**Deep Hallucination Classification**

Proiectul are ca scop clasificarea imaginilor de tipul "Deep Hallucination" în una dintre cele 96 de clase. Cele două metode de învățare automată utilizate în cadrul acestui proiect sunt Naive Bayes și KNN.

**Modelul Naive Bayes**

Detalii implementare:

- Fiecare imagine este citită inițial sub forma unui vector de pixeli, acest vector având forma 64x64x3, deoarece o imagine este compusă din 64 de linii, fiecare conținând 64 de coloane, astfel încât are 64x64 de pixeli, iar pentru fiecare dintre acești pixeli se cunosc cele 3 valori RGB.

- Vectorul corespunzător fiecărei imagini este redimensionat la o singură dimensiune, iar apoi fiecare vector rezultat este adăugat într-un vector de caracteristici care conține informațiile pentru fiecare imagine în parte. Acest proces este aplicat în mod identic pentru cele trei seturi de date: setul de antrenament, setul de validare și setul de testare.

- Valorile din cadrul acestor vectori de caracteristici sunt apoi discretizate, obținându-se astfel o formă prelucrată a celor trei vectori menționați mai sus.

- Se folosește clasificatorul Multinomial Naive Bayes (din cadrul bibliotecii sklearn) pentru a antrena modelul pe datele de antrenament (vectorul de caracteristici prelucrat și clasele corespunzătoare fiecărei imagini reprezentate în cadrul acestuia).

- După antrenare, modelul este folosit pentru a prezice clasele imaginilor de validare. Acuratețea modelului este evaluată utilizând datele de validare și este calculată ca raport între numărul de predicții corecte și numărul total de imagini de validare.

- După afișarea ratei de acuratețe a predicției făcute pentru imaginile de validare, sunt calculate și afișate pentru fiecare clasă în parte valorile de precizie și recall, precum și matricea de confuzie. Mai multe detalii despre acestea sunt prezentate mai jos.

- Același procedeu se repetă și pentru imaginile de test, unde modelul nostru încearcă în mod similar ca în cazul imaginilor de validare să prezică clasele corespunzătoare acestora, rezultatele prezicerii fiind în final scrise într-un fișier CSV.

Optimizarea hiperparametrilor și preprocesarea datelor:

- În ceea ce privește prelucrarea datelor, aspectul de care a trebuit ținut cont a fost numărul de intervale în care dorim să se facă discretizarea datelor. În final, a fost aleasă valoarea 17, deoarece aceasta obținea cel mai bun scor în cadrul competiției. Totuși, un lucru interesant de menționat este faptul că valoarea 6 a fost cea aleasă inițial, deoarece pentru datele de validare obținea cea mai mare acuratețe (cu o diferență infimă de doar 0.001), însă în cadrul competiției submisia în care numărul de intervale era 6 era cu aproximativ 0.01 mai slabă decât cea în care numărul acestora era 17.

Ratele de acuratețe obținute pentru diverse valori alese pentru numărul de intervale:

|  |  |
| --- | --- |
| Număr intervale | Acuratețe |
| 1 | 0.036 |
| 2 | 0.036 |
| 3 | 0.153 |
| 4 | 0.178 |
| 5 | 0.183 |
| 6 | 0.192 |
| 7 | 0.184 |
| 8 | 0.190 |
| 9 | 0.185 |
| 10 | 0.190 |
| 11 | 0.191 |
| 12 | 0.189 |
| 13 | 0.191 |
| 14 | 0.188 |
| 15 | 0.189 |
| 16 | 0.188 |
| 17 | 0.191 |
| 18 | 0.188 |
| 19 | 0.188 |
| 20 | 0.189 |
| 21 | 0.191 |
| 22 | 0.187 |
| 23 | 0.186 |
| 24 | 0.187 |
| 25 | 0.190 |
| 26 | 0.187 |
| 27 | 0.188 |
| 28 | 0.187 |
| 29 | 0.185 |
| 30 | 0.190 |
| 31 | 0.187 |
| 32 | 0.187 |
| 33 | 0.186 |
| 34 | 0.187 |
| 35 | 0.186 |
| 36 | 0.185 |
| 37 | 0.184 |
| 38 | 0.184 |
| 39 | 0.185 |
| 40 | 0.186 |
| 41 | 0.186 |
| 42 | 0.183 |
| 43 | 0.183 |
| 44 | 0.185 |
| 45 | 0.185 |
| 46 | 0.185 |
| 47 | 0.184 |
| 48 | 0.183 |
| 49 | 0.185 |
| 50 | 0.184 |

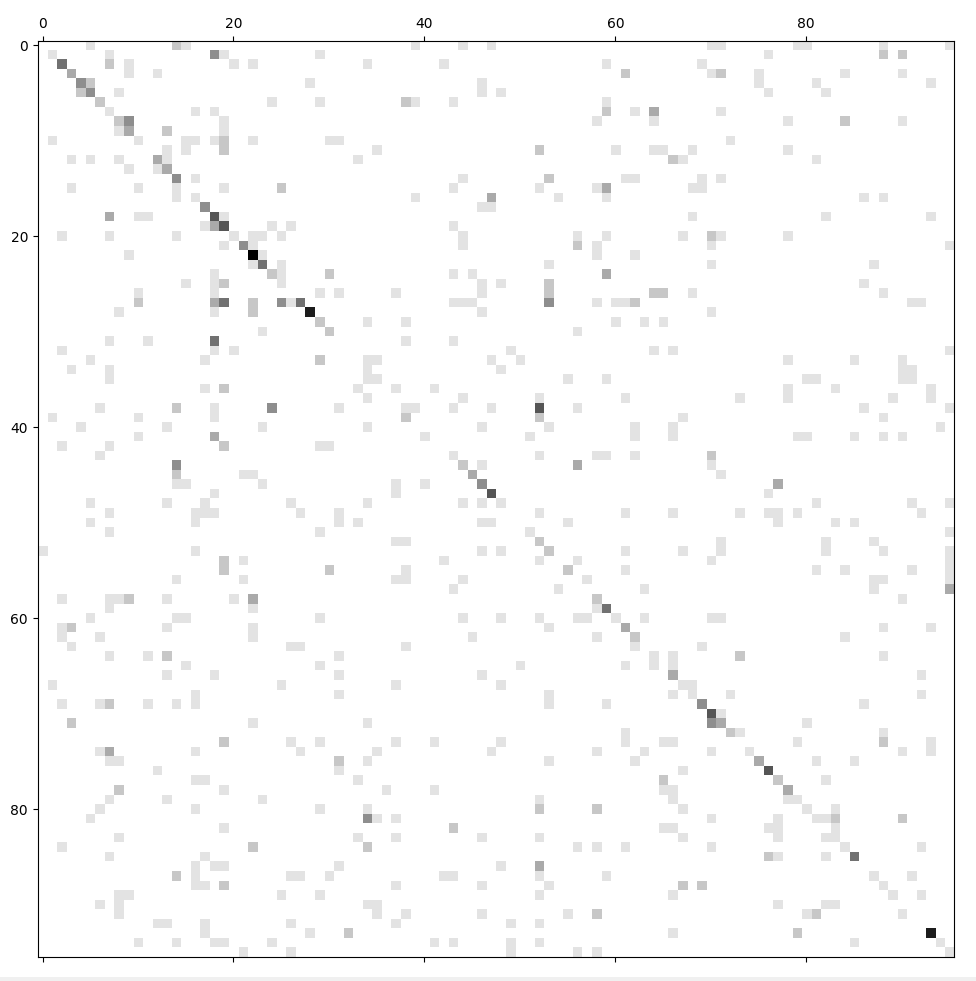
- De asemenea, se merită menționat și faptul că s-a încercat și o a doua normalizare, înaintea celei despre care am discutat până acum. Această normalizare consta în scăderea mediei valorilor trasaturilor (pixelilor) unei imagini și împărțirea rezultatului la deviația standard a imaginii. Cu toate acestea, am renunțat rapid la această abordare, deoarece în urma câtorva teste s-a observat că rata de acuratețe obținută pe testele de validare este mult mai redusă, fiind constantă indiferent de numărul de intervale (mai precis, 0.036).

Date despre acuratețe, precizie si recall:

- Cu scopul de a avea o mai bună viziune asupra predicțiilor făcute de către model, pentru fiecare clasă în parte, au fost calculate valorile True Positive, False Positive, True Negative și False Negative corespunzătoare, iar apoi aceste date au fost folosite pentru calcularea valorilor de precizie și recall (datele afișate mai jos sunt pentru un număr de 17 intervale):

|  |  |  |
| --- | --- | --- |
| Clasa | Precision | Recall |
| 0 | 0.000 | 0.000 |
| 1 | 0.250 | 0.077 |
| 2 | 0.385 | 0.357 |
| 3 | 0.273 | 0.231 |
| 4 | 0.571 | 0.364 |
| 5 | 0.308 | 0.364 |
| 6 | 0.222 | 0.222 |
| 7 | 0.040 | 0.100 |
| 8 | 0.143 | 0.154 |
| 9 | 0.214 | 0.429 |
| 10 | 0.111 | 0.091 |
| 11 | 0.000 | 0.000 |
| 12 | 0.429 | 0.250 |
| 13 | 0.188 | 0.600 |
| 14 | 0.167 | 0.333 |
| 15 | 0.000 | 0.000 |
| 16 | 0.067 | 0.100 |
| 17 | 0.286 | 0.667 |
| 18 | 0.158 | 0.400 |
| 19 | 0.154 | 0.462 |
| 20 | 0.250 | 0.067 |
| 21 | 0.500 | 0.333 |
| 22 | 0.321 | 0.692 |
| 23 | 0.455 | 0.500 |
| 24 | 0.222 | 0.182 |
| 25 | 0.083 | 0.100 |
| 26 | 0.000 | 0.000 |
| 27 | 0.556 | 0.139 |
| 28 | 0.800 | 0.571 |
| 29 | 0.143 | 0.286 |
| 30 | 0.222 | 0.500 |
| 31 | 0.000 | 0.000 |
| 32 | 0.000 | 0.000 |
| 33 | 0.000 | 0.000 |
| 34 | 0.056 | 0.167 |
| 35 | 0.143 | 0.111 |
| 36 | 0.000 | 0.000 |
| 37 | 0.000 | 0.000 |
| 38 | 0.083 | 0.045 |
| 39 | 0.000 | 0.000 |
| 40 | 0.000 | 0.000 |
| 41 | 0.000 | 0.000 |
| 42 | 0.000 | 0.000 |
| 43 | 0.077 | 0.125 |
| 44 | 0.182 | 0.182 |
| 45 | 0.500 | 0.375 |
| 46 | 0.211 | 0.333 |
| 47 | 0.400 | 0.667 |
| 48 | 0.111 | 0.091 |
| 49 | 0.000 | 0.000 |
| 50 | 0.000 | 0.000 |
| 51 | 0.500 | 0.250 |
| 52 | 0.069 | 0.286 |
| 53 | 0.105 | 0.167 |
| 54 | 0.000 | 0.000 |
| 55 | 0.400 | 0.167 |
| 56 | 0.000 | 0.000 |
| 57 | 0.000 | 0.000 |
| 58 | 0.118 | 0.154 |
| 59 | 0.238 | 0.625 |
| 60 | 0.250 | 0.077 |
| 61 | 0.200 | 0.273 |
| 62 | 0.167 | 0.250 |
| 63 | 0.000 | 0.000 |
| 64 | 0.100 | 0.100 |
| 65 | 0.000 | 0.000 |
| 66 | 0.167 | 0.375 |
| 67 | 0.100 | 0.167 |
| 68 | 0.167 | 0.143 |
| 69 | 0.400 | 0.286 |
| 70 | 0.240 | 0.857 |
| 71 | 0.214 | 0.250 |
| 72 | 0.500 | 0.400 |
| 73 | 0.000 | 0.000 |
| 74 | 1.000 | 0.071 |
| 75 | 0.600 | 0.250 |
| 76 | 0.462 | 0.667 |
| 77 | 0.167 | 0.250 |
| 78 | 0.300 | 0.300 |
| 79 | 0.143 | 0.143 |
| 80 | 0.167 | 0.091 |
| 81 | 0.111 | 0.067 |
| 82 | 0.000 | 0.000 |
| 83 | 0.125 | 0.111 |
| 84 | 0.167 | 0.091 |
| 85 | 0.500 | 0.455 |
| 86 | 0.000 | 0.000 |
| 87 | 0.200 | 0.083 |
| 88 | 0.067 | 0.83 |
| 89 | 0.500 | 0.125 |
| 90 | 0.000 | 0.000 |
| 91 | 0.000 | 0.000 |
| 92 | 0.000 | 0.000 |
| 93 | 0.533 | 0.533 |
| 94 | 0.500 | 0.091 |
| 95 | 0.071 | 0.167 |

- De asemenea, a fost construită și matricea de confuzie corespunzătoare predicțiilor făcute (cu cât un pătrațel este mai închis la culoare, cu atât valoarea corespunzătoare acestuia din cadrul matricei de confuzie este mai mare; un pătrațel alb înseamnă valoarea 0):



**Modelul KNN:**

Detalii implementare:

- Similar ca în cazul modelului Naive Bayes, inițial fiecare imagine este citită sub forma unui vector de pixeli, acest vector fiind de forma 64x64x3, întrucât o imagine este compusă din 64 de linii, fiecare conținând 64 de coloane, astfel având 64x64 de pixeli, iar pentru fiecare dintre acești pixeli cunoscându-se cele 3 valori RGB.

- Aceste imagini sunt normalizate prin scăderea valorii medii a fiecărei imagini din vectorul de caracteristici (valori ale pixelilor) și împărțirea rezultatului la deviația standard. Apoi, imaginile sunt adăugate într-un vector de caracteristici care conține informațiile pentru fiecare imagine în parte. Acest proces este identic pentru toate cele 3 seturi de date: setul de antrenament, setul de validare și setul de testare.

- Imaginile din setul de validare sunt apoi clasificate folosind metoda celor mai apropiați vecini. Pentru a face acest lucru, luăm fiecare imagine de validare în parte și calculăm distanța acesteia față de fiecare imagine din setul de antrenament (folosim distanța Euclidiană sau distanța Manhattan). După aceea, din totalul imaginilor de antrenament selectăm primele 'numarVecini' imagini sortate crescător după distanța obținută, și folosind informațiile din fișierul CSV de antrenament aflăm clasele corespunzătoare fiecăreia dintre aceste imagini. În final, selectăm clasa cu cele mai multe apariții între acești 'numarVecini' vecini și o adăugăm în lista noastră de predicții.

- Astfel, după ce am obținut pentru fiecare imagine din setul de validare o predicție, procedăm similar ca în cazul modelului Naive Bayes: afișăm rata de acuratețe obținută pentru setul de date, iar apoi, pentru fiecare clasă în parte, valorile de precizie și acoperire (recall), și de asemenea matricea de confuzie.

- Același procedeu se repetă și pentru imaginile de test, predicțiile făcute în acest caz fiind scrise într-un fișier CSV.

Optimizarea hiperparametrilor și preprocesarea datelor:

- De-a lungul numeroaselor rulări ale algoritmului, au fost încercate multiple variante pentru parametri, dar și pentru prelucrarea datelor.

- În ceea ce privește prelucrarea datelor, s-au încercat două abordări, inițial una în care imaginile erau folosite așa cum erau citite, fără modificări făcute asupra caracteristicilor corespunzătoare acestora, iar apoi s-a încercat normalizarea acestora, folosind procedeul menționat mai sus (cel cu valoarea medie și deviația standard).

- De asemenea, pentru calculul distanței dintre două imagini, au fost testate atât distanța Euclidiană, cât și distanța Manhattan.

- Dacă imaginile sunt clasificate fără a fi normalizate, distanța Euclidiană oferă valori ale acurateței mai bune, în timp ce folosirea distanței Manhattan va rezulta într-o acuratețe mult mai scăzută. Dacă imaginile sunt normalizate, distanța Manhattan este cea care oferă rezultate mai bune. În general, acuratețea cea mai bună se obține în cazul în care normalizăm valorile corespunzătoare imaginilor și folosim distanța Manhattan (cel puțin pe datele de validare). Cu toate acestea, în cadrul submisiei din competiție am încărcat o variantă care folosește normalizare și distanța Euclidiană, deoarece am observat faptul că distanța Manhattan în combinație cu normalizarea datelor oferă rezultate mai bune abia după încheierea competiției.

- Un alt parametru de care a trebuit ținut cont a fost numărul de vecini. În cadrul submisiei din competiție a fost folosit un număr de 26 de vecini, dar în general (în cazul în care folosim normalizare + distanța Manhattan) cel mai bun rezultat (pentru datele de validare) se obține pentru un număr de 9 sau 12 vecini.

În imaginile de mai jos se pot observa majoritatea valorilor testate pentru cazurile menționate mai sus.

Fără normalizare + distanța Euclidiană:

|  |  |
| --- | --- |
| Număr vecini | Acuratețe |
| 3 | 0.097 |
| 6 | 0.112 |
| 9 | 0.108 |
| 12 | 0.120 |
| 15 | 0.121 |
| 18 | 0.119 |
| 21 | 0.118 |
| 24 | 0.122 |
| 27 | 0.122 |
| 30 | 0.126 |
| 33 | 0.137 |

Fără normalizare + distanța Manhattan:

|  |  |
| --- | --- |
| Număr vecini | Acuratețe |
| 3 | 0.064 |
| 6 | 0.061 |
| 9 | 0.060 |
| 12 | 0.053 |
| 15 | 0.048 |
| 18 | 0.052 |
| 21 | 0.050 |
| 24 | 0.046 |
| 27 | 0.046 |
| 30 | 0.048 |
| 33 | 0.048 |
| 36 | 0.046 |
| 39 | 0.044 |
| 42 | 0.043 |
| 45 | 0.041 |
| 48 | 0.042 |
| 51 | 0.039 |
| 54 | 0.041 |
| 57 | 0.038 |
| 60 | 0.036 |
| 63 | 0.036 |
| 66 | 0.035 |
| 69 | 0.036 |
| 72 | 0.034 |
| 75 | 0.035 |
| 78 | 0.033 |
| 81 | 0.033 |
| 84 | 0.033 |
| 87 | 0.033 |
| 90 | 0.033 |

Cu normalizare + distanța Euclidiană:

|  |  |
| --- | --- |
| Număr Vecini | Acuratețe |
| 3 | 0.158 |
| 6 | 0.180 |
| 9 | 0.183 |
| 12 | 0.188 |
| 15 | 0.192 |
| 18 | 0.194 |
| 21 | 0.189 |
| 24 | 0.194 |
| 27 | 0.202 |
| 30 | 0.198 |
| 33 | 0.196 |
| 36 | 0.194 |
| 39 | 0.192 |
| 42 | 0.190 |
| 45 | 0.189 |
| 48 | 0.185 |
| 51 | 0.185 |
| 54 | 0.185 |
| 57 | 0.182 |
| 60 | 0.179 |
| 63 | 0.179 |
| 66 | 0.181 |
| 69 | 0.181 |
| 72 | 0.184 |
| 75 | 0.183 |
| 78 | 0.182 |
| 81 | 0.178 |
| 84 | 0.179 |
| 87 | 0.181 |
|  |  |
| 25 | 0.198 |
| 26 | 0.203 |
| 28 | 0.198 |
| 200 | 0.157 |
| 400 | 0.143 |
| 500 | 0.137 |

Cu normalizare + distanța Manhattan:

|  |  |
| --- | --- |
| Numar vecini | Acuratețe |
| 3 | 0.220 |
| 4 | 0.213 |
| 5 | 0.228 |
| 6 | 0.234 |
| 7 | 0.234 |
| 8 | 0.236 |
| 9 | 0.242 |
| 10 | 0.233 |
| 11 | 0.238 |
| 12 | 0.242 |
| 13 | 0.241 |
| 14 | 0.241 |

Date despre acuratete, precizie si recall:

Cu scopul de a avea o mai bună vizune asupra predicțiilor făcute de către model, pentru fiecare clasă în parte, au fost calculate valorile TruePositive, FalsePositive, TrueNegative, FalseNegative corespunzătoare, iar apoi aceste date au fost folosite pentru calcularea valorilor precizie și recall (datele afișate mai jos sunt pentru cazul în care se folosește normalizare, distanța utilizată este cea Euclidiană, iar numărul de vecini este de 26):

|  |  |  |
| --- | --- | --- |
| Clasa | Precision | Recall |
| 0 | 0.125 | 0.077 |
| 1 | 0.000 | 0.000 |
| 2 | 0.156 | 0.500 |
| 3 | 0.500 | 0.154 |
| 4 | 0.222 | 0.182 |
| 5 | 0.161 | 0.455 |
| 6 | 0.000 | 0.000 |
| 7 | 0.125 | 0.100 |
| 8 | 0.143 | 0.077 |
| 9 | 0.108 | 0.571 |
| 10 | 0.000 | 0.000 |
| 11 | 0.000 | 0.000 |
| 12 | 0.455 | 0.417 |
| 13 | 0.111 | 0.800 |
| 14 | 0.308 | 0.333 |
| 15 | 0.250 | 0.214 |
| 16 | 0.125 | 0.100 |
| 17 | 0.222 | 0.333 |
| 18 | 0.205 | 0.533 |
| 19 | 0.000 | 0.077 |
| 20 | 0.600 | 0.200 |
| 21 | 1.000 | 0.167 |
| 22 | 0.191 | 0.692 |
| 23 | 0.375 | 0.300 |
| 24 | 0.158 | 0.545 |
| 25 | 0.000 | 0.000 |
| 26 | 0.250 | 0.143 |
| 27 | 0.395 | 0.417 |
| 28 | 0.353 | 0.429 |
| 29 | 0.400 | 0.286 |
| 30 | 0.200 | 0.250 |
| 31 | 0.143 | 0.222 |
| 32 | 0.054 | 0.286 |
| 33 | 0.500 | 0.091 |
| 34 | 0.138 | 0.667 |
| 35 | 0.000 | 0.000 |
| 36 | 1.000 | 0.111 |
| 37 | 0.000 | 0.000 |
| 38 | 0.290 | 0.409 |
| 39 | 0.500 | 0.111 |
| 40 | 0.000 | 0.000 |
| 41 | 0.000 | 0.000 |
| 42 | 0.000 | 0.000 |
| 43 | 0.000 | 0.000 |
| 44 | 1.000 | 0.091 |
| 45 | 0.000 | 0.000 |
| 46 | 1.000 | 0.167 |
| 47 | 0.296 | 0.889 |
| 48 | 0.667 | 0.182 |
| 49 | 0.000 | 0.000 |
| 50 | 0.333 | 0.100 |
| 51 | 0.000 | 0.000 |
| 52 | 0.231 | 0.429 |
| 53 | 0.375 | 0.250 |
| 54 | 0.000 | 0.000 |
| 55 | 0.000 | 0.000 |
| 56 | 1.000 | 0.111 |
| 57 | 1.000 | 0.143 |
| 58 | 0.286 | 0.308 |
| 59 | 0.065 | 0.250 |
| 60 | 1.000 | 0.154 |
| 61 | 0.000 | 0.000 |
| 62 | 0.000 | 0.000 |
| 63 | 0.000 | 0.000 |
| 64 | 1.000 | 0.100 |
| 65 | 0.047 | 0.500 |
| 66 | 0.083 | 0.125 |
| 67 | 0.000 | 0.000 |
| 68 | 0.071 | 0.143 |
| 69 | 0.267 | 0.286 |
| 70 | 0.106 | 1.000 |
| 71 | 0.333 | 0.083 |
| 72 | 0.500 | 0.200 |
| 73 | 0.000 | 0.000 |
| 74 | 1.000 | 0.071 |
| 75 | 0.714 | 0.417 |
| 76 | 0.400 | 0.222 |
| 77 | 0.000 | 0.000 |
| 78 | 0.167 | 0.700 |
| 79 | 0.125 | 0.143 |
| 80 | 0.333 | 0.182 |
| 81 | 0.286 | 0.133 |
| 82 | 0.000 | 0.000 |
| 83 | 1.000 | 0.111 |
| 84 | 0.000 | 0.000 |
| 85 | 0.667 | 0.364 |
| 86 | 1.000 | 0.125 |
| 87 | 0.000 | 0.000 |
| 88 | 0.143 | 0.083 |
| 89 | 0.000 | 0.000 |
| 90 | 0.000 | 0.000 |
| 91 | 0.000 | 0.000 |
| 92 | 0.500 | 0.143 |
| 93 | 0.000 | 0.000 |
| 94 | 0.385 | 0.455 |
| 95 | 0.250 | 0.167 |

- De asemenea, a fost construită și matricea de confuzie corespunzătoare predicțiilor făcute (cu cât un pătrațel este mai închis la culoare, cu atât valoarea corespunzătoare acestuia din cadrul matricei de confuzie este mai mare; un pătrațel alb înseamnă valoarea 0):

