Project 2 - Image Classification
CS 4346.002
Erik Olvera and DeAndre Johnson

In this project, we worked with two datasets: handwritten digits and face images, both provided in text format. Each dataset consists of separate files for training, validation, and testing. For the digit dataset, the training labels consist of 5000 lines and the training images contain 140,000 lines. By dividing the total number of image lines by the number of labels, we determined that each digit image is 28 lines tall. The face training labels contain 451 lines and the training images contain 31,570 lines, resulting in each face image being 70 lines tall. These calculations allowed us to properly segment the large image files into individual images.

To load the data, we implemented a read_labels function that reads each label file, converts each line to an integer, and returns a clean list of labels. For the image files, we wrote a read_images function that reads all lines of a file, computes the correct number of lines per image based on the number of labels, and slices the text into separate images. Each image is stored as a list of strings representing its rows. This completed the data processing step, giving me structured lists for all digit and face images across the training, validation, and test sets.

After loading the data, we implemented two types of features for each image. The first feature type is a raw pixel feature, where each pixel is converted into a binary value: a 1 if the character is a # or +, and a 0 if it is whitespace. This produces a long binary vector representing the entire image in pixel form. The second feature type is a simple counting feature, where we count the total number of whitespace characters and the total number of symbol characters within each image, returning them as a two element vector.

For both digit and face datasets, we applied my feature extractors to every training image to produce a set of raw pixel feature vectors and a set of counting feature vectors. We then

combined these two sets by concatenating each image's raw pixel vector with its corresponding counting vector, forming a unified feature vector per image.

After generating the unified feature vectors for each image, we implemented two learning algorithms to classify both the digit and face datasets: Naive Bayes and Perceptron. Both algorithms operate directly on the binary pixel features and the counting features extracted earlier.

For the Naive Bayes classifier, we created a NaiveBayes class that tracks how many examples belong to each class and how often each feature value appears for that class. During training, the code updates these counts for every feature in each training example. Since our features include both binary values and count-based values, the implementation discretizes the counts so they can be handled in the same structure. When predicting, the classifier computes the log probability for each class by combining the smoothed prior with the smoothed likelihood of every feature, and the class with the highest score is returned.

For the Perceptron classifier, we implemented a Perceptron class that maintains a weight vector and bias for each class. The model predicts a class by taking the dot product between the feature vector and each class's weights. During training, the Perceptron runs for several epochs and updates the weights whenever the prediction is incorrect. It increases the weights for the correct class and decreases them for the incorrectly predicted class, shifting the decision boundaries so the model improves over time.

To evaluate both algorithms, we wrote an evaluate_classifier function that trains and tests the models using different percentages of the training data, from 10% up to 100%. For each percentage, the function randomly samples a subset of the training images, extracts their features,

trains the classifier, and measures accuracy on the full test set. This process is repeated five times, and we compute the mean accuracy, the standard deviation, and the average training time.

We used a run_experiments function to automate this process for both datasets. It collects the results for Naive Bayes and Perceptron and prints summary tables showing how accuracy and runtime change as more training data is used. This allows a direct comparison between the two algorithms under the same experimental setup.

Overall, both classifiers improved as more training data was used. Naive Bayes trains very quickly and performs consistently, while the Perceptron benefits from multiple epochs and can reach higher accuracy when enough data is available. These results show how our feature design and training approach impact performance and provide a basis for future improvements such as adding more advanced features or exploring other learning algorithms.

Results:

```
================================================================================
CS 4346 - IMAGE CLASSIFICATION PROJECT
================================================================================


================================================================================
EXPERIMENTS ON DIGIT DATASET
================================================================================

Training with 10% of data...
  Naive Bayes    - Accuracy: 73.36% (±1.59), Time: 0.1148s
  Perceptron     - Accuracy: 19.22% (±1.48), Time: 5.9081s
```

Training with 20% of data...

Naive Bayes - Accuracy: 76.22% (±1.02), Time: 0.2122s

Perceptron - Accuracy: 17.52% (±2.99), Time: 11.5457s

Training with 30% of data...

Naive Bayes - Accuracy: 75.72% (±1.16), Time: 0.3319s

Perceptron - Accuracy: 20.06% (±2.83), Time: 17.3026s

Training with 40% of data...

Naive Bayes - Accuracy: 76.20% (±0.74), Time: 0.4143s

Perceptron - Accuracy: 21.08% (±0.93), Time: 23.1629s

Training with 50% of data...

Naive Bayes - Accuracy: 76.26% (±0.61), Time: 0.5372s

Perceptron - Accuracy: 17.92% (±5.58), Time: 28.8786s

Training with 60% of data...

Naive Bayes - Accuracy: 77.02% (±0.75), Time: 0.6419s

Perceptron - Accuracy: 18.18% (±1.54), Time: 35.1365s

Training with 70% of data...

Naive Bayes - Accuracy: 77.04% (±0.44), Time: 0.7340s

Perceptron - Accuracy: 18.56% (±1.02), Time: 40.5511s

Training with 80% of data...

Naive Bayes - Accuracy: 76.74% (±0.20), Time: 0.9281s

Perceptron - Accuracy: 24.04% (±4.69), Time: 48.7021s

Training with 90% of data...

Naive Bayes - Accuracy: 76.80% (±0.22), Time: 1.0408s

Perceptron - Accuracy: 22.54% (±4.61), Time: 54.8367s

Training with 100% of data...

Naive Bayes - Accuracy: 76.90% (±0.00), Time: 1.1473s

Perceptron - Accuracy: 22.24% (±3.43), Time: 61.8729s

SUMMARY TABLE - DIGIT

%Data	NB Acc	NB Std	NB Time	P Acc	P Std	P Time
10	73.36	1.59	0.1148	19.22	1.48	5.9081
20	76.22	1.02	0.2122	17.52	2.99	11.5457
30	75.72	1.16	0.3319	20.06	2.83	17.3026
40	76.20	0.74	0.4143	21.08	0.93	23.1629
50	76.26	0.61	0.5372	17.92	5.58	28.8786
60	77.02	0.75	0.6419	18.18	1.54	35.1365
70	77.04	0.44	0.7340	18.56	1.02	40.5511
80	76.74	0.20	0.9281	24.04	4.69	48.7021
90	76.80	0.22	1.0408	22.54	4.61	54.8367
100	76.90	0.00	1.1473	22.24	3.43	61.8729

==========

EXPERIMENTS ON FACE DATASET

==========

Training with 10% of data...

Naive Bayes - Accuracy: 73.20% (±4.14), Time: 0.1101s

Perceptron - Accuracy: 49.60% (±1.44), Time: 0.8019s

Training with 20% of data...

Naive Bayes - Accuracy: 83.07% (±5.47), Time: 0.1204s

Perceptron - Accuracy: 56.00% (±7.76), Time: 1.6453s

Training with 30% of data...

Naive Bayes - Accuracy: 84.80% (±1.60), Time: 0.1622s

Perceptron - Accuracy: 55.73% (±5.71), Time: 2.4106s

Training with 40% of data...

Naive Bayes - Accuracy: 87.60% (±1.77), Time: 0.2264s

Perceptron - Accuracy: 52.67% (±5.42), Time: 3.2409s

Training with 50% of data...

Naive Bayes - Accuracy: 86.80% (±1.42), Time: 0.2893s

Perceptron - Accuracy: 56.27% (±9.31), Time: 4.0050s

Training with 60% of data...

Naive Bayes - Accuracy: 87.73% (±2.09), Time: 0.3075s

Perceptron - Accuracy: 55.87% (±7.22), Time: 4.8506s

Training with 70% of data...

Naive Bayes - Accuracy: 87.47% (±1.07), Time: 0.4192s

Perceptron - Accuracy: 50.80% (±3.11), Time: 5.7737s

Training with 80% of data...

Naive Bayes - Accuracy: 88.80% (±0.78), Time: 0.4781s

Perceptron - Accuracy: 56.80% (±8.85), Time: 6.5958s

Training with 90% of data...

Naive Bayes - Accuracy: 89.07% (±0.68), Time: 0.5294s

Perceptron - Accuracy: 55.87% (±7.51), Time: 7.5602s

Training with 100% of data...

Naive Bayes - Accuracy: 90.00% (±0.00), Time: 0.5722s

Perceptron - Accuracy: 53.33% (±6.56), Time: 8.1514s

SUMMARY TABLE - FACE

%Data NB Acc NB Std NB Time P Acc P Std P Time

10	73.20	4.14	0.1101	49.60	1.44	0.8019
20	83.07	5.47	0.1204	56.00	7.76	1.6453
30	84.80	1.60	0.1622	55.73	5.71	2.4106
40	87.60	1.77	0.2264	52.67	5.42	3.2409
50	86.80	1.42	0.2893	56.27	9.31	4.0050
60	87.73	2.09	0.3075	55.87	7.22	4.8506
70	87.47	1.07	0.4192	50.80	3.11	5.7737
80	88.80	0.78	0.4781	56.80	8.85	6.5958
90	89.07	0.68	0.5294	55.87	7.51	7.5602
100	90.00	0.00	0.5722	53.33	6.56	8.1514

==========

ALL EXPERIMENTS COMPLETED!

==========