UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Identificarea similarității vizuale a imaginilor folosind rețele neurale

propusă de

Petru-Marian Gafițescu

Sesiunea: iulie, 2019

Coordonator științific

Conf. Dr. Sabin-Corneliu Buraga

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Identificarea similarității vizuale a imaginilor folosind rețele neurale

Petru-Marian Gafițescu

Sesiunea: iulie, 2019

Coordonator științific

Conf. Dr. Sabin-Corneliu Buraga

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Sabin-Corneliu Buraga.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Gafițescu Petru-Marian** domiciliat în **România, jud. Suceava, Sat. Arbore(Com. Arbore), nr. 1012**, născut la data de **30 iunie 1998**, identificat prin CNP **1980630330527**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2019, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Identificarea similarității vizuale a imaginilor folosind rețele neurale** elaborată sub îndrumarea domnului **Conf. Dr. Sabin-Corneliu Buraga**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Identificarea similarității vizuale a imaginilor folosind rețele neurale**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

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

Absolvent **Petru-Marian Gafițescu**

Data: ..... Semnătura: .....

Cuprins

Rezumat

Această lucrare descrie scopul, implementarea și rezultatele aplicației web ”FlickrCloneFetch”, o aplicație web ce permite utilizatorului să se conecteze cu ajutorului contului de *Flickr*, să încarce în aplicație o imagine, iar apoi să vizualizeze imaginile similare din punct de vedere vizual. În funcție de preferințele utilizatorului, imaginile vor fi preluate de pe profilul său, imaginile publice ale contactelor sale sau imaginile publice relevante de pe platforma *Flickr*.

Modulul de *back-end* al aplicației este scris în Python, folosește Flask pentru a expune un API și Tensorflow și Keras pentru partea de rețele neuronale.*Front-end*-ul aplicației este realizat în Typescript, utilizând platforma Angular. Aplicația este găzduită în *cloud*, pe platforma Google Cloud Platform, această soluție fiind aleasă datorită fiabilității și posibilității de scalare.

Introducere

Ín zilele noastre, datorită faptului că majoritatea telefoanelor mobile au și funcția de cameră foto, imaginile au devenit principala formă de rememorare a evenimentelor trăite. Platforme întregi sunt dedicate fotografiilor, printre cele mai populare numărându-se Instagram, Flickr sau Pinterest. Multe din aceste platforme oferă posibilitatea de sortare, respectiv căutare a imaginilor în funcție de locație, mărime, data fotografiei sau descrierea adăugată de utilizator. Uneori aceste opțiuni nu sunt de ajuns, imaginile având mai multă relevanță din prisma conținutului decât după orice alte metadate. Totuși, opțiunea de a căuta imagini după similaritatea vizuală față de altă imagine, cu alte cuvinte, chiar după componența imaginii este o caracteristică rar întâlnită.

Aplicația "FlickrCloneFetch" își propune implementarea acestei caracteristici pentru platforma *Flickr*, oferind utilizatorului posibilitatea de a vedea imagini similare vizual cu o imagine încărcată atât de pe propriul profil, cât și de pe profilele contactelor sau chiar din imaginile publice de pe platformă.

Scopul "FlickrCloneFetch" este de a fi o unealtă utilă atât fotografilor care folosesc platforma (pentru a vedea lucrările din propriul profil care au aceeași tematică), cât și utilizatorului uzual (pentru a găsi fotografii legate de un anumit eveniment din viața sa, sau fotografii cu unele tematici alese).

Pentru realizarea funcționalității descrise au fost folosite diverse tehnologii web, precum Flak, Angular, împreună cu API-ul platformei *Flickr* și un model de rețea neurală convoluțională disponibil prin *framework*-ul Tensorflow.

Capitolul *Fundamente* va oferi o descriere a principalelor concepte, tehnologii și biblioteci folosite în implementarea aplicației.

Scopul urmărit, o privire de ansamblu asupra aplicației și comparația cu unele soluții deja existente se vor regăsi în capitolul *Scopuri și cerințe*.

Următorul capitol, *Analiză și proiectare* conține descrierea arhitecturii aplicației,

prezentarea componentelor necesare și a modului în care interacționează, precum și scenariile de utilizare ale aplicației.

În capitolul *Detalii de implementare* se prezintă aspecte privitoare la modul de implementare, motivarea alegerii modelului de rețea neurală, algoritmii și structurile de date folosite pentru a optimiza timpul de răspuns al aplicației, precum și procesul de integrare cu mediul *cloud*.

Capitolul *Manual de utilizare* prezintă modul de utilizare a aplicației , fiind urmat de *Concluzii*, o serie de concluzii legate de modul în care aplicația răspunde cerințelor și posibile direcții de dezvoltare ulterioară.

Capitolul 1

Fundamente

1.1 Rețea neurală

O rețea neurală este un ansamblu interconectat de elemente de procesare simple, *unități* sau *noduri*, a căror funcționalitate este ușor bazată pe neuronul biologic. Abilitatea de procesare a rețelei este stocată în conexiunile inter-unități, sau *ponderi* sinaptice, obținute dintr-un proces de adaptare la învățare dintr-un set de date de antrenare.[7]

În biologie, fiecare neuron primește informație prin dendrite și transmite informație prin axon, doar atunci când a primit suficientă informație de la neuronii conectați la dendritele sale. Astfel, un neuron va transmite informația mai departe atunci când cantitatea de informație primită depășește un anumit *prag*.

Neuronii artificiali sunt construiți pe același principiu: dendritele vor fi echivalate de conexiuni către unitate, iar axonul de către o conexiune pornind de la acea unitate. Fiecare sinapsă(conexiune inter-neuronală) va fi caracterizată de o anumită *pondere* astfel încât datele de intrare primite prin acea conexiune vor fi multipilicate cu acea pondere înainte de a fi transmise la neuronul artificial. Aici, semnalele ponderate sunt însumate pentru a forma o *activare*.

În cadrul nodului, dacă *activarea* va depăși un anumit *prag*, unitatea va transmite mai departe o valoare(de obicei 1), altfel nu transmite nimic. Compararea cu pragul de activare se face folosind o funcție de activare. Matematic, pentru datele de intrare reprezentate de $x_1..x_n$ și ponderilele reprezentate de $w_1...w_n$, neuronul artificial funcționează astfel:

$$output = \begin{cases} 0, \text{dacă } \sum_i x_i * w_i \leq prag \\ 1, \text{dacă } \sum_i x_i * w_i > prag \end{cases}$$

Procedeul descris (însumare, compararea cu pragul, apoi emiterea de 1 când pragul este mai mic și 0 altfel) reprezintă cel mai simplu model de neuron artificial, perceptronul (fig. 1.1).

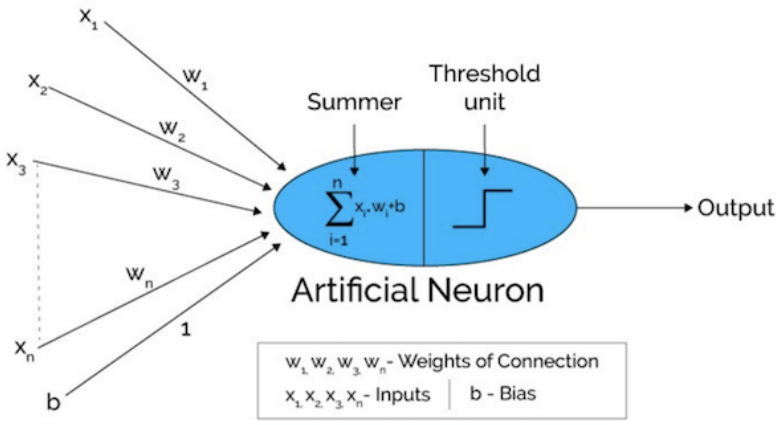

Figura 1.1: Un model de perceptron[5]

Modul de învățare a unui perceptron este foarte simplu: Dacă greșește în a clasifica un element (obține *output* 1 când valoarea elementului e 0, sau invers) creștem ponderile dacă activarea a fost prea mică comparativ cu pragul de activare, respectiv descreștem ponderile dacă pragul de activare a fost mai mic decât activarea. Astfel perceptronul își adaptează ponderile dinamic, în funcție de datele de antrenament, trecând printr-un proces de învățare.

De-a lungul istoriei, au fost folosite mai multe tipuri de neuroni, fiecare având avantajele și dezavantajele sale. Cum corpul neuronului este în principiu un sumator ponderat, ce se va schimba între tipurile de neuroni este funcția de activare.

Deoarece de unul singur un neuron nu are putere de decizie prea mare, o rețea neurală va folosi mai mulți, ieșirile unor neuroni constituind intrări pentru alți neuroni. Neuronii vor comunica între ei prin activarea/lipsa activării, astfel imitând transmiterea de semnale electrice din cadrul creierului uman.

O rețea neurală cu mai multe straturi de neuroni mai este numită și *MultiLayer-Perceptron*, deși de obicei sunt folosiți neuroni de tip sigmoid, nu perceptron. O rețea de tip este organizată tipic pe mai multe straturi, având urmatoarea structură:

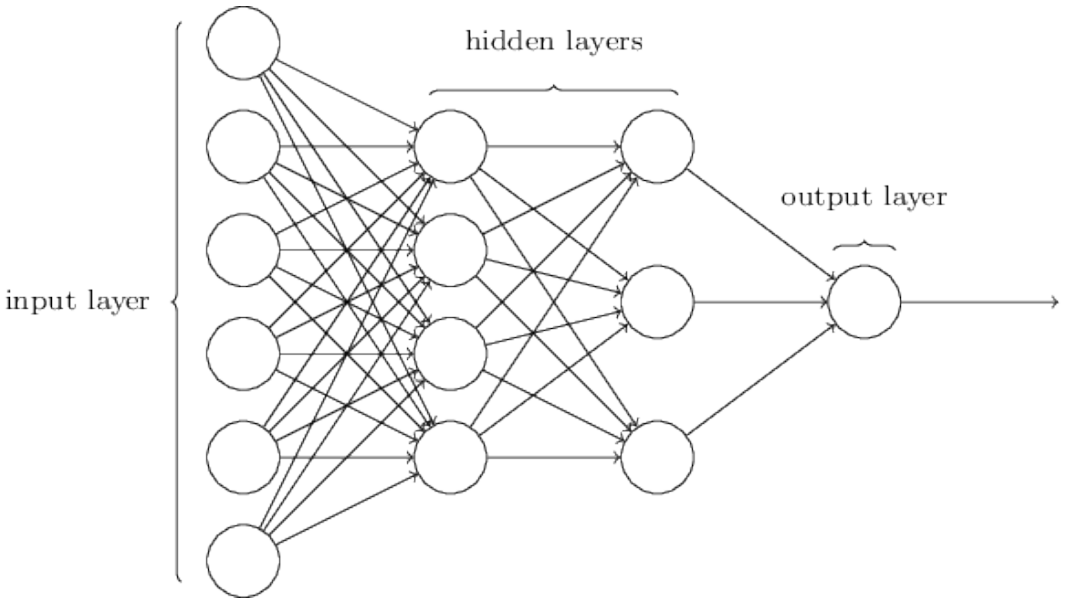

Figura 1.2: Structura unei rețele[9]

Straturile rețelei:

1. Stratul de intrare (*input*)

   Stratul de *input* este format din neuroni de intrare, care preiau datele de intrare și le transmit mai departe rețelei. De cele mai multe ori, neuronii de intrare nu au ponderi și nici funcții de activare, având un rol pasiv în cadrul rețelei.

2. Stratul/Straturile de procesare (*hidden*)

   Straturile *hidden* se ocupă de procesarea datelor de intrare. Există diferite moduri de a organiza straturile ascunse ale rețelei, în funcție de tipul rețelei: în cadrul unei rețele *feedforward* doar vor transmite datele prelucrate,iar în cadrul unei rețele convoluționale ce se ocupă de procesarea imaginilor vor exista straturi convoluționale și de *pooling* pentru a descoperi anumite trăsături în imagine, iar la rețelele de tip recurent, în straturile ascunse vor apărea conceptele de memorie și stări.

   În toate cazurile, ideea de bază este ca straturile *hidden* vor prelua datele de intrare neprelucrate de la stratul de input, apoi vor prelucra acele date conform ponderilor asignate și calculate în diverse moduri, pasând apoi datele procesate ultimului strat, cel de ieșire. Numărul de straturi*hidden*, organizarea lor, precum și tipurile de straturi și neuroni folosite depind atât de structura rețelei cât și de scopul în care este folosită aceasta.

3. Stratul de ieșire (*output*)

   Neuronii din stratul de ieșire pot fi proiectați diferit față de straturile anterioare, reprezentând actorii finali ai rețelei și având rolul de a produce outputul programului. Astfel, stratul de ieșire se coalizează și produce în mod concret rezultatul final, cu rol în eficientizarea și îmbunătățirea acestuia. Referitor la cele menționate anterior, pentru a înțelege mai bine rețeaua neurală, cele trei nivele de straturi(stratul de intrare, straturile ascunse și stratul de ieșire)vor fi privite împreună ca întreg.

Pe lângă straturile descrise, o rețea neuronală mai are o funcție de cost folosită pentru a aproxima diferența dintre valoarea prezisă de rețea și valoarea reală. Mecanismul de antrenare utilizat la perceptron nu funcționează și aici, deoarece eroarea nu se propagă decât pe ultimul strat ascuns. Antrenarea unei rețele neurale are ca scop principal reducerea valorii funcției de cost, existând mai multe tehnici pentru a îmbunătați acest procedeu. Un rol foarte important în antrenarea unei rețele îl are algoritmul de *Backpropagation*.

Backpropagation Când antrenăm un simplu perceptron, îi putem modifica instant ponderile în funcție de eroarea obținută (valoarea funcției de cost). În cazul unei rețele eroarea trebuie corectată la toate nivelele, nu doar la ultimul. Aici intervine algoritmul de *Backpropagation*. Algoritmul funcționează astfel:[4]

Pentru fiecare strat *l*, începând cu ultimul:

Pentru fiecare neuron *i* de pe stratul *l*:

1. Calculăm eroarea neuronului *i* de pe stratul *l*
2. Folosind eroarea calculată calculăm derivatele parțiale ale funcției de cost în funcție de pragul neuronului *i* și ponderile sale (derivatele parțiale ne arată cum variază funcția de cost în funcție de parametrii aleși)
3. Modificăm ponderile și/sau pragul pentru a corecta eroarea pentru acel neuron
4. Propagăm eroarea către neuronii de pe stratul *l-1* și repetăm pașii de mai sus

Atât în timpul antrenării, cât și în timpul folosirii unei rețele (trecerea datelor de intrare prin rețea pentru a obține date de ieșire), se folosește foarte mult înmulțirea de matrici (pentru a înmulți ponderile neuronilor cu datele de intrare). Pentru anumite cazuri de input, putem optimiza numărul de parametri și procesul de funcționare. În continuare vom prezenta un tip de rețea optimizată în mod special pentru imagini.

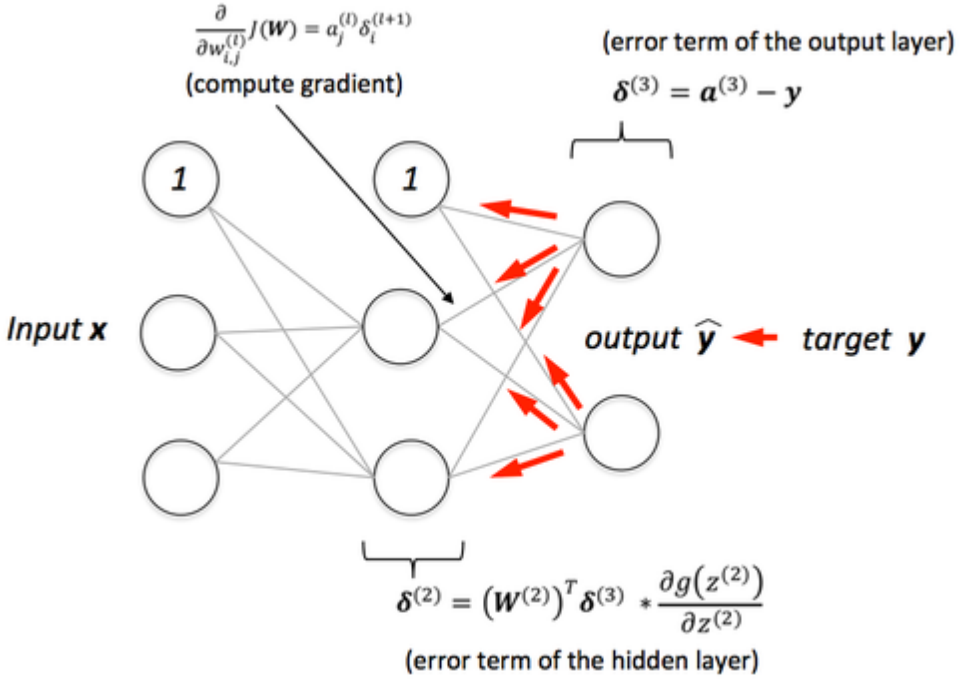

Figura 1.3: Algoritmul de Backpropagation[10]

1.2 Rețele neurale convoluționale

O rețea neurală convoluțională este un tip specializat de rețea neurală pentru procesarea datelor organizate sub o formă de tablou (un tablou unidimensional pentru datele temporale, un tablou bidimensional de pixeli pentru imagini) care folosește o operație matematică specializată numită convoluție în locul înmulțirii generale de matrici.[6]

O rețea convoluțională are anumite tipuri de straturi care nu apar într-o rețea clasică: straturi convoluționale și straturi de *pooling*, dar la final apar și straturi conectate(*fully-connected*) similare cu cele clasice. În cadrul straturilor convoluționale și celor de *pooling* neuronii din fiecare strat vor fi conectați doar la o mică regiune din stratul precedent, în loc de conexiuni între toți neuronii cum aveam la o arhitectură clasică.

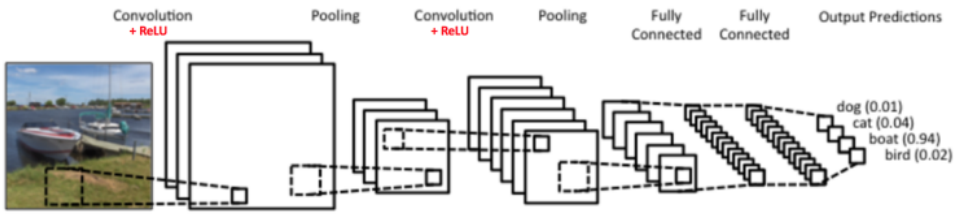

Figura 1.4: Arhitectura unei rețele convoluționale[1]

1.2.1 Convoluția

În cadrul unui strat de convoluție se folosește un filtru(sau *kernel*) pentru a extrage trăsături din cadrul imaginii. Filtrul se aplică unei mici regiuni din stratul precedent, realizând o mapare a imaginii în funcție de acel filtru. Cum imaginea este un tablou bidimensional de pixeli, un filtru va fi reprezentat tot de către un tablou bidimensional, mai mic decât cel de intrare. Aplicarea filtrului presupune calcularea produsului scalar între o parte din imagine de mărimea filtrului și filtru, efectuând această operație pentru toate părțile de aceeași mărime din imagine, luate la rând.

Un exemplu de aplicare a unui filtru pe o imagine:

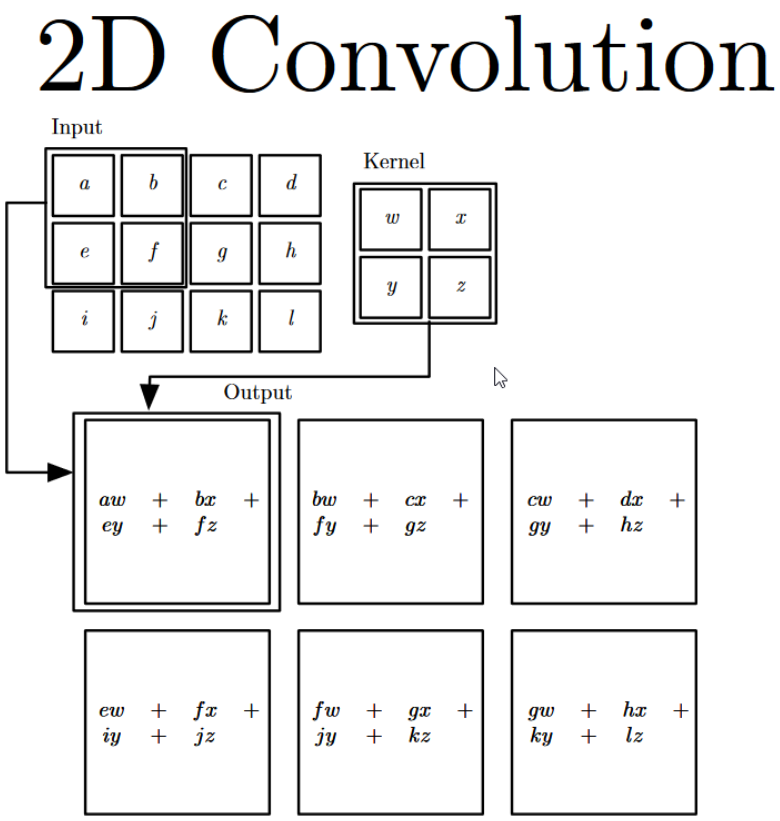

Figura 1.5: Operația de convoluție [6]

1.2.2 *Pooling*

Straturile de *pooling* (denumite și straturi de *subsampling* sau *downsampling*) vor prelua câte o mapare a imaginii dată de un filtru convoluțional și vor reduce dimensiunea fiecărei mapări, extrăgând astfel trăsăturile dominante. Pot exista mai multe tipuri de straturi de *pooling*: *MaxPooling*, *AveragePooling*, etc. De exemplu, în cazul operației

de *MaxPooling* ne alegem o zonă din mapare, din care extragem doar elementul maxim, astfel reducând dimensiunea mapării.

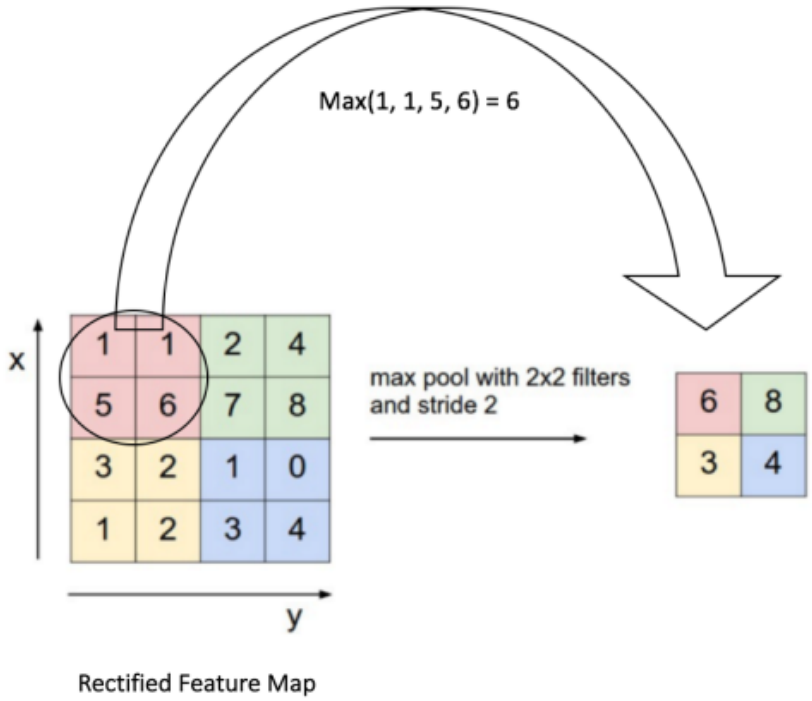

Figura 1.6: Exemplu de *MaxPooling*[1]

1.2.3 *Clasificare*

Adăugarea unui strat *fully-connected* ajută la clasificarea imaginii.Stratul de clasificare este de fapt un *MLP* tradițional, cu toți neuronii din el având conexiuni cu toți neuronii din stratul precedent. Stratul de clasificare va prelua trăsăturile de nivel înalt din imagine furnizate de straturile convoluționale și reduse ca dimensiune de straturile de pooling și va realiza o clasificare a imaginii bazată pe trăsăturile întâlnite.

Există multe tipuri de arhitecturi de rețele convoluționale, însă mai toate își au bazele pe operațiile descrise: convoluție, scalarea mapării(*pooling*), apoi clasificare.

1.3 Keras și Tensorflow

1.3.1 Tensorflow

Tensorflow este un *framework* cu sursă deschisă dezvoltat de Google și folosit pentru calcule numerice. Tensorflow oferă suport pentru învățare automată și *deep*

learning. Folosind diverse instrumente puse la dispoziție, utilizatorii pot construi modele de învățare automată la nivele diferite de abstractizare, de la modele construite pe baza unor serii de operații matematice la API-uri de nivel înalt pentru construirea de arhitecturi întregi prerealizate, cum ar fi rețele neurale sau regresori liniari.

1.3.2 Keras

Keras reprezintă un API de nivel foarte înalt dedicat rețelelor neurale, scris în Python și care poate rula pe mai multe platforme de învățare automată cum ar fi TensorFlow, CNTK sau Theano. A fost dezvoltat pentru a abstractiza la maximum procesul de creare a unei rețele neurale, având ca scop atât experimentarea rapidă, cât și construcția într-un mod facil a unei arhitecturi solide. Keras poate fi văzut ca o interfață simplă, consistentă, folosită pentru cazurile uzuale, ce se pliază peste un *back-end* optimizat de *deep learning* și învățare automată. Este extrem de modular, permițând reutilizarea de blocuri configurabile, dar și extensibil, utilizatorii având posibilitatea de a crea noi straturi, funcții de cost sau arhitecturi pentru idei de cercetare.

1.4 OAuth

OAuth este un protocol deschis ce permite o metodă pentru o aplicație client să acceseze diverse resurse protejate de pe un server fără ca utilizatorul să își dezvăluie credențialele de pe server aplicației client.[8]

Procesul de autentificare OAuth folosește două tipuri de *token*[3]:

1. *Request token*

   Folosit de aplicația client pentru a cere utilizatorului să autorizeze accesul la resursele protejate. *Token*-ul autorizat de utilizator este folosit o singură dată, este recomandat să expire după o limită de timp și va fi schimbat cu un *token* de acces.

2. *Access token*

   Folosit de aplicația client să acceseze resursele protejate în numele utilizatorului.

Pașii folosiți în protocolul OAuth:

1. Aplicația trimite o cerere semnată pentru a primi un *Request token*
2. Serviciul furnizează aplicației *token*-ul solicitat

3. Aplicația redirectează utilizatorul spre o pagină de autorizare, unde utilizatorul își dă acordul, apoi se revine în contextul aplicației

4. Printr-o nouă cerere semnată, aplicația schimbă *Request Token*-ul cu un *token* de acces și un secret al acestui *token*

5. Aplicația folosește t*token*-ul de acces și secretul acestuia pentru a accesa resursele protejate

Figura 1.7: Autentificarea prin protocolul OAuth[2]

1.5 Angular

Angular este un *framework* cu sursă deschisă dezvoltat de către Google pentru crearea de aplicații client SPA (*single-page application*) folosind HTML, CSS și TypeScript. O aplicație Angular este compusă din module, componente și servicii.

1.6 Flask

Flask este un *microframework* WSGI(*Web Server Gateway Interface*) pentru realizarea de aplicații web, scris in Python. Este cunoscut pentru simplitatea sa, dar și pentru posibilitatea de a se scala pentru aplicații oricât de complexe.

1.7 Google Cloud Platform

Google CLoud Platform (GCP), oferită de Google, reprezintă o suită de servicii *cloud computing* și unelte de infrastructură ce oferă utilizatorilor posibilitatea de a crea, găzdui și scala diverse aplicații. GCP oferă diverse servicii, în domenii precum: calcul, învățare automată, inteligență artificială, *Internet of Things*, stocare, baze de date relaționale și nerelaționale, *big data* , etc.

Capitolul 2

Scopuri și cerințe

Capitolul 3

Titlul celui de-al treilea capitol

Amet venenatis urna cursus eget. Quam vulputate dignissim suspendisse in est ante. Proin nibh nisl condimentum id. Egestas maecenas pharetra convallis posuere morbi. Risus viverra adipiscing at in. Vulputate eu scelerisque felis imperdiet. Cras adipiscing enim eu turpis egestas pretium aenean pharetra. In aliquam sem fringilla ut morbi tincidunt augue. Montes nascetur ridiculus mus mauris. Viverra accumsan in nisl nisi scelerisque eu ultrices vitae. In nibh mauris cursus mattis molestie a iaculis. Interdum consectetur libero id faucibus nisl tincidunt eget. Gravida in fermentum et sollicitudin ac orci. Suscipit adipiscing bibendum est ultricies. Etiam non quam lacus suspendisse. Leo urna molestie at elementum eu facilisis sed odio morbi. Egestas congue quisque egestas diam in arcu cursus. Amet consectetur adipiscing elit ut aliquam purus.

3.1 Titlul secțiunii 1

Eros donec ac odio tempor. Facilisi morbi tempus iaculis urna id volutpat. Faucibus in ornare quam viverra orci sagittis eu. Amet tellus cras adipiscing enim eu turpis egestas. Integer feugiat scelerisque varius morbi. Platea dictumst vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Bibendum arcu vitae elementum curabitur. Eu nisl nunc mi ipsum faucibus. Id aliquet lectus proin nibh nisl condimentum id venenatis a. Cras adipiscing enim eu turpis egestas pretium. Quisque non tellus orci ac auctor augue mauris augue. Malesuada pellentesque elit eget gravida cum. Ut lectus arcu bibendum at. Massa id neque aliquam vestibulum morbi blandit. Posuere ac ut consequat semper viverra nam. Viverra adipiscing at in tellus integer feugiat

scelerisque varius morbi. Morbi enim nunc faucibus a pellentesque sit amet porttitor eget. Eu feugiat pretium nibh ipsum consequat nisl vel. Nisl purus in mollis nunc sed.

3.2 Titlul secțiunii 2

Elementum sagittis vitae et leo duis ut diam quam nulla. Purus sit amet volutpat consequat mauris nunc. Tincidunt augue interdum velit euismod in pellentesque massa. Nunc sed augue lacus viverra vitae congue. Porttitor leo a diam sollicitudin. Faucibus pulvinar elementum integer enim. Adipiscing bibendum est ultricies integer quis auctor elit. Blandit aliquam etiam erat velit scelerisque in. A iaculis at erat pellentesque adipiscing commodo elit at. Erat nam at lectus urna duis. Consequat ac felis donec et. Fermentum posuere urna nec tincidunt praesent semper feugiat nibh sed. Proin gravida hendrerit lectus a. Pretium viverra suspendisse potenti nullam ac tortor vitae purus. Arcu cursus euismod quis viverra nibh cras pulvinar mattis. Gravida arcu ac tortor dignissim convallis aenean. Quam nulla porttitor massa id neque aliquam vestibulum morbi. Sed viverra ipsum nunc aliquet. Quis enim lobortis scelerisque fermentum dui faucibus in.

Concluzii

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc mattis enim ut tellus elementum sagittis vitae et. Placerat in egestas erat imperdiet sed euismod. Urna id volutpat lacus laoreet non curabitur gravida. Blandit turpis cursus in hac habitasse platea. Eget nunc lobortis mattis aliquam faucibus. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Viverra maecenas accumsan lacus vel facilisis volutpat est. Non odio euismod lacinia at quis risus sed vulputate odio. Consequat ac felis donec et odio pellentesque diam volutpat commodo. Etiam sit amet nisl purus in. Tortor condimentum lacinia quis vel eros donec. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt. Aliquam id diam maecenas ultricies mi eget mauris pharetra. Enim eu turpis egestas pretium.

Bibliografie

[1] ***. An intuitive explanation of convolutional neural networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

[2] ***. Oauth bible. http://oauthbible.com/.

[3] ***. Oauth core 1.0 revision a. https://oauth.net/core/1.0a/.

[4] Răzvan Benchea. Neural networks. https://sites.google.com/view/rbenchea/neural-networks?authuser=0.

[5] Adi Chris. From perceptron to deep neural nets. https://becominghuman.ai/from-perceptron-to-deep-neural-nets-504b8ff616e.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Kevin Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., 1997.

[8] Eran Hammer-Lahav. The oauth 1.0 protocol. RFC 5849, RFC Editor, April 2010. http://www.rfc-editor.org/rfc/rfc5849.txt.

[9] Michael A. Nielsen. *Neural Networks and Deep Learning*. De3termination Press, 2015.

[10] Sebastian Raschka. A visual explanation of the back propagation algorithm for neural networks. https://www.kdnuggets.com/2016/06/visual-explanation-backpropagation-algorithm-neural-networks.html.