Deep Hallucination Classification

Dobre Talida-Marina, grupa 242

April 2022

1 Cerinta Proiectului:

Clasificarea unor imagini in 7 categorii.

2 Prelucrarea Datelor:

Datele cu care lucram sunt impartite astfel:

- 3 fisiere cu extensia .txt, care contin id-ul si labelul imaginii(fisierul test contine doar id-uri). Setul de date este compus din 8000 de date de antrenare, 1173 de validare si 2819 pentru testare.
- 2 foldere cu imagini care vor fi utilizate pentru testare, validare si antrenare. Datele de antrenare si validare necesita separare.

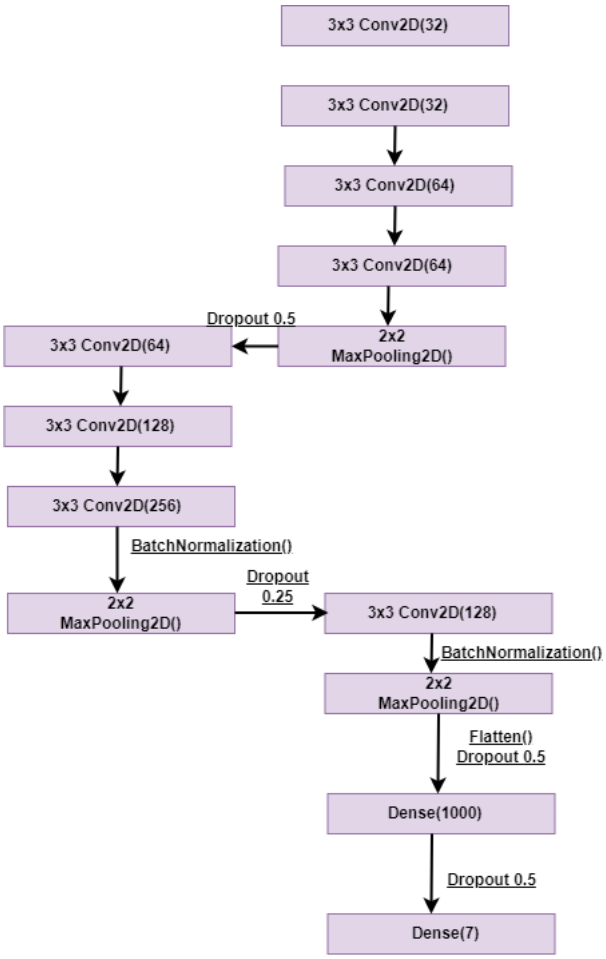
3 Cum am procesat datele :

Ca prim pas m-am ocupat de fisiere .txt, creand 3 liste de tip numpy array(train, test, validation), care contin fiecare subliste cu numele imaginii si categoria(exceptand test).
Pentru deschiderea imaginilor, am folosit functia Image.open(path + id). Am adaugat fiecare imagine intr-un numpy array corespunzator (*imagini_validation*, *imagini_train*, *imagini_test*). Am pastrat imaginile la forma initiala (16,16,3). Deoarece datele nu sunt imagini rezultate prin fotografie, ci sunt create prin alte metode, incercarea de a augumenta pozele (rotire, cresterea luminozitatii, crop, blurare si cresterea contrastului) a rezultat in scaderea acuratetii, astfel am ales sa las pozele in formatul lor original. Am folosit standardizare asupra datelor (preprocessing.MinMaxScaler) si normalizari, pentru a aduce pixelii la valori aflate in intervalul (0,1] (normalizare generata prin utilizarea functiei cv2.normalize()).
Am convertit fiecare categorie din vectorii care contin label-urile specifice fiecarui set de date intr-o matrice formata din valori binare(1 pentru pozitia corespunzatoare numarului ce reprezinta clasa), aceasta transformare fiind utila la construirea reteaua neuronala.

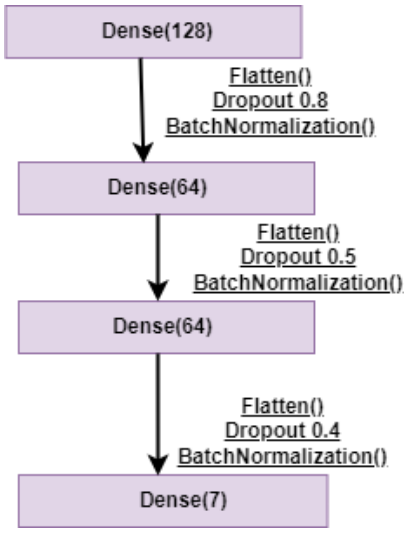
Exemplu:
[[0.0.0....0.1.0.] → corespunde categoriei 5
[0.0.0....0.1.0.] → corespunde categoriei 5
[0.0.0....0.0.1.] → corespunde categoriei 6
...
[0.1.0....0.0.0.] → corespunde categoriei 1
[1.0.0....0.0.0.]] → corespunde categoriei 0

4 Construirea Retelei Neuronale:

Primul model construit si cu cele mai bune rezultate este un CNN, care seamana foarte mult cu arhitectura AlexNet(care a fost de asemenea sursa de inspiratie). Am inceput de la o retea mai mica, ajungand prin adaugari si eliminari de layere la rezultatul actual. O extindere mai mare sau eliminarea unor layere scadea cu un procentaj semnificativ acuratetea, astfel am dedus ca modelul nu mai poate fi imbunatatit in aceste aspecte.
Pentru a preveni overfitting-ul am folosit functii precum BatchNormalization(), Dropout() (am variat procentajul neuronilor deconectati de la 0.5 la 0.25) si MaxPooling2D().
La compilarea retelei functia de optimizare care a dat rezultatele cele mai bune a fost Adam, iar functia de loss Categorical Crossentropy.
Schema Retelei Neuronale:



Al doilea model de retea neuronala este un MLP, Multi Layer Perceptron, care consta in cel putin 3 layere: input layer, hidden layer si output layer. Deoarece utilizeaza ca tehnica de invatare supervizata backpropagation, am ales sa implementez ca un alt model de rezolvare al task-ului, aceasta retea.
Pentru a preveni overfitting-ul am folosit functii precum BatchNormalization(), Dropout() (am variat procentajul neuronilor deconectati de la 0.8 la 0.5 iar apoi la 0.4).
La compilarea retelei functia de optimizare care a dat rezultatele cele mai bune a fost Adam, iar functia de loss Categorical Crossentropy.
Schema Retelei Neuronale:



Antrenare Modele:
Pentru prima retea am antrenat de trei ori consecutiv, alternand numarul de epoch si *batch_size* astfel:

- prima antrenare : epochs = 10, $batch\_size = 512$;
- a doua antrenare : epochs = 30, $batch\_size = 128$;
- a treia antrenare : epochs = 10, $batch\_size = 512$.

Functia de activare potrivita pentru acest model a fost Tanh. Pentru kernel size un numar > 3 ar fi fost prea mare.
Alte incercari de a aduce regularizari de tip l1, l2, $l1\_l2$ nu au adus rezultate mai bune si astfel am ales sa le omit.

Tot pe acest model de retea, dar pe antrenari diferite am obtinut rezultate multumitoare (nu cu acelasi succes precum cea anterioara). Ceea ce difera de incercarea anterioara este o a patra antrenare cu epochs = 10, $batch\_size = 512$ si adaugarea de regularizari($kernel\_regularizer, bias\_regularizer, activity\_regularizer$) de tipul l2, $l1\_l2$.

Pentru a doua retea am antrenat de doua ori consecutiv, alternand numarul de epoch si *batch_size* astfel:

- prima antrenare : epochs = 40, $batch\_size = 128$;
- a doua antrenare : epochs = 30, $batch\_size = 256$;

Functia de activare potrivita pentru acest model a fost Selu(alte functii de activare precum Relu si Tanh nu au avut rezultate mai bune). Pentru kernel size un numar > 3 ar fi fost prea mare.
Alte incercari de a aduce regularizari de tip l1, l2, $l1\_l2$ nu au adus rezultate mai bune si astfel am ales sa le omit si pentru acest model.

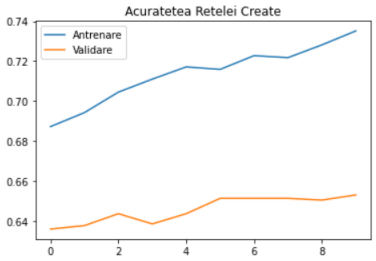
5 Rezultate:

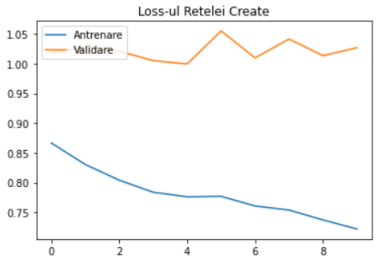
Pentru primul model CNN acuratetea se situeaza intre 0.62-0.65, iar pentru modelul cu regularizare intre 0.60-0.64.
Al doilea model MLP ofera rezultate mult mai mici decat primul model: 0.54-0.56.

Putem vedea cu claritate diferentele de acuratete si pierderile in urmatoarele grafice. De asemenea putem observa si matricile de confuzie ale celor 3 modele.

- CNN cu 3 antrenari si fara regularizare:
  Informatiile sunt corespunzatoare celor mai bune valori obtinute pe acest model → loss: 1.0267 - accuracy: 0.6530.
  Matricea de confuzie:
  [[156 12 7 5 14 15 7]
  [16 93 14 11 6 42 19]
  [3 1 122 4 4 7 1]
  [4 9 10 95 9 8 15]
  [1 1 7 3 119 7 5]
  [4 21 15 10 6 78 11]
  [12 6 4 22 9 20 103]]

  Grafice:

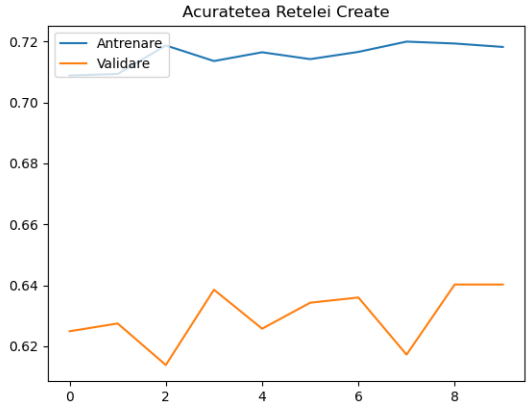


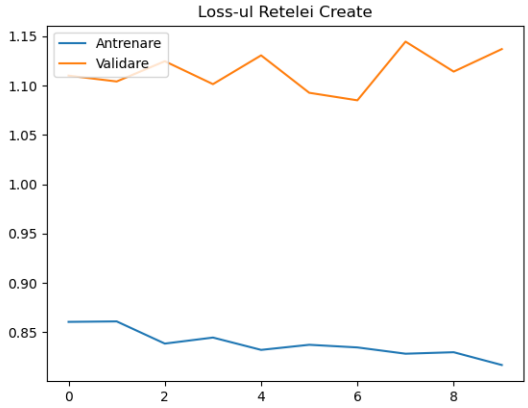


- CNN cu 4 antrenari si regularizare:
  Informatiile sunt corespunzatoare celor mai bune valori obtinute pe acest model → loss: 1.0843 - accuracy: 0.6402.

Matricea de confuzie:
[[155 12 10 9 13 6 11]
[18 105 14 17 8 10 29]
[2 3 123 4 6 3 1]
[2 11 12 97 7 3 18]
[2 3 8 6 116 1 7]
[4 39 17 10 7 50 18]
[17 9 5 25 7 8 105]]

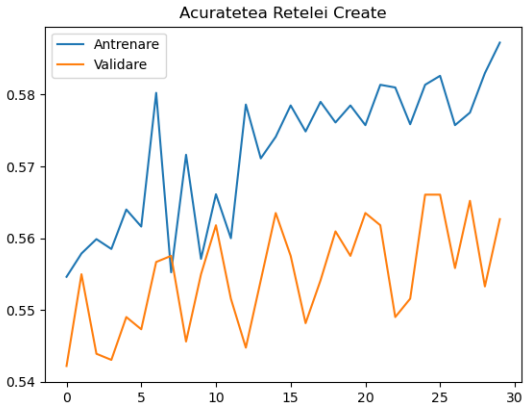
Grafice:

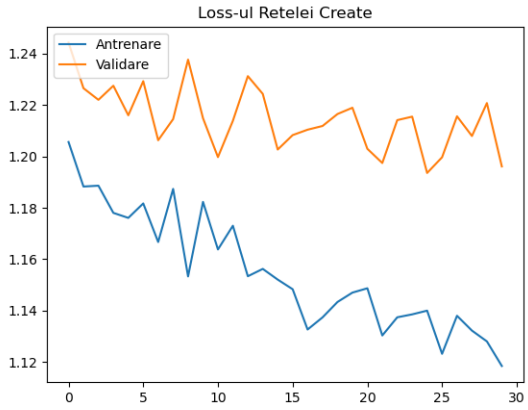




- MLP
  Datele sunt corespunzatoare celor mai bune valori obtinute pe acest model
  → loss: 1.2349 - accuracy: 0.5639.
  Matricea de confuzie:
  [[142 17 11 11 12 10 13]
  [22 74 20 30 3 28 24]
  [4 3 109 10 8 7 10]
  [1 14 8 99 4 7 17]
  [8 0 5 14 105 4 7]
  [8 25 16 19 6 56 15]
  [15 19 4 34 5 24 75]]

  Grafice:





6 Concluzii:

Modelele prezentate mai sus in urma testelor efectuate au dus la cele mai bune rezultate in indeplinirea task-ului proiectului. Este destul de vizibila diferenta intre cele doua modele CNN si MLP, cel din urma oferind o acuratete cu pana la 8% mai scazuta. Consider ca ambele modele se comporta multumitor pe datele primite, iar acuratetea rezultata este una satisfacatoare mai ales in cazul retelei de tip CNN, dar sunt sigura ca pot fi modificate ambele, incat sa ajunga la valori finale mult mai bune.