Aryan Raut (AR2300)

Abdeslam Boularias

CS-440

May 4, 2024

CS440 - Project Image Classification

Image Input

- Used the code provided on Berkeley's website to get all the inputs and feature extractions.

Algorithms

1. **Perceptron**

Features - The features used by this classifier were a set of pixel features. Each pixel could either be 0, or 1.

Weights - initialize the weights to be 0 in the beginning and then later modify them.

Training - have set the max iteration for perceptron (3) in the beginning. Once max iteration, stop training the program. also keep a list of weights to the features for each legal label. store these values in the python dictionary to store the lists where the key is the label and the values in the list of weights. Every iteration, loop over the training data, for each datum, and compute a list of numbers corresponding to all the legal labels using the equation below.

$$f(x_i, w) = w_0 + w_1\phi_1 + ... + w_j\phi_j$$

Now, pick the label with the highest f(xi,w) value as the guess. If the guess is correct then predict the right answer and no changes are required. If the guess is different then you need to update the weight list for the correct label and incorrect label(s) using the following equations.

$$\begin{aligned}
&fori = 1, 2, ..., j : weights[label][i] += \phi_i(datum)\\
&w_0 += 1\\
&fori = 1, 2, ..., j : weights[guess][i] -= \phi_i(datum)\\
&w_0 -= 1
\end{aligned}$$

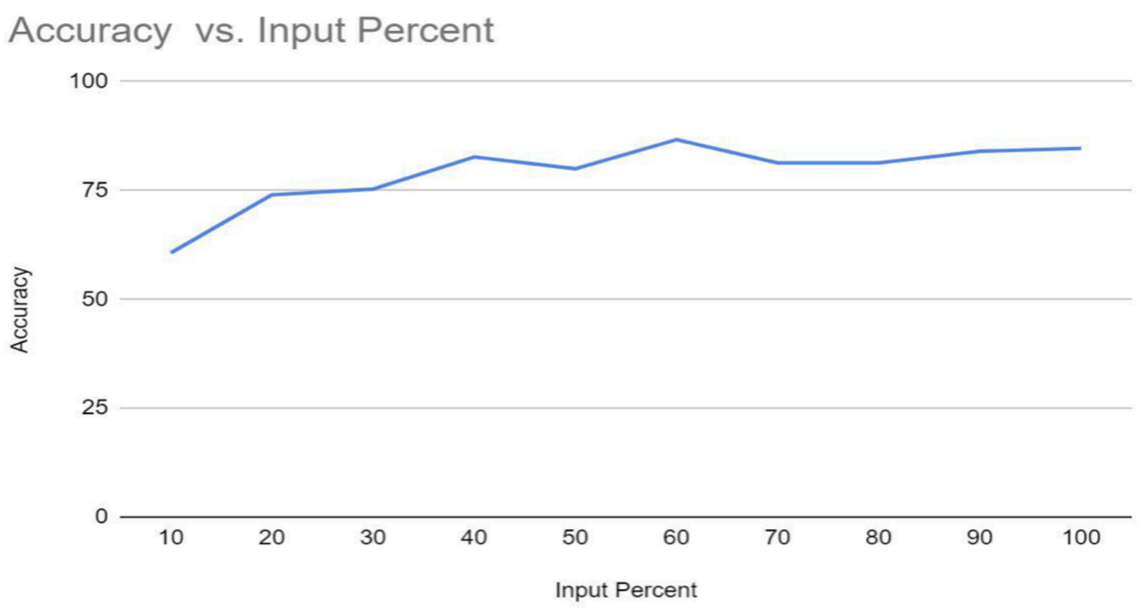
Classify - For classifying, calculate a list of numbers corresponding to all the legal labels using the following equation.

$$f(x_i, w) = w_0 + w_1\phi_1 + ... + w_j\phi_j$$

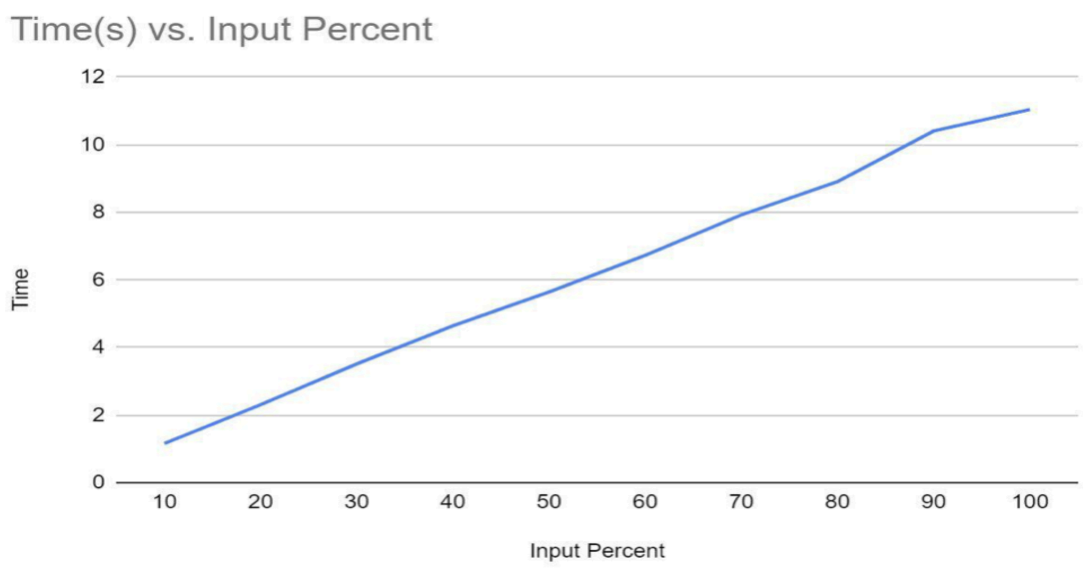
Pick the label with the highest f(x,w) as the guess.

The following are the results for both the digits and faces on 10%, 20%, 30% … 100% of the data using the Perceptron algorithm.

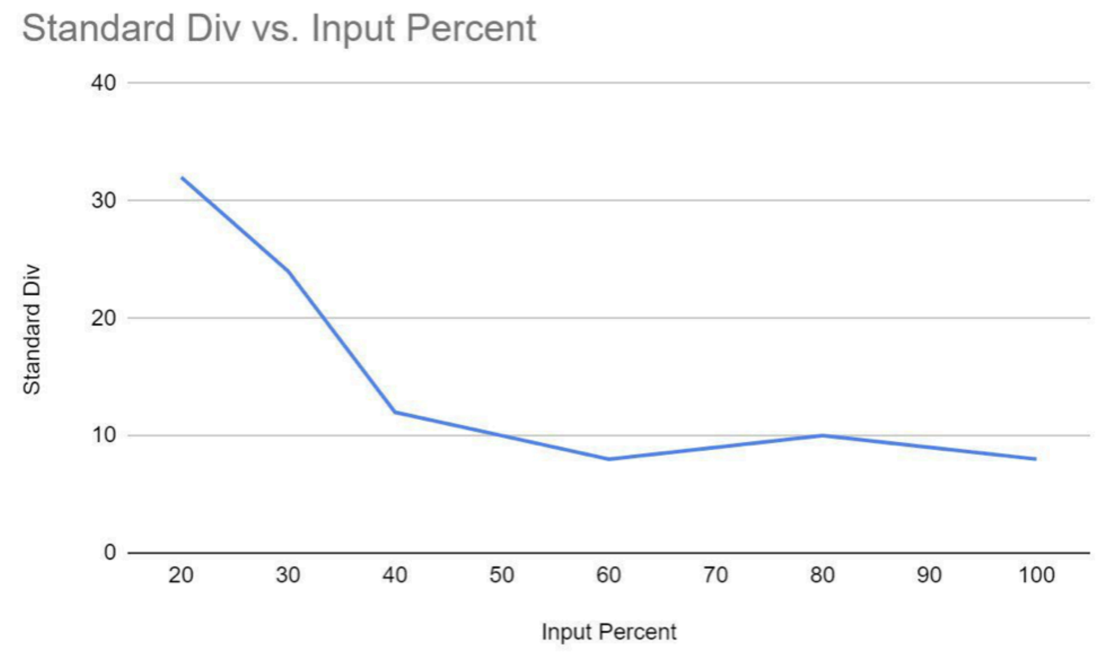
Perceptron Faces Accuracy



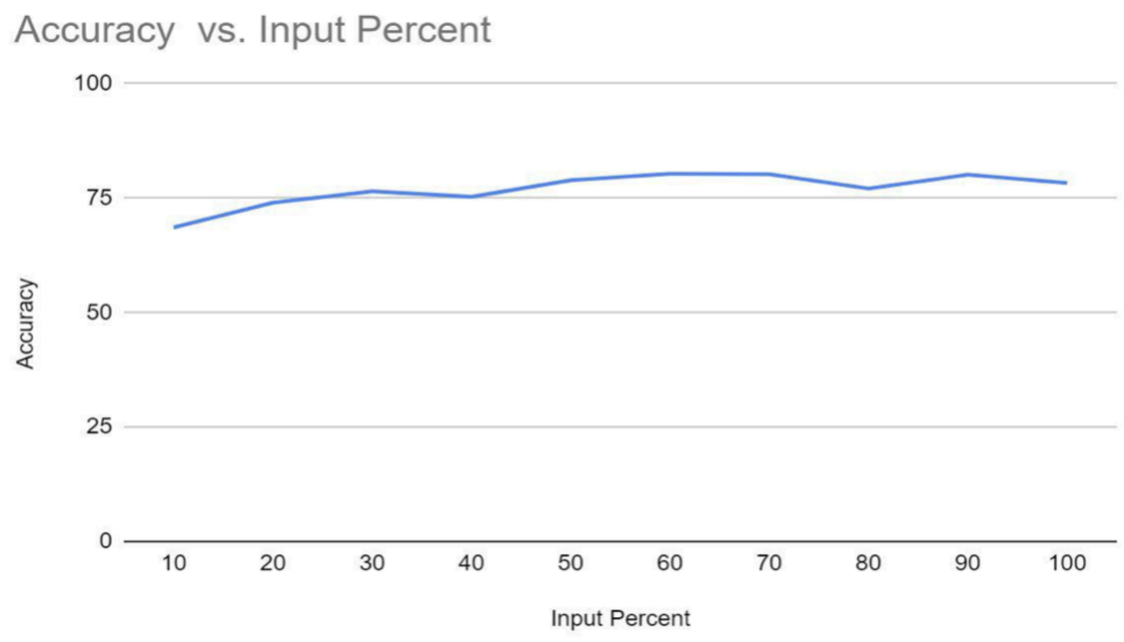
Perceptron Faces Timing



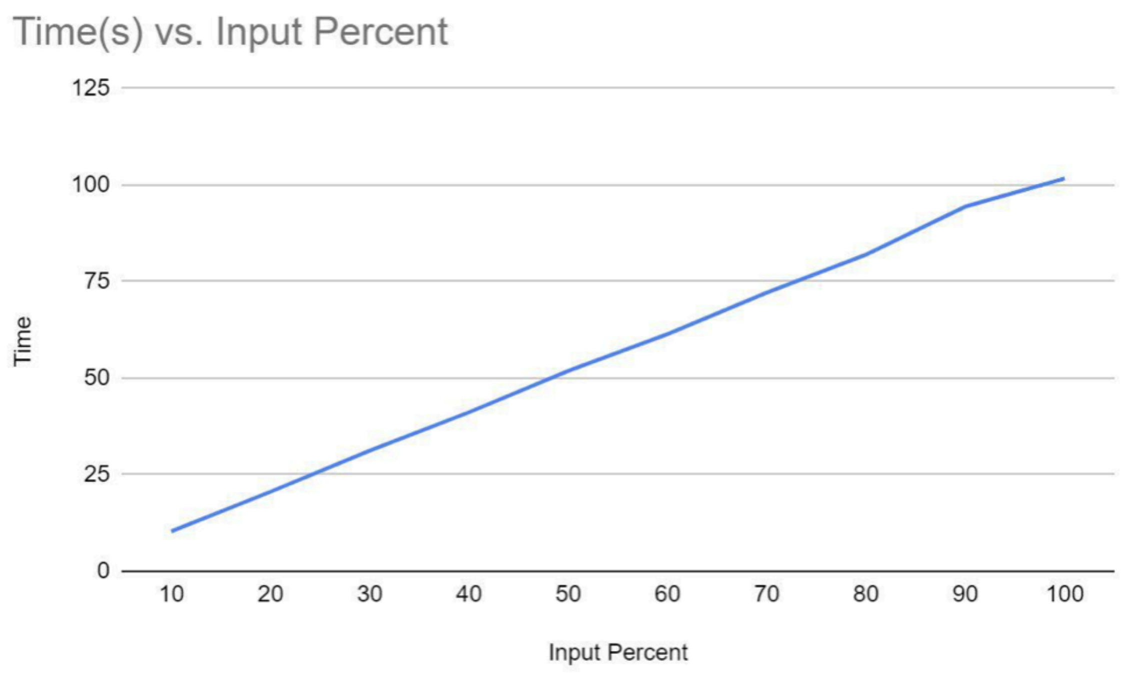
Perceptron Faces Standard Div



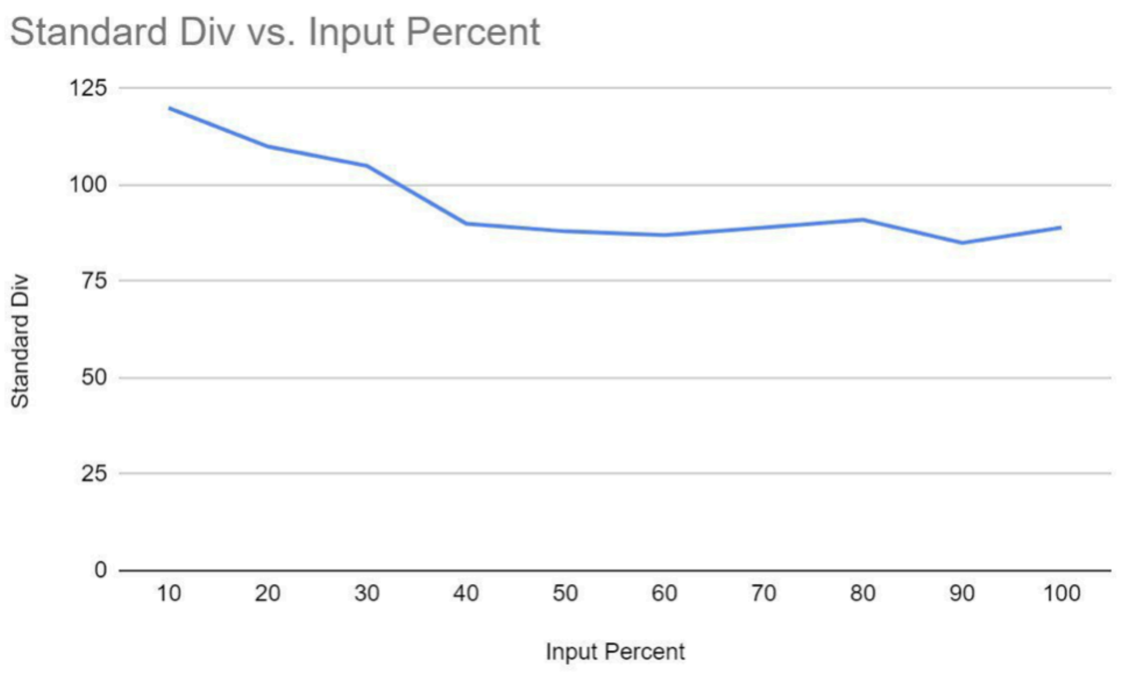
Perceptron Digits Accuracy



Perceptron Digits Timing



Perceptron Digits Standard Div



2. Neural Network

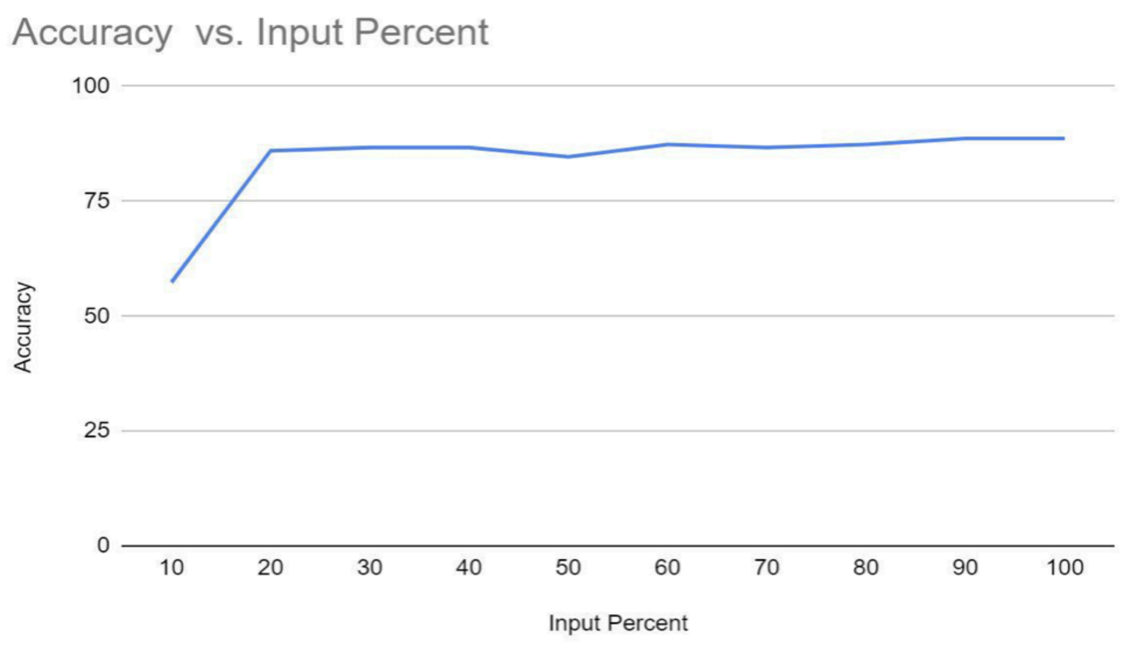
Features - Same as perceptrons, the features used by this classifier were a set of pixel features. Each pixel could either be 0 (white), or 1 (black/grey).

Weights - initialize the weights to be 0 in the beginning and then later modify them.

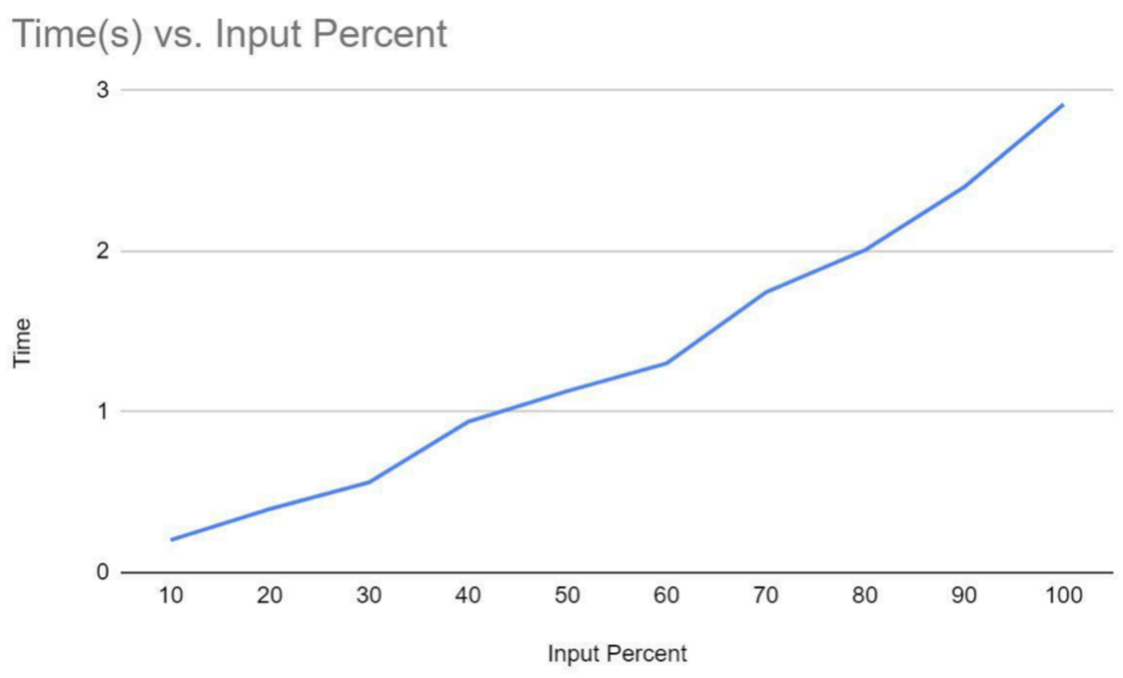
Training - involves providing the model with a large dataset of examples and their corresponding labels or target outputs, and using an optimization algorithm () to iteratively adjust the model's weights to minimize a loss function. Repeat over many iterations (epochs) on the dataset batches.

The following are the results for both the digits and faces on 10%, 20%, 30% … 100% of the data using the Perceptron algorithm.

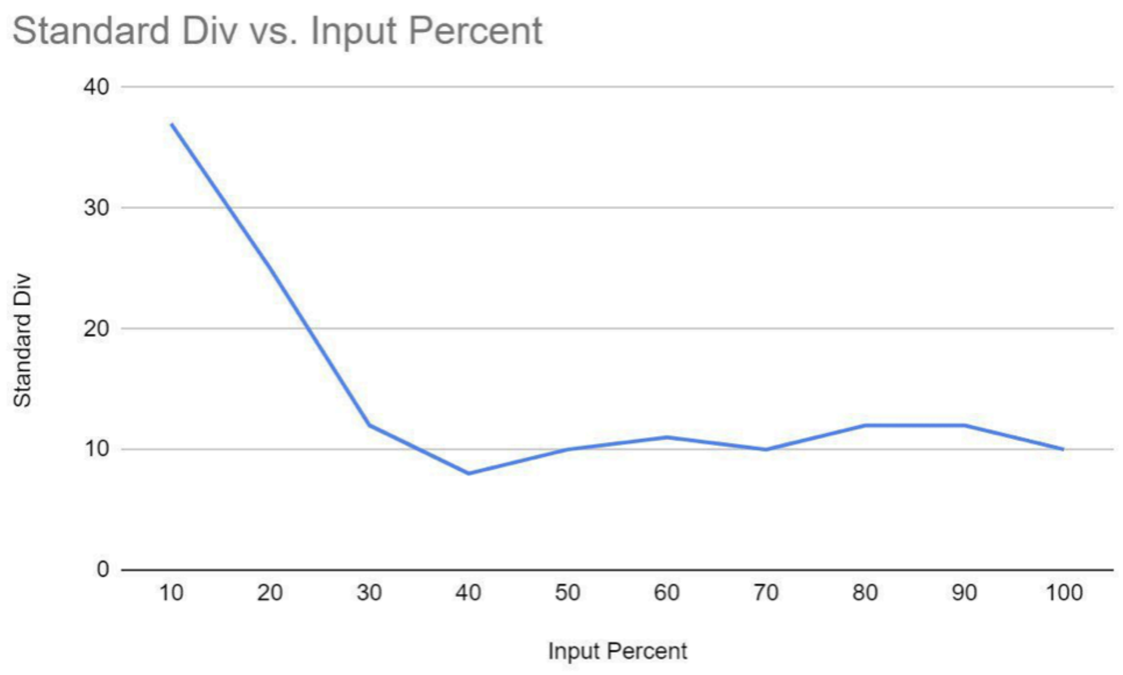
Neural Network Faces Accuracy



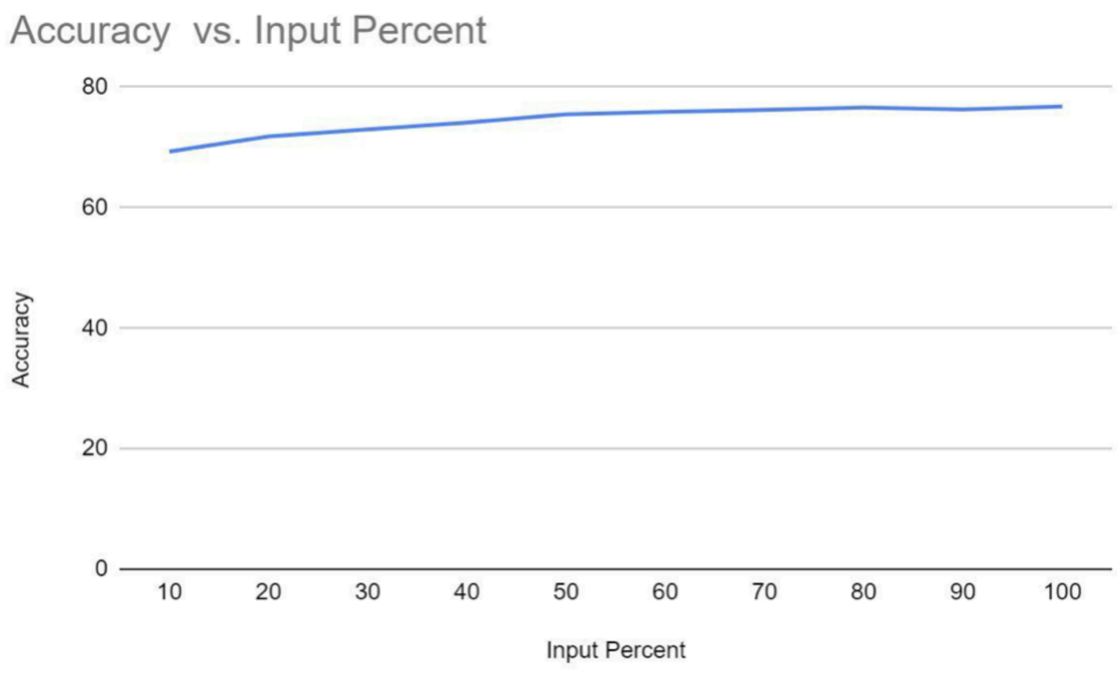
Neural Network Faces Timing



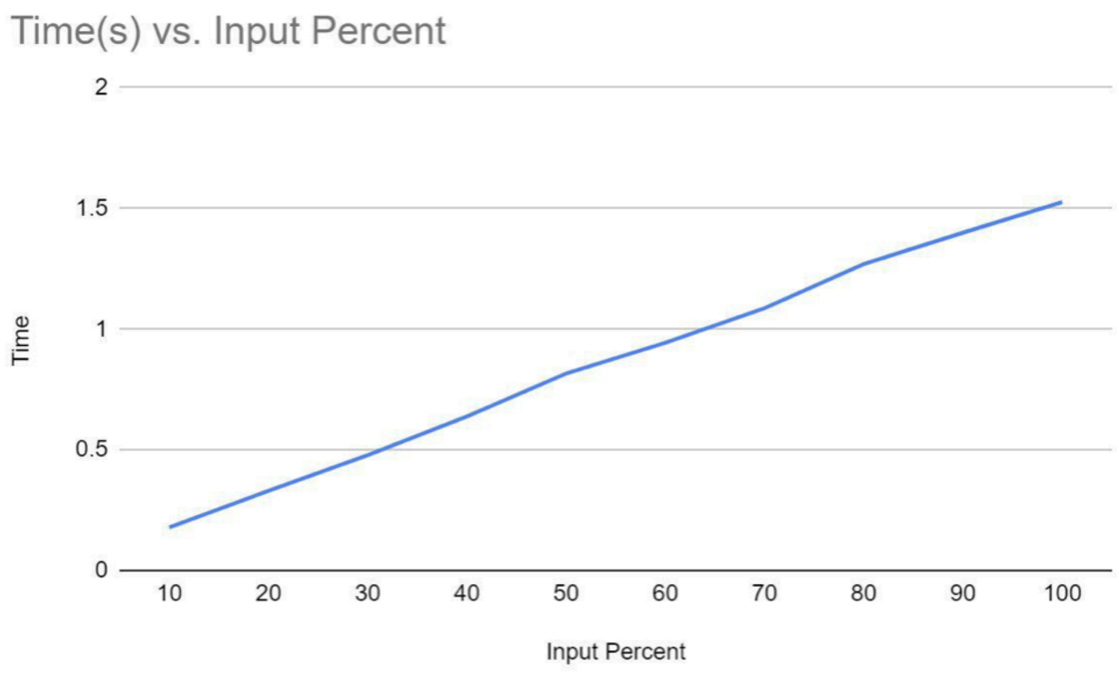
Neural Network Faces Standard Div



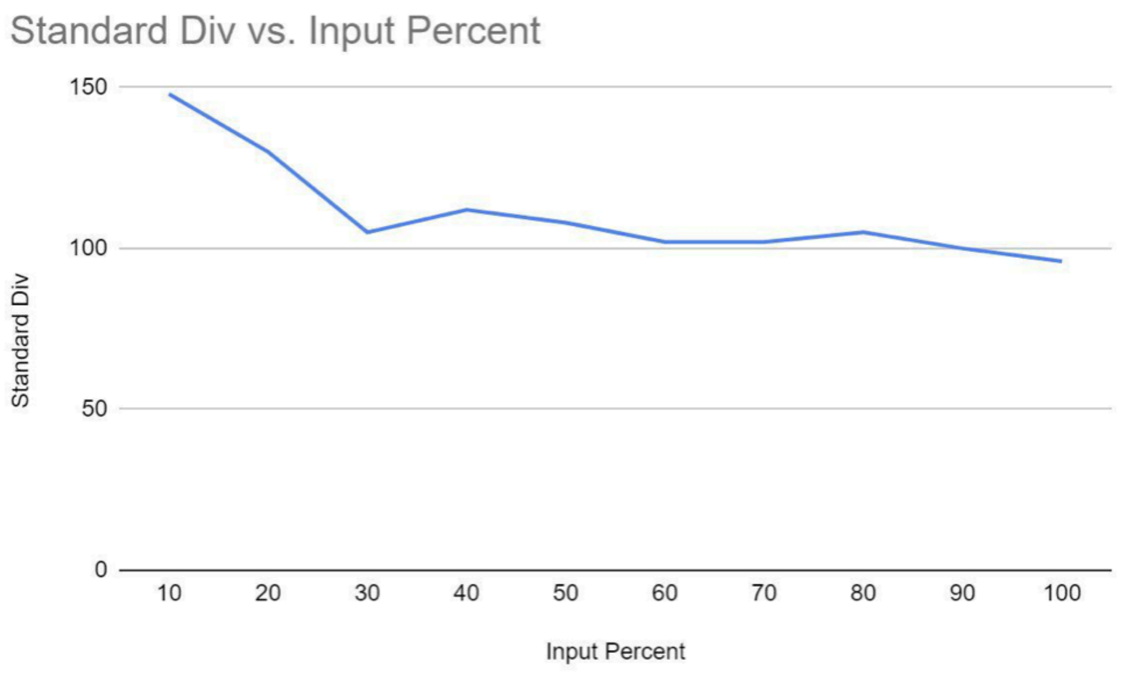
Neural Network Digits Accuracy



Neural Network Digits Timing



Neural Network Digits Standard Div



Discussion

We achieved an accuracy of over 75% across all algorithms, provided they were sufficiently trained. The training duration increased linearly with the volume of training data. Specifically, the Perceptron algorithm took the longest time to train. Interestingly, the increase in accuracy was not consistently correlated with the volume of data or the number of iterations; it typically started low, peaked midway through the training process, and then experienced fluctuations. In some instances, accuracy decreased after processing the full dataset. We noted that the standard deviation of accuracy was inversely related to the total amount of training data utilized. For instance, using about 50% of the data often led to a convergence in accuracy, which then stabilized and showed minimal improvement with additional data, indicating potential overfitting. Changes were rare but did occur. Key insights include recognizing that more training data does not necessarily equate to higher accuracy; instead, the quality of the data is crucial.

Testing is essential to determine data quality. Moreover, longer training times do not automatically result in better accuracy, suggesting that increasing the number of iterations indiscriminately may be inefficient.

Main Takeaway

The key takeaway is that increasing the volume of training data does not automatically enhance accuracy. Instead, accuracy tends to stabilize at a certain level, influenced by the features selected and the algorithm implemented. Therefore, when training our program, it's essential to conduct experiments to determine the optimal amount of training data needed to achieve this point of convergence. This approach helps avoid unnecessary expenditure of time on training without any improvement in accuracy.

Work Cited

Code Skeleton, Image Input, and features

http://inst.eecs.berkeley.edu/~cs188/sp11/projects/classification/classification.html