Monochrome Dreams Classification

Documentatie

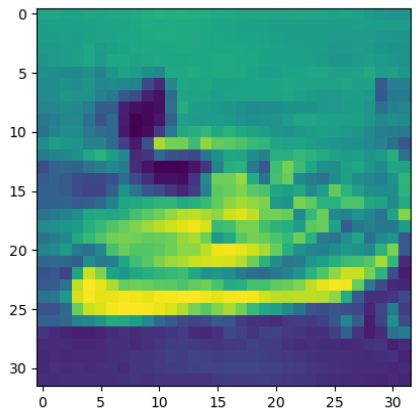
Paraschiv Alexandru-Andrei

Grupa 243

Pentru aceasta competitie am incercat 4 modele (**Naive Bayes, SVM, MLPClassifier si TensorFlow Sequential**), codul fiind scris in PyCharm(.py) si Google Colab(.ipynb transformat in .py ulterior). Dintre acestea, cele mai bune scoruri le-am obtinut prin SVM si TensorFlow Sequential.

Pentru modelele Naïve Bayes, SVM si MLPClassifier am folosit acelasi mod de citire si prelucare a datelor pana la inceperea modelarii modelului.

Astfel ca: Am o functie pentru a citi datele cu glob.glob, in care incarc pathname-ul imaginilor intr-o variabila si cu matplotlib.pyplot.imread citesc imaginea intr-un format binar si o adaug intr-un array (se imparte in mod automat fiecare pixel citit la 255). Apoi transform array-ul in numpy.array. Pentru a fi sigur ca citirea este buna, am afisat prima imagine si am comparat-o cu cea din datele oferite.



Dupa ce am citit toate imaginile din train, validation si test, iau shapeul acestora si le retin in trei variabile ca si numar de poze, lungime coloana si lungime linie (32x32).

Citesc informatiile din fisierele .txt. Pentru train.txt si validation.txt am cate un vector diferit pentru label si pentru numele imaginilor, iar pentru test.txt am doar un vector cu numele imaginilor. Apoi dau reshape la vectorul cu date pentru a il transforma din 3-dimensional array in 2-dimensiona array. Pentru reshape folosesc datele retinute anterior si aplic formula arr.respahe(nr_of_images, x_axis * y_axis) (adica pentru train de exemplu am 30001, 32*32).

Naive Bayes: Pentru acest model am folosit functia “values_to_bins” din laborator care primeste o matrice 2d si capetele de interval calculate cu linspace si calculeaza indexul intervalului corespunzator, folosind “digitize”. Linspace returneaza numere distantate uniform pe un interval dat.

Pentru acesta am folosit ca si parametric, 0 ca start si 1 ca si final, iar pentru distantarea “num_bins” am testat mai multe valori.

Am luat “num_bins” de la 3 la 14 si am obtinut valorile:

```
Accuracy for num_bins=3 is 0.358200
Accuracy for num_bins=4 is 0.375400
Accuracy for num_bins=5 is 0.385600
Accuracy for num_bins=6 is 0.387800
Accuracy for num_bins=7 is 0.389400
Accuracy for num_bins=8 is 0.387400
Accuracy for num_bins=9 is 0.391000
Accuracy for num_bins=10 is 0.388800
Accuracy for num_bins=11 is 0.391600
Accuracy for num_bins=12 is 0.392000
Accuracy for num_bins=13 is 0.390600
Accuracy for num_bins=14 is 0.390800
```

Am observant ca pentru num_bins valorile 11 si 12 sunt cele mai bune si am aplicat modelul pentru valoarea 12. Am discretizat multimea de antrenare, validare si testare folosind functia de mai sus, am definit un model MultinomialNB, am antrenat modelul pe datele de

train, am afisat un scor pentru datele de validare si am dat predict pentru datele de testare.

In cele din urma am afisat si predictiile, dar si matricea de confuzie, iar apoi am scris datele in fisier. Pe Kaggle obtinusem 0.38-0.39.

```
accuracy = 0.392
[8 1 5 ... 7 7 1]
[[ 68.  89.  27.  45.  86.  22.  43.  99.  91.]
 [ 17. 239.  23.  41.  15.  20.  28. 100.  44.]
 [ 23.  63. 154.  25.  47.  85.  34.  78.  24.]
 [ 21.  61.  10. 302.  31.  61.  26.  44.  22.]
 [ 22.  51.  45.  36. 184.  64.  21. 103.  28.]
 [  8.  15.  41.  54.  38. 324.  14.  51.  16.]
 [ 54.  63.  12.  46.  82.  26. 157.  66.  74.]
 [ 23.  53.  20.  32.  57.  20.  20. 280.  15.]
 [ 55.  45.  16.  46.  50.  42.  37.  34. 252.]]
```

SVM: In acest model am definit o functie compute_accuracy ca in laborator care calculeaza scorul predictiilor prin impartirea numarului de etichete corecte la numarul total de etichete.

Apoi am definit un model svm.SVC cu C = [1, 21], kernel = “rbf” si gamma = “scale” (default).

Pentru aceste valori ale lui C am obtinut:

```
SVM model accuracy for C=1 is 0.735000
SVM model accuracy for C=2 is 0.750200
SVM model accuracy for C=3 is 0.754400
SVM model accuracy for C=4 is 0.756200
SVM model accuracy for C=5 is 0.755800
SVM model accuracy for C=6 is 0.754200
SVM model accuracy for C=7 is 0.754200
SVM model accuracy for C=8 is 0.755400
```

```
SVM model accuracy for C=9 is 0.755800
SVM model accuracy for C=10 is 0.755400
SVM model accuracy for C=11 is 0.755600
SVM model accuracy for C=12 is 0.757000
SVM model accuracy for C=13 is 0.756800
SVM model accuracy for C=14 is 0.756600
SVM model accuracy for C=15 is 0.756000
SVM model accuracy for C=16 is 0.757000
SVM model accuracy for C=17 is 0.757400
SVM model accuracy for C=18 is 0.757600
SVM model accuracy for C=19 is 0.757800
SVM model accuracy for C=20 is 0.757600
SVM model accuracy for C=21 is 0.757200
```

Am observant ca pentru C = 4 si C = 9 sunt cele mai bune valori, pentru C > 11 fiind overfitting pe datele de testare.

Apoi am antrenat modelul pentru cele doua valori ale lui C, am luat scorul pentru datele de validare si am pastrat C = 4. La final am afisat acuratetea pe datele de validare, matricea de confuzie si am scris datele in fisier. Pe Kaggle am obtinut pentru acest model 0.76.

```
SVM model accuracy for C=4 is 0.756200
[[359.  19.  18.  14.  47.   4.  33.  42.  34.]
 [ 22. 436.  10.  10.   7.   6.   7.  23.   6.]
 [ 15.  28. 392.  16.  29.  27.   4.  18.   4.]
 [ 26.  16.  19. 417.  19.  23.  15.  25.  18.]
 [ 35.  20.  25.  26. 399.  10.   3.  25.  11.]
 [  7.   6.  16.  24.  15. 466.   5.  20.   2.]
 [ 26.  13.   8.  11.   8.   5. 470.   7.  32.]
 [ 41.  14.  18.  25.  27.  14.   9. 364.   8.]
 [ 25.   5.   4.   7.   3.   8.  40.   7. 478.]]
```

MLPClassifier: Cu acest model nu am reusit sa depasesc scorul anterior, obtinut cu modelul SVM. Am folosit clasificatorul MLP ca in laboratorul 7 si am incercat sa schimb parametrii ca sa evit overfitting/underfitting pentru o solutie buna. Am definit o functie “train_and_eval” care primeste modelul, il antreneaza si returneaza scorul pe datele de validare. Dupa peste 20 de combinatii fara succes, am pastrat una dintre cele mai bune definiri ale modelului si am afisat matricea de confuzie.

```
clf = MLPClassifier(hidden_layer_sizes=(50,100,50), activation='relu',
                    learning_rate_init=0.001, #rata de invatare este cea default
                    max_iter=2000, alpha=0.005) #numarul maxim de epoci si parametrul pentru regularizarea L2
```

```
Accuracy = 0.7056
[[377.  19.  35.   9.  30.   5.  32.  29.  34.]
 [ 42. 378.  29.   8.  21.   3.  16.  26.   4.]
 [ 21.  18. 418.   9.  27.  15.   7.  14.   4.]
 [ 26.  18.  39. 369.  21.  30.  19.  36.  20.]
 [ 48.  16.  45.  14. 365.  11.  10.  31.  14.]
 [ 11.   5.  41.  25.  23. 409.   7.  31.   9.]
 [ 46.   9.  17.   5.   9.   3. 457.   5.  29.]
 [ 59.  20.  28.  24.  31.  24.  15. 311.   8.]
 [ 42.   5.   6.  14.   7.   8.  43.   8. 444.]]
```

TensorFlow Sequential: Pentru ultimul model am folosit Google Colab in loc de PyCharm pentru a rula pe GPU dar si pentru ca am incercat sa instalez TensorFlow pe 2 calculatoare fara succes.

In acest model nu am mai folosit glob pentru citirea imaginilor pentru ca lua imaginile in ordine aleatoare. Am folosit cv2.imread care lua ca parametri pathname-ul fiecarei imagini si un flag care semnala ca avem poze monochromate(pe tonuri de gri). Am adaugat pozele intr-un array pe care l-am transformat ulterior in numpy.array.

Pentru ca nu am mai folosit matplotlib.pyplot.imread, dupa citire am impartit manual valorile pixelilor la 255. Apoi am luat shapeul datelor si am transformat vectorul de imagini din 3-dimensional in 4-dimensional.

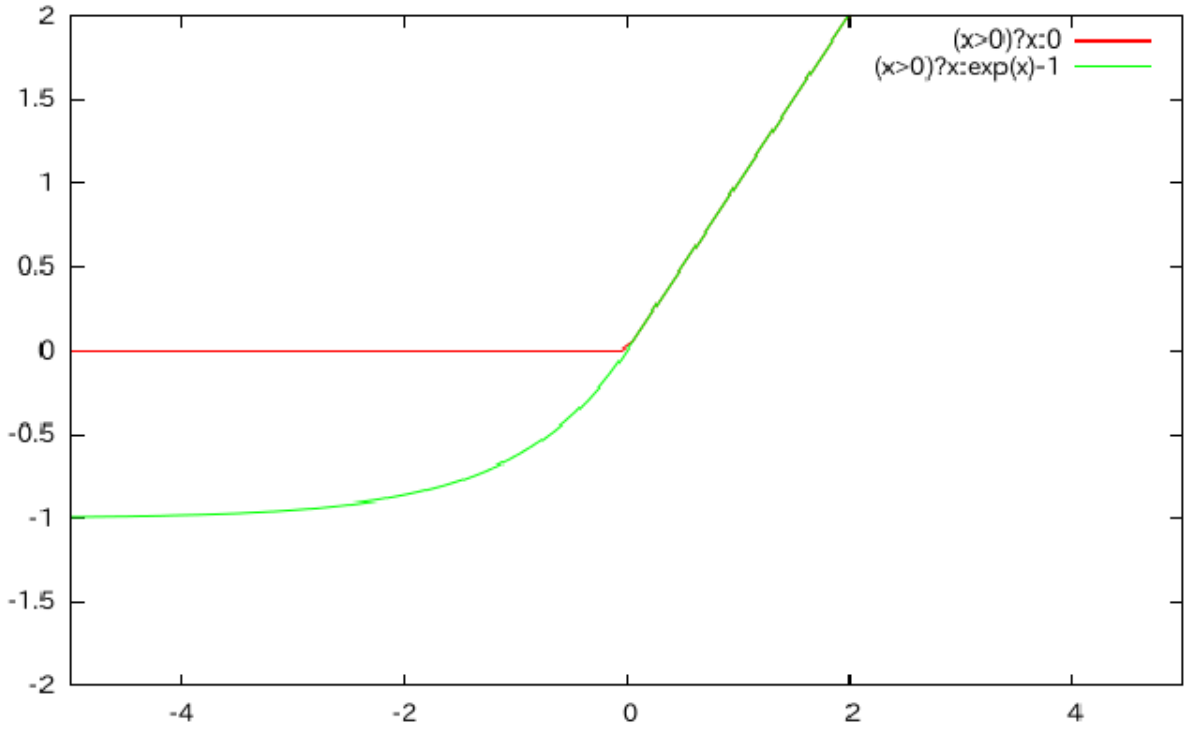
Dupa care, am inceput sa construiesc modelul Sequential cu arhitectura de la cursul 7: CONV-RELU-CONV-RELU-POOL-CONV-RELU-CONV-RELU-POOL-CONV-RELU-CONV-RELU-POOL-FC, insa rezultatul etichetelor prezise pe datele de evaluare nu era satisfacator.

Am incercat sa modelez singur arhitectura (aproximativ 14 ore continue) pentru a obtine un rezultat cat mai bun si pentru a evita overfitting/underfitting.

Am definit o convolutie cu 32 de filtre 5x5, dimensiunea de intrare 32x32x1, iar ca functie de activare am trecut de la “RELU” la “ELU”

pentru a avea un gradient mai bun pentru x < 0, desi are o functie exponentiala.

In imaginea urmatoare am se pot observa graficele pentru functia “RELU” (linia rosie) si functia “ELU” (linia verde) si diferenta acestora pentru valorile x < 0.



Am folosit o normalizare Batch ca in curs (mentine media aproape de 0 si deviatia standard aproape de 1) si am facut un pooling 2x2 cu stride default, pentru a scadea dimensiunea spatiala. Am repetat acest procedeu inca o data (am testat procedeul efectuat o singura data dar si de mai multe ori, pana am ajuns la concluzia de al repeat inca o data) .

Apoi am aplatizat datele cu Flatten fara sa le afectez dimensiunea, am folosit Dense cu 200 de unitati (am facut incercari de la 1024 = 32x32 de unitati pana la 200) si metoda de activare “ELU”, Dropout cu frecventa de 0.25 pentru a face diferiti neuroni random 0, iar la final Dense cu 9 unitati si functia de activare “softmax” pentru a lua 9 categorii si a avea un vector de probabilitati cu acestea.

Apoi am compilat modelul cu optimizatorul “adam” (initial incercasem foarte mult cu “sgd” fiind prezentat in curs si am schimbat), loss-ul cu “sparse_categorial_crossentropy” care calculeaza pierderea intre etichete si predictii facand un vector cu cele mai probabile categorii dintre cele 9 si metrics=]”accuracy”] care calculeaza acuratetea ca in functia definita la SVM.

Pentru ultima submisie de pe Kaggle am folosit datele prezentate mai sus si 50 de epoci si a putut fi observabil overfit-ul. Am obtinut pe datele de validare o acuratete de 0.92 si voiam sa fac antrenarea pe datele de antrenare + validare pentru a creste si mai mult.

```
Epoch 47/50
938/938 [==============================] - 5s 6ms/step - loss: 0.0326 - accuracy: 0.9894
Epoch 48/50
938/938 [==============================] - 5s 6ms/step - loss: 0.0326 - accuracy: 0.9881
Epoch 49/50
938/938 [==============================] - 5s 6ms/step - loss: 0.0326 - accuracy: 0.9876
Epoch 50/50
938/938 [==============================] - 5s 6ms/step - loss: 0.0369 - accuracy: 0.9881
157/157 [==============================] - 1s 3ms/step - loss: 0.4388 - accuracy: 0.9202
```

Valoarea la loss pe train era de aproximativ 0.037 si la validare de aproximativ 0.44. Cu toate acestea, Kaggle a fost destul de permisiv si nu m-a depunctat foarte mult pe datele reale.

Am afisat si matricea de confuzie pentru acest overfit.

```
[[506.   4.   3.   3.  20.   0.   6.  11.  17.]
 [  3. 503.   5.   4.   3.   1.   1.   5.   2.]
 [  3.   6. 474.  11.  15.  14.   4.   6.   0.]
 [  7.   4.  10. 512.  12.   4.   3.  14.  12.]
 [ 15.   4.  12.   3. 500.   6.   1.   4.   9.]
 [  3.   0.  15.  14.   9. 507.   4.   7.   2.]
 [  7.   0.   5.   3.   0.   5. 550.   3.   7.]
 [  6.  10.  10.   7.  12.  11.   3. 460.   1.]
 [  4.   0.   0.   3.   2.   2.   7.   3. 556.]]
```

Ulterior am reusit sa schimb datele, sa evit overfitting-ul si sa obtin un rezultat ca si “Late Submission” pe Kaggle, prin schimbarea frecventei la Dropout de la 0.25 la 0.8 si prin schimbarea numarului de epoci de la 50 la 35, de la 0.908 la 0.915.

```
Epoch 33/35
938/938 [==============================] - 5s 6ms/step - loss: 0.2633 - accuracy: 0.9085
Epoch 34/35
938/938 [==============================] - 5s 6ms/step - loss: 0.2595 - accuracy: 0.9104
Epoch 35/35
938/938 [==============================] - 5s 6ms/step - loss: 0.2528 - accuracy: 0.9125
157/157 [==============================] - 1s 4ms/step - loss: 0.2644 - accuracy: 0.9090
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:450:
  warnings.warn('`model.predict_classes()` is deprecated and '
[8 1 4 ... 7 4 8]
[[498.   1.   3.   3.  25.   2.   8.  13.  17.]
 [  5. 485.   8.   6.   2.   4.   1.  14.   2.]
 [  3.   1. 477.   7.  17.  17.   3.   7.   1.]
 [  6.   1.  12. 502.  12.  13.   2.  18.  12.]
 [ 15.   2.  15.  11. 481.  12.   3.   7.   8.]
 [  1.   0.   9.   6.   5. 529.   4.   7.   0.]
 [  6.   3.   1.   2.   0.   6. 553.   4.   5.]
 [  7.   2.   2.  13.   4.  11.   4. 477.   0.]
 [ 12.   1.   0.   3.   1.   5.  10.   2. 543.]]
```

Loss-ul pe train este de 0.2528 si la validare de 0.2644 ceea ce este mult mai bine decat rezultatele obtinute pentru ultima submisie.