Universitatea “Politehnica ” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Descrierea conținutului vizual pe terminalul mobil

Lucrare de disertație

prezentată ca cerință parțială pentru obținerea titlului de
Master în domeniul Imagisticii Digitale
programul de studii de Tehnici Avansate in Imagistica Digitală

Conducători științifici
Prof. dr. ing. Corneliu FLOREA
As. drd. ing Badea Mihai

Absolvent
George-Iulian NITROI

Anul 2020

Universitatea "Politehnica" din Bucureşti **Anexa 2**
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Programul de masterat **Tehnici avansate pentru imagistica digitală**

TEMA LUCRĂRII DE DISERTAŢIE
a masterandului **NITROI E. George-Iulian , 421-TAID.**

1. Titlul temei: Descrierea conținutului vizual pe terminalul mobil

2. Descrierea contribuţiei originale a masterandului (în afara părţii de documentare) şi specificaţii de proiectare:
Tema își propune realizarea unei aplicații pentru terminalul mobil capabilă să descrie imaginile provenite de la acesta. Se propune realizarea unui prototip funcțional si contribuțiile în 2 categorii de analiza a datelor si respectiv de implementare. Pentru analiza datelor se urmareste: implementarea unei rețele convoluționale care identifică obiectele din imagine și a unei rețele neuronale recurente(decoder) care preia obiectele identificate de prima rețea și generează o descriere pentru poza respectivă .
Din punct de vedere aplicativ se urmareste crearea unei aplicații utilizând Android studio capabilă să facă o poză sau să ia o poză din galeria foto a terminalului mobil, crearea unui server utilizând Firebase care să asigure comunicația intre terminalul mobil și calculator, respectiv afișare rezultatului utilizatorului.

3. Resurse folosite la dezvoltarea proiectului:
Python, Tensorflow, Android Studio, Java, XML, Javascript, Firebase

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
MLAV,PAIC,ASTM

5. Proprietatea intelectuală asupra proiectului aparţine: U.P.B.

6. Data înregistrării temei: 2019-11-13 10:51:54

Conducător(i) lucrare,
Prof. dr. ing. Corneliu FLOREA

semnătura: .....

As. drd. ing Badea Mihai

semnătura: .....

Student,

semnătura:.....

Responsabil program master,
Prof. dr. ing. Constantin VERTAN

semnătura:.....

Decan,
Prof. dr. ing. Cristian NEGRESCU

semnătura:.....

Cod Validare:

Declarație academica

Cuprins

Listă de figuri

Listă de acronime

AI: Artificial Intelligence(Inteligența artificială)

ML: Machine Learning

UI: User interface(Interfața grafică)

Cuda: Compute Unified Device Achitecture

BaaS: Backend-as-a-Service

API: Application Programming Interface (colecția de clase/funcții/interfețe cu ajutorul cărora exploatăm funcționalitatea unui framework sau a unei biblioteci)

RNN: Recurent Neural Network(Rețea neuronală recurentă)

GRU:Gated Recurrent Units

LSTM: Long-short term memory)

Introducere

În zilele noastre dezvoltarea tehnologiei este mai rapidă iar acest lucru se poate observa mai ales în domeniul științei calculatoarelor. Inteligența artificială(AI) este la momentul de fața unul din vârfurile de lance în acest domeniu oferind o perspectivă diferită asupra rezolvării unor probleme care în programarea clasică s-au dovedit de-a lungul timpului dificil de abordat. Deși nu este un domeniu nou, el având bazele la începutul anilor 50 când metodele statistice care stau la baza algoritmilor de azi sunt găsite și perfecționate, implementarea lor și utilizarea acestor algoritmi in mediile de producție s-a dovedit dificilă până la începutul anilor 2000.

Apariția internetului și accesul la acesta pentru publicul larg la sfârșitul secolului 20 a fost unul din principalii factori ce au dus la reducerea în prim plan al inteligenței artificiale. Internetul rezolvă problema insuficienței datelor cu care cercetătorii în domeniu s-au lovit până în acel punct, facilitând crearea unor baze de date care ulterior să poată servi antrenării diferitelor modele matematice disponibile în acel moment.

Un alt lucru ce a împiedicat multă vreme ca acești algoritmi sa fie fezabili a fost lipsa puterii de procesare. A fost nevoie de zeci de ani pentru ca industria semiconductoarelor să ajungă la nivelul în care aceste modele să poate fi utilizate pe probleme complexe, cu un număr considerabil de clase, în scenarii utile aplicațiilor noastre de zi cu zi. În istoria recentă, Nvidia face parte din ceea ce se poate numi “Big bang-ul” rețelelor neuronale adânci, când în 2009 un astfel de model este antrenat pe o placă video sporind viteza de procesare a acestora de aproximativ 100 de ori.

În 2020 putem găsi componente AI în majoritatea programelor/aplicațiilor pe care noi le folosim de zi cu zi. Proiectul curent își propune realizarea un aplicații pe terminalul mobil care să folosească toate avansurile tehnologice menționate mai sus. Folosind un terminal mobil cu Android aplicația va fi capabilă sa facă poze sau să folosească poze deja existente în galeria telefonului cărora ulterior sa le adauge o descriere generată de un model de ML(machine learning). Aplicația va fi una te tip server-client unde, terminalul mobil(smatphone-ul) se va comporta ca un client pentru un server ce așteaptă să primească o poză de la telefon și ii va răspunde cu ieșirea modelului, o propoziție ce descrie acea poză.

Obiectivele pe care și le propune acest proiect sunt următoarele:

• Dezvoltarea unei aplicații Android

• Implementarea topologiei server client intre terminalele mobile și un server cu o putere de calcul îndeajuns de mare încât să ruleze modelul.

• Antrenarea modelului de rețea neuronală cară sa facă descrierea imaginilor

• Testarea aplicației

Capitol 1 Noțiuni teoretice necesare dezvoltării practice

1.1 Topologia aplicației

Aplicația noastră are 3 componente software:

- O aplicație dezvoltată în Android Studio ce va servi ca front-end pentru aplicația noastră și va facilita utilizatorului un meniu intuitiv și ușor de folosit pentru a profita de facilitățile oferite.
- O componenta Firebase ce va servi ca back-end și pe care noi ne vom stoca modelul antrenat pentru a genera descrierea imaginilor
- Un model de rețea neuronală adâncă dezvoltat in Python cu ajutorul librăriei Tensorflow.

Figura 1-1 Topologie aplicație

Modelul obținut de noi în urma antrenării va fi prea mare pentru a putea fi stocat pe terminalul mobil. De asemenea nu orice terminal mobil are posibilitatea de a rula modele complexe pe arhitectura lor hardware așa că pentru a acomoda aplicația pe cat mai multe terminale mobile s-a luat decizia asupra acestei arhitecturi.

1.2 Terminalul mobil

Aplicația poate fi instalată și va rula pe orice telefon mobil, tabletă sau device ce rulează o distribuție de android mai noua decăt Nougat 7.0 și care dispune de o camera foto și de conexiune la internet. Cerințele acestea vor fi minimale întrucât marea parte a puterii de procesare va fi oferită de server.

1.2.1 Platforma Android

Android este o platformă software și un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux, dezvoltată inițial de compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance. Android permite dezvoltatorilor să scrie cod gestionat în limbajul Java și începănd cu anul 2017 Kotlin controlând dispozitivul prin intermediul bibliotecilor dezvoltate de Google. Aplicațiile scrise în C și în alte limbaje pot fi compilate în cod mașină ARM și executate, dar acest model de dezvoltare nu este sprijinit oficial de către Google. [1]

Deși Java a fost limbajul de programare care a dominat o buna perioadă de timp dezvoltarea aplicațiilor pe platforma Android, dezvoltatorii au posibilitatea să folosească Kotlin care în mai puțin de 3 ani de la integrarea în platforma Android a ajuns sa fie limbajul oficial preferat de Google pentru Android. Kotlin face ca dezvoltarea aplicațiilor sa fie mai concisă (mai mult cu mai puține linii de cod) și elimină necesitatea dezvoltatorilor să lege fiecare element din UI(user interface) cu codul din spate. Fiind un limbaj nou, el oferă un nivel de abstractizare similar cu cel al JavaScript-ului sau Python-ului. Cu toate acestea el se

poate integra ușor cu Java, dezvoltatorul având posibilitatea să folosească in Kotlin instanțe ale claselor scrise în Java și viceversa. [2]

1.2.2 Hardware

În momentul de față pe majoritatea terminalelor mobile pe lângă să asigure o comunicația audio fără fir sunt capabile să facă o multitudine de alte sarcini prin intermediul componentelor adiționale pe care le regăsim în formatul lor compact. Display-urle bazate pe tehnologii LED sunt capabile să redea imagini de calitate la rezoluții înalte și cu o gama largă de culori, procesoarele, memoria și stocarea sunt capabile să susțină sisteme de operare comparabile ca performanțe cu cele regăsite pe calculatoare, modem-urile integrate asigură comunicații de viteză și viteze înalte de transfer pe internet. Camerele foto integrate dispun de caracteristici precum: autofocus, senzor de lumină, flash, stabilizare de imagine digitală sau mecanică. De asemenea putem regăsi senzori adiționali precum: accelerometru, giroscop, busolă digitală, senzori de proximitate, magnetometru etc.

Terminalele mobile de ultimă generație dispun de acceleratoare grafice cu arhitecturi CUDA, arhitectura utilizată și pe plăcile video regăsite pe calculatoare. Aceste acceleratoare permite rularea unor redarea unor jocuri/aplicații cu înaltă calitate grafică și pun la dispoziție o suită de biblioteci de software de ML accelerată complet pe GPU.

1.3 Firebase

Firebase este un backend ca un serviciu (BaaS). Firebase elimină necesitatea dezvoltatorului de a se concentra pe dezvoltarea și managementul unui server, și pune la dispoziție dezvoltatorului un API generic pentru a ii ușura munca. Printre facilitățile oferite de Firebase noi vom utiliza următoarele:

- Firebase Analytics pentru a obține statistici despre folosirea aplicației
- Firebase Auth pentru a permite utilizatorilor sa se autentifice pe un acont propriu și pentru a le proteja pozele trimise către server
- Firebase Hosting pentru a ține scripturile în Python
- ML Kit pentru a integra modelul antrenat

1.4 Tensorflow

TensorFlow este o bibliotecă software gratuită și open-source pentru flux mare de date și programare tensorială pentru o gamă largă de sarcini. Este o bibliotecă simbolică de matematică și este, de asemenea, utilizată pentru aplicații de învățare automată, cum ar fi rețelele neuronale. Este utilizat atât pentru cercetare, cât și pentru producție la Google. Librăria este disponibilă pentru mai multe tehnologii de programare cum ar fi C, Java, Python și Javascript însă este mai stabilă pe ultimele 2. In cadrul proiectului nostru vom utiliza varianta librăriei din Python alături Keras.

Partea funcțională a TF este implementată în C++ și CUDA, în timp ce API-ul cel mai frecvent utilizat este creat în limbajul Python. Așadar, în TF se programează folosind Python, dar procesarea efectivă se realizează de către un engine dezvoltat in C++ / CUDA. [3]

1.4.1 Modul de funcționare

Un program TF are două componente esențiale:

- un model, ce conține totalitatea operațiilor ce se doresc a se efectua, precum și a datelor (tensorilor) ce se doresc a fi determinate. Modelul se implementează folosind o structură de tip graf, ce conține succesiunea de execuție a operațiilor și de evaluare a tensorilor (i.e. de determinare a valorilor variabilelor modelului).Acest graf este realizat conform principiului de programare data flow , de unde și denumirea de TensorFlow – o bibliotecă ce prelucrează tensori folosind operații structurate pe principiul data flow (pentru simplificare, prin tensor înțelegem un array multidimensional – scalar, vector, matrice 2D, 3D etc.) [3]
- sesiune – obiect ce permite execuția parțială sau completă a modelului menționat anterior. În cadrul unei sesiuni se rezolvă problema dorită cu ajutorul modelului definit în prealabil. [3]

1.4.2 Keras

Este API-ul la nivel înalt al TensorFlow-ului pentru construirea și antrenarea modelelor de ML. Este folosit pentru prototipare rapidă, cercetare de ultimă generație și producție, cu trei avantaje cheie:

- Ușor de utilizat: Keras are o interfață simplă, consistentă, optimizată pentru cazuri de utilizare obișnuită. Oferă feedback clar și acționabil pentru erorile utilizatorilor.
- Modulară și compozibilă: Modelele Keras sunt realizate prin conectarea blocurilor de construcții configurabile, cu câteva restricții.
- Ușor de extins: Scrierea blocurilor de construcții personalizate pentru a exprima noi idei de cercetare. Creați noi straturi, valori, funcții de pierdere și dezvoltați modele de ultimă generație.

1.5 Python

Python este un limbaj de programare interpretat, la nivel înalt, cu scop general. Creată de Guido van Rossum și lansată pentru prima dată în 1991, filozofia de proiectare a lui Python subliniază lizibilitatea codurilor prin utilizarea spațiilor pentru a delimita diferite structuri. Abordarea orientată spre obiect și multitudinea de funcții si librarii cu care el vine deja instalat face ca Python să fie in ziua de azi unul din limbajele de programare preferat pentru scripting, proiecte de dimensiuni mici sau back-end în aplicații. Suportă mai multe paradigme de programare, incluzând programarea procedurală, orientată pe obiecte și funcționale

Pe lângă librăriile cu care python vine deja instalat, în cadrul proiectului nostru vom mai utiliza, pe lângă Tensorflow, și alte librarii/pachete precum:

- NumPy: o librărie care oferă suport pentru operațiile cu tensori, și un set de funcții pe care le putem folosi pe aceștia
- Matplotlib: librărie ce permite plotarea datelor in grafice
- OpenCV: o bibliotecă de funcții informatice specializată pe vedere pe care o vom folosi în prelucrarea pozelor
- Pandas: o librarie folosită pentru manipularea datelor

Capitol 2 Modelul Rețelei Neuronale

Descrierea conținutului vizual(Image captioning) își propune descrierea conținutului vizual generând un text. La baza acestei idei a stat ideea de traducere bazată pe ML. Acolo întâlneam o arhitectura de tip autoecoder, unde prima jumătate a rețelei(encoder-ul) coda textul prezent la intrare într-un format cunoscut ca și vector-gând iar cea de-a doua parte a rețelei decoda semantica acestui vector într-o limbă noua. Acest vector-gând reprezintă semantica propoziției și permite ca aceasta sa fie decodat în altă limbă într-o propoziție a cărei lungime nu trebuie sa fie egala cu cea de la intrare, dar care păstrează informația primită. Atât encoder-ul cât si decoder-ul erau rețele neuronale recurente.

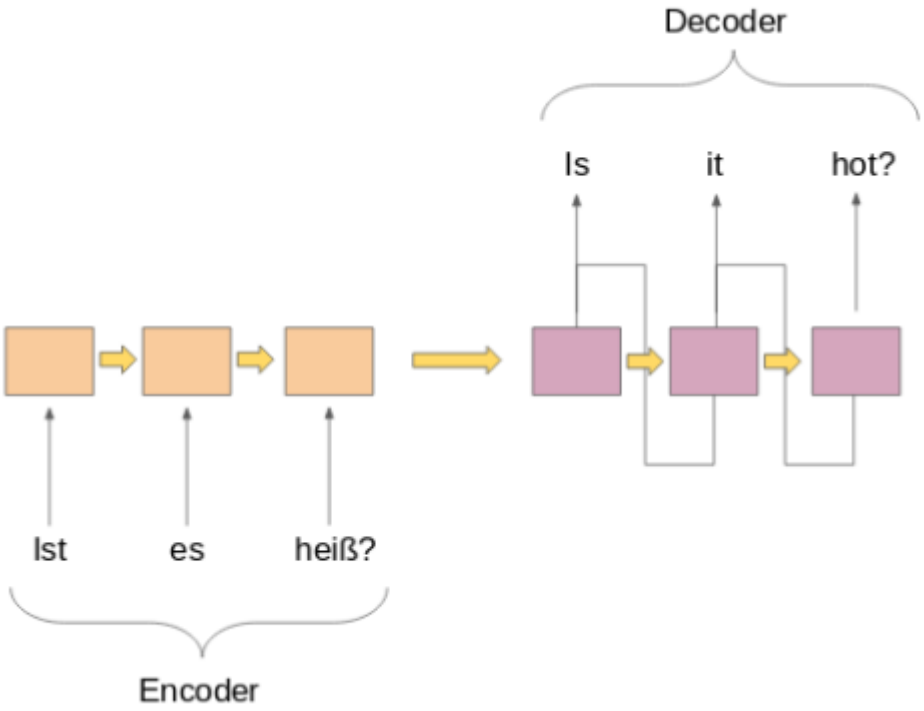

Figura 2-1 Machine Translation

Plecând de la aceeași idee vom înlocui encoder-ul cu un model de rețea neuronala convoluțională capabilă sa recunoască obiectele din imagine și care la ieșire va scoate un vector-gând similar cu cel prezent în arhitectura de traducere. Pentru a obține acest vector vom elimina din rețea ultimul strat de clasificare și vom redirecționa ieșirea penultimului strat către intrarea rețelei neuronale recurente. Cu toate acestea, dimensiunea internă a RNN-ului este de numai 512, de aceea avem nevoie de un strat intermediar complet (conectat) dens pentru a face maparea vectorului cu 4096 de elemente la un vector cu doar 512 elemente.

Decodorul folosește apoi acest vector împreună cu un marker de pornire „ssss” pentru a începe să producă cuvinte de ieșire. În prima iterație, este posibil să dea naștere cuvântului „big”. Apoi introducem acest cuvânt în decoder și sperăm că vom scoate cuvântul „brown” și așa mai departe. În cele din urmă, am generat textul „big brown bear sitting eeee" unde „eeee” marchează sfârșitul textului.

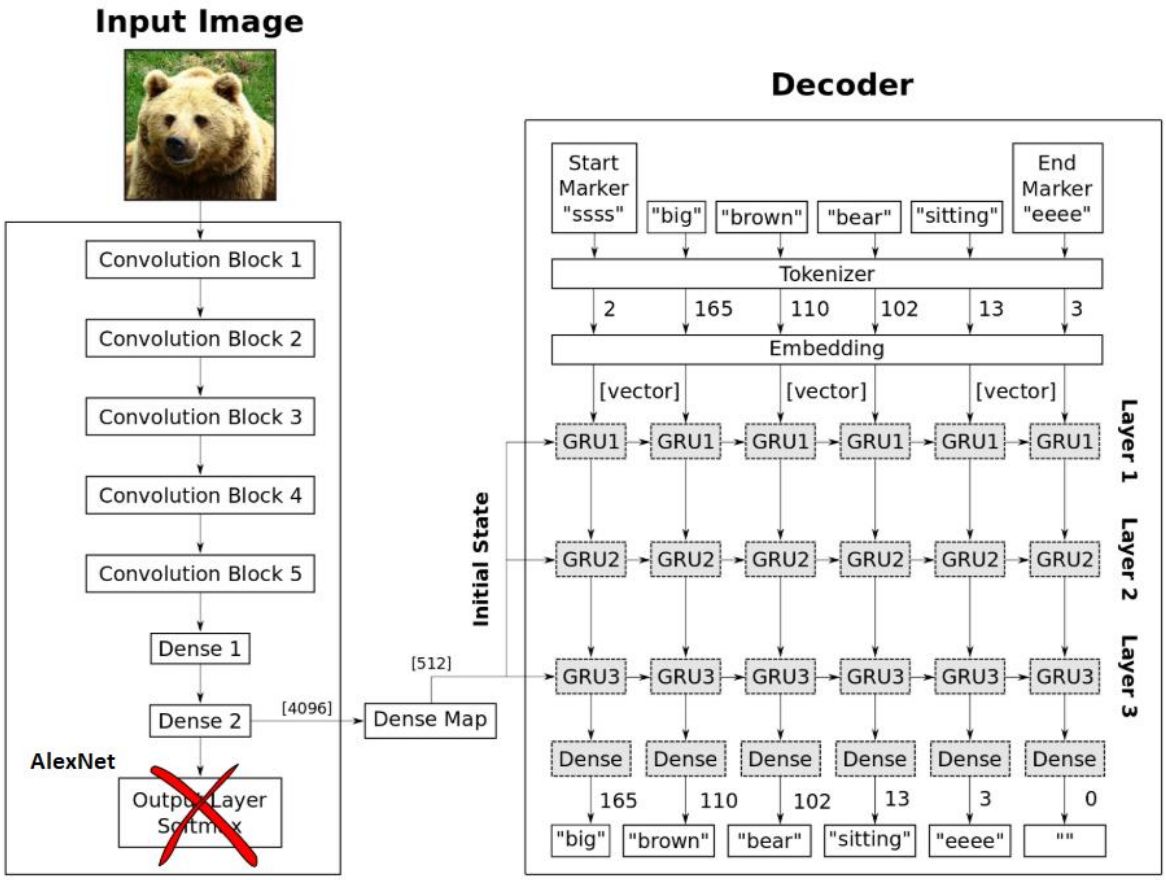

Figura 2-2 Topologie model Image Captioning [4]

Întrucât cele 2 arhitecturi diferă, AlexNet fiind unidirecțională(feed-forward), iar decoderul fiind o arhitectură recurentă, le vom antrena separat. Procedeul de antrenare este unul supervizat, în cazul rețelei convoluționale folosind "COCO data-set'' ce conține multe poze cu descrierile lor aferente, iar output-ul primei rețele va fi codat(tokenized). În cazul rețelei recurente alcătuită din 3 straturi GRU(Gated Recurrent Units), vom aplica același procedeu folosind descrierea pozelor codată in același format în care el o va primi in folosință de la rețeaua convoluțională.

2.1 Rețelele convoluționale

Este o arhitectură de rețea neuronală a capabilă sa extragă trăsături din imaginile propagate de-a lungul ei. Primele straturi convoluționale vor extrage trăsături de finețe, urmând ca pe măsură ce rețeaua se adâncește la nivelul unui strat sa putem observa trăsături ce descriu forme geometrice sau chiar obiecte de mici dimensiune. Acest tip de rețele au 3 straturi distincte:

Stratul convoluțional: este stratul de baza al rețelelor neuronale, fiind constituit dintr-o serie de filtre ce acoperă o mică porțiune din imagine. Operația matematica specifică este suma produsului element cu element intre fereastra acoperita de filtru și filtrul însăși.

Stratul convoluțional are proprietatea de a găsi caracteristici în imagine indiferent de poziția lor în imagine, datorită conectivității locale și a ponderilor comune pe fiecare porțiune a imaginii.

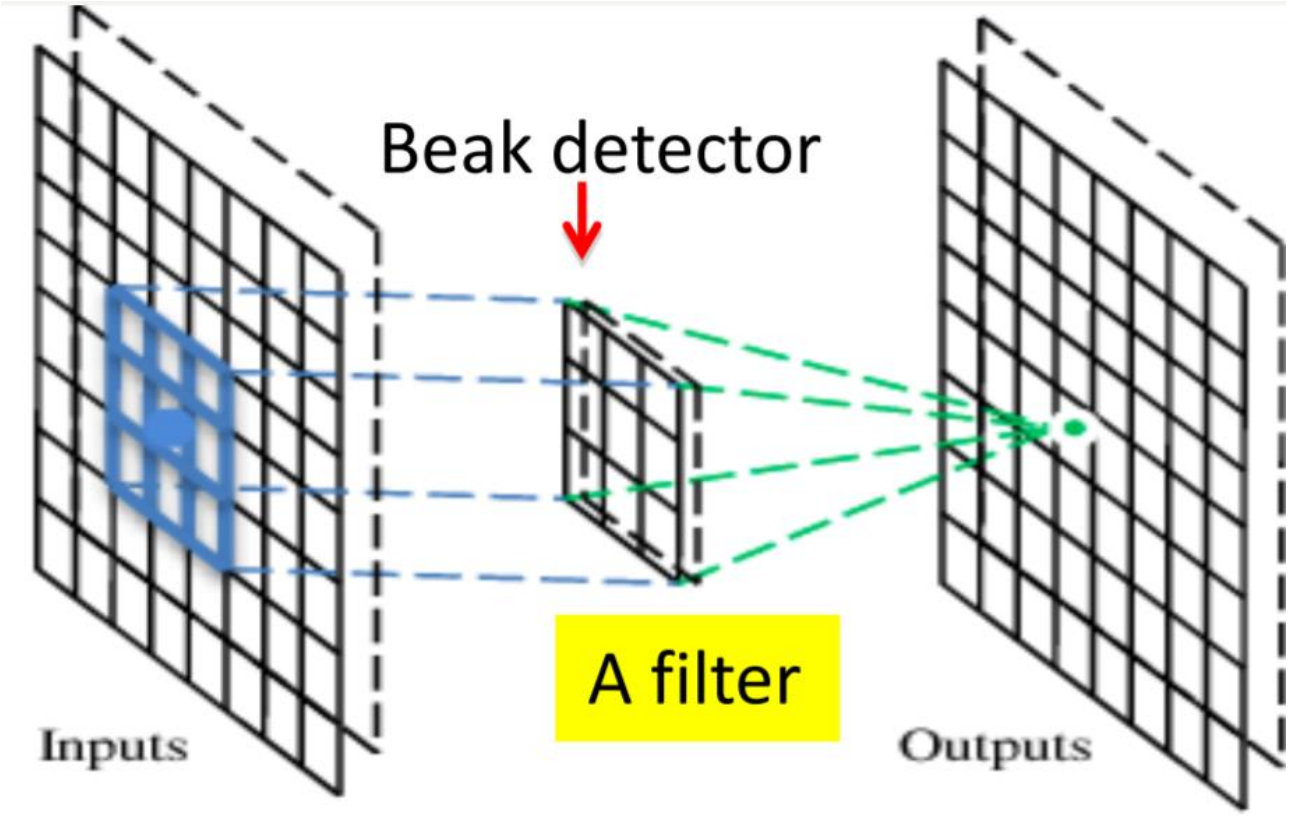

Figura 2-3 Convoluție [5]

Straturi de pooling al căror scop este să subeșantioneze informația in diferite stagii ale rețelei. Această subeșantionare nu va distorsiona informația din imagine, dar va reduce numărul de pixeli/calitatea imaginii pentru a putea fi mai ușor de lucrat cu aceasta.

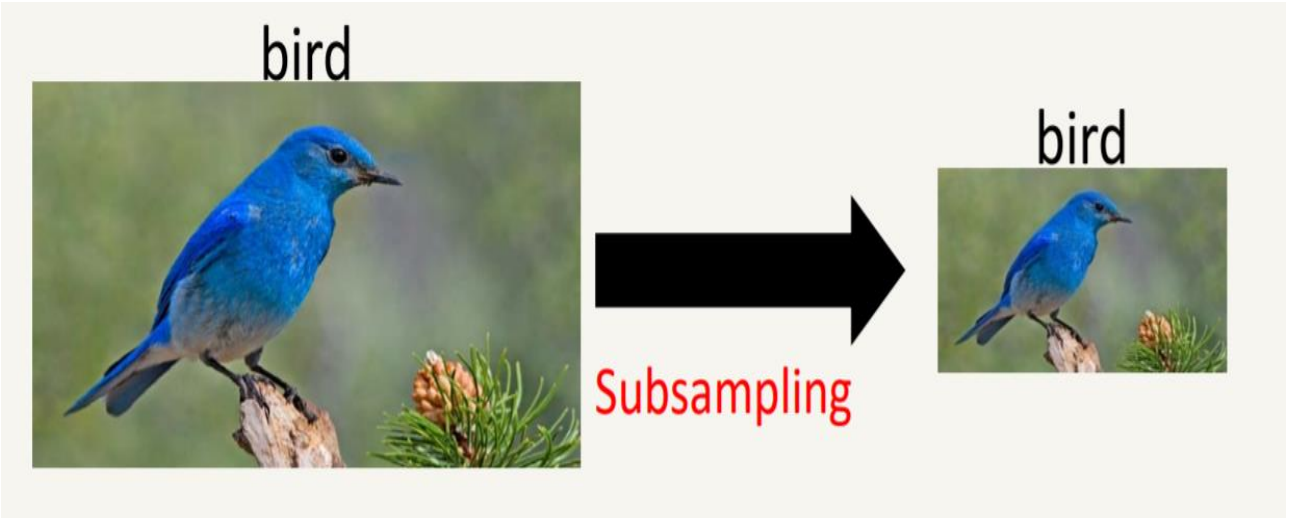

Figura 2-4 Subeșantionare [5]

De regulă aceste straturi sunt prezente într-o rețea la fiecare 1-2 straturi convoluționale și reduc la jumătate informația aflată în imagine la acel moment fie prin:

- Max-pooling-maximul dintr-o fereastră(2x2 de regulă)
- Avg-pooling- media dintr-o fereastră

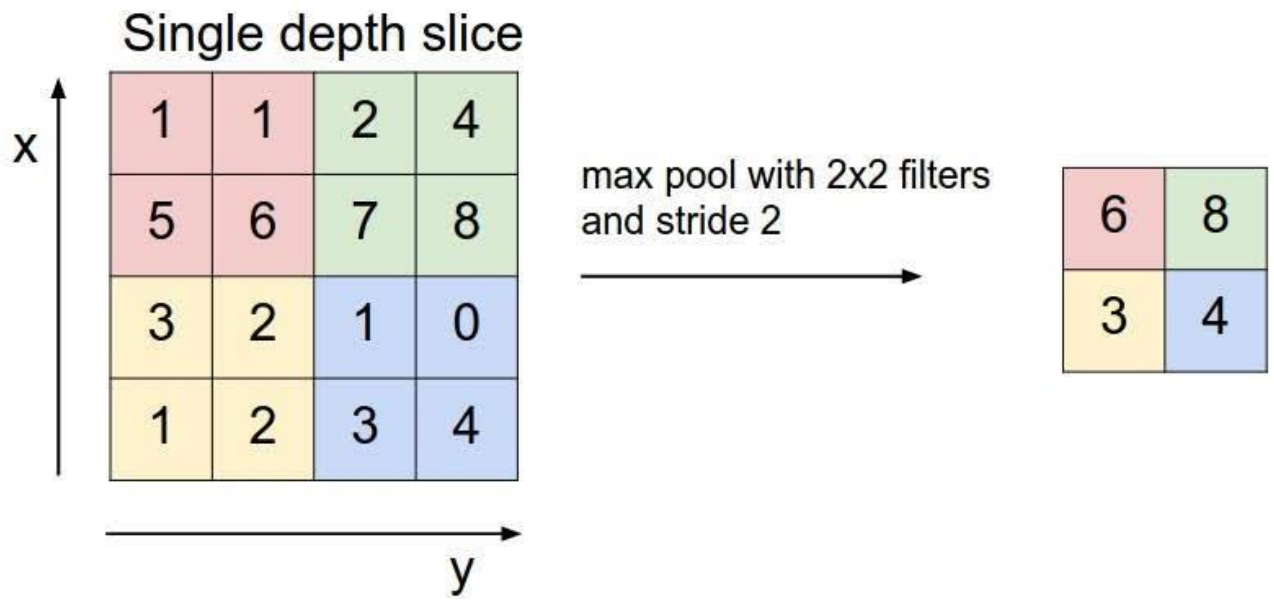

Figura 2-5 Max Pool [6]

Straturi Dense/Fully connected pe ultimele straturi dintr-o rețea convoluțională vom găsi straturi dense(fully connected) care preiau trăsăturile, determinate de straturile convoluționale și clasifică imaginile. Intre straturile convoluționale și acestea mai exista un strat de liniarizare a informație care liniarizează imaginile venite din straturile convoluționale pentru a putea fi procesate de straturile fully connected. Ultimul va un număr de neuroni egal cu numărul de clase după care se dorește clasificarea.

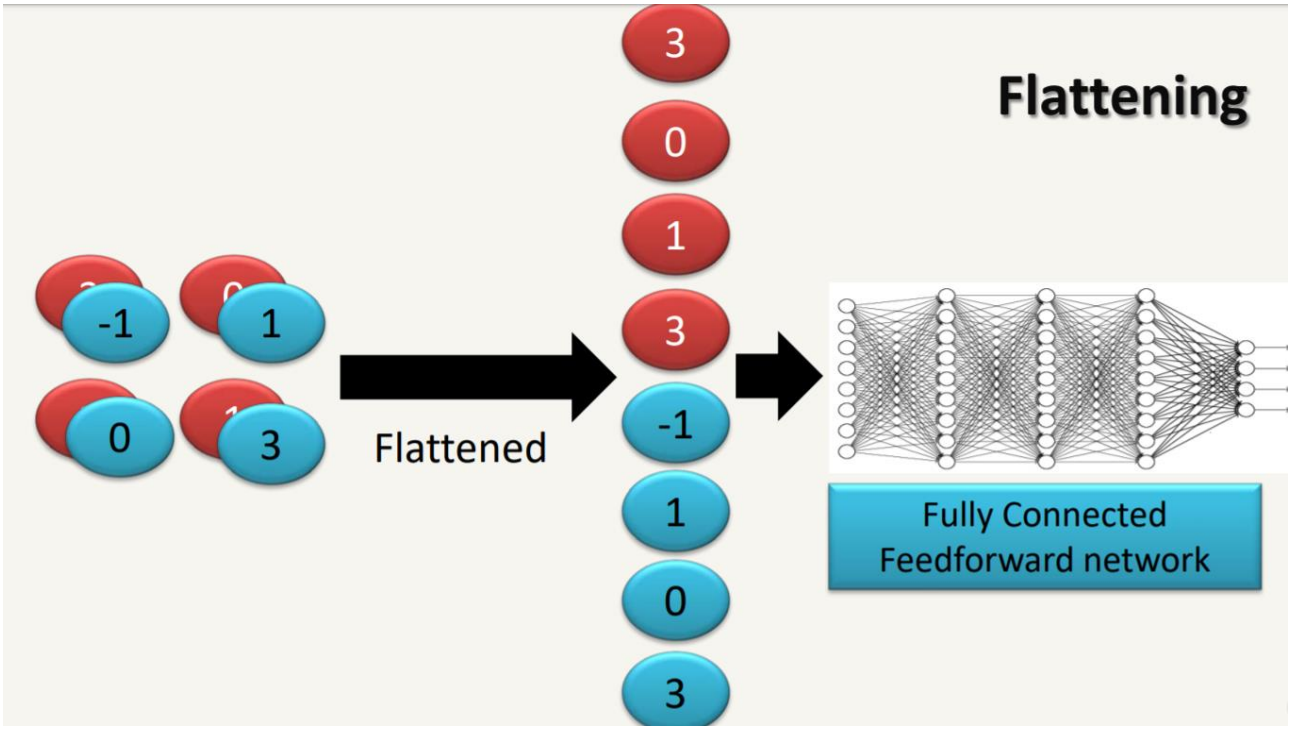

Figura 2-6 Liniarizarea straturilor convoluționale [5]

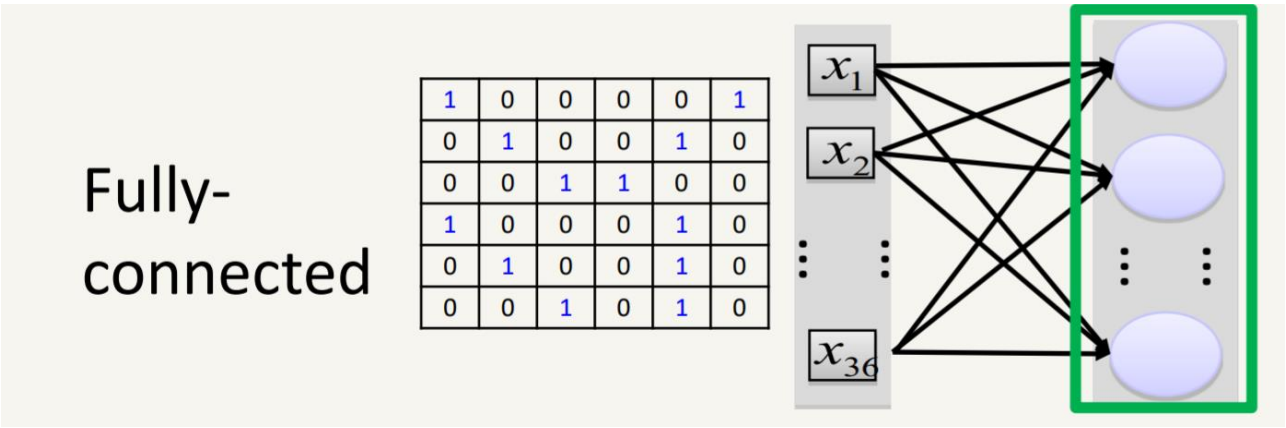

Figura 2-7 Fully connected [5]

Cele 3 tipuri de straturi menționate anterior se combină in ordinea specificată pentru a forma diferite arhitecturi de rețele neuronale convoluționale după cum se poate observa și in imaginea de mai jos.

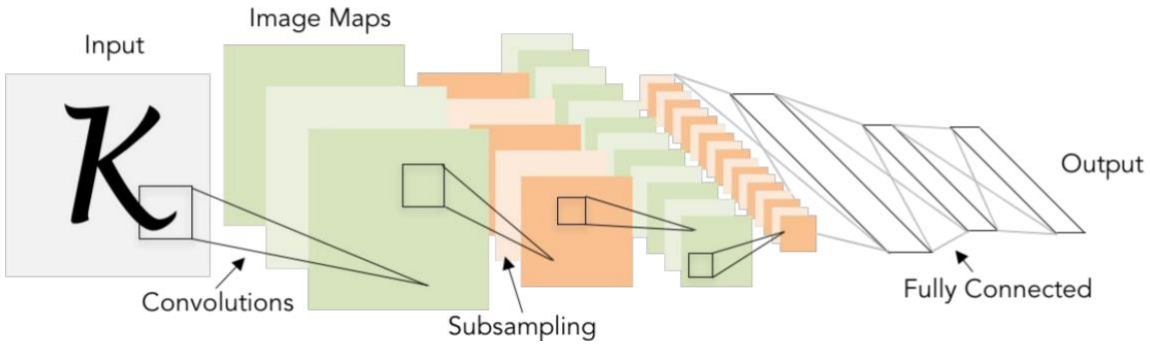

Figura 2-8 Rețea convoluțională [7]

2.2 Rețele recurente

Dacă o rețea feed forward, dens conectată primea la intrare ei toată informația, o rețea recurentă primește bucăți din informație la momente diferite de timp. Rețeaua e capabilă sa proceseze informația(cuvintele in cazul nostru) pe rând iar la momentul următor de timp, când aceasta va primi următorul cuvânt, să il proceseze, folosind aceleași ponderi și tinând cont de cuvintele pe care le-a primit anterior pentru a produce o ieșire care să clasifice întreaga secvență primită. Privită pe toată durata s-a de timp, adică pe lungimea vectorului de intrare(numar de cuvinte) ea poate fi convertită ca o rețea neuronală densă care după fiecare strat primește încă o bucată din informația de intrare, lucru ce se poate vedea și in Figura 2-10.

Principala problemă a acestor rețele este dată de lungimea secvenței. O dată cu trecerea timpului primele cuvinte care au intrat în rețea vor avea o pondere mai mică asupra rezultatului final fața de ultimele cuvinte. Această problemă se va reflecta și in procedeul de antrenare unde gradienții vor fi din ce in ce mai mici pe măsură ce ne întoarcem în timp pentru a modifica ponderile neuronilor.

Soluții precum LSTM(long-short term memory) și GRU au fost găsite pentru a păstra relevantă informația din pașii anteriori. Aceste unități sunt capabile să rețină informații de la aproximativ 1000 de cuvinte.

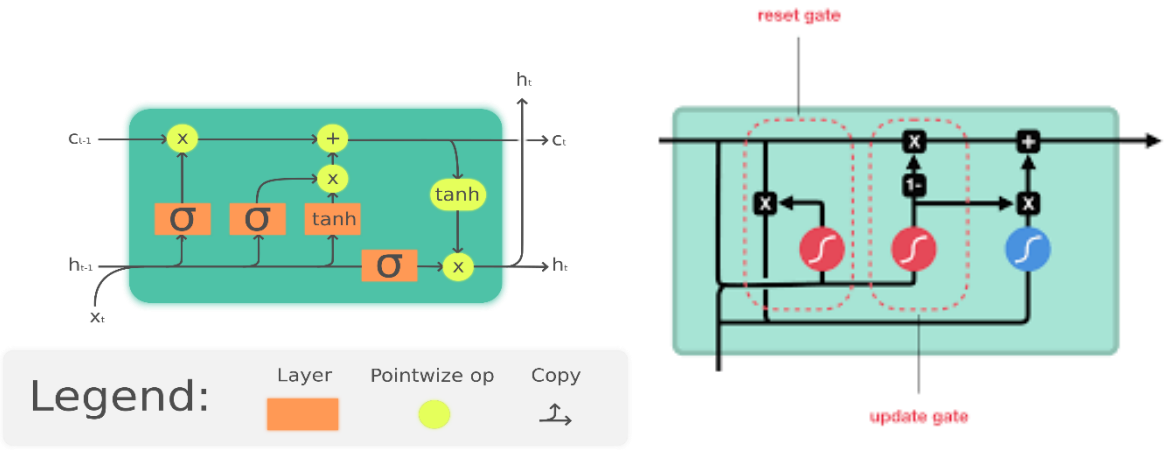

Figura 2-9 LSTM și GRU

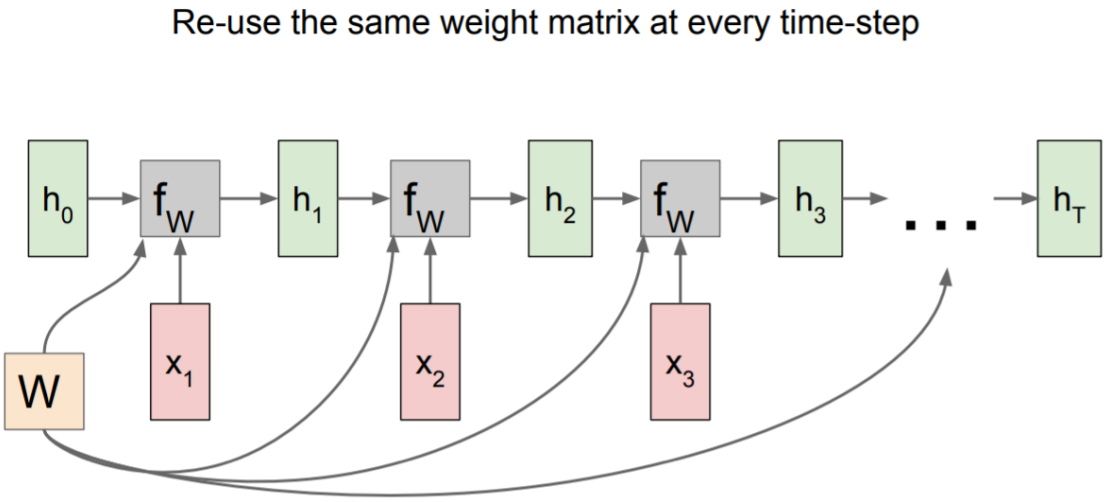

Figura 2-10 Rețea recurentă desfașurată in timp

În funcție de topologia rețelelor acestea pot avea diverse aplicații după cum se poate observa și in figura de mai jos.

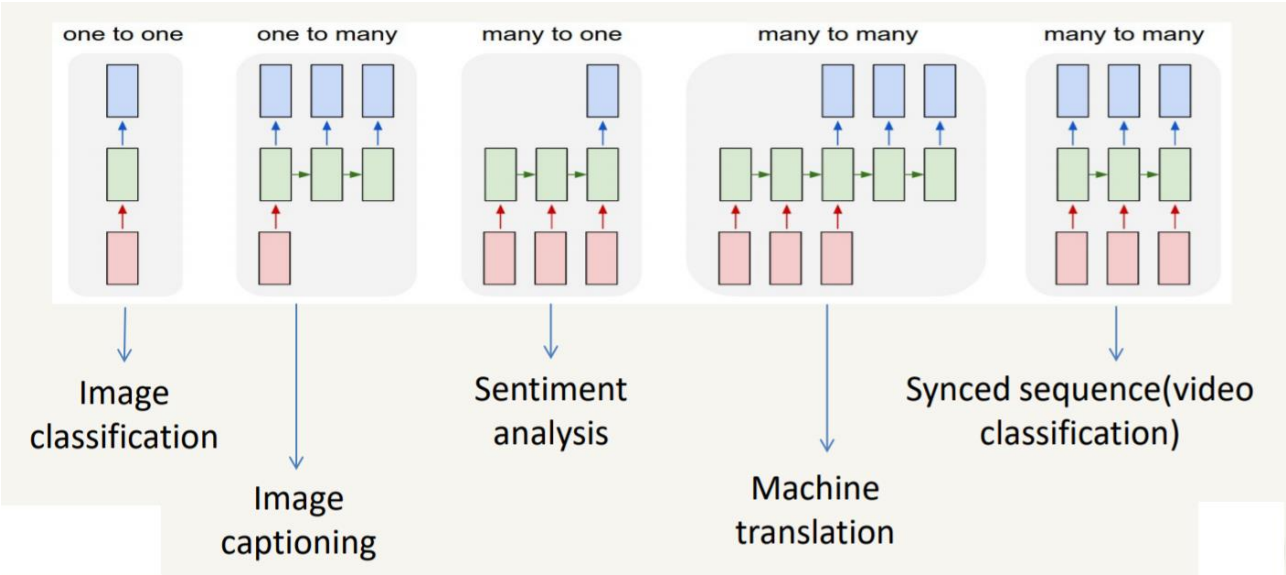

Figura 2-11Topologii de rețele recurente [8]

Bibliografie

[1] Google, „Androi Development,” Google, [Interactiv]. Available: https://developer.android.com/. [Accesat 18 1 2020].

[2] Android Authority, „Kotlin vs Java,” 18.10.2019. [Interactiv]. Available: https://www.androidauthority.com/kotlin-vs-java-783187/. [Accesat 18 1 2020].

[3] „Rețele neuronale în TensorFlow,” [Interactiv]. Available: http://myac.xhost.ro/inva/Lab9/InvA_Lab9.pdf. [Accesat 18 1 2020].

[4] HVASS-Labs, „Image Captioning,” [Interactiv]. Available: https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/22_Image_Captioning.ipynb.

[5] C. Florea, „Curs MLAV,” [Interactiv]. Available: http://www.master-taid.ro/Cursuri/MLAV_files/07_08_MLAV_ConvNets_CF.pdf.

[6] S. Tejani. [Interactiv]. Available: https://shafeentejani.github.io/assets/images/pooling.gif.

[7] J. J. S. Y. Fei Fei LI, „CS231,” Stanford, 2017.

[8] J. J. S. Y. Fei Fei Li, „cs231n_2017_lecture10,” 4 5 2017. [Interactiv]. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf.