Introducere** 4](#_Toc39746651)

[**1.1 Ce? De ce? Cum?** 4](#_Toc39746652)

[**1.2** **Motivația și structura lucrării** 4](#_Toc39746653)

[**2.** **Problema Științifică** 5](#_Toc39746654)

[**2.1 Descrierea problemei** 5](#_Toc39746655)

[**3.** **State of the art** 6](#_Toc39746656)

[**3.1 Probleme similare studiate** 6](#_Toc39746657)

[**4.** **Implementarea propusă** 11](#_Toc39746658)

[**5.** **Concluzii** 16](#_Toc39746659)

[**Bibliografie** 17](#_Toc39746660)

# **Introducere**

## **1.1 Ce? De ce? Cum?**

* Care este problema științifică?

- Detecția pietonilor reprezintă un caz special de detecție a obiectelor fiind legate de computer vision și procesare de imagini. Din perspectiva unui algoritm inteligent, putem spune că reprezintă o problemă de clasificare, dar dintr-o perspectivă mai detailată vorbim despre o sub divizie a detecției de obiecte.

* De ce este important?

- Detecția de pietoni reprezintă una din cele mai importante funcții a unui vehicul autonom. Mai mult decât atât, detectarea trebuie să fie în același timp rapidă cât și de încredere luând în considerare toți factorii care pot aparea când conducem o mașină.

* Care este metoda de rezolvare aleasă?

- Prima dată vom antrena rețeaua neuronală cu un anumit dataset. Prima data voi începe cu dataset-uri mai mici și obiecte de baza ca apoi aplicația să poată diferenția o imagine cu pietoni față de una fără.

## **Motivația și structura lucrării**

Principala motivație a acestei lucrări este prezentarea unui algoritm inteligent ce rezolvă problema detecției de pietoni.

O altă motivație pentru această lucrare consistă în creearea unei aplicații software care să fie intuitivă și ușor de folosit. Ținta acestei lucrări este de a ajuta fabricanții de mașini pentru a face îmbunătățiri sistemelor autonome de condus și de asemenea și pe șoferi care vor fi alertați în trafic de orice pericol.

# **Problema Științifică**

## **2.1 Descrierea problemei**

Detectarea obiectelor a fost un subiect în trend în ultimii ani datorită inovației în inteligența artificială, dar și a performanței hardware. Detectarea pietonilor reprezintă o problemă canonică cu aplicații diverse, principala fiind conducerea autonomă. În ciuda cercetărilor extensive realizate pe acest topic, documentările recente arată înbunatățiri majore față de metodele vechi deși sunt tot la o mare depărtare de acuratețea unui om. Un document de cercetare popular ce a evaluat cele mai bune metode existente citează urmatorul lucru “Rezultatele indică faptul că încă mai e nevoie de o îmbunatățire de zece ori mai mare pentru a putea atinge performanțele umane.” [1]

Problema prezentată este una extrem de complexă datorită varietății mari de factori și provocări, spre exemplu înălțimea, greutatea, îmbrăcămintea dintre multimile de pietoni. Astfel, metodele convenționale și non-inteligente construite în trecut s-au găsit a fi insuficiente pentru a rezolva această problemă într-o manieră generalizată, avînd o acuratețe mare doar în cazuri și medii specifice. Ca și rezultat, acest trend a luat o întorsătură spre variantele de algoritmi inteligenți care pot generaliza mult mai usor problema și în același timp să mențină ideile din implementări specifice.

Având în vedere că acest subiect consistă mai mult pe analiza imaginilor vizuale, folosința rețelelor neuronale convoluționale cu varietatea acestora au avut un impact major în ultima perioadă. Avantajele acestor tipuri de metode provine din faptul că aceste rețele au tendința de a învăța singure ceea ce este un lucru destul de important pentru problema actuală, eliminând nevoia de a fi modificate în parte ca și în cazul algoritmilor tradiționali de clasificare de imagini. Un dezavantaț major il prezintă nevoia unui set larg de date, lucru care nu este mereu disponibil.

# **State of the art**

## **3.1 Probleme similare studiate**

**Prima lucrare** parcursă a fost Pedestrians Detection [1], o lucrare gasită pe pagina dumneavoastră de github, datele pe care s-au lucrat au fost PascalVOC2012 ce este un dataset pentru recunoașterea claselor de obiecte și conține adnotații pentru 20 de clase dar la acest proiect a fost folosită doar pentru clasa person. Imaginile au fost imparțite 50-50 pentru antrenare și validare, iar numarul total de imagini a avut puțin peste 10.000. Algoritmii AI folosiți aici au fost YOLO ce este rețea neuronală convoluțională completă , are 75 de straturi convoluționale și am vazut ca s-a folosit și tensorflow-keras. Învățarea a fost facută pe Tiny Architecture și Normal Architecture. Pentru calculare s-a folosit formula IOU care se calculeaza ca Area of Overlap/Area of union.

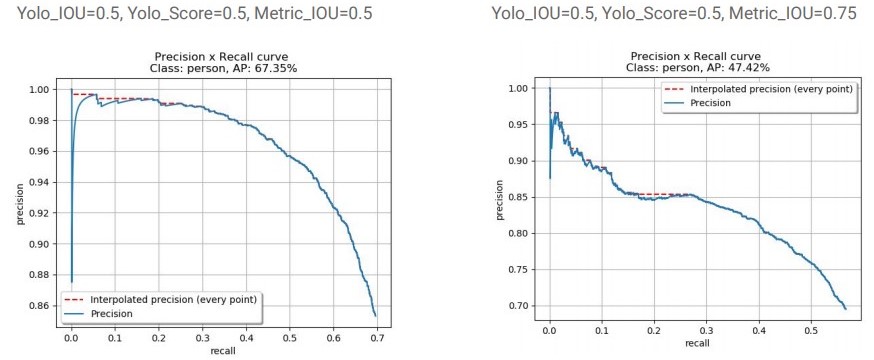


Fig. 1 : Rezultate obținute după antrenarea pe tiny architecture [1]

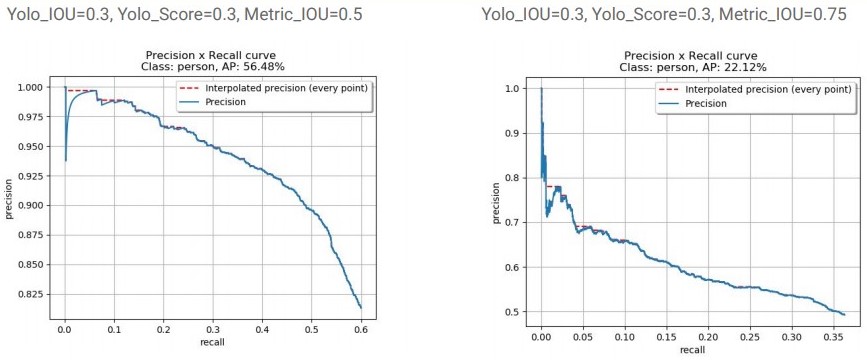
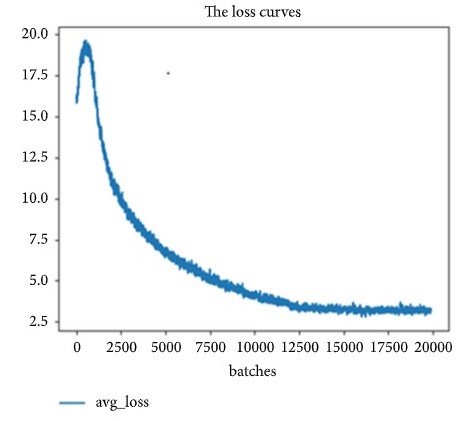


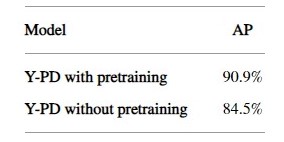
Fig. 2 : Rezultatele obținute cu normal architecture.[1]

**A doua lucrare** parcursă se numeste “An Efficient Pedestrian Detection Method Based on YOLOv2”[2] . Datele pe care s-a lucrat în acest articol sunt Inria Pedestrian Dataset ce este de obicei folosit pentru a evalua performanțele tehnicilor de detectare a persoanelor. Imaginile pozitive sunt normalizate la marimea de 64x128 pixeli cu imaginile negative brute. Setul de antrenare conține 614 imagini cu 1237 pietoni iar setul de test conține 228 imagini cu 589 pietoni.

Algoritmul de detecție folosit a fost YOLOv2, o versiune îmbunatățită a YOLO, este un model de detectare cu performanțe superioare aplicate sarcinilor generale de detectare. YOLOv2 ar putea rula la diferite dimensiuni folosind un novel, precum și tehnica de antrenre pe mai multe niveluri. Acestia au adus și unele modificări în structura și parametrii rețelei pentru a fi adaptată mai bine pentru pietoni. Rețeaua folosită are 23 de straturi convoluționale, 6 straturi de max pooling , 3 straturi de reorganizare și 1 strat de fusion.

Evaluarea a fost realizată prin prisma compromiterii dintre FPPI (false positives per image) și MR (miss rate). Din exemplele lor am aflat ca MR se calculează ca și FP/N unde FP este false positive și N este numarul total de sample-uri pozitive.





Tabel 1: Efectul pretraining-ului [2]

Fig. 3 : Curbarea loss fără pretraining [2]

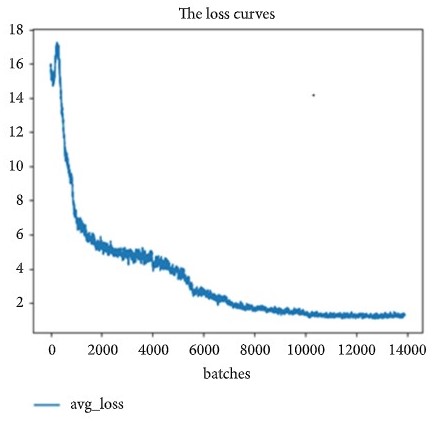
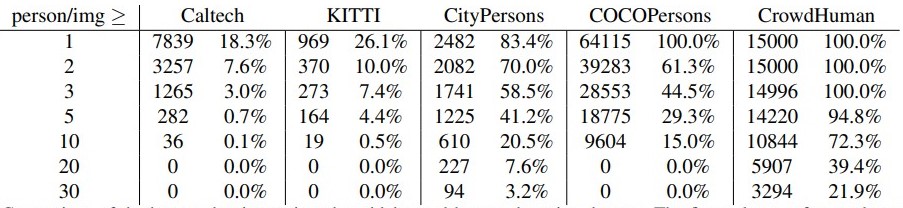


Fig. 4 : Curbarea loss cu pretraining [2]

**A treia lucrare** parcursă are titlul “CrowdHuman: A Benchmark for Detecting Human in a Crowd”[3] . Datele pe care s-au lucrat în acest articol se numesc CrowdHuman , este un dataset cu un total de 15.000 de imagini din care fac parte 340.000 de persoane și 99.000 de regiuni ignorate toate acestea fiind folosite pentru antrenare. Ca și densitate aceștia spun că sunt 22.6 persoane per imagine

Algoritmii de detectare folosiți au fost Faster R-CNN și RetinaNet amandoi bazați pe Feature Pyramid Network cu ResNet-50 și o rețea back-bone.

Pentru a evalua performanța testarii ei au folosit un server online de evaluare și au menționat că adnotările subsetului de testare nu vor fi puse la dispoziția publicului. Ni se mai spune că urmăm metrica de evaluare folosită pentru Caltech care este rata medie de pierdere a log-ului peste fals pozitive per imagine ce variază în [10−2 , 100] .



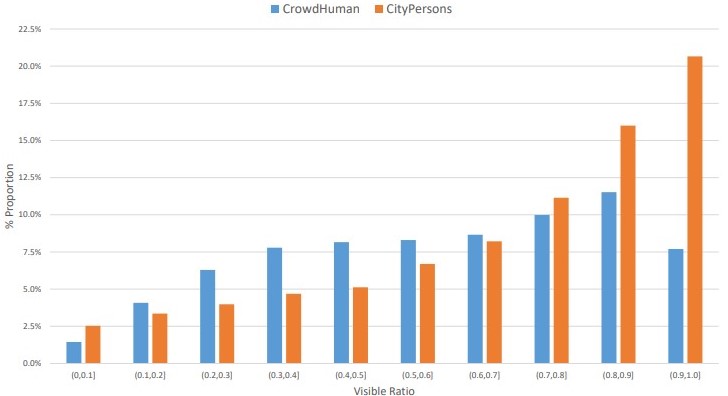
Tabel. 2: Comparația densității de oameni contra cel mai răspândit dataset de detectare umană. [3]

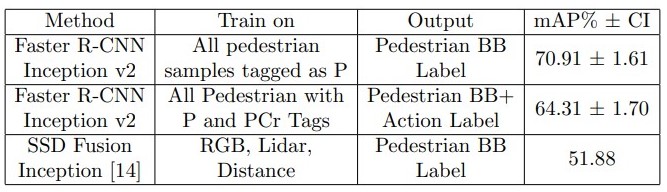
Fig. 5: Comparație între rațiile vizibile dintre Crowd Human și City Persons dataset. [3]

**A patra lucrare** parcursă este “Detection of pedestrian actions based on deep learning approach”[4]. Dataset-ul folosit în acest articol este JAAD , un dataset bazat pe trafic urban, locații variate ,mai multe ore ale zilei și diferite condiții meteo. Acest set de date furnizează bounding boxes (BB) pentru detectarea pietonilor.

Ca și algoritmi examineaza concurent pe partea de detecție folosind un detector public bazat pe Faster R-CNN. Trateaza de asemenea arhitectura Inception CNN pentru task-urile de clasificare folosind implementarea open-source TensorFlow.

Pentru învățare au folosit primele 70% date din dataset iar pentru partea de validare s-au folosit 10% din setul de învățare. S-a folosit soluția metoda de validare, care a împiedicat formarea modelului. Procesul de antrenare pe CNN a fost facut pe 200.000 de iterații folosind initial o rata de invățare de 0.0063. Au folosit wheights-urile pre-antrenate cu setul de date COCO cu loss fuction-ul de la Faster RCNN care este optimizat pentru multi-task loss fuction. Pentru setul de teste este independent față de setul de antrenament și conține ultimele 30% de date din setul JAAD.

Procesul de evaluare pentru toate modelele CNN este realizat de TensorFlow Deep Neural Network Framework.

Tabel 3: Performanța detecțiilor folosing unul sau mai multe label-uri de output. Un label reprezintă că toate sample-urile sunt tagate ca și pietoni. [4]

# **Implementarea propusă**

Ca și parte de testare am folosit mai multe rețele neuronale convoluționale și de asemenea mai multe metode de feature extracțion asupra setului de date.

Setul de date ales este Daimler Mono Pedestrian set de la Gavrila. Acesta este un set de date ce conține 4000 de imagini cu pietoni și 5000 de imagini fără pietoni, imagini obținute din video-uri și micșorate la o dimensiune de 18 x 36 pixeli.

Fig 13. Un exemplu de cum arată imaginile din dataset



**Primul program** de test a fost realizat cu aceste date pe care am aplicat algoritmul de feature extracțion HOG, după care am creat doua liste, una pentru features iar una pentru labels. După finalizarea de extragere a features-urilor acestea cât și label-urile au fost salvate în două fișiere separate de tip numpy. Aceste fișiere apoi vor fi folosite pentru antrenarea rețelei.

Histogram of oriented gradients sau mai simplu HOG [8], este un feature descriptor folosit în computer vision și procesare de imagini cu scopul de a detecta obiecte. HOG se focusează pe structura formei unui obiect și este diferit față de algoritmii de tip edge detection, deoarece în cazul acelor algoritmi ei doar identifică dacă un pixel este o margine sau nu, lucru de care este capabil și HOG. Acest lucru este realizat extrăgând gradientul și orientarea marginilor. Apoi aceste orientări sunt calculate în puncte localizate, ceea ce înseamnă că o imagine completă este împărțită în regiuni mai mici, iar pentru fiecare regiune se calculează gradientul și orientarea. Într-un final HOG va genera o Histogramă pentru fiecare regiune în parte. Aceste histograme sunt create folosind gradientul și orientarea valorilor pixelilor, de unde vine și numele de „Histogram of oriented gradients”.

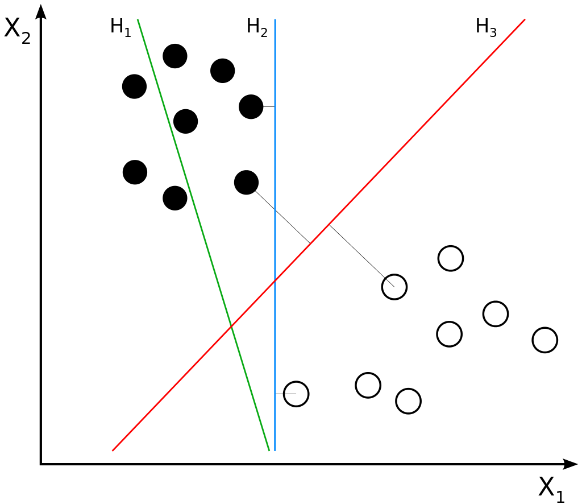
Algoritmul de învățare automată pentru acest test este SVM. Support-vector machine sau SVM constituie o tehnică eficientă de clasificare a datelor, tehnică ce presupune existența a doua seturi de date, unul de antrenare și unul pentru testare. Fiecare instanță din setul de antrenare este deja clasificată ca aparținând unei anumite clase, iar acest set de date este folosit pentru creearea unui model capabil să eticheteze instanțele din setul de test ca aparținând unei anumite clase. Un model SVM este o reprezentare a exemplelor ca și puncte în spațiu, mapate astfel încât exemplele categoriilor separate să fie împărțite printr-un decalaj clar cât mai larg.

Fig 9. Separarea hiperplanelor in SVM [6]

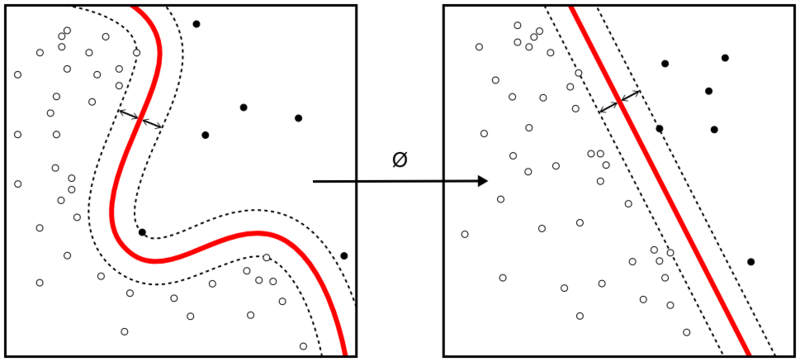
SVM pot fi folosite pentru rezolvarea diverselor probleme, precum clasificarea textelor și hypertextelor lucru ce poate reduce semnificativ nevoia unei instanțe de instruire etichetată, clasificarea de imagini este un alt topic ce poate fi acoperit de către SVM, clasificarea datelor provenite de la sateliți și recunoașterea scrisului de mână.

Fig 10. Kernel Machine-ul folosit de SVM [6]

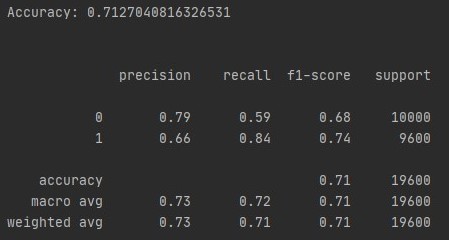
Așadar după finalizarea procesului de antrenare rețeaua a fost salvată într-un fișier separat. Pentru partea de test s-a făcut încărcarea algoritmului antrenat și ca date am folosit un set de testare, cu imagini separate față de cele folosite pentru antrenarea rețelei. Rezultatele obținute cu aceste configuratii au fost urmatoarele:

Fig 14. Rezultatele SVM cu HOG features

**Al doilea program** de test a fost realizat cu același set de date prezentat mai sus doar că de data aceasta am folosit FAST ca algoritm de feature extraction. Ca și în exemplul de mai sus după finalizarea acestui proces rezultatele au fost salvate în fișiere separate după care au fost introduse în rețea pentru antrenare.

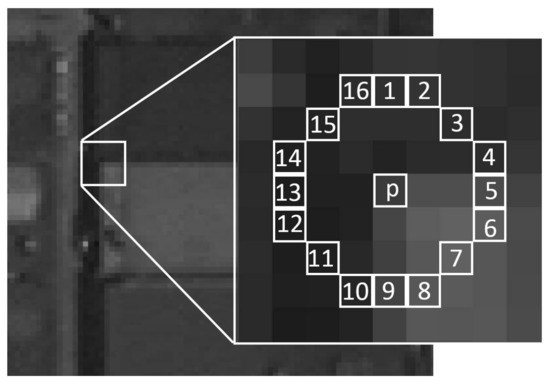
Features from accelerated segment test sau mai simplu FAST [9], este un corner detection method folosit pentru a extrage feature points care apoi să fie folosite pentru a urmări și mapa obiecte în multe task-uri ale computer vision. Un avantaj promițător al lui FAST este eficiența computațională a acestuia, cu referință la numele lui acesta este mult mai rapid decât binele cunoscute metode de feature extraction. Mai mult, când se aplică și tehnici de machine learning, se pot obține performante computaționale superioare. FAST este foarte potrivit pentru procesare video în timp real datorită performanței sale de viteză mare.

Fig 12. Pixelii folosiți de algoritmul FAST [9]

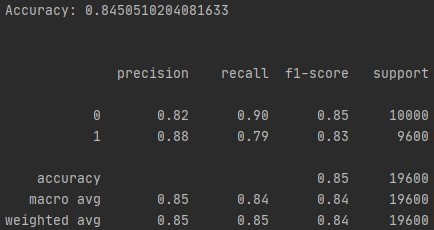
Algoritmul este aceeași folosit anterior cu aceleași configurații. De data aceasta rezultatele au fost considerabil mai bune față de datele antrenate cu HOG features. Mai jos sunt rezultatele obținute în acest test:

Fig 15. Rezultatele SVM cu FAST features

**Pentru al treilea** program de test am ales să folosesc un alt tip de rețea neuronală pentru a compara rezultatele obținute cu o extragere de features manuală și antrenarea acestora cu SVM și antrenarea datelor cu o rețea ce face automat extragerea de features.

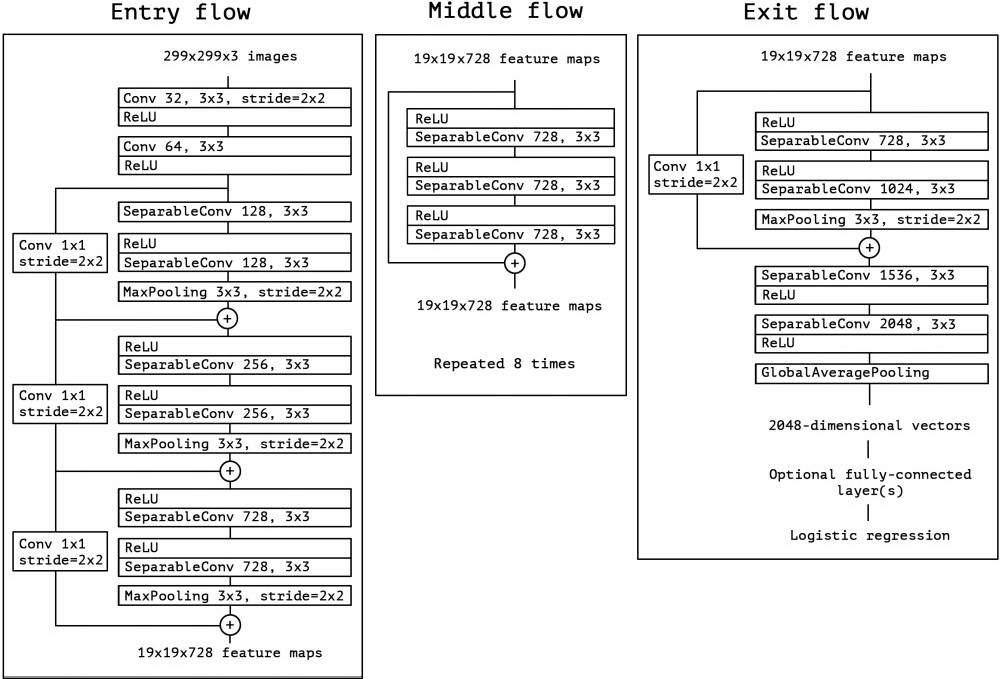
Rețeaua folosită este Xception [7], este o rețea neuronală creeată de Google, care este un model mai rafinat al rețelelor Inception, mai exact este o versiune îmbunătățită a lui Inception V3. Această arhitectură pe care ei au numit-o Xception surclasează puțin modelul Inception V3 pe dataset-ul ImageNet pentru care a fost creat Inception V3 și depășește semnificativ Inception pe un set de clasificare de date mai mare ce cuprinde 350 de milioane de date și 170 de clase. Astfel cum arhitectura Xception are același număr de parametrii ca si Inception V3, aceste îmbunătățiri în performanță nu sunt datorită unei măriri a capacității ci mai degrabă o utilizare mai eficientă a parametriilor. Deci arhitectura acestei rețele este bazată în întregime pe straturi convoluționale separabile în profunzime. Așadar există urmatoare ipoteză, aceea că maparea corelațiilor încrucișate și a celor spațiale în feature maps-urile rețelelor convoluționale pot fi decupate în întregime. Deoarece această ipoteză este o versiune mai puternică decât ipoteza arhitecturii Inception, ei au numit-o pe aceasta Xception care vine de la “Extreme Inception”. Rețeaua Xception are 36 de straturi convoluționale ce formează baza de feature extraction a rețelei. Cele 36 de straturi sunt structurate în 14 module, fiecare din ele având conexiuni reziduale în jurul lor, cu excepția primelor și ultimelor module. Pe scurt, arhitectura Xception este stivă liniară de straturi convoluționale sparabile în profunzime și cu conexiuni reziduale, lucru ce face această arhitectură să fie foarte ușor de definit și modificat.

Fig 11. Arhitectura rețelei Xception, datele merg întâi prin entry flow, apoi prin middle flow care este repetat de 8 ori iar la final ies prin exit flow. [7]

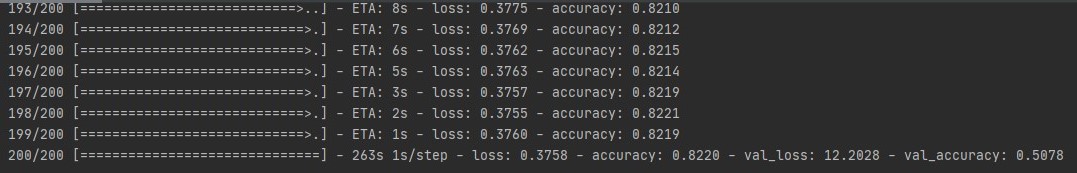
 Așadar pentru programul acesta datele au trebuit să fie modificate la o dimensiune de 71 x 71, această mărime fiind o cerință pentru datele de intrare în acest tip de rețea. După aceasta rețeaua mai aplică unele preprocesări asupra imaginilor după care le oferă rețelei pentru antrenare. Ca și un prim test am realizat o antrenare în 5 epoci cu câte 200 de pași pe fiecare epocă și 10 pași de validare. Rezultatele obținute au fost urmatoarele:

Fig 16. Rezultate obținute cu Xception

Exemplul de mai sus a fost implementat și rulat pe CPU single thread cu 9800 de imagini, 4800 cu pedestrian iar 5000 fără. Apoi am urmat și am schimbat implementarea pentru a rula pe GPU și multi thread și am mărit setul de date la 29000 de poze. După mai multe rulări cu aceste date, 5 epoci și 200 de pași pe epocă rezultatele au fost urmatoarele:

Fig 13. Rezultatele Xception cu gpu și multi thread

După finalizarea acestui proces am realizat și o parte de test a unui set de date făcut special pentru testare, set de date ce conținea 19600 de imagini care nu fac parte din setul de antrenare. Rezultatul testării a fost urmatorul:

Fig 14. Rezultate date de testare

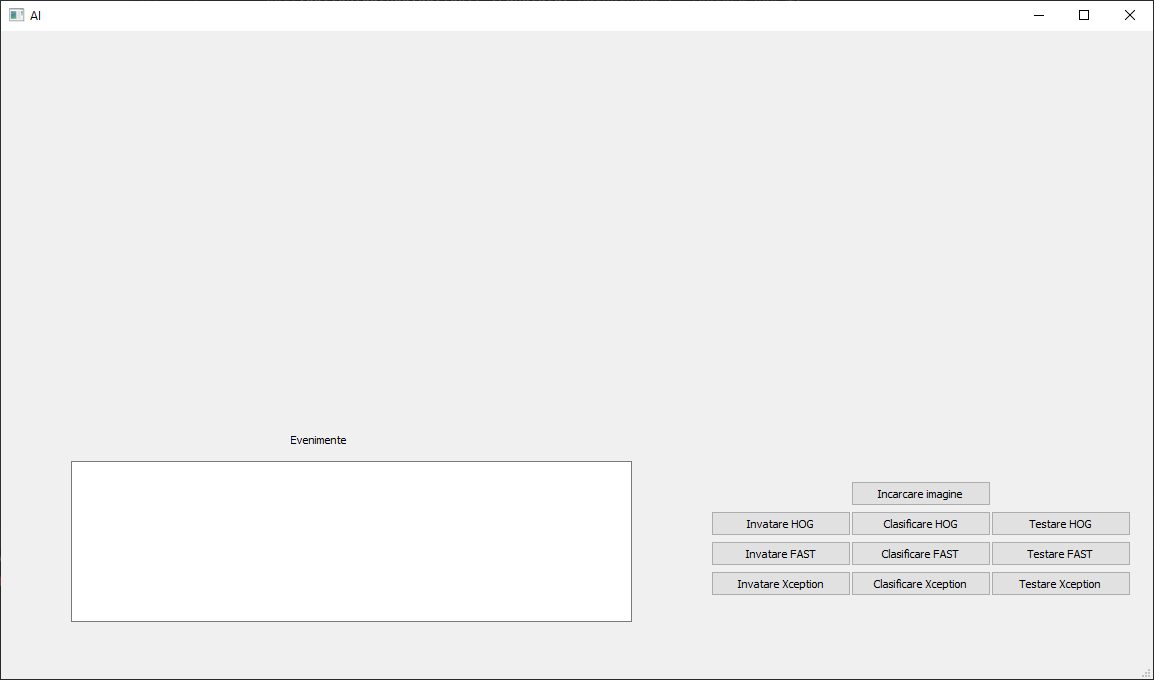
 Pentru această aplicație am realizat și o simplă interfață în PyQt pentru o mai ușoară utilizare a aplicației. Aceasta este o captură de ecran a interfeței:

Fig 15. Interfața aplicației

Aici avem câte un buton pentru fiecare acțiune ce se poate realiza atât cu algoritmul de învățare SVM cât și cu rețeaua Xception. În plus avem un buton de încărcare a unor imagini neantrenare pentru a fi trecute prin rețea și prin algoritmul SVM pentru a fi clasificate de către acestea. Clasificarea consta într-un simplu răspuns al rețelei cu ped sau non\_ped depinzând ce este în imaginea încărcată. Clasificarea se realizează în modul urmator, utilizatorul încarcă o imagine după care aceasta este pre-procesată pentru a putea fi oferită rețelei pentru clasificarea ei, după care încărcăm modelul rețelei antrenate de către noi și obținem un rezultat.

# **Concluzii**

Acest document oferă o pespectivă clară a problemei detecției de pietoni, prin compararea mai multor teste realizate pe mai multe tehnologii, pentru a demonstra că această problemă obține rezultate mai bune cu o anumită tehnologie folosită. Am ilustrat de asemenea că se poate să antrenăm o rețea și un clasificator cu un data set relativ mic folosing transfer learning și fine tuning.

Cu toate acestea, rezultatele finale sunt cum mult departe de cum ar trebui să fie pentru o aplicație folosită în viața de zi cu zi, iar acum voi enumera câteva din motivele pentru care am obținut aceste rezultate și unele îmbunătățiri ce pot fi aduse:

* În primul rând, nu am folosit imagini de rezoluție mare ci le-am modificat dimensiunea astfel încât să fie acceptată de rețea, lucru ce poate duce la pierderea calității, artefacte, blur etc. Acesta cred ca poate fi un prim pas pentru îmbunătățirea rezultatelor
* În al doilea rând, se poate folosi un dataset mult mai mare și mai specializat pentru o astfel de problemă, spre exemplu imagini din mai multe cadre sau poziții diferite.
* Alte îmbunătățiri ar putea fi detectarea de pietoni din filmări video sau chiar în timp real de pe o cameră, dar acest lucru necesită o arhitectură specială pentru această problemă și un dataset foarte mare de antrenare pentru a avea o acuratețe impecabilă.

# **Bibliografie**

[1]. Pedestrian Detection by Jurj Flaviu Andrei, Maxim Tudor, Moisi Teofana-Ionela

[2]. Liu, Z., Chen, Z., Li, Z., & Hu, W. (2018). An Efficient Pedestrian Detection Method Based on YOLOv2. Mathematical Problems in Engineering, 2018.

[3]. Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., & Sun, J. (2018). Crowdhuman: A benchmark for detecting human in a crowd. arXiv preprint arXiv:1805.00123.

[4]. Pop, D. O. (2019). Detection of pedestrian actions based on deep learning approach. Studia UBB Informatica.

[5]. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[6]. http://robotics.stanford.edu/~nilsson/MLBOOK.pdf

[7]. Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[8]. https://en.wikipedia.org/wiki/Histogram\_of\_oriented\_gradients

[9]. https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65