UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL INGINERIA SISTEMELOR

PROIECT DE DIPLOMĂ

Sistem de prelucrare a unui flux de imagini pentru detecția vizuală a unor elemente de siguranță obligatorii ale personalului uman

Martinescu Sorin-Alexandru

Coordonator științific:
Prof. univ. Dr. Ing. Andrei Hossu

BUCUREŞTI

2020

CUPRINS

1. SINOPSIS

Rolul sistemului este de a ajuta personalul ce se ocupă de supravegherea unui șantiere / unei zone de producție, în identificarea rapidă a persoanelor ce nu respecta normele de siguranță (mai exact, nu sunt echipate cu cască de protecție).

Pentru a ajunge la un rezultat satisfăcător, s-au testat mai multe metode de detecție, sistemul cu cele mai bune rezultate fiind livrat de algoritmul Faster R-CNN ce utilizează o arhitectura Inception V2.

După o analiza ulterioară a rezultatelor, s-a ajuns la următoarea concluzie: sistemul creat folosind tehnici din Deep Learning nu poate fi utilizat ca un sistem independent, mai degrabă ca o unealtă în ajutorul persoanelor responsabile de supravegherea unei zone de producție.

2. ABSTRACT

The system's purpose is to help the person who is supposed to monitor a site/production zone, in rapid identification of the person who does not respect the safety norms (more exactly, those persons don't use protection helmet).

To achieve a satisfying result were tested many methods of detection, the best system being based on the algorithm Faster R-CNN who used Inception V2.

After an analysis of the results, the conclusion was: the created system, using Deep Learning techniques, can't be used as an independent system, more like a tool.

3. MULȚUMIRI

Mulțumesc domnului Andrew Ng pentru toate informațiile livrate public prin intermediul platformei Coursera, fiind cel mai bun start în domeniul de Machine Learning/Deep Learning.

Mulțumesc domnului Evan (aka Edje Electronics) pentru informațiile livrate public prin intermediul platformelor Youtube si Github privind implementarea algoritmilor de detecție folosind Tensorflow.

Mulțumesc pentru libertatea acordată de dl. îndrumător, fără de care nu puteam explora atât de mult acest domeniu.

Mulțumesc mult lui Sabi Kun pentru răbdarea avuta.

1 INTRODUCERE

Problematica temei a plecat de la necesitatea respectării utilizării echipamentelor de protecție de către persoanele ce lucrează în medii unde aceste echipamente sunt obligatorii.

Sistemul propus, pentru a ajuta în soluționarea problematicii, poate detecta două entități: cască de protecție (cel mai popular echipament de protecție) și capul unei persoane.

Se utilizează ca entitate de identificare a persoanelor capul acestora, deoarece este ușor de identificat de algoritmii moderni și reprezintă partea protejată de casca de protecție.

Algoritmii utilizați sunt Faster R-CNN și SSD, ambele sisteme de detecție utilizând arhitectura bazată pe Inception V2.

Ulterior s-a încercat utilizarea altei arhitecturi pentru sistemul bazat pe SSD, dar rezultatul nu a putut concura cu performanțele sistemului bazat pe Faster R-CNN.

În urma rezultatelor celor doua modele, s-a ajuns la concluzia ca Faster R-CNN, oferă un model mai stabil și mai precis raportat la modelul produs de SSD ce are dificultăți considerabile în detecția celor mai simple cazuri.

1.1 Context

Rezultatul proiectul reprezintă un sistem capabil sa identifice două entități: casca de protecție și capul unei persoane.

Motivația dezvoltării unui sistem de detecție al căștilor de protecție o reprezintă importanța utilizării acestui echipament în asigurarea siguranței muncitorilor.

S-a dorit implementarea unui sistem ce poate fi utilizat de persoanele responsabile de supravegherea personalului, pentru o identificare mai rapidă și mai precisă a persoanelor ce nu respecta normele de siguranță.

S-a ales casca de protecție ca entitate ce va urma sa fie detectata, deoarece este unul dintre cele mai populare echipamente de protecție.

1.2 Problema

Sistemul de detecție propus va trebui sa asiste persoanele responsabile de monitorizarea muncitorilor pe șantiere/zone de producție (zone unde echipamentul de protecție este obligatoriu), facilitând identificarea persoanelor ce nu respecta normele de siguranță obligatorii.

Pentru ca setul de date utilizat pentru antrenarea modelelor este foarte mic, s-a ajuns la concluzia ca în stadiul actual, privind performanța de detecție, sistemele antrenate, în aceasta lucrare, nu pot funcționa fără un personal ce sa supervizeze detecția.

1.3 Obiective

Obiectivul proiectului consta în identificarea unui sistem de detecție bazat pe algoritmi complecși din Deep Learning, mai exact Faster R-CNN sau SSD, pentru a fi folosit ca unealta ajutătoare în identificarea persoanelor ce nu respecta normele de siguranța.

1.4 Structura lucrării

Analiza și specificarea cerințelor: în aceasta secțiunea va fi descrisa modul dorit de operare a platformei de detecție.

Abordări existente: în aceasta secțiune sunt prezentate abordările recente similare cu ideea sistemului de detecție propus.

Soluția propusă: în aceasta secțiune este prezentată o descriere complexă și completă a soluției propuse.

Detalii de implementare: aceasta secțiune începe cu o descriere teoretică a pieselor componente ales sistemului de detecție, urmată de descrierea implementării sistemului folosind API-ul Framework-ului Tensorflow.

2 ANALIZA ȘI SPECIFICAREA CERINȚELOR

Pornind de la problematica identificării persoanelor ce nu utilizează casca de protecție într-un mediu unde utilizarea acestui echipament este obligatorie, s-a ajuns la elaborarea primei cerințe de implementare: sistemul de detecție rezultat trebuie sa identifice toate entitățile dintr-un cadru (la un moment dat). Identificarea acestor entități trebuie făcut corect, valorile de Fals Pozitiv si Fals Negativ trebuind sa tinda la 0.

Deoarece activitatea desfășurata într-un șantier/ într-o zona de producție (în zone unde echipamentul de protecție e obligatoriu) este dinamica, este important ca sistemul de detecție sa proceseze cat mai multe cadre într-un timp cat mai mic. Pentru ca algoritmii ce au o precizie a detecției entităților dintr-un cadru foarte bună (mAP 70%+ sau un scor F1 mediu peste 0.95), necesita și un timp de procesare al cadrului mai mare, empiric, s-a dezvoltat cea de-a doua cerință a sistemului de detecție: posibilitatea analizării unui cadru provenit dintr-un flux video HD în maximum 1 secundă.

Cele două cerințe dezvoltate anterior reprezintă și metoda de selecție a algoritmului folosit în implementarea sistemului de detecție.

Pentru a facilita identificarea persoanelor ce nu utilizează casca de protecție, este necesar ca sistemul de detecție sa poată identifica componente umană. Aceasta cerința (de a identifica pe lângă casca de protecție și personalul uman), este foarte importantă, permițând crearea unui sistem, car cu o calibrare buna, ar putea funcționa autonom, ne mai fiind necesara intervenția personalului uman de supraveghere.

Funcționara autonoma poate fi realizata prin aplicarea peste outputul sistemului de detecție dezvoltat in aceasta lucrare a unei funcții/rețele neuronale care sa stabilească dacă persoanele poarta sau nu cască de protecție; însă acest task, de identificare a unei asemenea funcție, nu este acoperit in aceasta lucrare.

Un scenariu de utilizare al produsului îl reprezintă, asistarea persoanelor responsabile cu supravegherea șantierului/zonei de producție; sistemul de detecție preia fluxul video, și prin intermediul algoritmului de detecție va produce minim 1 cadru pe secundă, cadru în care entitățile ce trebuie sa fie detectate for fi înconjurate de un chenar dreptunghiular (un chenar per entitate), chenar (sau box) de culoare diferită, în funcție de entitatea identificată (exemplu: galben pentru casca și roșu pentru cap).

Outputul generat de sistem va ajuta supraveghetorul sa identifice mai ușor entitățile existente.

3 ABORDĂRI EXISTENTE

Majoritatea sistemelor existente ce se ocupă de detecția căștilor de protecție trec cu vederea componenta umană (aceste sisteme nu detectează prezenta personalului uman, se focuseaza strict pe detecția căștilor).

Cea mai robust implementare este bazată pe lucrarea [1], ce își propune dezvoltare un sistem de detecție ce identifica următoarele entități: cap ce poarta casca roșie, cap ce poarta casca alba, cap ce poarta casca albastră, cap ce poarta casca galbenă și cap ce nu poarta casca.

Lucrarea folosește un sistem de detecție bazat pe SSD, antrenat end-to-end, combinat cu un sistem RPA (atenție revers progresiva).

Rezultatele lucrării precizează că sistemul de detecție obține un scor 83.89% mAP pentru un input (cadru) de dimensiunea 512X512.

Din punct de vedere al implementării, lucrarea [1], oferă un rezultat robust, sistemul de detecție putând prelucra minimum un cadru pe secundă, și are o metoda de a evidenția componenta umană. După o analiza a setului de date utilizat în antrenare modelului[1], am ajuns la concluzia ca nu exista exemple (pentru training/test), cu casca/persoana umană, în altă poziție decât cea de stat în picioare (vertical) cu sau fără casca. Aceasta deficit poate genera în probleme de identificare sau în imposibilitatea identificării a entităților ce pot fi detectate de algoritm, un exemplu simplu îl reprezintă cazul în care o persoana este întinsă pe o suprafața orizontală, pentru ca în setul de date pentru training nu exista asemenea exemple, sunt șanse (destul de mari) ca acelei persoane sa nu i se identifice capul sau dacă poarta casca să nu se identifice nimic; aceasta problema fiind generata de diferența de distribuție a datelor în setul de date folosit pentru training comparativ cu exemplul dat.

Pentru a implementa un sistem de detecție bazat pe cerințele formulate în secțiunea anterioară, pot fi utilizați o serie de algoritmi din Deep Learning : state-of-the-art în momentul actual in Deep Learning sunt Faster R-CNN, YOLO (ultima versiune a algoritmului, YOLO V3), SSD, Retina Net, DSSD. Mai există câteva sisteme de detecție mai exotice, dar pentru problema noastră, algoritmii enumerați anterior sunt suficienți.

Din punct de vedere al preciziei detecției, Retina Net si Faster R-CNN sunt unii din cei mai robuști algoritmi de detecție.

Din punct de vedere al numărului de cadre procesat pe secunde, YOLO și SSD, într-o secundă (pe același hardware), pot procesa nu număr mult mai mare de cadre comparativ cu Retina Net si Faster R-CNN (pot procesa cel puțin de doua ori mai multe cadre pe secunde, comparație făcută între modelele clasice de YOLO și SSD și algoritmii optimizați pentru rularea pe dispozitive mobile derivați din Faster R-CNN si Retina Net).

Pentru sistemul de detecție propus s-a decis sa se facă o comparație între doua modele generate cu Faster R-CNN si SSD; altfel zis, comparația are loc între un model precis dar lent și un model mai puțin precis dar mai rapid.

Modelele generate vor fi apoi comparate, și evaluate pe baza cerințelor descrise.

Scopul lucrării în sine este identificarea unui algoritm de detecție, ce poate obține performanțe bune în detecție, în același timp fiind antrenat pe un set de date mic.

Comparat cu setul de date utilizat în lucrarea [1], setul de date dezvoltat pentru lucrarea curentă (derivat din setul de date folosit pentru lucrarea [1]), este ca dimensiune sub 10% din dimensiunea setului de date utilizat în [1].

Ca observație, deoarece modelul bazat pe SSD ce utilizează Inception V2 are probleme privind antrenarea pe un set de date mic, s-a încercat și o alta arhitectura pentru rețeaua de convoluție utilizata de sistemul de detecție SSD, rețeaua folosită fiind bazată pe ResNet 101.

Ca observație secundară, în antrenarea modelelor folosite ulterior pentru detecția de cap de om și cască de protecție, nu s-a folosit procesul de augmentare de date; deși aceasta metoda este recomandata pentru modelele de sisteme de detecție ce folosesc algoritmul SSD, s-a dorit identificarea modelului ce poate face detecții acceptabile pe un set de date mic.

Implementarea propusa in aceasta lucrare diferă de modul de identificare al entităților din lucrarea [1], și își propune să îmbunătățească setul de date pentru a preveni (pe cat de mult posibil) problema identificată în lucrarea [1].

4 SOLUȚIA PROPUSĂ

Sistemul propus este implementat, în doua variante, folosind 2 algoritmi, state-of-the-art din Deep Learning, Faster R-CNN si SSD.

Deoarece setul de date prelucrat are o dimensiune relativ mica (aproximativ 300 de imagini pentru training), pentru antrenarea celor doua modele, este folosită tehnica Transfer Learning.

Aceasta tehnica în cazul rețelelor neuronale consta în preluarea un model deja antrenat pentru un task similar; acestui model îi este eliminat stratul de ieșire (dar pot fi eliminate și straturi ascunse). După eliminari, se adaugă noi straturi, în funcție de taskul la care va fi utilizata noua rețea. Antrenarea se poate realiza în doua moduri, fie vor fi antrenate doar noile straturi adăugate (soluție recomandată pentru un set de date mic), fie va fi antrenată rețeaua integral.

Comparativ cu sistemul deja existent, descris în secțiunea anterioară, unde entitățile detectate erau modelat sub forma: cap cu casca alba, cap cu casca roșie, cap cu casca albastră, cap cu casca galbenă și cap fără casca, în cazul soluției propuse în aceasta lucrare, entitățile detectabile vor fi cap și casca.

Acest detaliu încearcă sa mimeza modul uman de identificare al obiectelor.

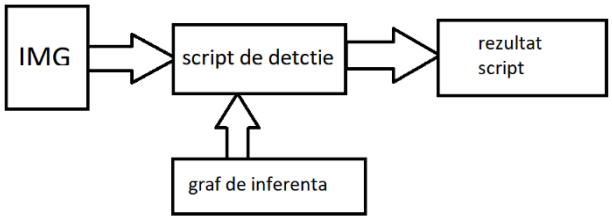

Figura 1: mode de realizare a detecției.

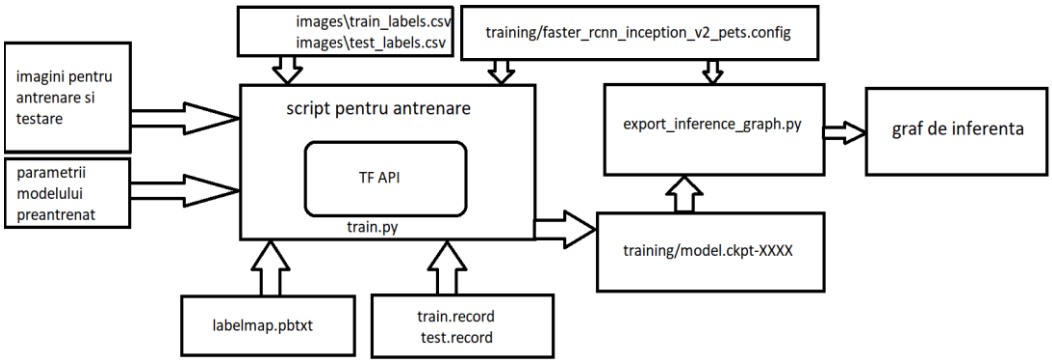

Figura 2: antrenare model si generare graf de inferență.

Pentru a putea realiza procesul de training, figura 2, utilizând API-ul oferit de Tensorflow, trebui sa fie pregătit setul de date pentru training (imaginile și libelurile corespunzătoare), trebui descărcat modelul pe care dorim să-l folosim (parametrii pre antrenați; se utilizează un model

pre antrenat deoarece setul de date utilizat pentru acest task este de dimensiuni forate mici), trebui configurat fișierul labelmap.pbtxt (unde se precizează obiectele ce urmează a fi detectate), trebuie configurat fișierul .config corespunzător modelului ales (trebuie precizat de unde se iau datele pentru training/testare, dimensiunea batch-ului și alte aspecte privind antrenarea acestuia). Odată pornit procesul de training, acesta durează până când s-a obținut un rezultat corespunzător; când s-a obținut un rezultat bun, procesul de training este oprit.

Rezultatul trainingului este salvat pe parcursul procesului de training in formatul model.ckpt-XXXX, unde XXXX reprezintă valoare pasul la care s-a obținut o noua ce mai mica valoare pentru funcția totala de pierdere.

Având modelul antrenat, folosind cel mai bun rezultat, se generează graful de inferența.

Graful de inferența va fi folosit mai departe pentru realizarea de detecții.

Detecția se poate realiza sub următoarea formă, figura 1: se da o imagine ca input unui script (conceput pentru a realiza detecția); acest script utilizează graful de inferența și pe baza acestuia realizează detecția; outputul scriptului depinde de scopul acestuia, poate fi pur si simplu imaginea de input peste care s-au aplicat o serie de boxuri unde exista obiectele (boxuri însoțite de categoria obiectului identificat), sau poate fi pur și simplu un fișier cu valori numerice ce reprezintă coordonatele, dimensiunile și categoriile boxurilor detectate.

5 DETALII DE IMPLEMENTARE

Înainte de a implementa modelele, am analizat Framework-urile existente pentru acest task. Cele mai populare fiind Tensorflow (soluția celor de la Google) și Gluon CV (soluția celor de la Amazon, bazată pe MxNet).

Deoarece o importanță deosebită în implementare o are documentația Framework-ului, s-a utilizat ca Framework de implementare al modelelor de detecție Tensorflow, deoarece dispunea de o documentație și o comunitate de dezvoltatori mai bine definite.

5.1 CNN

Pentru a putea înțelege modul în care sistemele de detecție operează, lucrarea începe analiza fiecărei componente în parte.

CNN reprezintă componenta ce detectează caracteristici într-o imagine, o alternativa modernă la soluții realizate prin crearea de filtre manual, cum ar fi HOG (histograme pentru gradienți orientați).

Inputul unui CNN este o imagine / cadru, de o dimensiune H x W x C (unde H este înălțimea, W este lățimea, iar C reprezintă adâncimea sau numărul de canale; exemplu pentru o imagine HD RGB, dimensiunea inputului va fi 1280 x 720 x 3, 3 vine de la numărul de canale RGB).

Outputul CNN-ului este dependent de utilizarea rețelei, acesta putând avea diferite forme. Cel mai utilizata forma de output este vectorul; utilizarea are loc în procesele de clasificare.

Un CNN poate avea în structura sa următoarele componente:

- strat de convoluție (obligatoriu)
- strat de pooling (nu este obligatoriu, dar utilizarea acestuia este recomandată)
- strat dens (sau fully connected)
- strat de aplatizare

Dintre aceste straturi cel obligatoriu (fără de care nu s-ar mai numi Convolutional NN) este startul de convoluție.

Stratul de pooling este opțional, dar pentru rezultate mult mai bune se recomanda utilizarea acestuia. În general se face o alternanță între straturile de convoluție și cele de pooling.

Pentru a înțelege mai bine ce scop are stratul de convoluție, trebuie mai întâi înțeleasă operația de convoluție.

Operația de convoluție (în Deep Learning, notată cu '*') se realizează între un input (de obicei imagine / outputul unui strat de convoluție/pooling) de dimensiunea H x W x C (unde H reprezintă înălțimea, W reprezintă lățimea iar C reprezintă adâncimea sau numărul de canale în cazul imaginii) și un Kernel de dimensiunea Hf x Wf x C (unde Hf reprezintă înălțimea kernelului, Wf

reprezintă lățimea și C numărul de canale). Se observa ca numărul de canale pentru inputul operației de convoluție și filtrul (kernelul) este același.

Pentru a ilustra operația de convoluție vom lua cel mai simplu caz, în care C = 1, și vom considera ca avem ca input matrice I și ca filtru, filtrul F:

I = [1 2 3 4; 4 5 6 8; 9 10 11 12] (o imagine 3 x 4 x 1)

F = [1 0; 0 1] (un filtru 2 x 2 x 1)

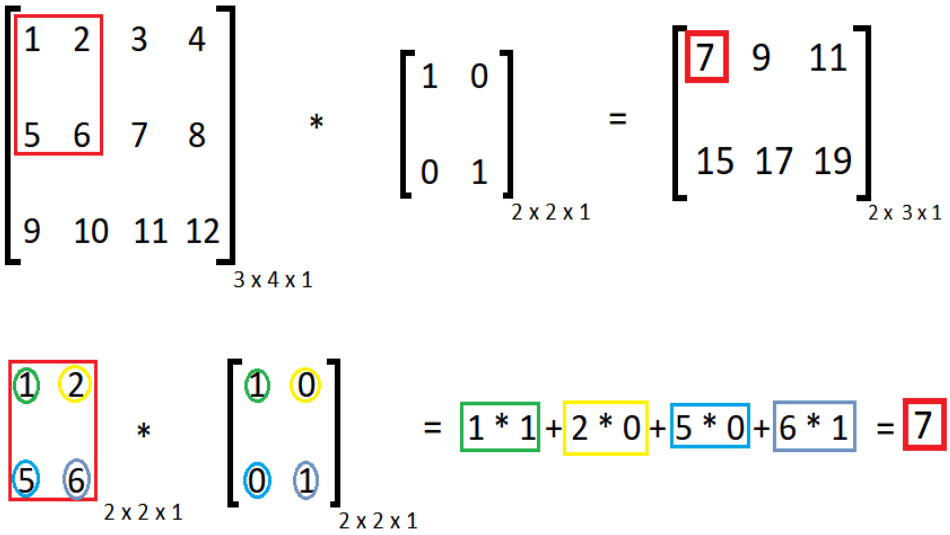

Figura 3: exemplu operație simplă de convoluție, unde C = 1.

Operația de convoluție începe prin suprapunerea filtrului peste colțul de sus (stânga) al imaginii. După realizarea convoluției între kernel și porțiunea din imagine ce corespunde dimensiunilor kernelului, filtrul se deplasează la dreapta o poziție (o valoare; se mai spune care are pasul sau stride egal cu 1) și continua acest proces pana ajunge în marginea din dreapta. După ce a ajuns în partea din dreapta și a realizat convoluția cu porțiunea din imagine ce corespunde dimensiunilor kernelului, filtrul se mută din nou în partea din stânga, dar de data aceasta se regăsește cu o poziție (o valoare mai jos) mai jos fata de poziția inițială. Acest proces continua pana filtrul ajunge în partea dreapta jos a imaginii.

Rezultatul fiecărei operații de convoluție dintre filtru își o porțiune din imagine ce corespunde dimensiunilor kernelului sunt salvate într-o matrice. Acea matrice reprezintă rezultatul procesului de convoluție. Elementele sunt salvate în matrice pe parcurs ce se realizează convoluția, de la stânga la dreapta de sus în jos.

Pasul sau stride reprezintă pasul filtrului de convoluție peste imagine.

Valoarea implicită a pasului de deplasare este 1, aceasta afirmație se poate interpreta în domeniul Deep Learning ca stride egal cu 1.

Padding reprezintă o margine de valori de zero ce se adaugă pe marginile imaginii (înconjurând imaginea). Scopul padding-ului este de a evita pierderile informaționale din marginile unei imagini, prin aplicarea unui filtru convoluțiunile.

Pentru cazul in care sunt mai multe canale, rezultatul convoluției va avea un singur canal.

Se va realiza operația de convoluție independent pe fiecare canal, iar la final rezultatele obținute pentru fiecare canal se vor aduna.

Înțelegând conceptual operația de convoluție, poate fi explicat stratul de convoluție.

Un stratul de convoluție are un input și un output.

Inputul unui strat de convoluție este o imagine sau outputul altui strat de convoluție/pooling.

Stagiul intermediar al unui strat de convoluție are următoarea formă: daca inputul are forma H x W x C, filtrele au forma Hf x Wf x C (este setat un padding P si un stride S) și se aplica Cf filtre, atunci outputul va avea forma: [(H - Hf + 2P) / S + 1] x [(W - Wf + 2P) / S + 1] x Cf.

Mai multe detalii privind o implementare matematica a acestui strat poate fi regăsită în [12].

Se observa ca numărul de canale ale outputului corespunde cu numărul de filtre; acest rezultat se obține prin suprapunerea rezultatelor celor Cf filtre.

După obținerea stagiului intermediar (prin convoluție cu filtrele), peste acesta se aplica o funcție de activare, obținându-se outputul final al stratului de convoluție.

Cea mai populară funcție de activare folosită la momentul actual pentru straturile ascunse în rețelele convoluție este ReLU.

Ca observație, aplicarea unei funcții de activare peste stagiul intermediar nu modifica forma acestuia (structura matriceala).

Stratul de aplatizare este important deoarece (chiar dacă nu aplica nici o funcție asupra datelor de input), acesta schimba forma datelor pentru a fi compatibile cu straturile dense (mai exact cu formatul unei rețele neuronale artificiale sau pe scurt, unei rețele neuronale).

Stratul de pooling este util în reducerea dimensiunii outputului stratelor convoluțiunile.

Reducerea se realizează pentru dimensiunile H și W.

Cele mai comune straturi de pooling sunt stratul de max pooling și stratul average pooling.

In practica cel mai folosit strat pentru aplicarea operației de pooling este stratul max pooling.

Stratul de pooling funcționează pe un principiu similar cu stratul de convoluție cu următoarele diferite:

1. Stratul de pooling păstrează numărul de canale.
2. Stratul de pooling nu utilizează un filtru, în sine, ci folosește o fereastra de dimensiunea unui filtru (dimensiunea ferestrei fiind precizată ca parametru, cea mai populară dimensiune fiind 2 x 2);la fel ca la stratul de convoluție, aceasta fereastra se deplasează peste input și în funcție de tipul stratului de pooling, rezultatul suprapunerii ferestrei cu

o parte a inputului stratului poate fi valoarea maximă a acelei zone (pentru max pooling) sau valoarea medie a acelei zone (pentru avg pooling).

3. Stratul de pooling nu folosește funcții de activare.

Stratul dens este un strat standard, cea mai largă utilizare fiind în structura unei rețele neuronale artificiale.

Acestea pot utiliza diverse funcții de activare, cea mai populară funcție de activare pentru straturile dense ascunse fiind ReLU.

Stratul de output este unul versatil.

Ca structura, păstrează simplitatea stratului dens, cu singura observație ca acesta poate utiliza diverse funcții de activare, alegerea acestora depinzând de tipul de problemă pe care rețeaua, din care acest strat face parte, încerca sa o rezolve.

Dacă se dorește ca rețeaua din care acest strat face parte să facă operația de clasificare, se pot aplica doua funcții de activare, softmax în cazul în care se dorește sa se realizeze clasificarea unei entități când avem mai multe clase (numărul claselor > 2), sau sigmoid când sa realizează clasificare cu numărul de clase egal cu 2.

Cele doua funcții de activare precizate anterior sunt cele mai preferate pentru taskul precizat, dar mai exista și alte funcții (alegerea unei funcții de activare depinzând și de setul de date).

Mai exista un caz, în care nu se aplica nici o funcție de activare peste acest strat.

Aceasta soluție se folosește când se vrea a s realiza operația de regresie.

5.2 INCEPTION V2

Începând cu anul 2012, rețelele convoluțiunile au început fi aplicate pentru o serie de taskuri diverse de Computer Vision, cum ar fi detecția de obiecte [2], segmentarea de obiecte[3], urmărirea obiectivelor[4] și multe altele.

Cea mai populară arhitectura, în anul 2012, este AlexNet [5] ce a câștigat competiția ImageNet [6] din acel an.

Observând potențialul rețelelor convoluțiunile, cercetători au început sa dezvolte noi arhitecturi.

Începând cu anul 2014 calitatea noilor arhitecturi s-a îmbunătățit considerabil, cele mai populare arhitecturi fiind VGGNet [7] și GoogLeNet [8].

Aceste îmbunătățiri în arhitectura rețelelor a permis implicat și o serie de îmbunătățiri în task-urile de Computer Vision ce foloseau rețele convoluțiunile și chiar, noile modele, au permis utilizarea rețelelor convoluțiunile pentru taskuri noi de Computer Vision, cum ar fi propunerea de zone în detecție [9], unde rețelele mai vechi cum ar fi AlexNet obțineau rezultate inferioare față de soluțiile proiectate de ingineri.

Rețeaua Inception este o arhitectura speciala și foarte performantă.

Privind numărul de parametrii este inter GoogLeNet și VGGNet, având mai mulți param ca GoogleLeNet dar mai puțini ca VGGNet.

Un dezavantaj al rețelei Inception a fost dificultatea de a realiza modificări în rețea, o scalare proastă poate duce la pierderea avantajelor rețelei.

Înainte de intra în arhitectura rețelei Inception, autorii lucrării de cercetare [10] au propus 4 principii importante (din punctul lor de vedere) privind o implementare adecvata.

Chiar dacă aceste principii sunt speculative și au nevoie de mai mult suport experimental, abaterile de la ele au dus la scăderea calității rețelei.

Aceste patru principii au se rezumă astfel:

1. A se încearcă eliminarea bottleneck-urilor arhitecturale, dimensiunea reprezentării ar trebui sa scadă lin pornind de la stratul de input pana la stratul de output, unde outputul este reprezentarea folosită pentru taskul de Computer Vision.
2. Utilizarea unei reprezentări de dimensiune mai mare este mai ușor de a fi prelucrata de rețea, ajuta la o mai buna înțelegere a caracteristicilor învățate de filtre, iar rețeaua se va antrena mai rapid.
3. Se poate realiza agregare spațială, fără pierderi sau cu pierderi minime în capacitatea de reprezentare a rețelei, peste caracteristicile de dimensiune mica.
4. Trebuie găsit un echilibru între lățimea rețelei și grosimea acesteia.

Sunt recomandate a fi utilizate doar în situații mai ambigue, chiar dacă utilizarea acestora are sens.

Odată cu folosirea unor filtre de dimensiune din ce în ce mai mare și puterea computațională va crește, un exemplu simplu este cazul în care se utilizează un filtru 5 x 5 comparativ cu un filtru 3 x 3 [10]; printr-un simplu calcul se constata ca un filtru 5 x 5 este de 2.78 ori mai scump din punct de vedere computațional comparat cu un filtru 3 x 3.

Autorii lucrării [10] au încercat exploatarea proprietății de translație a invarianței (un din proprietățile de bază ale unei rețele convoluțiunile) și au ajuns la concluzia ca pot înlocui în filtru de dimensiuni 5 x 5 cu doua filtre de convoluție de dimensiuni 3 x 3.

După ce autorii lucrării [10] au observat ca pot înlocui un filtru 5 x 5 cu doua 3 x 3 s-au gândit, ce avantaje ar apărea din înlocuirea filtrelor 3 x 3 cu doua filtre 2 x 2. Au ajuns la concluzia ca în urma acestei înlocuiri, ar realiza cu 11% mai puține operații.

Autorii au mers mai departe și au încercat utilizarea de filtre asimetrice, implementarea fiind de forma: se aplica mai întâi un filtru n x 1 și apoi încă un filtru 1 x n.

Prin aceasta metoda, pentru înlocuirea unui filtru 3 x 3 cu doua filtre, un filtru 3 x 1 și unu filtru 1 x 3, au realizat cu 33% mai puține operații, deci s-a obținut un câștig computațional mult mai mare comparativ cu metoda în care un filtru 3x3 era înlocuit de doua filtre 2 x 2.

Autorii au mai ajuns la concluzia ca, în practică, aceste filtre nu lucrează foarte bine pentru laserele inițiale ale rețelei; pentru rezultate bune recomanda ca n = 7.

O componenta arhitecturala interesanta o reprezintă clasificatorii auxiliari, introdusă de [8].

În [10], s-a ajuns la concluzia, comparând doua rețele, una ce utiliza clasificatori auxiliari și alta ce nu utiliza, ca pană cele doua rețele nu depășesc o acuratețe mare, cele doua in training performa la fel.

Odată de ating o acuratețe mare, rețeaua ce utilizează clasificatorii auxiliari depășește în performanța rețeaua ce nu utilizează aceste componente arhitecturale.

În [11] este precizat ca aceste componente permit obținerea unei mai bune convergente și a unui proces de învățare mai stabil.

Un alt avantaj ar fi combatere problemei de vanishing gradient pentru rețelele foarte adânci.

În arhitecturile clasice, pentru a reduce dimensiunea pentru harta de caracteristici, se aplica un strat de pooling (maximum sau average); aceasta operație, însă, implica un bottleneck.

Pentru a elimina bottlenek-ul, autorii lucrării [10] au propus următoarea metoda de reducere dimensională pentru harții de caracteristici, figura 4.

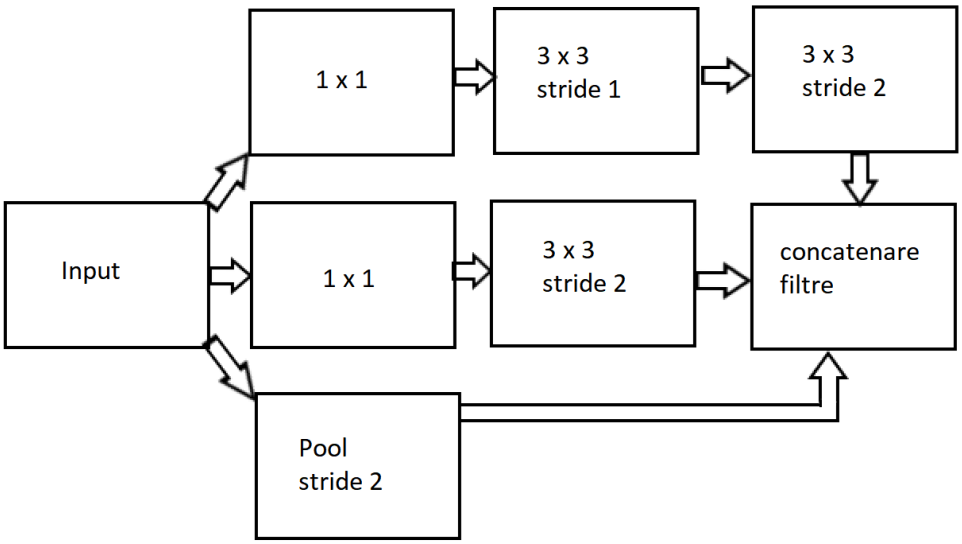

Figura 4: reducere a harții de caracteristici.

Inception V2 reprezintă o arhitectura puternica ce îmbina componentele și proprietățile enunțate anterior.

Costul computațional al acestei rețele este aproximativ de 2.5 ori mai mare comparat cu GoogLeNet; astfel obține o eficient computațională mult mai bună comparativ cu VGGNet.

Arhitectura rețelei cuprinde 42 de straturi.

5.3 FAMILIA R-CNN

5.3.1 R-CNN

Autorii [13] își propun implementarea unui algoritm de detectie de obiecte ce sa utilizeze o rețea convoluțiunile, acest algoritm urmând să aibă performanțe pe PASCAL VOC comparabile cu algoritmii ce utilizează caracteristici HOG.

Problema se poate împarți în două etape:

1. demitizarea obiectelor într-o imagine
2. clasificarea acestor obiecte.

Exemplu: sa zicem ca avem într-o imagine 3 obiecte, algoritmul le va identifica unde se afla în imagine și va spune spre exemplu ca obiectul 1 este o pisică, obiectul 2 este tot o pisică iar al treilea obiect este un câine.

Un mod de a localiza obiectele într-o imagine este tratând problema ca pe una de regresie.

Însă în practică, sa dovedit ca un model ce tratează problema de detecție ca pe una de regresie [14], nu se comportă atât de bine, obținând în VOC 2007 un mAP de 30.5%, comparat cu mAP-ul de 58.5 % obținut de algoritmul final propus de aceasta lucrare [13].

O alta alternativa în localizarea obiectelor într-o imagine este utilizarea unor CNN-uri cu metoda sliding window. Nu s-a utilizat aceasta metoda fiind o provocare localizarea precisă a obiectelor.

În cele din urma autorii [13] au hotărât, pentru a rezolva problema localizării, sa utilizeze recunoaștere utilizând regiuni [15], o paradigma ce fusese deja cu succes implementată pentru taskuri de detecție de obiecte [16].

De aici provine și numele algoritmului R-CNN (combinarea a paradigmei recunoaștere utilizând regiuni cu o rețea convoluțiunile).

Algoritmul preia o imagine, pentru aceasta propune în jur de 2000 de regiuni, pentru fiecare regiune este aplicată tehnica de potrivire prin deformare (sau affine image warping) pentru a obține un input fix pentru rețeaua convoluțiunile folosită, rezultatul CNN-ului fiind preluat de mai multe SVM-uri pentru a clasifica fiecare regine.

Sistemul de detecție propus are trei componente:

1. primul modul are rolul de a propune regiunile, un set de candidați pentru sistemul nostru de detecție.
2. al doilea modul este reprezentat de o rețea convoluțiunile ce are rolul de a extrage un vector de caracteristici (de dimensiune fixa) pentru fiecare regiune propusă de primul modul
3. al treilea modul este reprezentat de un set de SVM-uri liniare(un SVM specific pentru fiecare clasa). Odată ce caracteristicile unei regiuni sunt extrase de CNN, clasificarea va fi realizata de SVM-uri, existând un SVM liniar optimizat per clasa

Pentru a antrena acest algoritm, s-a propus o metodă în doi pași:

1. Pasul 1: Antrenarea rețelei convoluțiunile pe un datase auxiliar de dimensiuni foarte mari.
2. Pasul 2: Constă în înlăturarea stratului de output folosit pentru pre antrenare și folosirea rețelei pre antrenate pentru scopul dorit, în cazul de fata, integrarea rețele în sistemul de detecție R-CNN.

Ca observație minora, s-a ales un IoV >= 0.5.

Făcând o analiză a erori sistemului de detecție, s-a implementat o metoda pentru a reduce eroarea de localizare a algoritmului.

Aceasta metoda este inspirată de metoda folosită în algoritmul DPM[17] și consta în prezicerea unui nou box pentru fiecare detecție utilizând un box regresor (un box regresor specializat pentru fiecare clasa).

5.3.2 Fast R-CNN

Chiar dacă R-CNN obține niște performanțe remarcabile pentru taskul de detecție de obiecte, acest sistem de detecție are câteva dezavantaje:

1. Pentru a se realiza detecția sistemul este antrenat în mai multe etape: etapa în care este pregătită rețeaua convoluțiunile, urmată de etapa de pregătire a clasificatorilor (a SVM-urilor).
   La final, ca ultima etapă sunt antrenați regresorii pentru a prezice boxurile (localizarea obiectelor în imagine).
2. Trainingul ocupă foarte multă stocare și necesita o putere computațională mare.
3. Sistemului de detecție R-CNN, ce utilizează ca rețea convoluțiunile bazată pe arhitectura VGG16 , îi are nevoie de 47 de secunde pentru a procesa o imagine/cadru (pe un singur GPU Nvidia K40).

R-CNN este un sistem de detecție lent deoarece pentru fiecare regiune propusă aplica rețeaua convoluțională.

Soluția propusă în [18] se inspiră din lucrarea ce propune sistemul SPPnets [19].

SPPnets realizează o harta de caracteristici pentru întreaga imagine, iar apoi pentru fiecare regiune propusă se extrage din harta de caracteristici un vector de caracteristici care mai departe sa fie utilizat pentru clasificare.

Extragerea vectorului de caracteristici pentru o anumită regiune propusa se realiza prin aplicarea unui max pooling în acea regiune, obținându-se un output de dimensiune fixă (exemplu 5 x 5).

Aceasta metoda urmarea sa facă sistemul de detecție R-CNN mult mai rapid.

La testarea sistemului de detecție SSPnet , acesta este de 10 pana la 100 ori mai rapid comparativ cu sistemul de detecție R-CNN. De asemenea și timpul pentru training era redus de aproximativ 3 ori pentru sistemul de detecție SSPnet.

SSPnet și R-CNN păstrează dezavantajul de training în mai multe etape.

În [18] este propusă o noua metoda, ce sa rezolve dezavantajele sistemelor de detecție descrise anterior.

Aceasta metoda își propune să aducă îmbunătățiri, atât în viteza la testare cât și la acuratețea detecției, și se numește Fast R-CNN.

Printre avantajele aduse de aceasta noua tehnica se numără antrenarea sistemului într-o singura etapa, îmbunătățiri în viteza și acuratețea, în etapa de training pot fi update toate straturile rețelei convoluționale.

În mare sistemul de detecție Fast R-CNN are următoare funcționare:

1. Pentru imaginea (inputul) peste care aplica o serie de straturi de convoluție și de pooling pentru a crea o harta de caracteristici.
2. Pentru fiecare regiune propusă, un strat RoI (un strat inspirat de sistemul SSPnet) extrage un vector de caracteristici de lungime fixă.
3. Acest vector de caracteristici este preluat în paralele de doua secvențe de straturi dense, o secvența ce are ca output un strat cu funcția de activare softmax pentru a prezice ce obiect se afla în acea regiune (K + 1 clase de obiecte, unde 1 vinde de la background), iar cealaltă secvența se ocupă de regresor ce pentru un vector de caracteristici dat returnează 4 valori pentru fiecare din cele K clase (cele 4 valori reprezintă coordonatele boxului în care se afla obiectul detectat).

Stratul RoI (region of interest) utilizează la baza sa un strat de max pooling ce preia și transforma orice regiune data (din harta de caracteristici) într-o harta de caracteristici de dimensiuni H x W (unde H si W sunt hiper parametrii stratului RoI).

Acest strat este similar cu stratul de pooling, efectuând operațiile de mai sus pentru fiecare canal în parte, și este inspirat de startul spatial pyramid pooling folosit în sistemul SSPnet [19].

Pentru a folosi o rețea preantrenata cu Fast R-CNN trebuie caca:

1. Ultimul strat de pooling sa fie înlocuit cu un strat RoI, cu precizarea ca hiper parametrii H și W ai stratului RoI trebuie sa fie compatibil cu primul strat dens al rețelei.
2. Ultimul strat dens și stratul cu funcția de activare softmax (în cazul VGG) vor fi înlocuite cu cele doua secvențe de straturi (secvența pentru clasificare și secvența pentru regresorul ce generează boxurile).
3. Rețeaua este modificata sa ia ca input o serie de imagini și o serie de RoI-uri corespunzătoare acelor imagini.

Pentru a putea utiliza cele doua seturi de straturi pentru output, Fast R-CNN utilizează funcția multi-task loss.

Ca observație, prin plecarea de SVD trunchiat pe straturile dense, acestea sunt comprimate, accelerând procesul de detecție.

Folosind SVD se poate reduce timpul de procesare a unei imagini cu minimum 30%, acest trade of in calitatea detecției realizate de sistem fiind slab penalizat, pierderile de acurateț fiind foarte mici. Aceste pierderi pot fi estimate ca fiind 0.3% din mAP-ul obținut de sistem fără aplicarea de

SVD.
În [18] a fost făcut o comparatie intre R-CNN, SPPnet si Fast R-CNN, toate aceste sisteme folosind arhitectura VGG16.

Fast R-CNN fără SVD procesează imagini de 146 de ori mai rapid, comparativ cu R-CNN, iar Fast RCNN cu SVD procesează imagini de 213 de ori mai rapid comparativ cu R-CNN.

Încă un aspect important, timpul luat pentru training este redus de la 84 de ore la 9.5 ore (o reducere de 9 ori).

Comparativ cu SPPnet, Fast R-CNN procesează imagini de 7 ori mai rapid fără SVD și de 10 ori mai rapid cu SVD.

Timpul de training este de 2.7 ori mai mic pentru Fast R-CNN, comparație făcută față de sistemul SSPnet.

Un avantaj major comparativ cu R-CNN este faptul ca Fast R-CNN nu are nevoie de spațiu pentru a calcula caracteristici, problema ce făcea ca R-CNN sa consume cantități enorme de spațiu (sute de GB de spațiu de stocare pe disk).

5.3.3 Faster R-CNN

[20] Este un sistem de detecție format din doua module, primul modul ocupându-se de propunerea regiunilor iar cel de-al doilea modul este sistemul de detecție Fast R-CNN [18].

În [20] s-a hotărât implementarea unui modul cu scopul de a propune regiuni deoarece vechile metode folosite de sisteme, printre care cea mai populară metoda numită Selective Search [16], introduceau un bottleneck în sistemul de detecție.

În general aceste sisteme de propunere de regiuni au fost concepute pentru a lucra pe CPU, comparativ cu restul algoritmului de detecție Fast R-CNN ce profita de puterea computațională pusă la dispoziție de GPU.

Astfel, în [20] pentru rezolvarea acestei probleme, modulul ce ar trebui sa propună regiuni este o rețea întreg convoluțiunea, numita RPN (Region Proposal Network).

Pe scurt vorbind, în Faster R-CNN, RPN îi indica sistemului de detecție Fast RCNN unde se afla obiectele.

Aceasta rețea, RPN, comparativ cu modulele mai vechi ce se ocupau de propunerea de regiuni este aproximativ fără cost, din punct de vedere computațional; aceasta caracteristică a permis rezolvarea bottleneck-ului întâlnit la algoritmul Fast R-CNN.

Cu un mAP vizibil mai mare (cu minim 2%), Faster R-CNN are un timp de procesare al imaginii mult mai bun comparativ cu sistemul Fast R-CNN (ce utiliza Selective Search).

Pe hardware-ul utilizat în lucrarea [20], Selective Search procesează o imagine în aproximativ 1.5 secunde, comparativ cu RPN ce are nevoie de doar 10 ms (cu observația ca în Faster R-CNN, RPN si Fast R-CNN au câteva caracteristici convoluțiunile / straturi comune).

Revenind la sistemul de detecție Fast R-CNN ce utilizează Selective Search, doar etapa de Fast RCNN (în cazul în care este aplicat SVD) procesează un input în 223 ms.

Faster R-CNN are nevoie de doar 198 ms, atât pentru propunerea de regiuni, FPN, cat și pentru etapa de detectare, Fast R-CNN.

Aceasta performantă este rezultatul implementării sistemului Faster R-CNN și a faptului ca modulul FPN și modulul Fast R-CNN împart o serie de caracteristici convoluțiunile.

Un alt aspect important ce permite asemeni rezultate este utilizarea de 300 de regiuni, în cazul sistemului Faster R-CNN comparativ cu 2000 în cazul modulului Selective Search și Fast R-CNN.

În comparația anterioara arhitectura utilizata pentru rețeaua convoluțiunea în cadrul sistemului / modulului Fast R-CNN a fost VGG.

Ca un rezumat pentru cazul în care este utilizata arhitectura VGG, Fast R-CNN cu Selective Search atinge performanța de 0.5 cadre pe secundă pe când Faster R-CNN atinge performata de 5 cadre pe secundă.

Folosind arhitectura ZF [21], Faster R-CNN atinge performanța de procesare de 17 cadre pe secundă.

Sistemul ce utilizează ZF obține pe PASCAL VOC 2007 un mAP de 58.7% (pentru implementarea în doua etape).

Fast R-CNN (în funcție de complexitatea implementării) obține un mAP în intervalul 66.9% - 70.0%, iar Faster R-CNN (în funcție de complexitatea implementării) obține un mAP în intervalul 68.5 % - 78.8 %.

Chiar dacă implementarea cu ZF nu obține un mAP comparabil cu implementările bazate pe VGG, obține un timp de procesare al imaginilor superior sistemelor ce utilizează VGG, putând spune ca este un sistem de detecție ce rulează la viteze apropiate de real-time.

5.4 SSD

Până în momentul redactării lucrării [22], sistemele performanțe de detecție de obiecte necesitau un timp de procesare foarte mare.

Sistemul de detecție SSD introdus în [22] reprezintă prima rețea neuronala cu scop de a fi un obiect detector.

Pe setul de date PASCAL VOC 2007 SSD obține 59 FPS cu un mAP de 74.3 %.

Ca termen de comparație Faster R-CNN obține 7FPS cu un mAP de 73.2 % iar sistemul de detecție YOLO obține 45 FPS cu un mAP de 63.4 %.

Pentru început, SSD este un sistem de detecție ce face parte din familia single-shot, la fel ca sistemul de detecție YOLO.

SSD este bazată pe o rețea convoluțiunea ce are ca output o mulțime fixa de boxuri, pentru fiecare box existând un scor ce identifica clasa obiectului aflat în interiorul boxului.

Peste outputul rețelei este aplicată operația de non maximum suppression. Rezultatul acestei operații reprezintă outputul final al sistemului de detecție SSD.

Straturile principale ale arhitecturii SSD formează rețeaua de baza și sunt bazate pe arhitecturi convoluțiunile existente, utilizate pentru clasificarea de imagini, cu acuratețe foarte buna.

La rețeaua de baza se adaugă o serie de straturi pentru a realiza detecția.

La finalul rețelei de baza se adaugă o serie de straturi de convoluție cu scopul de a diminua progresiv dimensiunea outputului rețelei de baza pentru a permite detecția la mai multe scale.

Fiecare strat adăugat după rețeaua de baza poate produce o serie de predicții folosind filtre convoluțiunile.

Spre exemplu un kernel de dimensiuni 3 x 3 x p (unde p este numărul de canale al inputului peste care este aplicat) poate produce fie un scor privind categoria din care face parte obiectul fie valori privind localizarea fata de cadrul de input.

Sistemul de detecție SSD folosește o serie de boxuri predefinite, cu diferite scale.

S-a ajuns la concluzia ca utilizarea de mai multe boxuri duce la mai bune performanțe, făcând ca taskul de prezicere a boxurilor pentru obiecte de către rețeaua convoluțională sa fie mai simplu.

Cel mai complex model de SSD (SSD 512) a obținut în PASCAL VOC 2007 81.6 % mAP comparativ cu cel mai bun de Faster R-CNN ce a obținut 78.8 % mAP.

Ca observație, pentru un set de date mic, SSD are nevoie de un proces numit augmentare de date (un proces aplicat setului de date); în urma utilizării procesului, performanțele sunt cu mult îmbunătățite fapt demonstrat în [22].

La momentul în care se face inferența, este esențial ca pasul de non-max suppression sa fie făcut pe cat posibil mai eficient.

În [22] s-a ajuns la concluzia (la momentul când a fost redactata lucrarea) ca SSD300 este singurul sistem de detecție real-time ce obține un mAP mai mare de 70%. Aplicând o serie de modificări, este posibil ca și sistemul de detecție SSD 512 să poată funcționa în mod real-time.

300 și 512 din numele sistemelor de detecție enunțate anterior vin de la dimensiunea inputului, SSD 300 are input de forma 300 x 300 x c iar SSD 512 are input de forma 512 x 512 x c, unde c este numărul de canale al inputului.

5.5 SETUL DE DATE

Setul de date utilizat este o combinație între un data set specializat pentru detecția de căști de protecție (setul de date este public pe Kaggle și a fost folosit de asemenea în lucrarea [1]) și o serie de imagini preluate din setul de date Open Images.

În construirea setului de date folosit pentru training, am ales doar 300 de imagini din setul de date folosit și în lucrarea [1].

Aceasta decizie a fost susținută de câteva cerințe pe care imaginile pentru training trebuiau sa le îndeplinească:

1. Imaginile alese au foarte multe entități (de preferat 5+)
2. Imaginile alese sunt realizate în tot felul de condiții de iluminare (iluminare pe timp de zi, de noapte, iluminare interioara).
3. Persoanele se afla la diferite distanțe față de camera (s-au preferat imagini unde subiecții taskului de detecție sa se afla la distanțe foarte mari, pentru a simula pe cat de bine posibil situațiile unui șantier/unei zone de producție).
4. S-au preferat imaginile unde căștile de protecție erau acoperite de murdărie, pentru a simula pe cat de bine posibil situația unui mediu de lucru adecvat.
5. S-au preferat imaginile unde subiecții umani se afla in diferite pozații, altele decât cea verticala.

S-au ales doar imaginile ce au depășit aceste cerințe (cel puțin 3) deoarece s-a dorit ca algoritmul de învățare sa fie supus la situații foarte dificile de detecție.

În cazul în care o imagine aleasă a s-a încadrat cu greu în aceste cerințe, aceasta a fost augmentata prin rotirea ei la 90 și 180 de grade în ambele direcții; pe scurt, imaginea a fost înlocuită de un colaj în care imaginea originală se afla în următoarele poziții: original, rotire în oglinda vertical, rotire în oglinda orizontal, rotire la stânga și la dreapta cu 90 de grade.

Deoarece setul de date din care s-au selectat cele 300+ de imagini era defectuos în imagini cu oameni ce nu purtau cască de protecție sau in imagini cu oameni ce poarta alte echipatele/articole vestimentare pe cap, s-au mai selectat din setul de date Open Images încă 50+ de imagini pentru a încerca alăturarea acestui minus.

Având cele 300+ imagini din setul de date dedicat pentru detecția de căști plus încă 50+ de imagini preluate din Open Image a urmat etapa de creare de etichetare pentru imaginile din setul de date dezvoltat.

S-a hotărât etichetarea acestora de la 0 pentru a asigura ca nu exista urme de eroare umană.

Etichetarea a fost realizata folosind tool-ul LabelImg și a fost finalizată în 11h+ (de etichetat continuu).

Ca observație, s-a ales utilizarea unui set de date atât de mic pentru training, deoarece s-a încercat sa se identifice ce sistem de detecție poate realiza performate bune cu un set de date altfel considerat în lumea Deep Learning-ului un set de date mic chiar insuficient.

5.6 IMPLEMENTARE FOLOSIND API-UL TENSORFLOW

Pentru implementarea sistemului de detecție, s-a utilizat API-ul pus la dispoziție de Tensorflow.

Pentru antrenare s-au ales doua sisteme de detecție state-of-the-art, SSD și Faster R-CNN.

Pentru a realiza o comparație corecta între cele doua sisteme de detecție antrenate pe data setul dezvoltat pentru taskul propus, s-a folosit aceeași arhitectura pentru rețeaua de convoluție; ambele sisteme utilizează Inception V2.

De asemeni, aceste sisteme de detecție, sunt deja prea antrenate pe un set de date comun.

Modelele pre antrenate sunt preluate de pe pagina de GitHub a celor de la Tensorflow, din secțiunea detection_model_zoo.md.

Trainingul modelelor s-a realizat pe platforma Windows.

Înainte de a putea utiliza API-ul oferit de Tensorflow pentru antrenarea modelului în scopul dorit, trebuie pregătite câteva tool-uri pe platforma de lucru: Anaconda (pentru a putea utiliza python in Windows), CUDA si cuDNN (pentru a putea folosi Tensorflow GPU) și API-ul Tensorflow.

Având tool-urile, acestea trebuie configurate; configurarea lor trebuie făcută cu mare atenție, pentru că își cele mai mici abateri de la modul lor de a fi configurate poate duce la erori funcționale greu de depistat.

Ca observație, pentru tool-urile CUDA si cuDNN, se recomanda utilizarea versiunilor compatibile cu versiunea de Tensorflow utilizata; verificarea utilizării de versiuni compatibile poate fi făcută pe site-ul celor de la Tensorflow.

Având stația de lucru pregătea, trebuie pregătite imaginile pentru training.

Acest pas se poate realiza cu mai multe metode, dar este recomandat utilizarea tool-urilor pentru pregătire dispuse de API-ul celor de la Tensorflow.

Având datele pregătite, se poate lansa antrenarea.

Acest proces este relativ lung, în funcție de hardware-ul de care se dispune.

În cazul lucrării de față s-a utilizat o placă video Nvidia 1050 MaxQ cu 3GB de memorie dedicata.

Ca observație, sunt anumite modele ce necesita cantități foarte mari de memorie dedicata video. De aceea, dacă un model nu reușește sa ruleze cu configurația predefinita, se poate încerca micșorarea valorii batch-ului pana la 1.

Daca modelul nu poate fi antrenat nici cu un dimensiunea batch-ului egala cu 1, sunt de ales două opțiuni, să se schimbe modelul cu unul cu o complexitate mai mica sau sa se mărească capacitatea memoriei dedicate video (prin înlocuirea plăcii video existente sau prin utilizare de SLI sau alte metode similare).

Nu se recomanda antrenarea pe procesor, deoarece procesul de training ar dura extrem de mult.

Procesul de training s-a finalizat, dacă valoarea erorii totale a scăzut sub un anumit prag sau s-a așteptat un timp îndelungat pentru training, sau algoritmul a început sa diverge, sau algoritmul nu mai învață (atingerea unei anumite valori a funcției toantele de pierderi și fluctuarea în jurul acestei valori).

Aplicând pașii enumerați anterior, s-a folosit API-ul dispus de Tensorflow, cu setul de date pregătit pentru acest task pentru antrenarea a doua modele, unul bazat pe SSD cu Inception V2 și celălalt bazat pe Faster R-CNN cu Inception V2.

Modelul bazat pe Faster R-CNN a fost antrenat puțin peste 35000 de pași.

Durata pentru training pentru acest model a fost de 3.5 ore, ce mai buna valoare înregistrat pentru total loss fiind de 0.2. Trainingul a fost oprit după 3.5 ore deoarece timp de 30 de minute a fluctuat în jurul acestei valori de 0.2.

Modelul bazat pe SSD a fost antrenat în 250000 de pași.

Durata pentru training a fost aproximativ de 4.5 ore, cea mai buna valoare a total loss fiind de 1.5. După 3 ore de training, modelul nu a dat semne ca ar mai învață, dar a fost prelungit trainingul, sperând că va reuși sa obțină valori mult mai bune.

Pentru că rezultatele obținute de sistemul de detecție SSD ce utiliza Inception V2, nu au fost foarte bun, s-a încercat antrenarea sistemului de detecție folosind o rețea convoluțională bazate pe arhitectura ResNet 101.

Pentru a putea face o comparație, acest model a fost prea antrenat pe același set de date, COCO, ca modelul ce utiliza Inception V2.

Surprinzător, după puțin peste 6400 de pași, modelul bazat pe ResNet 101 a obținut o valoare minima pentru total loss de 0.04. Aceasta performanța a fost obținută pentru un timp de training de sub 1 o ore și jumătate.

Ca observație, sistemul SSD ce utiliza ResNet 101, după 15 minute de training obținea rezultate apropiate de sistemul SSD ce utiliza Inception V2 după 4.5 ore de training.

Sunt șanse ca setul de date foarte mic sa fi fost insuficient pentru antrenarea modelului SSD cu Inception V2; chiar este precizat în [22] că, sistemele de detecție bazate pe SSD au nevoie de un număr destul de mare de date de training, și în cazul în care nu sunt suficiente date să se aplice data augmentation (etapa pe care în antrenarea sistemelor de detecție pentru lucrarea de fata a fost sărită intenționat).

6 EVALUAREA REZULTATELOR

Pentru a testa cele trei sisteme de detecție, s-a folosit graful de inferența (pentru fiecare model în parte) generat în urma trainingului împreuna cu un script (dedicat detecției; acest script este unul ajutător, open source, destinat testării modelelor de detecție).

Acel script preia flux video de la camera web, în format 720p (1280x720), și cu ajutorul grafului de inferența corespunzător fiecărui model în parte, genera detecții pe fluxul video.

Pentru început, cele trei sisteme antrenate pe setul de date dezvoltat pentru taskul lucrării s-au încadrat în limita impusă, de a detecta un cadru în mai puțin de o secundă.

Singurul sistem ce a avut probleme cu aceasta constrângere, privind procesarea de cadru video în maxim o secundă, a fost sistemul SSD ce utiliza arhitectura ResNet 101, dar era oarecum de înțeles, deoarece aceasta arhitectura este mult mai complexă comparativ cu Inception V2, privind numărul de parametrii.

Inception V2 are sub 30 de milioane de parametrii, pe când ResNet 101 depășește cu puțin 44 de milioane de parametrii.

La polul opus se afla sistemul de detecție SSD ce utilizează Inception V2.

Din punct de vedere al timpului necesar petrecut pentru a procesa un cadru, putem spune ca acest sistem este unul real-time. Pentru ochiul umana nu sunt urme de lag / sacadare privind procesarea cadrelor. Detecția este rapidă și outputul fluxului video procesat de script are minim 25 de cadre pe secundă.

Sistemul bazat de detecție ce utilizează metoda Faster R-CNN cu Inception V2 a fost puțin mai rapid ca sistemul bazat pe SSD cu ResNet 101, rezultatul outputului aplicat de scriptul de detecție pe fluxul video având peste 3 cadre pe secundă; nu este un rezultat fantastic, dar îndeplinește condiția de a putea procesa minim 1 cadru pe secundă.

Cea mai importantă cerința dezvoltată anterior este calitatea predicției și a detecției.

De aceasta cerința nu pot trece sistemele bazate pe SSD, nici cel ce utilizează Inception V2 și nici cel ce utilizează ResNet 101.

Primul sistem evaluat la aceasta cerința este sistemul bazat pe SSD ce utilizează ResNet 101 pentru ca la cerința anterioara a obținut cele mai prost rezultat.

Fiind supus la un task de detecție simplu, cu doi subiecți la o distanța de maxim 3 metri de camera, sistemul de detecție identifica corect capetele celor două persoane, dar identifica o casca pe capul persoanei din stânga; acesta detecție este greșită, singurul subiect de utilizează casca fiind subiectul din dreapta, figura 5.

Pentru anumite poziții ale capului subiectului din stânga, sistemul de detecție nu mai identifica casca, dar eroarea de identificare exista.

Figura 5: doi subiecți, SSD ResNet 101.

Încă o problema detectata apare când un subiect se deplasează la o distanța mai mare de 5 metri de camera, mai ales în zone întunecate.

În figura 6, subiectul din stânga poarta casca iar cel din dreapta nu poarta casca.

Sistemul de detecție detectează bine casă și capul subiectului din stânga dar are probleme cu detecția subiectului din dreapta, detectând pentru acesta cap și casca, chiar dacă acesta nu poarta casca.

Figura 6: 2 subiecți, SSD ResNet 101.

Sistemul de detecție bazat pe SSD ce utilizează Inception V2, deși câștigător în urma primei evaluări, obține cele mai proaste performanțe privind detecția.

Cel mai bun exemplu de detecție proastă este în figura 7.

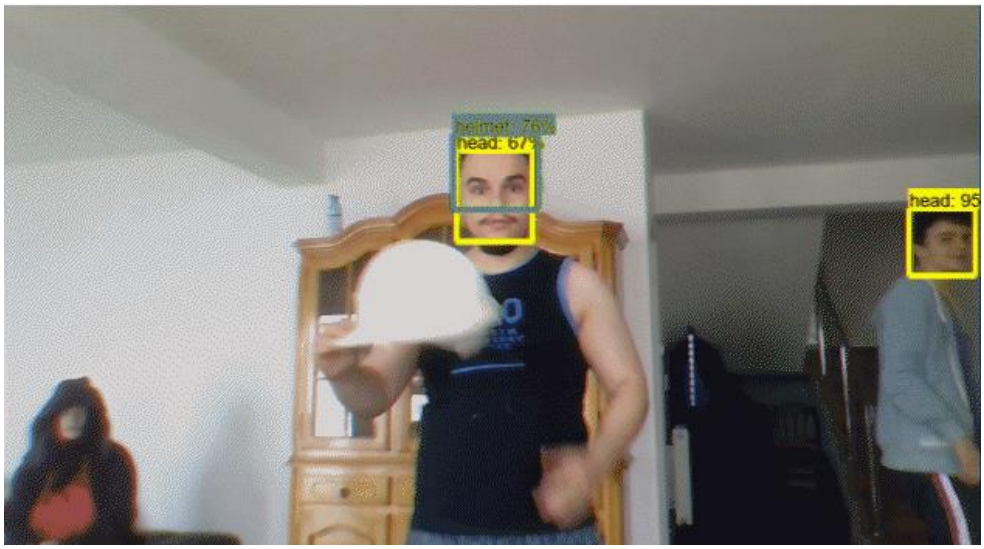

Figura 7: trei subiecți, SSD Inception V2.

Subiectul din stânga nu este detectat, subiectul din mijloc este detectat (îi este detectat capul), dar îi este detectata o casca pe capul acestuia (chiar dacă nu purta nici o cască); casca pe care subiectul 2 o are în mână nu este detectat.

Singurele detecții corecte pe care acest sistem le-a avut sunt capul subiectului din mijloc și capul subiectului din dreapta.

În figura 8, sistemul nu are probleme în a depista capetele celor doi subiecți, dar are

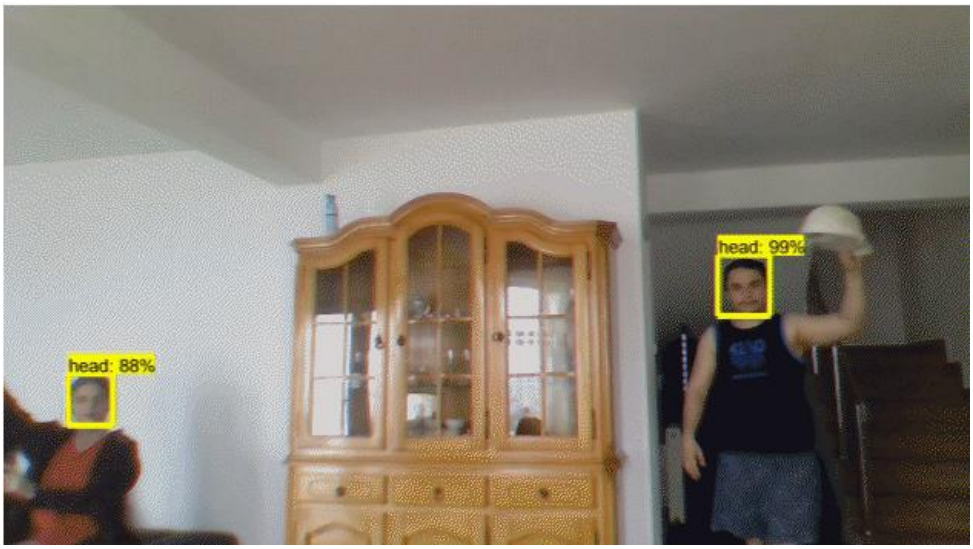

Figura 8: doi subiecți, SSD Inception V2.

probleme în a identifica cască de protecție.

Chiar dacă aceasta se afla în mână subiectului din dreapta și este foarte vizibilă, și într-o iluminare relativ buna, nu poate să o identifice.

Aceasta problema este persistentă și chiar dacă se poate spune ca este un sistem de detecție real-time, din punctul de vedere al calității detecției, este un eșec.

Singurul sistem ce a depășit constrângerea pe care celelalte doua sisteme n-au putut-o depăși, este sistemul ce utilizează Faster R-CNN cu arhitectura Inception V2.

În figura 9, se observa calitatea detecției acestui sistem.

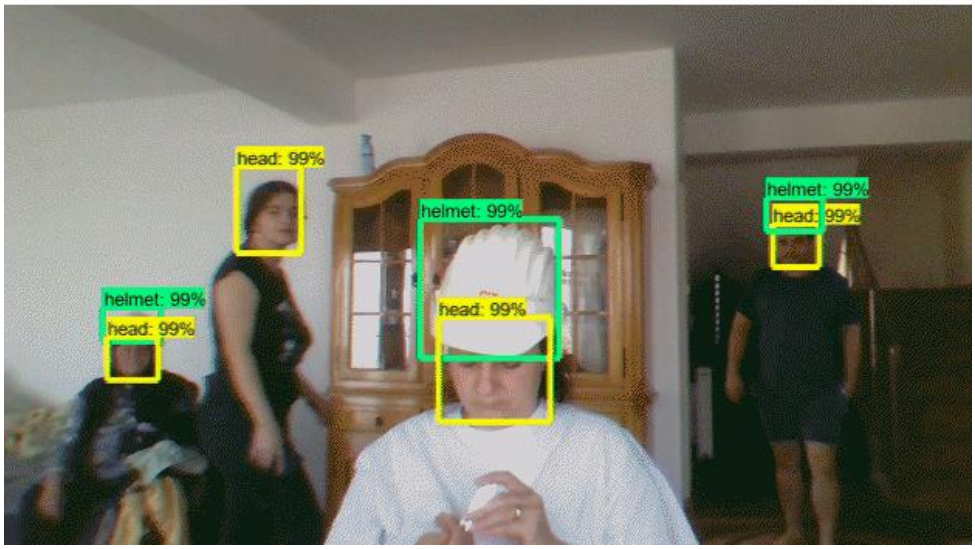

Figura 9: numerotare subiecții de la stângă la dreapta, Faster R-CNN Inception V2.

Subiectul 1 poarta casca și este identificat și capul și casca corect, chiar dacă acesta purta pe cap un articol vestimentar ce a derutat celelalte doua sisteme de detecție (subiectul 1 purta un batic negru).

Subiectului 2 îi este identificat corect capul; acesta nu poarta casca.

Subiectului 3 îi este identificat corect capul (chiar dacă poziția în care se afla cu capul este relativ un mai complexă, o parte din cap fiind acoperita de casca), și casca.

Subiectului 4 îi este identificat corect capul (chiar dacă se afla la o distanța la care sistemul bazat pe SSD și ResNet 101 ar fi avut probleme de detecție); de asemeni subiectul poarta casca și chiar dacă se afla într-o zona slab iluminata, mult mai slab iluminata comparativ cu figura 4 (unde subiectul din dreapta ținea o cască în mana), reușite sa o detecteze cu acuratețe de 99%.

Utilizând un set de date mic, au început sa se vadă problemele.

Chiar dacă sistemul de detecție bazat pe Faster R-CNN cu Inception V2 are o detecție buna, ocazional mai apar erori în detecție.

În figura 10 se poate observa o astfel de identificare eronată.

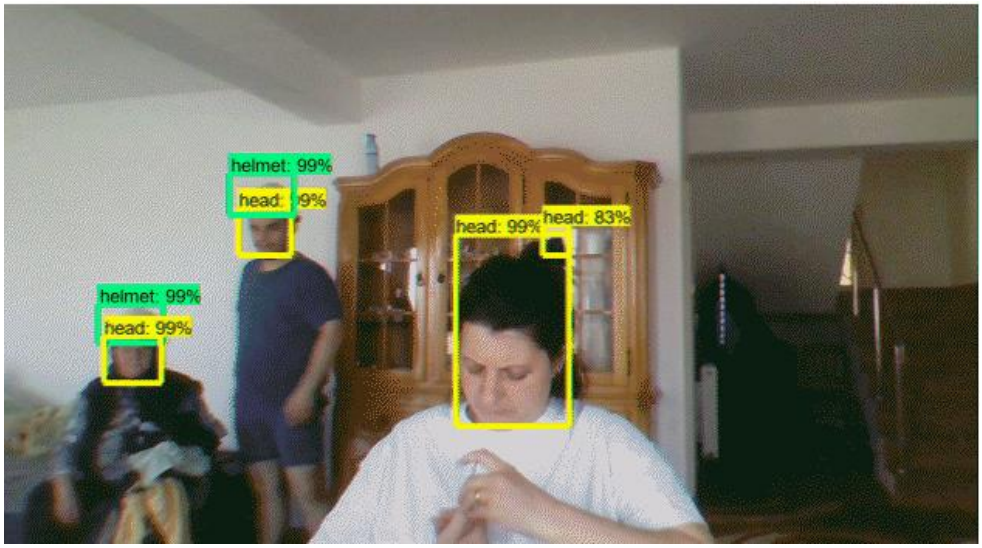

Figura 10: trei subiecți, Faster R-CNN Inception V2.

Se poate observa în figura 6 că în cazul subiectului 3, pe lângă capul real al acestuia (identificat corect), mai este identificat incorect un cap (capul incorect identificat este un coc, o coafură clasică); aceasta eroare nu tine mai mult de 0.5 secunde, dar apare.

Mai sunt cazuri cu diverse obiecte, ce pot păcăli aceste sisteme de detecție, dar pentru cazul de utilizare generală, cel mai robust este sistemul de detecție bazat pe Faster R-CNN cu Inception V2.

Ca un sumar, modelul ce utilizează SSD cu ResNet 101 identifica corect toate capetele, dar are probleme în identificarea căștilor de protecție, în fiecare caz a identificat cu o casca mai mult decât era în cadru, obținând o precizie de 0.5 și un recall de 1 salarizând un scor F1 egal cu 0.66. Pentru detecția capetelor, scorul F1 este 1.

Modelul bazat pe SDD cu Inception V2, având cea mai proastă detecție, identifica corect aproape toate capetele (singurul neidentificat putând fi trecut cu vederea, deoarece subiectul din stânga figura 7, purta foarte multe articole vestimentare ce au derutat algoritmul de detecție), dar are o precizie 0 privind detecție căștilor de protecție, ducând la un scor F1 = 0. Pentru detecția capetelor, scorul F1 este egal cu 1, dacă se omite cazul în care subiectului din stângă, figura 7, nu i s-a identificat capul; dacă nu se omite acest caz, scorul F1 pentru detecția capetelor este egal cu 0.88.

Dintre toate sistemele de detecție Faster R-CNN are si precizia si recall egale cu 1 pentru detecția căștilor de protecție, salarizând un scor F1 egal cu 1. Pentru detecția capetelor, dacă se omite cazul în care în care cocul a fost detectat drept cap, sistemul are un scor F1 egal cu 1, dar dacă nu se omite acest caz, scorul F1 este egal cu 0.92.

7 CONCLUZII

După cum s-a prezentat capitolul anterior, cel mai satisfăcător sistem de detecție obținut in urma antrenării pe setul de date dezvoltat pentru taskul curent, este Faster R-CNN ce utilizează Inception V2.

Acest sistem reușește sa proceseze mai mult de un cadru pe secundă, având și o calitate a detecției extraordinar de buna, comparativ cu competitorii acestuia.

După alegere acestui model ca fiind cel mai satisfăcător pentru acest task, s-au analizat erorile generate de sistemul de detecție, principalele surse fiind numărul mic de imagini din setul de date utilizat pentru training și varietatea subiecților din imagini (deși in imagini, subiecții se aflau în diferențe posturi și în diferite medii ambiante, aceste cazuri din setul de date sunt insuficiente pentru a permite sistemului de detecție sa fie robust).

Luând în considerare ca majoritatea rețelelor convoluțiunile sunt antrenate pe seturi de date cu milioane de imagini, antrenarea unui model pre antrenat pe 300 de imaginii, pentru un task altul decât a fost conceput, este relativ insuficientă.

Acest neajuns a fost cel mai bine reflecta în performanțele sistemelor de detecție ce folosesc SSD. Acestea pentru a putea realiza o detecție temeinică necesita un set de date mare, de aceea în lucrarea [22] se recomanda utilizare de data augmentation in cazul în care setul de date deja existent este mic.

Ca observație, setul de date folosit pentru acest task a fost limitat la aproximativ 300 de imagini pentru training, deoarece s-a încercat identificare unui sistem de detecție ce poate obține rezultate bune cu un set de date mic.

Aceasta alegere a fost motivata de stadiul curent al seturilor de date, existând foarte puține date pentru trainingul unui sistem de detecție specializat (spre exemplu detecția unui obiect care nu este comun, cum ar fi detecția măștilor de gaze, utilizate de pompieri).

Sistemul de detecție Faster R-CNN ce utilizează Inception V2 poate fi antrenat în continuare cu un set de date îmbunătățit ce sa conțină cazurile la care modelul actual detectează și clasifica greșit. Acest proces nu este unul foarte costisitor și va permite obținerea unui sistem de detecție robust, capabil sa ruleze fără supervizare.

Din punct de vedere al erori de detecție, cel mai bun model de detecție identificat, Faster R-CNN cu Inception V2 antrenat pe setul de date conceput pentru taskul lucrării nu poate fi lăsată să funcționeze fără supervizare, cel puțin cum se află în stadiul curent, dar prin continuarea trainingului pe un set de date mai complex se poate obține un sistem robust.

Deploymentul sistemului de detecție (realizarea inferenței) poate fi realizat pe orice sistem ce deține o placă video similară (sau mai puternica) cu placa utilizata (o alternativa ieftină este placa video Nvidia GT 1030), 6 GB de RAM (minim), și un procesor ce obține un scor de minim 600 în GeekBench 5 (pentru single core). De asemenea este recomandat deploymentul pe o platforma

ce ruleaza windows, sa daca se doreste deplyment pe o platforma linux, se recomanda utilizarea containerului de Doker special dezvoltat pentru Deep Learning.

8 BIBLIOGRAFIE

[1] J. Wu, N. Cai, W. Chen, H. Wang, G. Wang , Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset, Automation in Construction, 106(2019), 10.1016/j.autcon.2019.102894

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[3] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3431–3440, 2015.

[4] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In Advances in Neural Information Processing Systems, pages 809–817, 2013.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. 2014.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[9] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2155–2162. IEEE, 2014.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567v3 [cs.CV] 11 Dec 2015. Paper: https://arxiv.org/pdf/1512.00567.pdf

[11] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeplysupervised nets. arXiv preprint arXiv:1409.5185, 2014.

[12] J. Wu. Convolutional neural networks, May 14, 2020. Paper: https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf

[13] R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv:1311.2524v5 [cs.CV] 22 Oct 2014. Paper: https://arxiv.org/pdf/1311.2524.pdf

[14] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networksfor object detection. In NIPS, 2013

[15] C. Gu, J. J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In CVPR, 2009.

[16] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013.

[17] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. TPAMI, 2010.
[18] R. Girshick. Fast R-CNN, arXiv:1504.08083v2 [cs.CV] 27 Sep 2015. Paper: https://arxiv.org/pdf/1504.08083.pdf
[19] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.
[20] S. Ren, K.He, R. Girshick, J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497v3 [cs.CV] 6 Jan 2016. Paper: https://arxiv.org/pdf/1506.01497.pdf
[21] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in European Conference on Computer Vision (ECCV), 2014
[22] W. Liu, D.Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: Single Shot MultiBox Detector. arXiv:1512.02325v5 [cs.CV] 29 Dec 2016. Paper: https://arxiv.org/pdf/1512.02325.pdf