Performance Analysis of Hyperdimensional Computing for Character Recognition

Alec Xavier Manabat, Celine Rose Marcelo, Alfonso Louis Quinquito and Anastacia Alvarez
Electrical and Electronics Engineering Institute
University of the Philippines - Diliman
Quezon City, Philippines

***Abstract*—The Internet of Things (IoT) is an increasingly expanding network of devices. The whole premise of IoT is to provide comfort and convenience, requiring functions such as recognition. This is done through data - first collected by nodes, and then processed to provide meaningful information, commonly using machine learning. However, conventional machine learning techniques such as deep-learning require complicated operations, which may be computationally heavy for small devices. This study explores the capabilities of hyperdimensional computing (HDC), as a less complex architecture for data classification. The effect of dimensionality (and therefore energy) in the performance of HDC, done through a character recognition application, was explored. Results show that a dimensionality of 4,000-bit one-shot learning system yields an average accuracy close to that of its 12,000-bit counterpart at 0% distortion, and an average accuracy of 89.94% with 14.29% distortion. This study provides insights towards optimizations of HDC applications.**

***Index Terms*—hyperdimensional computing, HoloGN, character recognition**

I. Introduction

Digital devices work because of the switching behavior of thousands to millions of transistors in them. Decades of development have led to the reduction of these transistors to nanometer dimensions, allowing high computational power and low energy usage over a small chip area. While there are efforts to further scale down process technology, better computational architectures are needed to maximize the capabilities of these devices.

The Internet of Things (IoT) is a system that connects everyday objects to the internet. The IoT forms a network of mostly small devices like wearables, smartphones, and sensor nodes. These enable applications such as environmental monitoring, traffic management, and electricity distribution. These technologies are embedded with sensing capabilities which collect and process data into useful information, often modelling cognitive functions like data classification. Due to the portability of these devices, their longevity and resource management rely on the efficiency of algorithm, energy, and area of their design. The emergence of the IoT poses these new challenges, which are different from when the devices were only expected to perform calculations on numbers with high accuracy [1].

Neural nets and other conventional machine learning (ML) techniques are used in IoT to implement the cognitive functions. They are used in applications such as image recognition and numeric prediction. However, these update parameters and weights iteratively, requiring high energy and processing time that would not be ideal for local processing in the IoT.

On the other hand, hyperdimensional computing (HDC) models cognitive functions by using less complex operations and algorithms. This architecture uses vectors with high dimensionality, which bring about properties of neural representation: robustness and randomness. These hyperdimensional vectors can associate concepts, classify objects, and recognize patterns. Furthermore, the simple operations on hypervectors decrease computational requirements and high dimensionality enable single learning iteration.

Optical character recognition (OCR) is widely used nowadays, when there is a great need for making text searchable and accessible. One application of OCR is automatic plate recognition [2], which can be used in unattended parking lots, security control of restricted areas, and traffic law enforcement. HDC can be used in IoT-connected cameras for local processing and recognition, while potentially providing longer node lifetime. When implemented on IoT devices, HDC may solve the power and efficiency issues in the current state of ML, with high robustness against errors that may be caused by technology scaling [3].

This paper aims to extend the findings demonstrated through a character recognition application, adapted from [4]. Through this study, the authors aim to gain insight on possible energy optimization and its effect on the performance of the HDC. The input images are encoded to random hyperdimensional vectors for recognition while varying parameters: dimensionality, learning dataset size, image/character size, mode of learning, and bit distortion.

II. Related Work

A. Concepts of Hyperdimensionality

Connectionism refers to an approach in the study of human cognition which mathematically models the brain as a network of interconnected neurons [5]. One of the classes of connectionist network architectures is Vector Symbolic Architecture (*VSA*). VSAs model these connections through *binding* and *bundling* operators [6]. In VSAs, entities (i.e. roles, fillers, compositions) are solely represented as vectors.

As suggested by the large number of neurons and synapses present in the nervous system, it is plausible that our brain's circuits deal with ultrawide words [7]. These may be represented by vectors with **high dimensionality**, considered

hypervectors when dimensionality is in the thousands. These hypervectors have properties which imitate that of neural representation.

Robustness is shown by the brain by recalling successfully even when given partial specifications [8]. Suppose you are given the words "sport", "bat", and "pitcher". The brain will most probably retrieve the concept of "baseball" in spite of the loose description.

B. Operations in the Hyperspace

1) Distance Measurement: The dissimilarity of two hypervectors can be computed by measuring the **Hamming distance** or counting the indices that have dissimilar bits. By convention, two hypervectors are similar if their distance is less than half of the dimensionality.

2) Binding: A hypervector can be binded with information from another hypervector by component-wise exclusive-or (XOR) or permutation. The resulting hypervector is dissimilar to its arguments.

3) Bundling: Hypervectors are bundled into a single hypervector by component-wise addition, and majority function to binarize the sum hypervector. The bundled hypervector is similar to its arguments.

C. HoloGN Encoding on Character Recognition

In HDC, meaningful entities or parameters (e.g. pixel, frequency bin, etc.) are assigned index hypervectors. Values/data are stored in these hypervectors through *permutation*.

In [4], a character recognition application was implemented using HoloGN. The dataset includes $7x5$ images of Latin letters with either a black or white pixel value. Each of the 35 pixels are given random index hypervectors (HVs) which are stored in the item memory. Information of the pixel value is encoded using these index HVs by shifting them circularly - right when the color is white, and no shift otherwise. These information-storing HVs are then bundled into a single HV by componentwise addition. The bundled HVs are then binarized by thresholding - setting the element to 1 if the component value is greater than or equal to 17 (halfway of 0 and 35), and to 0 otherwise. The resulting hypervector characterizes the image, accounting for the value of each pixel.

III. Methodology

A. System Design

The index hypervectors and datasets for both training and testing are generated in MATLAB to provide the necessary inputs for simulation. Next, the system is modelled to provide results, particularly in accuracy.

The system is first initialized by filling the item memory with the generated pixel index hypervectors. These are random and orthogonal to each other (Hamming distance is $\approx \frac{D}{2}$). A dimensionality of $D = 10000$ bits is used for the base case. In the training phase, the associative memory is populated with hypervectors characterizing each letter of the alphabet. This phase is also known as the formation of the knowledge-base. Next, the testing phase involves the similarity measurement of input data (images to be recognized) against the letter hypervectors stored in the associative memory. Finally, the data gathered are analyzed in the evaluation phase.

B. Implementation of Modules

A block diagram of the modules are presented in Fig. 1.

1) Encoder: This module takes a pixel index and its corresponding value (either 0 or 1; black and white, respectively) as its inputs. It then fetches the pixel's index hypervector (*indexHV*) from the item memory (assuming that the system has already been initialized). As in HoloGN encoding, the *indexHV* is shifted according to the pixel value. As there are only two possible values, the *indexHV* is either circularly shifted to the right once (for white), or not shifted at all (for black). The shift operation is modelled by shift registers and the bundling operation requires accumulators and thresholding blocks.

This process of translating pixel values into HoloGN-encoded hypervectors (*holoHVs*) is done for all pixels in the image. Afterwards, these *holoHVs* are bundled to form the characteristic hypervector for an input image during the testing phase (*queryHV*). The *queryHV* will then be passed onto the **similarity test** with the characteristic letter hypervectors (*letterHVs*).

For the training phase, producing *letterHVs* still follows the process above, but these are instead **stored** into the associative memory as the learned patterns where *queryHVs* are compared against.

2) Item Memory: This memory block contains the *indexHVs* for each pixel. For $7x5$ images, there are 35 rows in this block.

3) Associative Memory: This memory block, on the other hand, contains the 26 *letterHVs* stored during the training phase. In the testing phase, similarity measurements between the *queryHV* and each *letterHV* are computed. The *letterHV* corresponding to the minimum Hamming distance (least number of different bits) is chosen as the recognized letter.

4) Supervised Controller: This module is a buffer module; responsible for holding the letterHVs of a single letter coming from the Encoder, and accumulates them to produce a single letterHV for that specific letter. This module is only active when running a supervised learning (rather than one-shot learning) mode of simulation. The accumulation process composes of a elementary matrix addition and binarization of the resulting hypervector by thresholding.

C. Simulation Phases

1) Training Phase: This first phase of the character recognition application involves storing learned patterns (in general, the original and undistorted letters) as *letterHVs* in the associative memory for reference in the testing phase.

The system also implements the three scenarios in [4], using the same $7x5$ black and white images of letters as base training set. The training set for supervised learning includes images with distortions of up to 15 bits, as mentioned in [4].

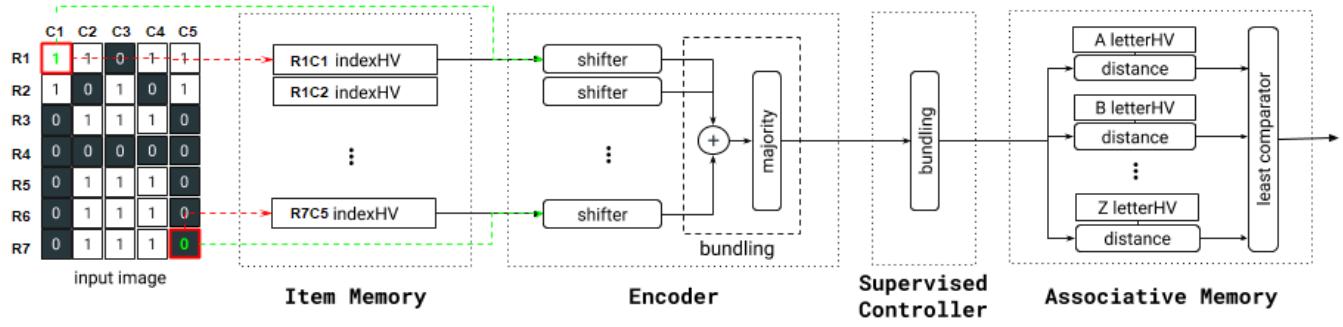


Fig. 1: Block diagram of the HD Character Recognition System.

2) Testing Phase: The testing phase is when test images are encoded to *queryHVs*, and least distance is measured versus stored *letterHVs* for recognition. The system is tested using varying dimensionality to gauge the efficiency of the approach, along with possible optimizations.

IV. Results and Analysis

This section discusses the characterization of the HD character recognition system. Accuracy across various scenarios and datasets are presented.

A. Hypervector Properties

In this subsection, the characteristics of the generated and encoded hypervectors will be discussed. The relation of distance and dimensionality is the main focus of the following analyses.

1) Item Memory: For the generation of 35 initial hypervectors (*indexHVs* in the previous diagrams), there are exactly $D/2$ 1s and exactly $D/2$ 0s. Their placements on the hypervector are then randomized. The occurrences of unique distances of the *indexHVs* are shown in Fig. 2. As seen in the graphs, the mean of the distribution is located at approximately $0.5D$. However, they differ in terms of the significance of each standard deviation. At $D = 10$, $std = 15.60\%$ of D, distances are scattered (there are hypervectors which differ by $0.2D$ and $0.8D$ bits) , which is shown in the rather flat normal distribution. This is in contrast to $D = 10000$, where $std = 0.48\%$ of D. Unique distances are more concentrated around the mean, as signified by the spike-like normal distribution. This evidence supports the purpose of operating in high dimensions, since orthogonality between random hypervectors is greatly assured.

2) Associative Memory: As described in the methodology section, the character recognition system is trained with the use of HoloGN encoding. The probability distribution of *letterHVs* is shown in Fig. 3 under one-shot learning. The distributions have a mean of $0.28D$ across the different dimensionalities. This shows that the 26 hypervectors contained in the associative memory (which wholly describe each letter) are about equal parts similar and different from each other. All letters have background and foreground pixels (white and black, respectively), and their similarity comes from the intersection

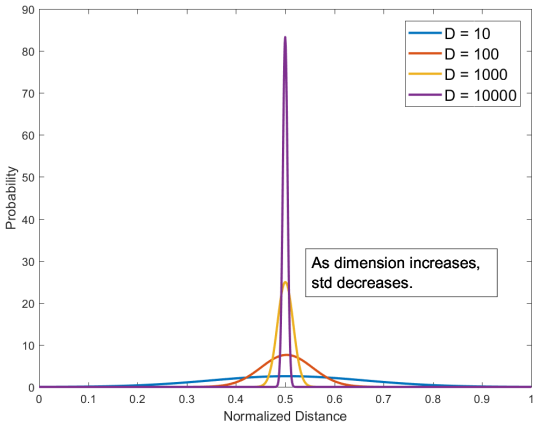


Fig. 2: Item Memory, 35 HVs, distance distribution across varying dimensionalities.

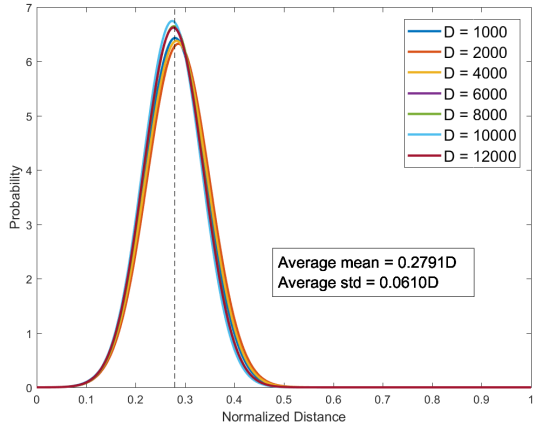


Fig. 3: Associative Memory, 26 HVs, distance distribution across varying dimensionalities.

of these. On the other hand, the difference, logically, is due to the distinguishing features of each letter. The graph shows that *letterHVs* are mostly non-orthogonal to each other, and they differ due ti the noise introduced during encoding (training).

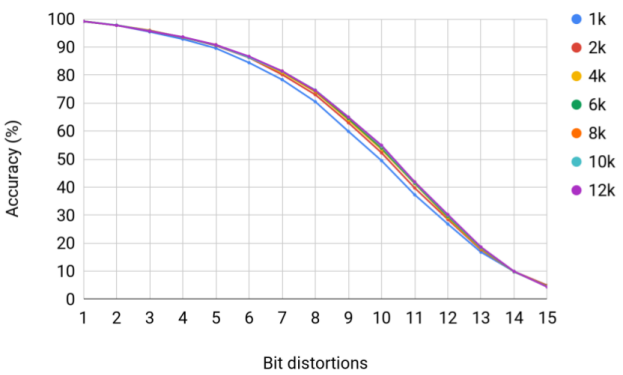


Fig. 4: Scenario 1: One-shot Learning, Varying Distortion

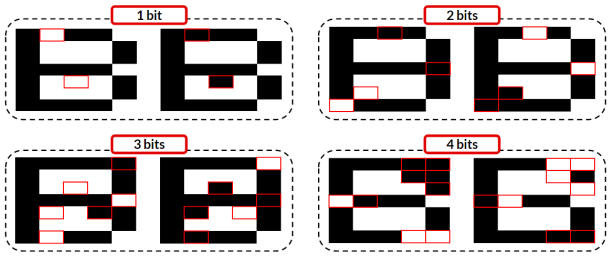


Fig. 5: Illustration of increase in "effective" bit distortion

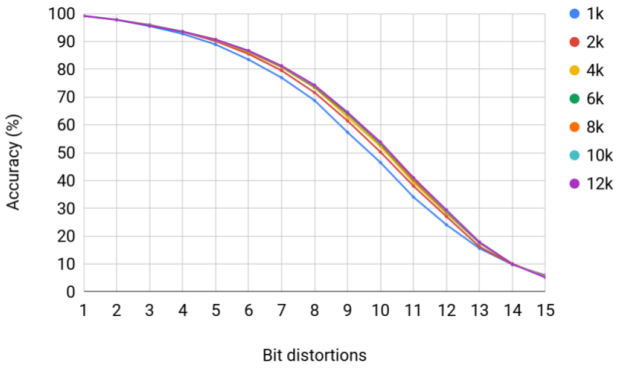


Fig. 6: Scenario 2: Supervised Learning, Varying Distortion

B. Original Scenarios

The results for each of the base tests are presented below. These are performed across varying dimensions to determine the minimum number of bits that would provide satisfactory performance. Cutting down on dimensions is a major step towards optimization.

1) Scenario 1: One-shot Learning, Varying Distortion: Presented in Fig. 4 are the accuracies averaged over all letters for one-shot learning. Each letter is subjected to 1000 tests for each distortion level. A system with $D = 4000$ performs nearly as well as that with $D = 12000$ (maximum dimensions, generally maximum accuracy), as seen in the graph. For each distortion level, the accuracies of $D = 4000$ are within 1.47% of $D = 12000$ (as opposed to $D = 1000$, where the maximum difference is 5.47%).

At low (1-5 bits) and high (13-15 bits) distortion levels, the accuracies across dimensionalities are close to each other. However, the performance of $D = 1000$ starts to deviate from 6-12 bit distortion levels.

2) Scenario 2: Supervised Learning, Varying Distortion: The results for supervised learning and varying distortion are shown in Fig. 6. A training set size of 500 samples are set for each letter, while 1000 patterns are presented for recall. The accuracies of $D = 4000$ are within 1.62% of $D = 12000$ (as opposed to $D = 1000$, where the maximum difference is 7.34%).

The graph follows the trend of the one-shot system, but the accuracies in the supervised system are actually marginally lower. The possibility of an increase in the "effective" number of distortions exists in this mode of learning. In Fig. 5, consider the pairs of letters for each bit distortion. For example, the letters on the left of each pair are part of the training set, and the letters on the right are part of the testing set. In the worst case, these letters will differ by *twice* the number of bit distortions (in red rectangles on the figure), thus decreasing the recall accuracy. This happens when the flipped pixels are mutually exclusive from each other.

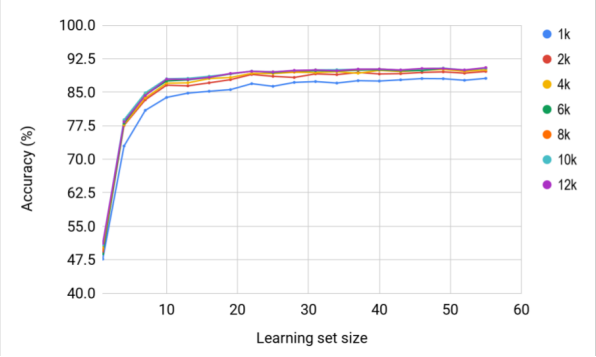
3) Scenario 3: Supervised Learning, Varying Learning Set Size: The graphs for this scenario are presented in Fig. 7. The number of bundles is swept from 1 to 55 presented patterns. To approach the average accuracy from the one-shot learning simulations, a certain number of bundles is required. For a distortion level of 5 bits, the maximum average accuracy of about 90% is reached starting at 15 bundles. For 10-bit distortion, the maximum accuracy only reaches about 50% at 55 bundles, whereas the one-shot average accuracy is at 54%. Results are unreliable at a distortion level of 15 bits (42.86% of pixels flipped).

As seen in the previous graphs, all dimensionalities follow the same trend. In Fig. 7(b), both bundle size and distortion level are varied for $D = 10000$. The training set size at which the system approaches the one-shot accuracy increases as distortion level increases.

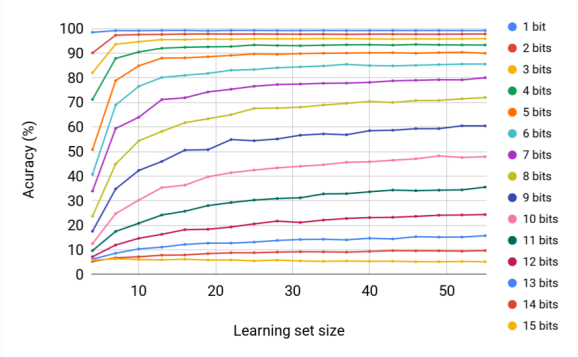
C. MNIST Dataset

Despite the lower accuracy of supervised learning and the overhead of bundling patterns to train the system, it is still necessary especially for applications where there is no *single, correct representation*. An example of this is handwriting recognition. The MNIST database contains 60,000 training images and 10,000 testing images of handwritten digits from 0-9. These images are composed of 28x28 pixels with 256 gray levels.

HoloGN encoding produces orthogonal hypervectors each time, which means that it cannot properly express the relation of continuous values. To verify this, two tests were made: one on unedited images, preserving original gray levels ($\{g \mid 0 <= g <= 255\}$) and another on binarized images ($\{g \mid g = 0$ or $g = 1\}$) which are thresholded at $g = 128$.



(a) 5 bit distortion



(b) 1-15 bit distortions for 10k dimensions

Fig. 7: Scenario 3: Supervised Learning, Varying Learning Set Size.

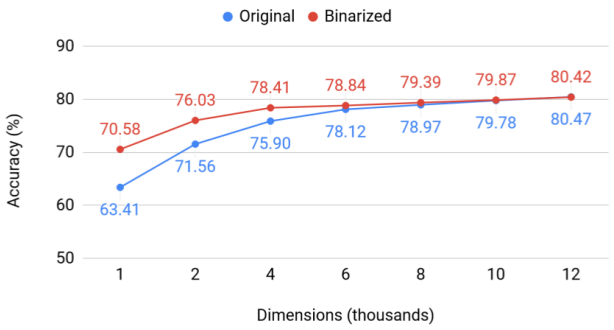


Fig. 8: Recognition accuracy on MNIST original and binarized images.

Fig. 8 shows the accuracy of the system with original and binarized images. The average accuracy over all digits approaches about 80% at higher dimensionalities. Along with the overhead of shifting hypervectors multiple times during encoding, the recognition accuracy of the system tested and trained with original images is lower than that with the binarized images. Given these, the current approach of HoloGN is not that suitable for applications that rely on the relations of values in a continuous range.

V. Conclusion

The characterization done to a character recognition system using HoloGN encoding shows the potential of hyperdimensional computing; highlighting its high scalability, good performance, and robustness to noise. HDC maintains its high accuracy with increasing image size as demonstrated by the MNIST 28x28 B&W image dataset simulations. Robustness to noise is also observed from the varying bit distortion levels on various simulations.

Varying dimensions shows us that it is not necessary to increase the dimensionality further to reach maximum accuracy as the accuracy saturates to a range within +/- 2% difference after a certain dimensionality. This could cut the unnecessary power consumption and area use when implemented on hardware. A 4,000-bit one-shot learning system, for example, yields a comparable performance its 12,000-bit counterpart at 0% distortion, and shows an average accuracy of 89.94% with 14.29% distortion.

VI. Recommendations

This study has presented dimensionality reduction as a primary method of optimization. This allows cutting down on area and power in hardware implementations while maintaining an accuracy that is close to that of higher dimensionalities. However, there are more techniques to further save resources and target a specific metric to improve upon (e.g. area, power, clock cycles, accuracy). Exploring other architectures of encoders and memory instances, along with the use of sparse hyperdimensional computing, on other applications and technologies is recommended.

For a deeper understanding of the characteristics of the produced hypervectors and the classification accuracy, further studies are needed on the mathematical analysis of the algorithm and the hyperspace itself.

References

[1] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey, "High-Dimensional Computing as a Nanoscalable Paradigm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2508–2521, sep 2017.

[2] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic License Plate Recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 1, pp. 42–53, mar 2004.

[3] V. Chandra and R. Aitken, "Impact of Technology and Voltage Scaling on the Soft Error Susceptibility in Nanoscale CMOS," in *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*. IEEE, oct 2008.

[4] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Sekerciogglu, "Holographic Graph Neuron: A Bioinspired Architecture for Pattern Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1250–1262, June 2017.

[5] D. A. Medler, "A Brief History of Connectionism," *Neural Computing Surveys*, 1998.

[6] S. D. Levy and R. Gayler, "Vector Symbolic Architectures: A New Building Material for Artificial General Intelligence," 2008.

[7] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, Jun 2009. [Online]. Available: https://doi.org/10.1007/s12559-009-9009-8

[8] D. A. Norman and D. G. Bobrow, "Descriptions: An Intermediate Stage and in Memory and Retrieval," in *Cognitive Psychology*, no. 11. Academic Press, Inc., 1979, pp. 107–123.