Recunoașterea cifrelor scrise de mână folosind rețele neuronale adânci

Autori: Golban Igor, Melinte Tudor, Onița Daniela Marcela
Profesor coordonator: Lect. univ. dr. Bîrluțiu Adriana
Instituție: Universitatea "1 Decembrie 1918" din Alba Iulia

Abstract

We present an application for recognizing handwritten digits in real time. It uses deep learning techniques and it is built using Keras and TensorFlow libraries combined with OpenCV. The application works as follows. First, using the built-in camera, it reads the image from the middle green square and it crops it into a different window converting its pixels into greyscale. This allows to easily find the outer edges of largest form creating another square surrounding and tracking it (like face recognition). Second, the form is centered and scaled. Third, the neuronal network analyzes and predicts the output in real time giving certain percentage to each digit. The digit with the highest percentage is chosen. The output can be seen in real time under the main window.

1. Introducere

La nivelul acestei lucrări este descrisă baza pentru dezvoltarea unei aplicații complexe de recunoaștere a cifrelor scrise de mână. Aplicația citește o cifră pe care utilizatorul o pune în fața camerei web, analizează și prezice în timp real procentajul pentru fiecare cifră, de la 0 la 9. Predicția corectă a cifrei se va alege pe baza celui mai mare procentaj returnat.

Aplicația a fost implementată în limbajul de programare Python. Pentru definirea și antrenarea rețelei au fost folosite librăriile Keras și TensorFlow, iar pentru recunoașterea cifrelor a fost folosită biblioteca OpenCV.

Aplicația poate fi folosită pentru citirea codurilor poștale de pe scrisori, sau citirea cifrelor de pe diferite documente, cum ar fi facturi. De asemenea, printr-o implementare ulterioară corespunzătoare, aplicația ar putea fi folosită pentru detectarea și recunoașterea fețelor, pentru identificarea obiectelor, clasificarea acțiunilor umane în videoclipuri, găsirea

imaginilor similare dintr-o bază de date de imagini, etc.

2. Tehnologii folosite

2.1 Rețele neuronale adânci (en. Deep Learning)

Rețele neuronale adânci sunt un subcâmp al mașinilor instruibile (*Machine Learning*), care utilizează algoritmi inspirați din structura și funcțiile creierului. Acești algoritmi sunt mai buni și mai ușor de folosit comparativ cu alți algoritmi de învățare automată. Deep Learning este o inovație a învățării automate datorită performanței ridicate raportată la cantitatea de date.

Relația dintre Deep Learning, Machine Learning și inteligența artificială este prezentată in figura de mai jos:

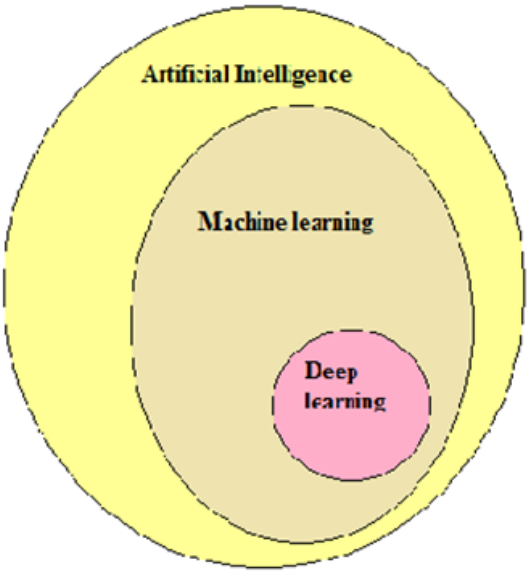

Figura 1: Legătura dintre deep learning, machine learning și inteligența artificială

Rețelele neuronale sunt inspirate din rețelele neuronale biologice. Acestea sunt folosite pentru a estima funcții care depind de un număr mare de inputuri. Rețelele neuronale reprezintă sisteme de interconectări de „neuroni” care schimbă mesaje între ele.

Modelele sunt construite prin setarea unui strat de intrare (*input layer*), care comunică cu unul sau mai multe starturi ascunse (*hidden layers*), care la rândul lor sunt legate de layer-ul de ieșire (*output layer*). [3]

Legătura dintre deep learning și rețelele neuronale este faptul că deep learning reprezintă un set de tehnici de învățare în rețele neuronale, în care există mai multe straturi.

2.2 Rețele neuronale convoluționale (*Convolutional neural networks - CNN*)

Rețelele neuronale convoluționale sunt o categorie a rețelelor neuronale, cu aplicabilitate în probleme de recunoaștere a imaginilor sau în probleme de clasificare. Acestea sunt un tip de rețele neuronale artificiale feed-forward în care modelul de conectivitate între neuroni este inspirat de organizarea cortexul vizual al animalelor.

CNN au câteva straturi/nivele de convoluții cu funcții de activare neliniare, precum ReLu (*Rectifield Linear Units*) sau tanh (*hyperbolic tangent*). Convoluțiile se folosesc peste stratul de intrare, pentru a calcula stratul de ieșire. Peste fiecare strat se aplica diferite filtre, după care se combină rezultatele acestora. În timpul etapei de antrenare, un CNN învață în mod automat valorile filtrelor sale în funcție de sarcina sa. De exemplu, dacă utilizăm CNN pentru clasificarea imaginilor, în primul strat se poate învăța detectarea marginilor pixelilor, apoi se vor detecta forme simple în al doilea strat, iar apoi aceste forme vor ajuta la detectarea caracteristicilor de nivel superior, cum ar fi formele faciale în straturile superioare. Ultimul strat este un clasificator care utilizează aceste caracteristici de nivel înalt. [10]

3. Partea experimentală

3.1 Setul de date folosit

A fost utilizat un set de date, MNIST [9], disponibil online, pentru a antrena rețeaua neuronală adâncă. Subsetul MNIST face parte dintr-un set mai mare, disponibil de la NIST (*National Institute of Standards and Tehnology*). Setul conține cifre scrise de mână, sub forma unui set de antrenare de 60.000 de exemple, precum și un set de test de 10.000 de exemple.

Imaginile din setul de date conțin nivele de gri, ca rezultat al tehnicii anti-aliasing utilizat de algoritmul de normalizare. Acestea au fost centrate într-o imagine de 28x28 prin calcularea centrului de masă al pixelilor și traducerea imaginii, astfel încât să poziționeze acest punct în centrul câmpului 28x28.

Figura 2 de mai jos prezintă câteva exemple de cifre din setul de date MNIST.

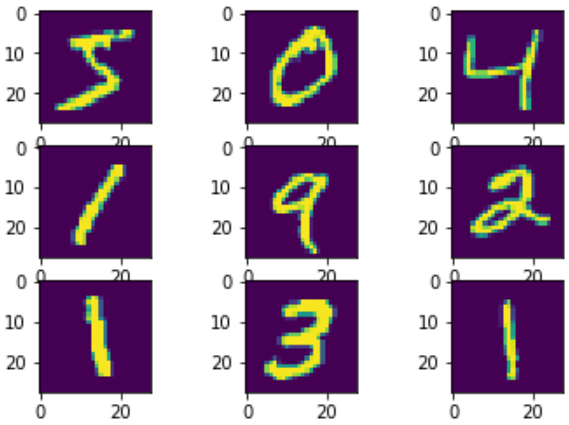

Figura 2: Exemplu de cifre din setul de date MNIST

3.2 Limbajul de programare și a librăriile utilizate

Limbajul folosit pentru implementarea problemei tratate la nivelul acestei lucrări este Python [5]. Acesta este un limbaj de programare popular, care oferă atât posibilitatea programării structurate, cât și a celei orientate pe obiect și include și elemente din paradigma funcțională. Este un limbaj de scripting, deci este interpretat, nu compilat, astfel economisind mult timp în procesul de dezvoltare și debugging.

Principalele biblioteci utilizate sunt: Keras [6] și OpenCV [8].

- **Keras** este o bibliotecă pentru rețele neuronale de nivel înalt API, scrisă în Python și capabilă să ruleze fie cu biblioteca TensorFlow [7], fie cu Theano. În acest caz Keras rulează cu TensorFlow.
- **TensorFlow** este o bibliotecă open source pentru calcule numerice, care utilizează grafice de flux de date (data flow). Nodurile din grafic reprezintă operații matematice, în timp ce marginile graficului reprezintă vectori de date multidimensionale (tensori), care comunică între ele.
- **OpenCV** este lansat sub o licență BSD, deci este gratuit atât pentru uz academic, cât și comercial. Are interfețe C++, C, Python și Java și suportă Windows, Linux, Mac OS, iOS și Android. OpenCV a fost proiectat pentru eficiență de calcul și cu un puternic accent pe aplicații în timp real. Biblioteca OpenCV conține peste 2500 de algoritmi, care pot fi utilizați pentru a detecta și recunoaște fețe, pentru a identifica obiecte, clasifica acțiunile umane în videoclipuri, mișcările camerei, obiecte în mișcare, extrage modele 3D ale obiectelor, găsirea imaginilor similare dintr-o bază de date de imagini, eliminarea ochilor roșii din imaginile realizate cu ajutorul flash-ului, etc.

4. Descrierea aplicației

4.1 Structură

Aplicația este formată din din doua părți:

- Formarea și antrenarea rețelei neuronale;
- Recunoașterea cifrei.

Rețeaua este formată din trei straturi dense care corespund în totalitate procesului de Machine Learning, unde toate output-urile unui layer sunt asociate cu input-urile următorului (Figura 4). Funcția de activare ReLU este folosită în primele două layere, iar pentru al treilea layer (de output) este folosită funcția softmax. Această funcție de activare este proiectată pentru a converti orice vector cu valori reale într-un vector de probabilități care este determinat pentru fiecare *j* neuron în felul următor:

$$\sigma(\vec{z})_j = \frac{exp(z_j)}{\sum_i exp(z_i)}$$

Figura 3: Determinarea vectorului de probabilități pentru fiecare *j* neuron

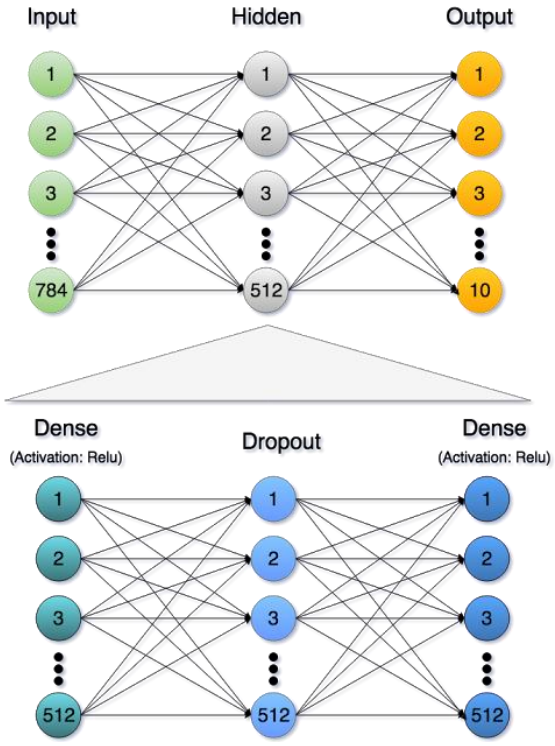

Figura 4: Schema modelului rețelei neuronale folosite

Definirea straturilor rețelei, formarea și antrenarea rețelei, scrise în limbajul de programare Python sunt redate in printscreen-ul de mai jos, în timp ce schema modelului rețelei este ilustrată în Figura 4.

```
#
#
#

inp = Input(shape=(height * width,)) # Our input is a 1D vector of size 784

hidden_1 = Dense(hidden_size, activation='relu')(inp) # First hidden ReLU layer
drop_1 = Dropout(0.2)(hidden_1)
hidden_2 = Dense(hidden_size, activation='relu')(drop_1) # Second hidden ReLU layer

out = Dense(num_classes, activation='softmax')(hidden_2) # Output softmax layer


model = Model(input=inp, output=out) # To define a model, just specify its input and output layers

#
#
#

model.compile(loss='categorical_crossentropy', # using the cross-entropy loss function
              optimizer='adam', # using the Adam optimiser
              metrics=['accuracy']) # reporting the accuracy

#
#
#

model.fit(X_train, Y_train, # Train the model using the training set...
          batch_size=batch_size, nb_epoch=num_epochs,
          verbose=1, validation_split=0.1) # ...holding out 10% of the data for validation
model.evaluate(X_test, Y_test, verbose=1) # Evaluate the trained model on the test set!

#
#
#

```

Figura 5: Printscreen al codului de formare și antrenare al rețelei neuronale

4.2 Rezultate experimentale

Figura 6 ilustrează rezultatul aplicației. Atunci când este detectată o cifră, se calculează probabilitatea ca cifra detectată să fie una din cele 10 cifre posibile (de la 0 la 9). Predicția cifrei corecte se realizează pe baza celei mai mari probabilități detectate.

În imaginea de mai jos s-a detectat probabilitatea cea mai mare pentru cifra 5, asta însemnând că rețeaua face predicția că cifra arătată camerei este cifra 5, deci predicția este corectă. Celelalte cifre au o probabilitate mică în acest caz.

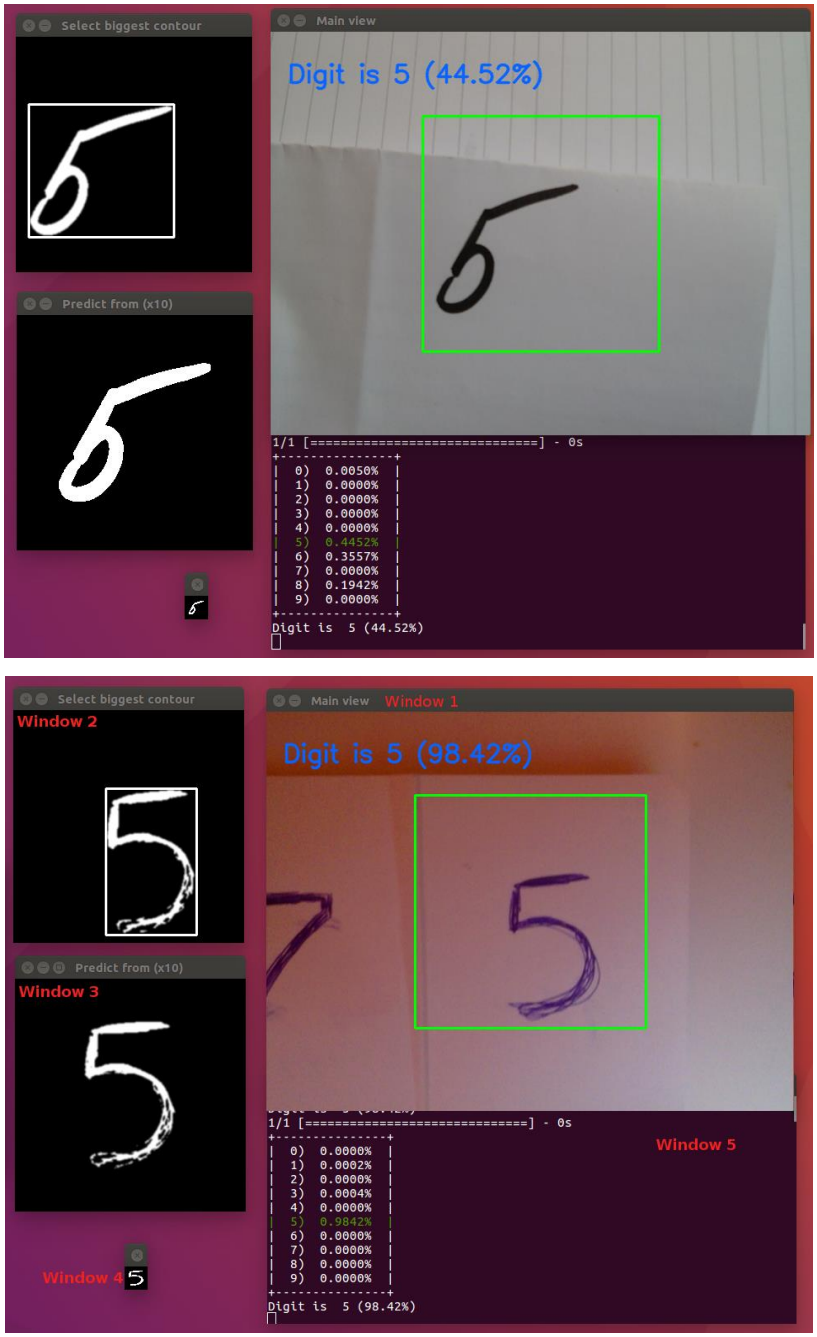

Figura 6: Printscreen al aplicației

Window 1 – Main View

- Rezultatul recunoașterii formei din interiorul chenarului verde (text albastru)

Window 2 – Select biggest contour

- Convertește imaginea din color in alb-negru
- Selecteaza cea mai mare forma din chenarul verde din Main view

Window 3 – Predict from

- Imaginea din care se face predicția

Window 4 –

- Ceea ce vede rețeaua neuronală

Window 5 – Terminalul

- Rezultatul detaliat al recunoașterii formei din interiorul chenarului verde

5. Concluzii și direcții viitoare de dezvoltare

Am prezentat o aplicație în timp real pentru recunoașterea cifrelor scrise de mână. Această aplicație poate fi considerată un punct de pornire pentru aplicații complexe de recunoaștere a obictelor, recunoașterea formelor, recunoaștere facială, etc.

Prin extinderea acestei aplicații, s-ar putea citi nu doar cifre, ci și litere, cuvinte sau fraze. O posibilă utilizare ar fi în identificarea unei persoane după scrisul specific, validarea semnăturilor și a falsului în scris (Art. 291 Codul Penal).

Bibliografie

[1] Jiawei Han and Micheline Kamber, "Data mining – Concepts and Techniques", Editura Morgan Kaufmann, 2006

[2] Lucrare de licență Onița Daniela Marcela, „Extragerea cunoștințelor din seturi mari de date utilizând tehnici de data mining și agenți inteligenți"

[3] Lucrare de licență Cristea Radu, "Detecția automată a anomaliilor în mamografii"

[4] Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep learning", 2017

[5] https://www.python.org/

[6] https://keras.io/

[7] https://www.tensorflow.org/

[8] http://opencv.org/

[9] http://yann.lecun.com/exdb/mnist/

[10]LeCun Y., Bengio Y., Hinton G.E. (2015) Deep learning. Nature, Vol. 521, pp 436-444.