MADRAS INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY
CHENNAI - 600 044

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5613 - SOCIALLY RELEVANT PROJECTLABORATORY

Semester: 6

Batch: 2

Team No: 5

Title: BraillePulse

S.NO	NAME & ROLLNO	CONTRIBUTION
1	Hemnath N M 2021506026	Dataset narration with Sample data/images
2	Dharshan S 2021506018	Assigning class label for classification, Performance metrics and Comparison table of different models, and language-based clustering for Braille character language identification."
3	Abinaya S 2021506002	Classification and performance metrics of different models for braille character identification and recognition
4	Arun N 2021506013	Clustering of braille images and performance metrics of different cluster models

SOCIALLY RELEVANT PROJECT

BRAILLEPULSE

Definition:

The project aims to develop an accessible educational website dedicated to teaching and providing resources about Braille. The core functionality of the site includes an interactive platform where users can convert Braille into text, audio, and video formats, and vice versa. This initiative addresses the educational gap for visually impaired individuals and those interested in learning Braille, enhancing their ability to communicate and access information effectively. The website will serve as a comprehensive tool, empowering users by facilitating the learning of Braille through modern, user-friendly digital solutions. Additionally, it aims to raise awareness and promote inclusivity for Braille literacy in digital spaces, making learning opportunities more accessible to a broader audience.

Dataset narration with Sample data/images

Dataset contains Braille images of several languages with format of

File type - JPG/JPEG

File Resolution - 52PX x 73PX

In each language, our dataset contains braille images of

- Alphabets
- Formatting
- Punctuation
- Symbols
- Numbers

Example Languages:

- These are some of the languages most commonly used

1. English
2. Arabic
3. Bengali
4. Chinese
5. French
6. Italian
7. Japanese
8. German
9. Portuguese
10. Spanish
11. Tamil

Sample Screenshots:

BraillePulse Public

Pin | Unwatch 1

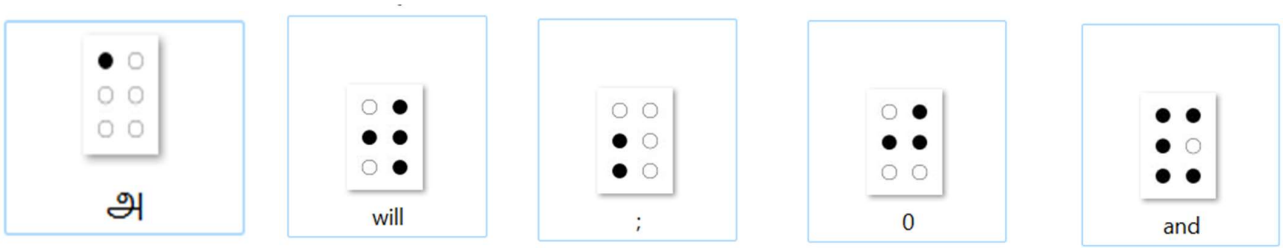
main | 1 Branch | 0 Tags | Go to file t | Add file | Code

ArunAVFC commit 891b6f0 · 2 months ago 172 Commits

Albanian	Add files via upload	2 months ago
Algerian/Alphabets	commit	2 months ago
Arabic	uploaded	2 months ago
Belarusian	uploaded	2 months ago
Bengali	Bengali braille files	2 months ago
Bulgarian	uploaded	2 months ago
Chinese	no	2 months ago
Croatian	committed	2 months ago
Czech	Czech braille files	2 months ago
Devanagiri(Hindi,Marati,Nepali)	uploaded	2 months ago
Dutch	Dutch braille files	2 months ago
English	uploaded	2 months ago

Hemnath31 Add files via upload 60978aa · 2 months ago History

Name	Last commit message	Last commit date
..		
Formatting	Add files via upload	2 months ago
Letters	Add files via upload	2 months ago
Punctuation	Add files via upload	2 months ago
readme.nd	Create readme.nd	2 months ago



Classification

1)Braille character language identification

- Identifies which language a braille belongs to (Tamil ,Greek..)

2)Braille character identification

- Identifies what the braille character is (a,b,c,….)

1)Braille character language identification

Class labels for classification

Assigning labels based on language folders, and creates a mapping of label indices to language names for classification training

```
Label: 0 Language: Arabic
Label: 1 Language: Bengali
Label: 2 Language: Croatian
Label: 3 Language: Dutch
Label: 4 Language: English
Label: 5 Language: Greek
Label: 6 Language: Hungarian
Label: 7 Language: Icelandic
Label: 8 Language: Kazakh
Label: 9 Language: Lithuanian
Label: 10 Language: Malayalam
Label: 11 Language: Portugese
Label: 12 Language: Russian
Label: 13 Language: Spanish
Label: 14 Language: Tamil
Label: 15 Language: Ukrainian
```

Classification Models

1. CNN: A deep learning model specialized in capturing spatial hierarchies in data, widely used for image classification tasks.
2. FNN: A feedforward neural network architecture consisting of input, hidden, and output layers, often used for basic classification tasks.
3. RNN: Recurrent Neural Network, designed to process sequential data by maintaining internal memory, commonly employed for tasks like time series analysis and natural language processing.
4. VGG16: A convolutional neural network architecture known for its simplicity and effectiveness, particularly in image classification tasks, trained on large-scale datasets like ImageNet.

CNN MODEL

```
Epoch 1/10
C:\Users\Dharshan.S\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass
an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model i
nstead.
  super().__init__(
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.5565 - loss: 75.7175 - val_accuracy: 0.7663 - val_loss: 0.7481
Epoch 2/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.7999 - loss: 0.6048 - val_accuracy: 0.7751 - val_loss: 0.6395
Epoch 3/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.8139 - loss: 0.5360 - val_accuracy: 0.7751 - val_loss: 0.6244
Epoch 4/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - accuracy: 0.8223 - loss: 0.5189 - val_accuracy: 0.7988 - val_loss: 0.6180
Epoch 5/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - accuracy: 0.8366 - loss: 0.5052 - val_accuracy: 0.7811 - val_loss: 0.6198
Epoch 6/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - accuracy: 0.7948 - loss: 0.7345 - val_accuracy: 0.7544 - val_loss: 0.6812
Epoch 7/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8050 - loss: 0.5422 - val_accuracy: 0.7633 - val_loss: 0.6401
Epoch 8/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8041 - loss: 0.5331 - val_accuracy: 0.7663 - val_loss: 0.6288
Epoch 9/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - accuracy: 0.8186 - loss: 0.4806 - val_accuracy: 0.7840 - val_loss: 0.6357
Epoch 10/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - accuracy: 0.8321 - loss: 0.4879 - val_accuracy: 0.7751 - val_loss: 0.6301
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       0.00      0.00      0.00        12
           2       0.00      0.00      0.00         5
           3       0.22      0.33      0.27         6
           4       1.00      1.00      1.00       314
           5       0.09      1.00      0.16         5
           6       0.20      0.12      0.15         8
           7       0.00      0.00      0.00        14
           8       0.00      0.00      0.00         4
           9       0.00      0.00      0.00         5
          10       0.00      0.00      0.00         8
          11       0.00      0.00      0.00        10
          12       0.00      0.00      0.00         7
          13       0.00      0.00      0.00         7
          14       0.06      0.20      0.09         5
          15       0.25      0.17      0.20         6

    accuracy                           0.78       422
   macro avg       0.18      0.24      0.18       422
weighted avg       0.77      0.78      0.77       422
```

FNN MODEL

```
Epoch 1/10
C:\Users\Dharshan.S\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `in
put_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instea
d.
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5151 - loss: 1486.2255 - val_accuracy: 0.7249 - val_loss: 320.0974
Epoch 2/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5829 - loss: 357.8799 - val_accuracy: 0.7604 - val_loss: 277.6897
Epoch 3/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7073 - loss: 188.6100 - val_accuracy: 0.1272 - val_loss: 284.5540
Epoch 4/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6379 - loss: 161.2100 - val_accuracy: 0.7426 - val_loss: 102.1641
Epoch 5/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6760 - loss: 141.6963 - val_accuracy: 0.7426 - val_loss: 120.2134
Epoch 6/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7178 - loss: 97.1183 - val_accuracy: 0.7515 - val_loss: 85.2525
Epoch 7/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6653 - loss: 102.7553 - val_accuracy: 0.7367 - val_loss: 57.9737
Epoch 8/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6819 - loss: 75.4102 - val_accuracy: 0.7160 - val_loss: 42.5726
Epoch 9/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7451 - loss: 41.7501 - val_accuracy: 0.5444 - val_loss: 35.6900
Epoch 10/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6736 - loss: 51.2356 - val_accuracy: 0.5710 - val_loss: 46.2414
<keras.src.callbacks.history.History at 0x1bca84d4dd0>
```

RNN MODEL

```
Epoch 1/10
C:\Users\Dharshan.S\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarnir
`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer
  super().__init__(**kwargs)
43/43 ━━━━━━━━━━━━━━━━━━━━ 91s 2s/step - accuracy: 0.4352 - loss: 2.0702 - val_accuracy: 0.7249 - val_loss: 1.0937
Epoch 2/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 94s 2s/step - accuracy: 0.7472 - loss: 0.9497 - val_accuracy: 0.7426 - val_loss: 0.8953
Epoch 3/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 93s 2s/step - accuracy: 0.7673 - loss: 0.8019 - val_accuracy: 0.7544 - val_loss: 0.7967
Epoch 4/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 89s 2s/step - accuracy: 0.7583 - loss: 0.7900 - val_accuracy: 0.7722 - val_loss: 0.7523
Epoch 5/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 93s 2s/step - accuracy: 0.7980 - loss: 0.6643 - val_accuracy: 0.7722 - val_loss: 0.7232
Epoch 6/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 92s 2s/step - accuracy: 0.7644 - loss: 0.7281 - val_accuracy: 0.7751 - val_loss: 0.6982
Epoch 7/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 92s 2s/step - accuracy: 0.8068 - loss: 0.6160 - val_accuracy: 0.7781 - val_loss: 0.6769
Epoch 8/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 92s 2s/step - accuracy: 0.7904 - loss: 0.6341 - val_accuracy: 0.7751 - val_loss: 0.6516
Epoch 9/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 94s 2s/step - accuracy: 0.8069 - loss: 0.5800 - val_accuracy: 0.7663 - val_loss: 0.6410
Epoch 10/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 93s 2s/step - accuracy: 0.7992 - loss: 0.5789 - val_accuracy: 0.7722 - val_loss: 0.6179
]: <keras.src.callbacks.history.History at 0x1bd0ed67350>
```

TRAINING MODEL-VGG16

```
Epoch 1/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 154ms/step - accuracy: 0.6292 - loss: 5.2767 - val_accuracy: 0.7515 - val_loss: 1.4635
Epoch 2/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 147ms/step - accuracy: 0.7895 - loss: 1.0711 - val_accuracy: 0.7574 - val_loss: 1.2638
Epoch 3/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 139ms/step - accuracy: 0.8028 - loss: 0.9441 - val_accuracy: 0.7899 - val_loss: 1.1468
Epoch 4/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 146ms/step - accuracy: 0.8032 - loss: 0.9153 - val_accuracy: 0.7899 - val_loss: 1.0777
Epoch 5/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 158ms/step - accuracy: 0.8186 - loss: 0.8388 - val_accuracy: 0.7840 - val_loss: 0.8785
Epoch 6/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 154ms/step - accuracy: 0.8034 - loss: 0.7091 - val_accuracy: 0.7781 - val_loss: 0.8960
Epoch 7/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 145ms/step - accuracy: 0.8292 - loss: 0.7461 - val_accuracy: 0.7722 - val_loss: 0.9896
Epoch 8/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 144ms/step - accuracy: 0.8139 - loss: 0.8851 - val_accuracy: 0.7604 - val_loss: 1.2846
Epoch 9/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 6s 148ms/step - accuracy: 0.8039 - loss: 0.6990 - val_accuracy: 0.7633 - val_loss: 1.2799
Epoch 10/10
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 151ms/step - accuracy: 0.8167 - loss: 0.7437 - val_accuracy: 0.7604 - val_loss: 1.0885
<keras.src.callbacks.history.History at 0x1bd0ec6b6d0>
```

COMPARISON TABEL FOR DIFFERENT MODELS

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 2s 134ms/step
                        Model  Accuracy  Precision    Recall  F1-score
0  Feedforward Neural Network  0.611374   0.139046  0.201018  0.134351
1    Recurrent Neural Network  0.779621   0.156189  0.203125  0.172699
2   Transfer Learning (VGG16)  0.777251   0.145909  0.213542  0.150259
```

LANGUAGE IDENTIFICATION

Language prediction in Braille character images using various machine learning models, including CNN, FNN, and Transfer Learning. It prompts users to select a model, loads the input image, makes predictions, and prints the predicted language**.**

```
Choose a model for prediction:
1. CNN
2. FNN
3. Transfer Learning
Enter your choice (1/2/3):  1
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Predicted Language: 5
```

LANGUAGE-BASED CLUSTERING FOR BRAILLE CHARACTER LANGUAGE IDENTIFICATION

MODELS FOR CLUSTERING

1. **K-Means:** A centroid-based clustering algorithm that partitions data into 'k' clusters by iteratively updating cluster centroids.

2. **Hierarchica**l: A clustering algorithm that creates a tree of clusters, where each node represents a cluster and branches represent the merging of clusters.

3. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise): A density-based clustering algorithm that groups together points based on density, forming clusters separated by areas of low density.

No of clusters depends on number of language(16)

K-MEANS CLUSTERING

```
Cluster 0
Number of images in this cluster: 247
Indices of images in this cluster: [ 151  166  170  176  182  199  206  209  215  221  223  224  226  230
  247  248  251  254  257  259  260  272  284  289  292  314  329  346
  347  358  367  371  376  377  382  383  386  391  404  419  421  422
  428  437  452  454  470  472  473  479  485  488  491  493  496  512
  530  539  544  572  574  583  599  605  620  623  634  644  647  649
  650  656  664  668  674  676  682  686  689  692  700  728  737  745
  752  760  770  772  775  785  787  830  836  841  848  857  862  863
  866  883  887  893  899  902  916  923  946  949  953  962  983  985
  989  994  998 1006 1007 1010 1016 1034 1036 1037 1042 1052 1061 1067
 1069 1078 1079 1082 1091 1094 1106 1111 1115 1118 1130 1145 1148 1151
 1154 1160 1166 1169 1172 1184 1190 1193 1196 1202 1211 1214 1216 1220
 1223 1234 1241 1244 1253 1259 1261 1268 1270 1276 1277 1283 1313 1327
 1328 1331 1352 1355 1361 1370 1376 1384 1388 1394 1397 1400 1403 1406
 1417 1418 1421 1423 1426 1427 1429 1430 1436 1439 1447 1451 1457 1460
 1466 1469 1471 1477 1480 1481 1486 1490 1496 1499 1501 1508 1514 1522
 1523 1525 1531 1532 1535 1540 1553 1559 1561 1564 1573 1574 1577 1583
 1589 1595 1606 1613 1616 1619 1625 1634 1637 1646 1652 1655 1667 1670
 1672 1676 1679 1685 1688 1693 1694 1697 1703]
Cluster 1
Number of images in this cluster: 220
Indices of images in this cluster: [ 148  152  164  167  188  193  202  214  217  220  232  233  236  244
  245  253  266  275  277  280  286  293  298  299  311  313  325  326
  341  344  353  356  365  373  379  380  385  392  398  407  412  418
  425  433  440  442  443  469  475  481  490  497  499  500  503  515
  517  521  523  526  535  542  548  557  563  568  569  581  584  586
  589  593  596  604  608  614  616  617  622  629  631  632  641  659
  662  667  671  679  680  688  706  710  719  727  740  764  767  776
  779  794  800  806  809  811  815  818  827  833  842  844  845  850
  854  860  869  871  881  890  901  911  914  917  926  932  935  938
  943  944  950  955  959  977  979  995  997 1001 1003 1004 1013 1015
 1022 1031 1040 1049 1051 1058 1066 1070 1088 1090 1103 1117 1120 1127
 1144 1157 1162 1163 1175 1186 1195 1204 1205 1226 1231 1232 1243 1256
 1262 1265 1297 1301 1304 1307 1322 1325 1334 1337 1339 1364 1367 1373
 1409 1412 1415 1420 1424 1433 1438 1442 1444 1445 1463 1472 1474 1493
 1504 1505 1511 1513 1517 1526 1538 1547 1552 1556 1562 1568 1570 1586
 1607 1615 1622 1640 1643 1649 1661 1669 1673 1699]
Cluster 2
Number of images in this cluster: 125
Indices of images in this cluster: [ 147  150  162  177  186  189  213  219  225  231  243  255  261  264
  267  288  291  303  306  330  345  357  366  390  402  423  441  444
  462  465  468  480  507  555  558  567  570  612  621  633  678  702
  711  714  723  735  759  765  789  801  846  852  858  873  882  888
  891  897  939  945  963  975  984  987  993  996 1008 1017 1032 1038
 1050 1056 1059 1071 1086 1089 1095 1098 1101 1131 1134 1143 1149 1152
 1158 1164 1173 1191 1194 1206 1209 1227 1275 1287 1302 1320 1326 1335
 1341 1344 1371 1377 1389 1419 1428 1443 1455 1482 1488 1491 1494 1506
 1533 1542 1545 1575 1593 1599 1605 1620 1641 1662 1680 1683 1692]
Cluster 3
```

```
Cluster 13
Number of images in this cluster: 45
Indices of images in this cluster: [1812 1814 1815 1820 1831 1832 1833 1834
 1841 1842 1843 1844 1845 1982 1984 1989 2000 2001 2002 2003 2004 2005
 2006 2007 2008 2009 2078 2081 2083 2084 2089 2100 2101 2102 2103 2104
 2105 2106 2107]
Cluster 14
Number of images in this cluster: 103
Indices of images in this cluster: [ 153  156  165  195  204  216  222  252
  396  408  417  429  456  459  492  510  516  522  531  537  546  549
  561  588  591  594  615  645  660  693  720  756  762  771  783  792
  795  810  825  849  861  864  879  909  921  924  957  966  969  981
  990 1011 1023 1104 1107 1110 1113 1119 1128 1137 1146 1161 1167 1170
 1200 1212 1215 1218 1230 1236 1263 1281 1284 1317 1356 1365 1380 1386
 1392 1398 1401 1407 1425 1440 1452 1470 1500 1503 1530 1536 1581 1590
 1626 1629 1632 1635 1668]
Cluster 15
Number of images in this cluster: 126
Indices of images in this cluster: [ 157  169  172  175  190  205  241  250
  328  337  340  352  355  370  406  448  464  484  529  550  562  598
  610  619  635  640  646  655  665  673  701  703  715  734  739  751
  757  805  808  823  826  838  847  856  874  880  886  896  913  934
  940  952  958  965  968  982 1000 1018 1021 1025 1033 1039 1063 1076
 1081 1093 1099 1114 1129 1138 1159 1174 1177 1207 1219 1222 1225 1237
 1249 1255 1267 1273 1298 1300 1306 1309 1315 1321 1342 1369 1378 1390
 1459 1468 1475 1478 1483 1498 1502 1519 1528 1534 1537 1550 1576 1580
 1594 1624 1627 1630 1633 1636 1639 1645 1657 1660 1663 1666 1681 1687]
```

HIERARCHICAL CLUSTERING

```
Hierarchical Clustering Clusters:
Cluster 0
Number of images in this cluster: 237
Indices of images in this cluster: [ 146  151  158  166  170  173  176  179  181  194  196  197  206  209
  211  215  221  223  224  227  230  239  247  248  251  254  257  260
  269  284  287  314  317  320  323  326  335  347  371  376  377  383
  386  391  394  397  404  409  413  419  421  422  428  430  437  452
  454  470  472  479  485  491  493  496  518  530  539  544  572  587
  599  620  623  634  649  650  656  664  668  674  676  682  686  700
  716  728  737  752  758  760  763  766  770  775  785  791  793  796
  797  799  802  830  841  848  853  862  863  866  883  887  893  899
  902  916  917  923  953  956  962  967  983  988  989  994  998 1010
 1016 1024 1034 1037 1052 1061 1069 1078 1079 1082 1091 1106 1111 1115
 1118 1130 1145 1148 1151 1154 1160 1166 1169 1172 1180 1183 1193 1214
 1216 1220 1223 1244 1253 1259 1261 1268 1270 1276 1277 1283 1313 1327
 1328 1331 1346 1352 1355 1358 1361 1363 1370 1376 1388 1393 1394 1397
 1400 1403 1406 1414 1417 1421 1423 1427 1430 1436 1439 1451 1453 1457
 1460 1466 1469 1471 1477 1480 1481 1490 1499 1501 1508 1522 1532 1535
 1540 1544 1553 1559 1574 1577 1580 1583 1589 1606 1613 1625 1634 1637
 1646 1648 1655 1667 1670 1675 1676 1679 1685 1688 1694 1697 1703]
Cluster 1
Number of images in this cluster: 256
Indices of images in this cluster: [ 149  154  159  168  182  200  207  212  242  263  272  281  290  301
  302  305  308  316  322  329  334  349  350  368  387  388  389  395
  400  403  414  416  420  426  431  434  435  439  453  463  467  471
  477  482  483  487  489  494  495  506  508  509  512  514  524  527
  533  541  545  551  553  554  556  560  564  565  566  571  573  575
  577  578  590  592  601  607  618  624  627  630  636  637  644  648
```

```
Cluster 11
Number of images in this cluster: 45
Indices of images in this cluster: [1812 1814 1815 1820 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840
 1841 1842 1843 1844 1845 1982 1984 1989 2000 2001 2002 2003 2004 2005
 2006 2007 2008 2009 2078 2081 2083 2084 2089 2100 2101 2102 2103 2104
 2105 2106 2107]
Cluster 12
Number of images in this cluster: 88
Indices of images in this cluster: [ 157  169  190  205  256  262  265  268  310  319  328  337  355  370
  401  406  448  484  598  610  625  640  646  655  673  703  715  739
  751  757  808  823  856  880  901  940  952  958  982 1000 1018 1021
 1033 1039 1063 1081 1093 1129 1159 1174 1177 1207 1219 1222 1225 1237
 1249 1255 1267 1273 1300 1306 1309 1315 1321 1342 1369 1378 1399 1459
 1468 1483 1498 1519 1534 1537 1624 1627 1630 1636 1639 1645 1657 1660
 1663 1666 1681 1687]
Cluster 13
Number of images in this cluster: 132
Indices of images in this cluster: [ 147  153  156  162  165  177  195  204  216  222  252  312  324  333
  339  363  378  390  393  396  399  408  417  423  429  456  459  492
  510  516  522  531  537  546  549  561  588  591  594  615  645  660
  693  720  723  756  759  762  771  783  792  795  810  825  849  858
  861  864  879  909  921  924  957  966  969  981  990 1011 1023 1104
 1107 1110 1113 1119 1122 1128 1137 1146 1161 1170 1200 1215 1227 1230
 1236 1263 1281 1284 1299 1314 1317 1329 1356 1359 1365 1380 1386 1392
 1398 1401 1407 1410 1413 1425 1434 1440 1446 1452 1455 1458 1470 1488
 1494 1500 1503 1527 1530 1536 1545 1548 1563 1572 1578 1581 1590 1626
 1629 1632 1635 1647 1668 1686]
Cluster 14
Number of images in this cluster: 70
Indices of images in this cluster: [ 338  359  446  449  455  458  460  461  476  478  536  602  626  629
  635  638  680  683  698  701  704  761  782  788  812  824  851  872
  884  892  895  898  908  910  919  947 1046 1055 1057 1060 1085 1109
 1121 1124 1139 1142 1199 1208 1217 1229 1250 1271 1274 1286 1298 1316
 1319 1379 1385 1391 1454 1484 1502 1529 1565 1571 1610 1628 1631 1691]
Cluster 15
Number of images in this cluster: 108
Indices of images in this cluster: [ 150  186  189  213  225  231  243  255  264  267  288  291  303  306
  330  345  357  366  402  432  441  444  447  462  465  468  480  507
  555  558  567  570  612  621  633  678  702  711  714  735  765  789
  801  816  840  843  846  852  873  882  888  891  897  939  945  963
  975 1050 1056 1059 1071 1086 1089 1095 1098 1101 1134 1143 1149 1152
 1158 1167 1191 1206 1209 1212 1218 1287 1302 1335 1344 1371 1377 1389
 1404 1419 1428 1437 1443 1461 1482 1491 1506 1518 1521 1533 1542 1575
 1593 1599 1605 1641 1653 1662 1680 1683 1689 1692]
```

DBSCAN CLUSTERING

```
Cluster 0
Number of images in this cluster: 8
Indices of images in this cluster: [  46   79  128 1747 1889 1926 1935 2067]
Cluster 1
Number of images in this cluster: 5
Indices of images in this cluster: [  48  134 1752 1897 2054]
Cluster 2
Number of images in this cluster: 5
Indices of images in this cluster: [  49  142 1759 1893 2051]
Cluster 3
Number of images in this cluster: 5
Indices of images in this cluster: [  55  126 1745 1904 2055]
Cluster 4
Number of images in this cluster: 5
Indices of images in this cluster: [  66  138 1755 1912 2059]
Cluster 5
Number of images in this cluster: 5
Indices of images in this cluster: [  70  130 1749 1916 2060]
Cluster 6
Number of images in this cluster: 5
Indices of images in this cluster: [  71  133 1751 1918 2062]
Cluster 7
Number of images in this cluster: 5
Indices of images in this cluster: [  74  140 1757 1929 2070]
Cluster 8
Number of images in this cluster: 5
Indices of images in this cluster: [  75  129 1748 1922 2063]
Cluster 9
Number of images in this cluster: 7
Indices of images in this cluster: [  78  135 1753 1888 1924 1934 2065]
Cluster 10
Number of images in this cluster: 5
Indices of images in this cluster: [  82  136 1754 1932 2073]
Cluster 11
Number of images in this cluster: 5
Indices of images in this cluster: [  83  123 1743 1933 2074]
Cluster 12
Number of images in this cluster: 273
Indices of images in this cluster: [  85   86   87   88   89   90   91   92   93   94   95   96   97   98
   99  100  101  102  103  104  105  106  107  108  109  110  111  112
  113  114  127 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714
```

2)Braille character identification

CLASSIFICATION USING CNN

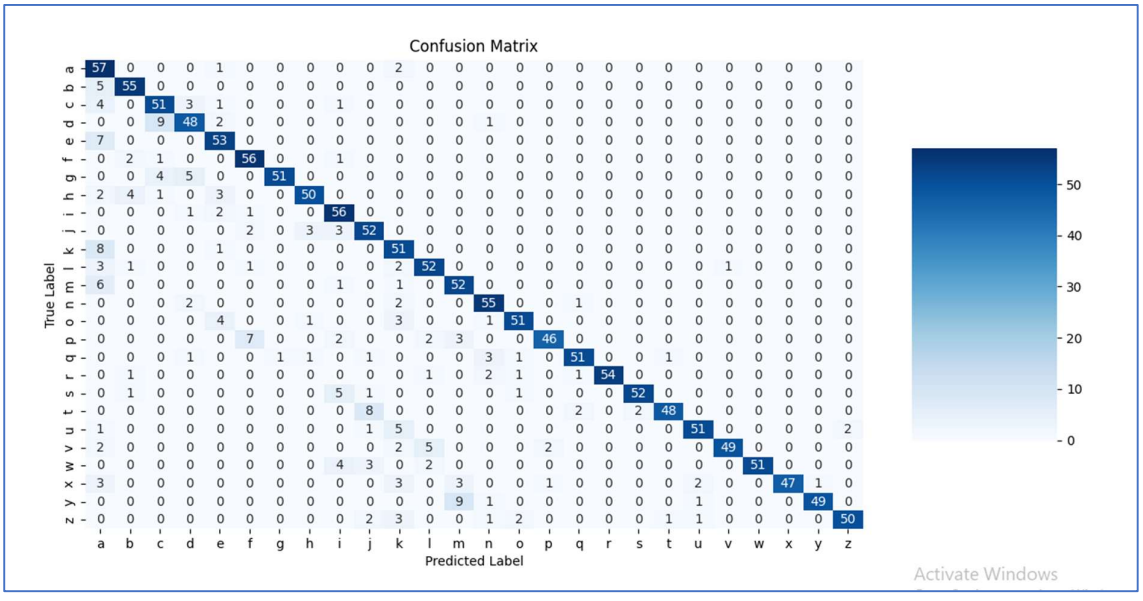
```
Model saved to disk.
Label Encoder saved to disk.
[1m 1/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m5s[0m 603ms/step - accuracy: 0.8125 - loss: 1.1841
[1m 2/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m5s[0m 634ms/step - accuracy: 0.8281 - loss: 1.2143
[1m 3/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m3s[0m 514ms/step - accuracy: 0.8438 - loss: 1.1421
[1m 4/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m2s[0m 477ms/step - accuracy: 0.8457 - loss: 1.1299
[1m 5/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m2s[0m 458ms/step - accuracy: 0.8466 - loss: 1.1803
[1m 6/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m1s[0m 449ms/step - accuracy: 0.8426 - loss: 1.2588
[1m 7/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m1s[0m 442ms/step - accuracy: 0.8364 - loss: 1.3487
[1m 8/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m0s[0m 436ms/step - accuracy: 0.8320 - loss: 1.4151
[1m 9/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m0s[0m 446ms/step - accuracy: 0.8294 - loss: 1.4554
[1m10/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m0s[0m 498ms/step - accuracy: 0.8269 - loss: 1.4942
[1m10/10[0m [32m ━━[0m[37m ━━━━━━━━━━━━━━━━━━━━[0m [1m5s[0m 507ms/step - accuracy: 0.8249 - loss: 1.5260
Test Accuracy: 0.8044871687889099
```



```
WARNING:absl:Compiled the loaded model, b
[1m1/1[0m [32m ━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m12s[0m 12s/step
Predicted Label: a
```

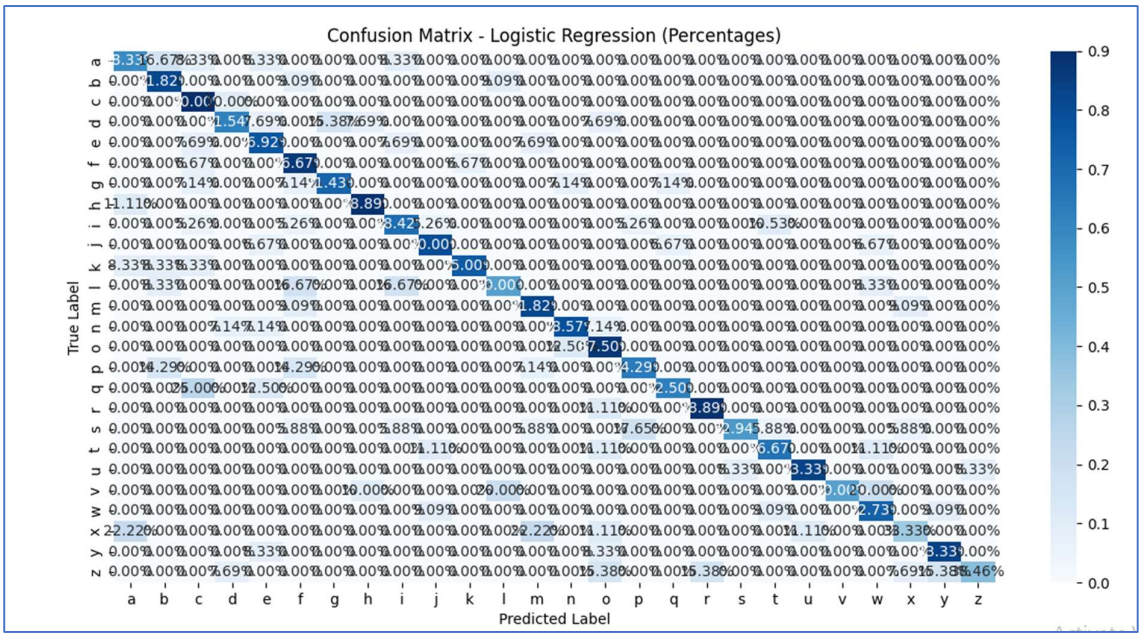
CLASSIFICATION USING K NEAREST NEIGHBOR

```
KNN Classifier saved to disk.
Label Encoder saved to disk.
Accuracy: 0.8576923076923076
Precision: 0.8759315327643641
Recall: 0.8576923076923076
F1-score: 0.8611901010079145
```



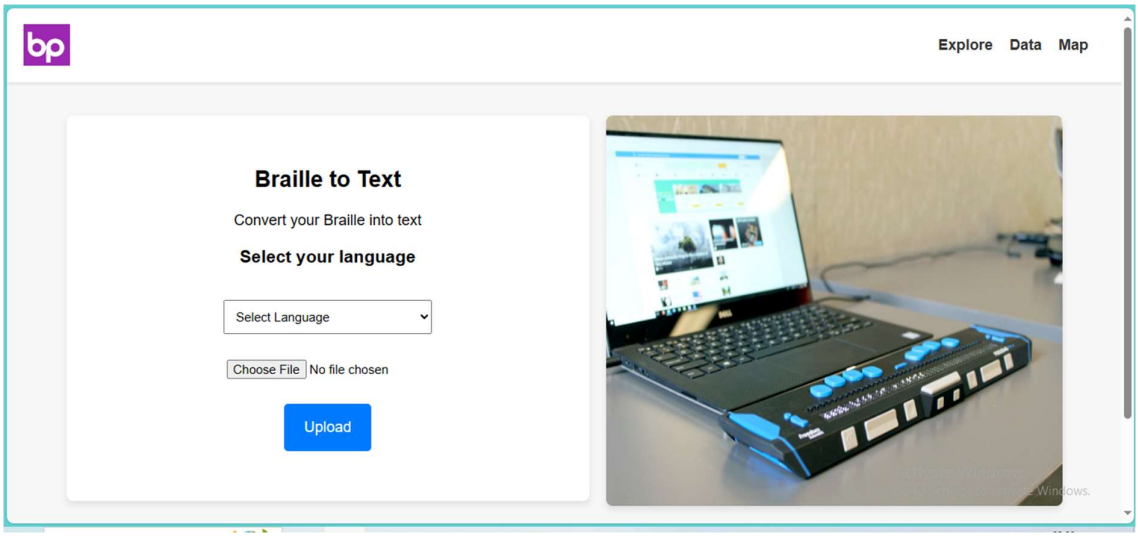
CLASSIFICATION USING LOGISTIC REGRESSION

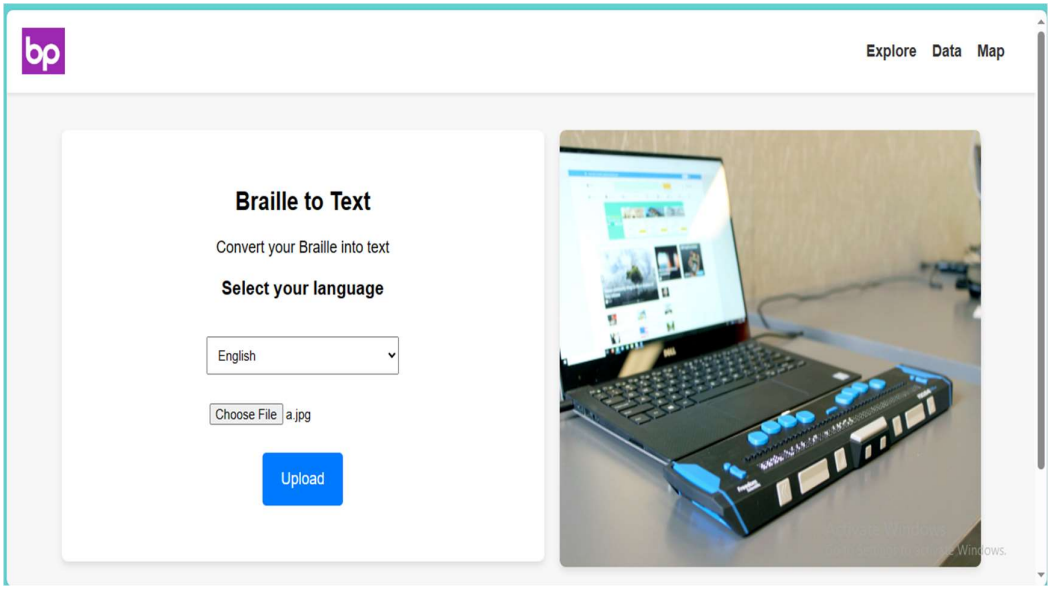
```
Logistic Regression Metrics:
Accuracy: 0.7019230769230769
Precision: 0.7293669403539313
Recall: 0.7019230769230769
F1-score: 0.6989455003603899
```

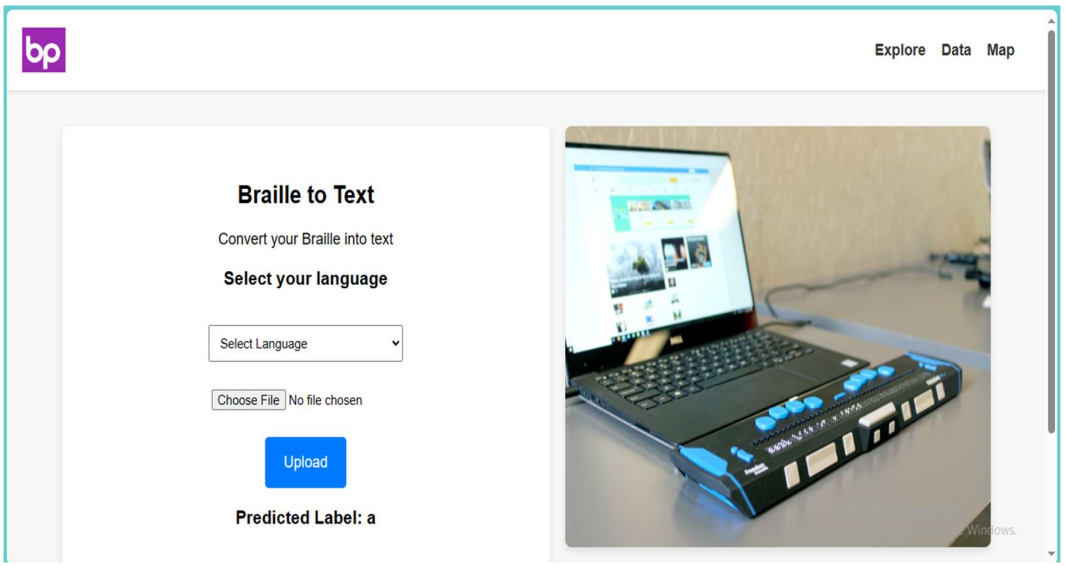


INTEGRATING TRAINED CNN MODEL IN BRAILLE TO TEXT MODULE:

brailletotext.html







Clustering of braille images and performance metrics of different models

In "Braille Pulse," each language in our dataset has been clustered into four distinct categories: alphabets, symbols, formatting elements, and numbers.We employ clustering algorithms such as k-means, spectral, and hierarchical clustering to organize Braille characters into meaningful categories.Additionally, we utilize silhouette scores to evaluate the effectiveness of our clustering methods. Silhouette scores measure how well each data point fits within its assigned cluster compared to other clusters.

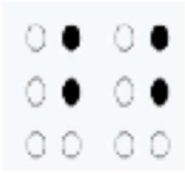
K-Mean Clustering model :

K-means clustering is a method used to divide a dataset into groups, or clusters, based on similarities between data points. It works by first guessing where the center of each cluster might be, then assigning data points to the nearest cluster center. After that, it adjusts the cluster centers to better fit the data points assigned to them, and repeats this process until the clusters are as tight and well-separated as possible.

```
Silhouette Score: 0.12588782608509064
```

Cluster 0 - Alphabets

Cluster 1 - Formatting



Cluster 2 - Numbers

Cluster 3 - Symbols

Hierarchial Clustering model :

Hierarchical clustering is a technique for grouping similar data points into clusters, forming a tree-like structure called a dendrogram. It starts by considering each data point as a separate cluster and then iteratively merges or splits clusters based on their similarities. There are two main approaches: agglomerative, where each data point begins in its own cluster and pairs of clusters are merged until everything is in one cluster, and divisive, where all data points start in one cluster and clusters are split recursively

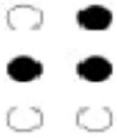
.

```
Silhouette Score (Hierarchical Clustering): 0.11682574450969696
```

Cluster 0 - Alphabets

Cluster 1 - Formatting

Cluster 2 - Numbers



Cluster 3 - Symbols

Spectral Clustering model :

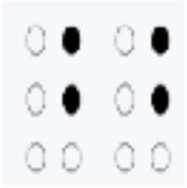
Spectral clustering is a method for clustering data points based on the similarity between them. Unlike traditional clustering algorithms like K-means or hierarchical clustering, spectral clustering works by transforming the data into a lower-dimensional space using the eigenvectors of a similarity matrix. This transformation helps to uncover the underlying structure of the data, making it easier to separate into clusters. Spectral clustering is particularly useful for datasets with complex structures or non-linear separations, where traditional methods may struggle.

```
Silhouette Score (Spectral Clustering): 0.13133947551250458
```

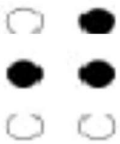
```
E:\anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory le
th MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable
=1.
  warnings.warn(
```

Cluster 0 - Alphabets

Cluster 1 - Formatting



Cluster 2 - Numbers



Cluster 3 - Symbols

CONCLUSION

In summary, this document involves dataset narration, class labeling, model performance evaluation, comparison table creation, and clustering analysis for language identification and character recognition using Braille images of different languages. This holistic approach enables effective language recognition and lays the groundwork for further advancements in assistive technology.

Ref: https://github.com/Dharshan465/BraillePulse.git