```
$HistoryLength = 0;
```

First download the CCAT_10 corpus to home directory. From there the code is as below.

```
c10Train = FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10train", "*"}]];
c10Test = FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10test", "*"}]];
allC10TrainFiles =
  FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10train", "*", "*"}]];
allC10TestFiles = FileNames[
   FileNameJoin[{$HomeDirectory, "c10", "C10test", "*", "*"}]];
AbsoluteTiming[allTrainWritings =
   Map[StringTake[Import[#, "Text"], {50, -50}] &, allC10TrainFiles];]
AbsoluteTiming[allTestWritings =
   Map[StringTake[Import[#, "Text"], {50, -50}] &, allC10TestFiles];]
allTrainWritingsPartitioned = Partition[allTrainWritings, 50];
allTestWritingsPartitioned = Partition[allTestWritings, 50];
MaxMemoryUsed[]
```

```
{1.31799, Null}
```

```
{1.30665, Null}
```

```
74 511 336
```

```
letterRules1 = {"d" | "p" -> "b", "k" | "q" | "x" | "z" -> "c", "f" -> "s",
    "h" | "j" -> "g", "n" -> "m", "y" | "i" | "v" | "w" -> "u",
    "?" | "!" | ":" -> ".", ";" | "," -> " ", "(" | ")" | "-" | "+" | "[" | "]" ->
     "(", "\t" | "\n" -> " ", "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ->
     "0"};
letterRules2 = Thread[{"a", "e", "o", "u", "b", "c", "g", "l", "m", "r", "s", "t",
      ".", "(", " ", "0"} -> IntegerDigits[Range[0, 15], 4, 2]];
```

```
AbsoluteTiming[traintextLetters = Map[Characters[ToLowerCase[RemoveDiacritics[#]]] &,
    allTrainWritingsPartitioned, {2}];]
AbsoluteTiming[traindigitseqs = Map[Developer`ToPackedArray[
       (IntegerDigits[Flatten[# /. letterRules1 /. letterRules2], 2, 2] /.
         IntegerDigits[__] :> Nothing)] &, traintextLetters, {2}];]

AbsoluteTiming[testtextLetters = Map[Characters[ToLowerCase[RemoveDiacritics[#]]] &,
    allTestWritingsPartitioned, {2}];]
AbsoluteTiming[testdigitseqs = Map[Developer`ToPackedArray[
       (IntegerDigits[Flatten[# /. letterRules1 /. letterRules2], 2, 2] /.
         IntegerDigits[__] :> Nothing)] &, testtextLetters, {2}];]

trainauthors = Map[FileNameTake, c10Train];
ltlen = Length[trainauthors];
testauthors = Map[FileNameTake, c10Test];
MaxMemoryUsed[]
```

```
{0.213011, Null}
```

```
{3.66352, Null}
```

```
{0.19793, Null}
```

```
{3.65633, Null}
```

```
365 410 840
```

```
makePositionsC = Compile[{{shifts, _Integer, 2}, {k, _Integer}},
    Module[{posns},
     posns = FoldList[Mod[2 * #1 + #2, 2^k] &, Reverse@shifts];
     Most[Reverse[Map[{2^k, 1} + {-1, 1} * Reverse[#] &, posns]]]
    ], RuntimeOptions → "Speed", CompilationTarget → "C"];

FCGR[chars_, k_] := Module[
  {posns, newposns},
  newposns = Round[makePositionsC[Cases[chars, {_Integer, _Integer}], k]];
  Normal[SparseArray[Apply[Rule, Tally[newposns], {1}], {2^k, 2^k}]]
 ]

pixLevel = 7;
AbsoluteTiming[
 trainimages1a = Table[FCGR[traindigitseqs[[j, k]], pixLevel], {j, ltlen},
    {k, Length[traindigitseqs[[j]]]}];]
AbsoluteTiming[testimages1a = Table[FCGR[testdigitseqs[[j, k]], pixLevel],
    {j, ltlen}, {k, Length[testdigitseqs[[j]]]}];]
MaxMemoryUsed[]
```

```
{2.75244, Null}
```

```
{2.7492, Null}
```

```
499 620 624
```

```
expon = 1 / 10;
trainimages1 = Map[(# / N[Max[#]])^expon &, trainimages1a, {2}];
testimages1 = Map[(# / N[Max[#]])^expon &, testimages1a, {2}];
MaxMemoryUsed[]
trainSetLabels = Flatten[
   Table[ConstantArray[trainauthors[[j]], Length[trainimages1[[j]]]], {j, ltlen}]];
trainImages = Apply[Join, trainimages1];
testSetLabels = Flatten[
   Table[ConstantArray[testauthors[[j]], Length[testimages1[[j]]]], {j, ltlen}]];
testImages = Apply[Join, testimages1];
Length[testImages]
MaxMemoryUsed[]
```

```
2 724 617 144
```

```
500
```

```
2 724 617 144
```

```
nearestImages[ilist_, vals_, dn_, dnum_, keep_] :=
 Module[
  {idata = ilist, dcts, top,
   topvecs, uu, ww, vv, udotw, norms},
  dcts = Map[FourierDST[# - Mean[Flatten[#]], dnum] &, idata];
  top = dcts[[All, 1 ;; dn + 1, 1 ;; dn + 1]];
  topvecs = Map[Flatten, top];
  {uu, ww, vv} =
   SingularValueDecomposition[topvecs, keep];
  udotw = uu.ww;
  norms = Map[Sqrt[#.#] &, udotw];
  udotw = udotw / norms;
  {Nearest[udotw → Transpose[{udotw, vals}], Method → "KDTree"], vv}]

processInput[ilist_, vv_, dn_, dnum_] :=
 Module[
  {idata = ilist, dcts, top,
   topvecs, tdotv, norms},
  dcts = Map[FourierDST[# - Mean[Flatten[#]], dnum] &, idata];
  top = dcts[[All, 1 ;; dn + 1, 1 ;; dn + 1]];
  topvecs = Map[Flatten, top];
  tdotv = topvecs.vv;
  norms = Map[Sqrt[#.#] &, tdotv];
  tdotv = tdotv / norms;
  tdotv]

keep = 28;
dn = 60;
dst = 2;
AbsoluteTiming[{nfY, vvY} =
  nearestImages[trainImages,
   trainSetLabels,
   dn, dst, keep];
testvecsY =
  processInput[testImages, vvY, dn, dst];]
MaxMemoryUsed[]
```

```
{0.625419, Null}
```

```
2 724 617 144
```

```
nnbrs = 16;
nbrScores = Table[nbrlist = nfY[testvecsY[[j]], nnbrs];
   dists = 1 / Map[Norm[testvecsY[[j]] - #] &, nbrlist[[All, 1]]];
   Thread[{nbrlist[[All, 2]], dists / Total[dists]}], {j, Length[testvecsY]}];
nbrScoresCollected = Map[Take[SortBy[#, -#[[2]] &], UpTo[4]] &,
   Map[Normal[GroupBy[#, First]] &, nbrScores] /.
    (val_ → vlist : {{val_, _} ..}) :⇢ (val → Total[vlist[[All, 2]]])];
allFTTPCAScores = (testauthors /. nbrScoresCollected) /. Thread[testauthors → 0.0];
results = Transpose[{testSetLabels, nbrScoresCollected}];
firsts = Cases[results, {a_, {a_ → _, ___}}] // Length
seconds = Cases[results, {a_, {_, a_ → _, ___}}] // Length
tot = firsts + seconds
```

```
325
```

```
122
```

```
447
```

```
preprocFunc = Identity;
SeedRandom[11 112 222];
AbsoluteTiming[
 cfunc = Classify[Map[preprocFunc, trainImages] -> trainSetLabels,
    Method → "LogisticRegression", PerformanceGoal → "Quality"];
]
MaxMemoryUsed[]
AbsoluteTiming[resultprobsraw =
   (testauthors /. Map[cfunc[#, "Probabilities"] &, Map[preprocFunc, testImages]]) /.
    Thread[testauthors :⇢ 0];
]
MaxMemoryUsed[]
```

```
{6.40604, Null}
```

```
2 724 617 144
```

```
{3.95109, Null}
```

```
2 724 617 144
```

```
results = Transpose[{testSetLabels, resultprobsraw}];
results1 = Map[{#[[1]], Ordering[#[[2]]][[-1]]} &, results] /.
   Thread[Range[Length[testauthors]] → testauthors];
t1 = Tally[results1];
results2 = Map[{#[[1]], Ordering[#[[2]]][[-2]]} &, results] /.
   Thread[Range[Length[testauthors]] → testauthors];
t2 = Tally[results2];
results3 = Map[{#[[1]], Ordering[#[[2]]][[-3]]} &, results] /.
   Thread[Range[Length[testauthors]] → testauthors];
t3 = Tally[results3];
results4 = Map[{#[[1]], Ordering[#[[2]]][[-4]]} &, results] /.
   Thread[Range[Length[testauthors]] → testauthors];
t4 = Tally[results4];
c1 = Total[Cases[t1, {{s_, s_}, v_} :> v]];
c2 = Total[Cases[t2, {{s_, s_}, v_} :> v]];
{c1, c2}
{c1, c2} / 500.
{(c1 + c2), (c1 + c2) / 500.}
```

```
{411, 66}
```

```
{0.822, 0.132}
```

```
{477, 0.954}
```

So 82.2% correct and another 13.2% are the second guess.