이학박사학위논문

Molecular AI with DNA Computing - Molecular Machine Learning for in vitro Pattern Classification

DNA 컴퓨팅기반 분자 인공지능 - in vitro
패턴인식을 위한 분자수준 기계학습

2019년 12월

서울대학교 대학원
자연과학대학 협동과정 뇌과학 전공
백다솜

Ph.D Dissertation

Molecular AI with DNA Computing - Molecular Machine Learning for in vitro Pattern Classification

DNA 컴퓨팅기반 분자 인공지능 - in vitro 패턴인식을 위한 분자수준 기계학습

Christina Baek

December 2019

Interdisciplinary Program of Brain Science
College of Natural Sciences
Seoul National University

Abstract

Recent research in DNA computing has demonstrated that biological substrates can be used for computing at a molecular level. However, in vitro demonstrations of DNA computations use preprogrammed, rule-based methods which lack the adaptability that may be essential in developing molecular systems that function in dynamic environments. In this dissertation, we introduce an in vitro molecular algorithm that 'learns' molecular models from training data, opening the possibility of 'machine learning' in wet molecular systems. The algorithm enables enzymatic weight update by targeting internal loop structures in DNA and ensemble learning, based on the hypernetwork model. This novel approach allows massively parallel processing of DNA with enzymes for specific structural selection for molecular learning in an iterative manner. This dissertation presents the journey in which an in vitro molecular learning was designed and implemented as my core research project. We describe how the molecular algorithm was designed to correlate to one which could be implemented in a test tube, an intuitive method of DNA data construction, where we dramatically reduce the number of unique DNA sequences needed to cover the large search space of feature sets, computing weight update operations using enzymes and the sequential process by which five iterations of learning were implemented in vitro. By combining DNA computing and machine learning, we anticipate that the proposed research direction in this dissertation makes a step closer to developing molecular computing technologies for future access to more intelligent

molecular systems.

keywords: Molecular Computing, Molecular Machine Learning, DNA Computing, Self-organizing Systems, Pattern Classification

student number: 2014-31418

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 DNA Computing

DNA computing is a fast-developing interdisciplinary field which uses DNA molecules to perform computations rather than traditional silicon chips. DNA is the information media. The processing power of molecular information is used instead of conventional digital components for a revolutionary form of computing. DNA is a biomolecule which has complementary base pairing properties that allow for both specificity in molecular recognition, self-assembly and for massively parallel reactions, which can take place in minute volumes of DNA samples. As a naturally programmable molecule occurring in biology, pioneering work by Adleman first demonstrated the use of DNA hybridization as a parallel computing tool to solve an algorithm, a simple version of the traveling salesman problem (Adleman, 1994). DNA was used to find the most efficient way through seven cities connected by 14 one-way flights us-

ing DNA strands which represented each flight and generating every possible route through the combination of these DNA strands (Adleman, 1998). This is a problem which challenges even a super computer. It is an example of a class of intractable computational problems where the computing time grows exponentially with the problem size, called the NP-complete or non-deterministic polynomial time complete problems. As the common computer uses deterministic sequential algorithms to solve NP problems, the time and space required increases exponentially with the problem size. Adleman consequently proposed a method of using DNA to solve this problem in a non-deterministic manner, where through brute force search using the parallel computing properties of DNA, all possible solutions could be generated almost instantaneously, in a massively parallel manner. Theoretically, billions of DNA molecules perform chemical reactions simultaneously providing immense computing power and storage capacity to DNA as an information media for a new form of computing.

The dissertation will first give an overview of the DNA computing field, discussing the advantages and limitations of DNA computing and present some relevant related works to our research. The motivation and purpose of the study is also clearly outlined. In the following sections, a theoretical overview which encompasses the molecular learning model proposed in my research, thorough experimental design details and results of both the in vitro and in silico experiments are presented. Finally, the dissertation will lead up to the preliminary results of the full five iterations of molecular learning implemented in the laboratory to achieve DNA computing in practice.

1.1.1 The DNA Molecule

DNA or DeoxyriboNucleic Acid, is a polymer, consisting of monomers called deoxyribonucleotides. It is found in every living cell (in vivo) and is the medium to store the genetic information in all living organisms (Weiner & Watson, 1987). The nucleotides which make up the DNA come in four forms, as the deoxyribose and phosphate are constant and they only differ in their bases; adenine (A), thymine (T), guanine (G) and cytosine (C). These nucleotides have specific binding properties, called complementary base pairs, between (A, T) and (G, C). Each base nucleotide base is held together with a hydrogen bond allowing sequences or single-stranded oligonucleotides to form the double stranded DNA molecules (Stryer, 1995).

1.1.2 DNA Computing Model

The main components of a DNA computing model consists of the DNA fragments and enzymes as an input. These inputs undergo controllable biochemical reactions which give a set of output solution DNA fragments (Huang & He, 2011; N. C. Seeman, 2003).

DNA computers can perform computations by manipulating DNA strands reacting in test tubes. The DNA double strands complementary hybridization rule is the cornerstone for DNA computing. Because DNA can bind in a predicable manner, the hybridization of DNA allows a powerful "search" ability, moreover in a massively parallel manner as vast amounts of DNA molecules in a tiny space can hybridize or perform calculations, simultaneously and instantaneously.

The methods of programming DNA computers are common to the techniques molecular biologists use in the lab and in a sense are standard procedures which can be used for the many innovative applications of DNA computing (Steele & Stojkovic, 2004).

DNA molecules can be:

- Synthesized – desired strands can be created
- Separated – strands can be sorted and separated by length
- Merged – pour two test tubes of DNA into one to form union
- Extracted – extract strands that contain a given pattern
- Melted/Annealed – breaking/bonding two DNA molecules with complementary sequences
- Amplified – make copies of DNA strands
- Cut—cut DNA restriction enzymes
- Rejoined – rejoin DNA strands with —sticky ends

1.1.3 Advantages

DNA computing provides an alternative source of computational power, where DNA and molecular biology hardware is used instead of typical silicon based technology. This is possible as the DNA's physical properties are exploited to store information and perform calculations. First, due to the DNA properties of directionality and programmability, exponentially large numbers of different

combinations or sequences of nucleic acids allow immense information density (Greengard, 2017). Furthermore, sequencing DNA oliogmers to suit the design of computing models to be implemented in vitro is very efficient and is a cheap resource. With the development of functional DNAs, such as aptamers and DNAzymes, more effective targeted recognition and signal conversion is becoming achievable in the design of a DNA computer (Heckel, Shomorony, Ramchandran, & David, 2017; Yazdi, Yuan, Ma, Zhao, & Milenkovic, 2015). One key advantage is the enormous parallelism achievable with DNA. Parallel computation can be define as the the different calculations that can be performed simultaneously in time or in space, or temporal and spatial parallelism. The computational ability of DNA can be easily increased according to the initial pool which contains the number of different strands for hybridization to occur.

Consider the travelling salesman problem implemented by Adleman as an example. A conventional computer may use the method of setting up and searching a tree, measuring each complete branch sequentially and preserving the shortest one. If there were 17 cities, approximately 200 trillion paths would have to be calculated. As the number of cities increase, brute force to calculate all the possible paths would overwhelm the best of supercomputers. Furthermore, the memory and time needed to theoretically calculate this would be too much to realistically implement. In the case of 20 cities, 45 million GBytes of memory would be needed, and a 100 MIPS (million instructions per second) computer would take two years to calculate all the possible paths. However with a DNA computer, the data size needed is relative small compared

to molecules present in a nanomole of material and searches can be calculated all in parallel and not sequentially (Kumar, 2015; El-Seoud, Mohamed, & Ghoneimy, 2017)

All living organisms use DNA as its fundamental medium for storing information. It is considered to be the main building block of living things, where DNA absorbs or transmits the data of life over generations for a billion years. Approximately 10 trillion DNA molecules can fit in a size of a marble, and all these molecules available to compute 10 trillion calculations simultaneously (N. C. Seeman, 2003). This amount of storage is estimated to be a computer holding 10 terabytes of data (X. Song & Reif, 2019). A recent study on nucleic acid databases give a thorough account on the capability of DNA as a storage media which outcompetes most conventional digital storage media (X. Song & Reif, 2019).

Conventional digital storage media suffers from material degradation and technology obsolescence resulting in limiting lifespans (Greengard, 2017). DNA offers a new and promising nonvolatile storage medium credit to its long-term durability, immense storage capacity and volumetric density (Zhirnov, Zadegan, Sandhu, Church, & Hughes, 2016). Through development of nanotechnology, it is possible to encode nonbiological information in synthetic oligonucleotides or genome DNA of living organisms. Earliest attempts to create synthetic databases used coding redundancy and sequencing to mitigate errors and recover data from large pools (Church, Gao, & Kosuri, 2012; Goldman et al., 2013). There have been further development to enhance coding capacity (Heckel et al., 2017) and methods of detecting errors for error free data recon-

struction. Now, both in vitro (Yazdi et al., 2015) and in vivo (Farzadfard & Lu, 2018) rewritable data storage is achievable with the use of targeted DNA editing tools. Two examples of include, the storage of a JPEG photograph, a set of Shakespearean sonnets and an audio file of Martin Luther King, Jr's speech "I Have a Dream" in DNA digital data storage (Junnarkar, n.d.) and the recording of a DNA-based in vivo memory system enables dynamic temporal events in the genomes of cells to be used for novel applications such as molecular biosensing to tracing cell lineages (Sheth & Wang, 2018) . Advancements in synthetic DNA databases go hand in hand with the development of DNA computing to support massively parallel database operations (Reif, 1995) and make further progress on revealing the power of molecular near-data processing (Carmean et al., 2018).

DNA computing also maintains an extremely low power consumption rate. It is possible to calculate 2*1019 operations per joule with DNA computers whereas latest supercomputers can only achieve a maximum of 109 operations per joule (Selvam & Suganya, 2014)

Another fundamental advantage of DNA computing is the use of a computing medium which is naturally biocompatible and water-soluble. DNA in all its biological functions are in aqueous electrolyte solutions making which allows for easy accommodation for soluble DNA molecules (Jones, Seeman, & Mirkin, 2015). This benefit of biocompatibility with living tissue or organisms brings closer proximity to implementing molecular computations in vivo for promising biomedical applications.

1.1.4 Limitations

Despite the promising prospect of DNA computing since its first demonstration in 1994 by Adleman, there are ongoing limitations, primarily regarding the in vitro implementation of DNA computing models (Ogihara & Ray, 1997). A key limitation is the dependency on the certainty produced by DNA chemistry, where exact processes must be assumed. DNA synthesis itself is prone to errors (Deaton et al., 1997) and errors such as mismatching during DNA hybridization still infiltrates exact DNA computations to be performed (Parker, 2003). The process of DNA computations can be regarded messy. Furthermore, as much it is an advantage to work with massive data with DNA computing, the probability of error also exponentially increases (Arkin, 2008).

Another stunting limitation of DNA computing would be inevitably, the long time consumption required to perform the laborious experiments usually involved in implementing DNA computations. The amount of the intensive human resources and mechanical intervention to perform DNA operations can take hours to days for the final computational results to be produced.

This contributes to more errors taking place, due to human error or maintenance of experimental protocols for long periods of time. Errors can take place at any step and care must be taken to ensure the quality of materials being used, such as oligonucleotides and enzymes, to importantly reduce chance of contamination (N. Seeman et al., 1998).

The molecular biology techniques used also sometimes create difficulties. The fidelity of the Polymerase Chain Reaction (PCR) for example, in the efficiency and reliability of extraction techniques pose an issue (Kaplan, Cecchi,

& Libchaber, 1999). The molecular biology protocols used may be sufficient for estimating experimental parameters or use in molecular biology applications, however more improvements or adjustments in design my be required to function in DNA computing demonstrations where the precision, reliability and reproduce-ability of computational results is of key importance.

As a result, although countless algorithms have now been developed for DNA computing, there is still a comparative lack of experimental results implementing DNA computing in a laboratory. The issues of error, in what can be called a messy computational environment and that of practicality. The very chemical nature of a DNA computer (N. Seeman et al., 1998) and inevitable dependence and sensitivity to the chemical conditions impose a range of obstacles to overcome to for implementing DNA computing. Thus, the dominating discussion in implementing DNA computing should be that of practical issues and though slow and expensive, progress is being made for the realization of DNA computers in the future.

It is important to note here, that the DNA computers can currently only perform rudimentary calculations. Simple logic-gates which are pre-programmed for solving more and more complex problems. This issue is more thoroughly discussed in Chapter 1.3.

1.2 Related Works

Research exploring and exploiting DNA properties in DNA computing provide the core nanotechnologies required to build DNA devices that are capable of decision-making at a molecular level. Such examples are the implementation

of logic gates (Brown, Lakin, Stefanovic, & Graves, 2014; Mao, LaBean, Reif, & Seeman, 2000; Seelig, Soloveichik, Zhang, & Winfree, 2006), storing and retrieving information (B.-T. Zhang & Jang, 2005; B.-T. Zhang & Kim, 2006), simple computations for differentiating biological information (Benenson et al., 2001), classifying analogue patterns (Yurke, Turberfield, Mills, Simmel, & Neumann, 2000), training molecular automatons (Stojanovic & Stefanovic, 2003) and even playing games (Pei, Matamoros, Liu, Stefanovic, & Stojanovic, 2010). In particular, molecular computing based on enzymes has also attracted much attention as enzymes can respond to a range of small molecule inputs, have advantage in signal amplification, and are highly specific in recognition capabilities (Katz & Privman, 2010).

These molecular computation approaches were generally preprogrammed, rule-based, or used logic gates for simple forms of computations which may not exceed the ability of reflex action from the perspective of intelligence. Such as in the work of (Lim et al., 2010; Mills Jr, Turberfield, Turberfield, Yurke, & Platzman, 2001) where a perceptron algorithm was designed with a weighted sum operation and (L. Qian, Winfree, & Bruck, 2011) where a feedforward and recurrent neural network was constructed with cascading nodes using DNA hybridization; although these studies realized pre-defined perceptrons, the idea of learning, where computational weight parameters were updated to train the model was lacking.

Another state-of-the-art molecular pattern recognition work using the winner-take-all model has been recently published, demonstrating molecular recognition using MNIST digit data and DNA-strand-displacement (Cherry & Qian,

2018). This work recognizes patterns into defined categories of handwritten digits ‘1’ to ‘9’ using a simple neural network model called the winner-take-all model. Though similar to our study, a key difference is that this work focuses on ‘remembering’ patterns during training for recognition and our study focuses on online learning of patterns for classification, where learning refers to the generalization of data following multiple iterations of weight update during molecular learning. Another key difference is the focus of our work to implement a complete in vitro molecular learning experiment, in wet lab conditions. This is further discussed in the results section as a comparative study with our work (Chapter 2.5).

Another related area of research includes the implementation of dynamic reaction networks. in vitro biochemical systems, transcriptional circuits have been used to form complex networks by modifying the regulatory and coding sequence domains of DNA templates (Kim & Winfree, 2011). A combination of switches with inhibitory and excitatory regulation are used as oscillators similar to that which are found as natural oscillators. Another study also use chemical reactions inspired from living organisms to demonstrate assembling of a de novo chemical oscillator, where the wiring of the corresponding network is encoded in a sequence of DNA templates (Montagne, Plasson, Sakai, Fujii, & Rondelez, 2011). These studies use the synthetic systems to further understand the complex chemical reactions found in nature to deepen our understanding of the principle of biological dynamics. A key similarity to our work is the use of modular circuits to model more complex networks. However, it is important to note that these studies are all demonstrated in silico,

although it illustrates the potential of in vitro transcriptional circuitry. Computational tools are also being developed, one example being the EXPonential Amplification Reaction (EXPAR), to facilitate the assay design of isothermal nucleic acid amplification (J. Qian et al., 2012). This method helps accelerate DNA assay design, identifying template performance links to specific sequence motifs.

These dynamic system programming paradigms could be valid approaches to implement machine learning algorithms, as programmable chemical synthesis and the instruction strands of DNA dictate which reaction sequence to perform. We ponder that this kind of powerful information-based DNA system technology could also be manipulated to perform defined reactions in specific orders similar to what our study strives to do, thus, implementation operations in vitro to demonstrate molecular learning with the hypernetwork or other machine learning algorithms (Fu et al., 2018).

Recent work by (Salehi, Liu, Riedel, & Parhi, 2018), implement mathematical functions using DNA strand displacement reactions. This study demonstrates considerably more complex mathematical functions to date, can be designed through chemical reaction networks in a systematic manner. It is similar to our work in that it strives to compute complex functions using DNA though a key difference is that the design and validation of this work were presented in silico whereas our work focuses on in vitro implementation of molecular learning. However, the mass-action simulations of the chemical kinetics of DNA strand displacement reactions may be key in developing in vitro learning implementations, as learning consists of target mathematical operations which

need to be performed with DNA in a systematic manner to manipulate DNA datasets. Consequently, operations or computational constructs are crucial in implementing machine learning algorithms, from simple perceptrons to neural networks, and this is proposed by this system and thus shares our interests in building systemic molecular implementations of chemical reactions for molecular machine learning. Further examples include a study where an architecture of three autocatalytic amplifiers interacts together to perform computations (T. Song, Garg, Mokhtar, Bui, & Reif, 2017). The square root, natural logarithm and exponential functions for x in tunable ranges are computed with DNA circuits.

Finally, as DNA computing has the advantage of biocompatibility with living tissue or organisms, implementing molecular computations has also led to promising biomedical applications. An example of this is the logic gates used to specifically target cells or tissue types, thereby minimizing side effects and releasing chemicals at a specific location (Douglas, Bachelet, & Church, 2012). A DNA 'nanorobot' using a DNA origami box with a logic gate locked lid was used to deliver a specific molecular payload to certain cell types. Further studies showed similar functioning nanorobots in live cockroaches targeted specific cells, with multiple robots containing different logic operations (Amir et al., 2014). Nanostructures have also been used to efficiently deliver cytotoxic drugs to targeted cancerous cells as a therapeutic drug delivery system for various cancers (Chang, Yang, & Huang, 2011; Q. Zhang et al., 2014).

1.3 Problem Setting and Purpose of Research

Molecular learning algorithms for pattern recognition in vitro with DNA molecules may be a step towards more advanced systems with higher complexity, adaptability, robustness, and scalability. This could be useful for solving more advanced problems, and be more applicable to use with more intelligent molecular learning devices in order to function in dynamic in vitro and in vivo environments. Some in vitro and in silico studies which aim to create more complex molecular computing systems include associative recall and supervised learning frameworks using strand-displacement (Chen, Deaton, & Wang, 2003; M. R. Lakin & Stefanovic, 2016; M. Lakin, Minnich, Lane, & Stefanovic, 2012; Lee, Lee, Kim, Deaton, & Zhang, 2011; Lim et al., 2002).

There are many difficulties in implementing molecular learning in wet lab experimental settings. DNA may be a more stable biomolecule compared to others such as RNA, however it still requires storage in appropriate solutions and temperature and it is prone to contamination, manipulation techniques often result in heavily reduced yield, and performing and analyzing the molecular biology results can be tedious and time consuming. Furthermore, applying learning algorithms familiar in machine learning bears critical differences to the current demonstration of DNA computing, such as predefined or rule-based models and logic gates.

Our previous study displayed in vitro DNA classification results (Lee et al., 2017) by retrieving information from a model that was trained with a pseudo-update like operation of increasing the concentration of the matched DNA strands. However, the adaptability and scalability of the model was limited, due

in part to the restrictions in the representation of the model by creating single strand DNA (features) with a fixed variable length. Here, this refers to a fixed length of DNA sequence which encodes the variables. Additionally, from the machine learning perspective, updating only with the positive term has critical limitations not common in conventional machine learning methods (Bishop, 2006), such as in the log-linear models or neural networks, which require both the positive and the negative terms to perform classification. The accuracy was also somewhat guaranteed because the training and test set were not divided and the features (pool of DNA) were manually designed to have small errors.

In this dissertation, we introduce a self-improving molecular machine learning algorithm, the hypernetwork (Lee et al., 2011; B.-T. Zhang, 2008; B.-T. Zhang & Kim, 2006), and a novel experimental scheme to implement in vitro molecular machine learning with enzymatic weight update. We present the preliminary in vitro experimental results of proof-of-concept experiments for constructing two types of DNA datasets, the training and test data, with the self-assembling processes of DNA with hybridization and ligation, and DNA cleavage with a nuclease enzyme for the weight update stage of the molecular learning. Our study provides a new method of implementing molecular machine learning with weight update including a negative term.

First, we consider a natural experimental situation typical to machine learning, where we separate the training and test data to evaluate the model. Secondly, by adopting a high order feature extraction method when creating single stranded DNA, a higher-order hypothesis space may be explored which allows for discovery of better solutions even with simple linear classifiers. Sec-

ondly, by adopting a high order feature extraction method when creating single stranded DNA, a higher-order hypothesis space may be explored which allows for discovery of better solutions even with simple linear classifiers. Thirdly, unlike previous methods which only increased the weight of the model, our proposed method considers both the positive and negative terms of the weight update in the model for learning using an enzymatic weight update operation in vitro. This method is inspired by the energy-based models which use the energy-based objective function to solve problems, and is represented with exponentially proportional probabilities and consists of positive and negative terms to calculate the gradient (Zhou, Sohn, & Lee, 2012). Lastly, by encompassing the concept of ensemble learning, the model uses its full feature space for the classification task over five iterations of weight update and learning and also guarantees best performance by voting the best classified labels between each ensemble model. These four aspects are the crucial differences that distinguish the research in this dissertation from previous demonstrations of molecular learning, where without these assumptions of machine learning based aspects, learning, adaptability, and scalability of the model is limited. We show in the results section that the performance of our model gradually increases with the continual addition of training data.

Please note, the work presented in Chapters 1 to 4 is published in my journal in Molecules (Baek, Lee, Lee, Kwak, & Zhang, 2019).

Chapter 2

The DNA Hypernetwork Model

2.1 Theoretical Overview

Biological brains are excellent at online learning and associative recall from multimodal stream data, but no machine learning algorithm approaches this capability in the field of DNA computing. The goal here was to develop a brain-like, highly-flexible learning architecture that self-organize complex memory structures incrementally and dynamically using DNA as its media source. For this, we simplified and adapted idea of machine learning to suit implementation of in vitro molecular learning, which use massively-associative molecular assembly computing to simulate the DNA hypernetworks of sparse population.

2.2 A Molecular Learning Model - The Hypernetwork

The hypernetwork is a graphical model with nodes and connections between two or more nodes called hyperedges (Figures 2.1 and 2.2) (Lee et al., 2011; B.-T. Zhang, 2008). The connections between these nodes are strengthened or weakened through the process of weight update or error correction during learning (B.-T. Zhang, 2008). We use the term 'hypernetwork' as we refer to hypergraphs which is a generalization of a graph where an edge can join any number of vertices. The hypergraph generally contains nodes and vertices and is a set of non-empty subsets termed hyperedges.

This model was inspired by the idea of in vitro evolution, and provides a clear framework for molecular computing to be realized for molecular learning in a test tube. The probability of hyperedges, or weights, are represented by the concentration of DNA species in the tube. In addition, the idea of weight update is implemented here with specific enzymes, which use the gradient descent in a natural way for autonomous weight update or error correction. Gradient descent is an optimizing procedure commonly used in many machine algorithms to calculate the derivative from the training data before calculating update. All of these reactions occur in a massively parallel manner. Related models have also been previously discussed from the constructive machine learning perspective (Heo, Lee, Lee, & Zhang, 2013; Sakellariou, Tria, Loreto, & Pachet, 2015).

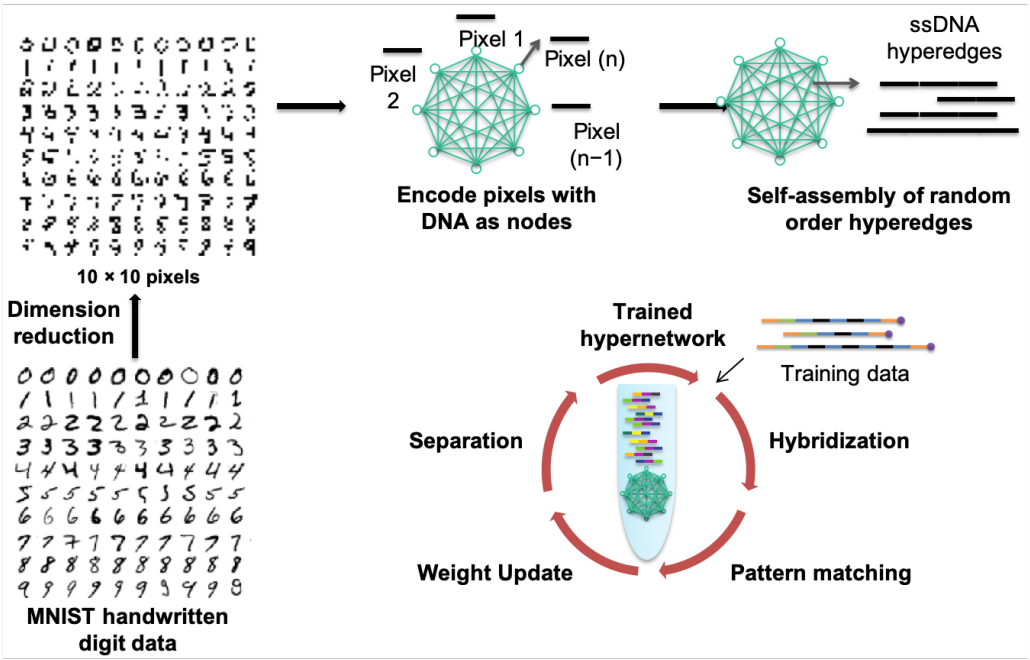


Figure 2.1: Hypernetwork with nodes and hyperedges with a conceptual overview of molecular machine learning with DNA processes. Each node represents a pixel which is encoded to a unique DNA sequence. These pixel DNA are self-assembled to form random order hyperedges.

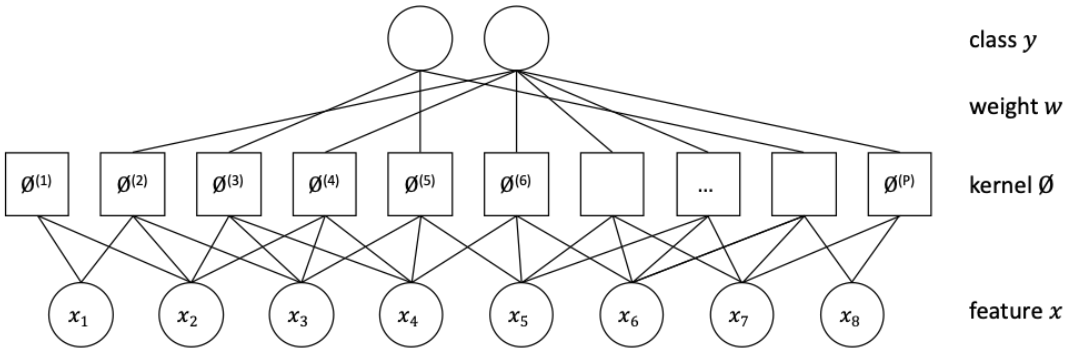


Figure 2.2: Graphical representation of the hypernetwork model. The graphical structure of the hypernetwork model. This is the factor graph representation of the model in Figure 2.1. Note, one kernel corresponds to one factor.

2.3 Theoretical Definitions

In this paper, the hypernetwork is interpreted as a maximum entropy classifier with an exponential number of hyperedges as input (Nigam, Lafferty, & McCallum, 1999).

$$p(y|x;w) = \frac{1}{Z}\exp(\sum_i w_i \phi^{(i)}(x,y)) \tag{2.1}$$

$$\phi^{(i)}(x,y) = \delta(y,\tilde{y}^{(i)}) \prod_{j\in C^{(i)}} \delta(x_j,\tilde{x}_j^{(i)}), \tag{2.2}$$

where *Z* is the normalization term, weight $w^{(i)}$ is the parameter corresponding to $\phi^{(i)}$, and $C^{(i)}$ is the set of indices of input features corresponding to ith hyperedge. The ith hyperedge consists of the set of input variables $\{\tilde{x}_j^{(i)}\}_{j\in C^{(i)}}$, the output variable $\tilde{y}^{(i)}$, and weight $w^{(i)}$. δ is the identity function. If the whole predefined variables of the ith hyperedge are matched to the corresponding variables of an instance, $\phi^{(i)}$ becomes 1 (Figure 2.2).

2.4 In Vitro Algorithm Design

Our DNA dynamics can be described from the machine learning perspective as presented in Algorithm 1 in Figure 2.3. Please note, M refers to the number of epochs and N, to the total number of images observed. Line 2 is essentially an indexing or numbering system to define for example, the first epoch m_1 to consist of images 1 - 10 and so on.

The DNA processes in the paper and Algorithm 1 to the molecular experimental scheme (Figure 2.4) is matched as following ways:

1. Initializing hyperedge in each epoch corresponds to line 4
2. Figure 2.4 hybridization corresponds to line 7–8
3. Figure 2.4 positive and negative update with addition and nuclease, respectively corresponds to line 9
4. Merging hyperedges in each epoch corresponds to line 10

In the in vitro implementation of Algorithm 1, the updating of calculated positive or negative term occurs in a slightly different order. In the case of negative weight update, the nuclease cleaves the perfectly matched DNA strands, which occur from the hybridization of complementary DNA hyperedges from training data for '6' and '7', the hybridization being when the negative weight term is calculated. In the case of the positive weight update, the resulting DNA concentrations of each hyperedge from cleavage and purification is amplified, where the positive term was also calculated from the initial hybridization process.

The hybridization rate of DNA datasets to a) construct hyperedges, theta being the hyperdges made from the data and b) to calculate the positive and negative term of Equation 3.1, is much faster due to the massively parallel nature of DNA computing, compared to the sequential matching of data in silico (Chapter 2.5). DNA data representation through the use of sticky ends and ligation enzyme is almost instantaneous too, due to the use of common complementary strands used to ligate single variable DNA to form free-order hyperedges. This step approximates the kernel function in Equation 2.1. The weight of hyperedges in silico is approximated by the relative concentrations of DNA

hyperedges, the relative weights of DNA hyperedges being the probabilistic weight calculated in silico, and the updating of weights in Equation 3.1 occurs through the addition of DNA and S1 nuclease enzyme cleavage of DNA, thereby increasing and decreasing the concentration of best matched DNA hyperedges respectively (Chapter 2.4).

The hypernetwork is a suitable model employing molecular machine learning for the following reasons. First, it is a non-linear classifier, which can search the exponential search space of the combination of hyperedges (B.-T. Zhang, 2008; B.-T. Zhang, Ha, & Kang, 2012), unlike the maximum entropy classifier, which has a single variable as input and is a linear classifier.

Secondly, the hypernetwork can be relatively easily implemented in DNA computing as the model utilizes constructive DNA properties such as self assembly and molecular recognition for the generation of hyperedges, and to perform learning operations (Lee et al., 2017). Massively-parallel processes can also be exploited with DNA, which means that the search of a large search space is much faster and applicable to the experimental setting.

Algorithm 1 Online Learning of Hypernetworks

Input stream: $(X, Y) = \{(X^{(1)}, y^{(1)}), \cdots, (X^{(M)}, Y^{(M)})\}$

$(X^{(m)}, y^{(m)}) = \{(x^{(((m-1)\cdot N/M+1):(m\cdot N/M))}, y^{(((m-1)\cdot N/M+1):(m\cdot N/M))})\}$

Initial feature set and weights: $(\Phi, w) \leftarrow$ empty

for $m = 1 : M$ **do**

  $(\Phi', w') \leftarrow sample(X^{(m)}, Y^{(m)})$

  $(\Phi, w) \leftarrow (\Phi, w) \cup (\Phi', w)$

  $\Delta w_{positive} \leftarrow positiveTerm(x^{(m)}, y^{(m)}, \Phi, w)$

  $\Delta w_{negative} \leftarrow negativeTerm(x^{(m)}, y^{(m)}, \Phi, w)$

  $w \leftarrow w + \eta(m) \cdot (\Delta w_{positive} - \Delta w_{negative})$

  $(\Phi, w) \leftarrow prune(\Phi, w)$

end for

Figure 2.3: Algorithm of online learning of hypernetworks.

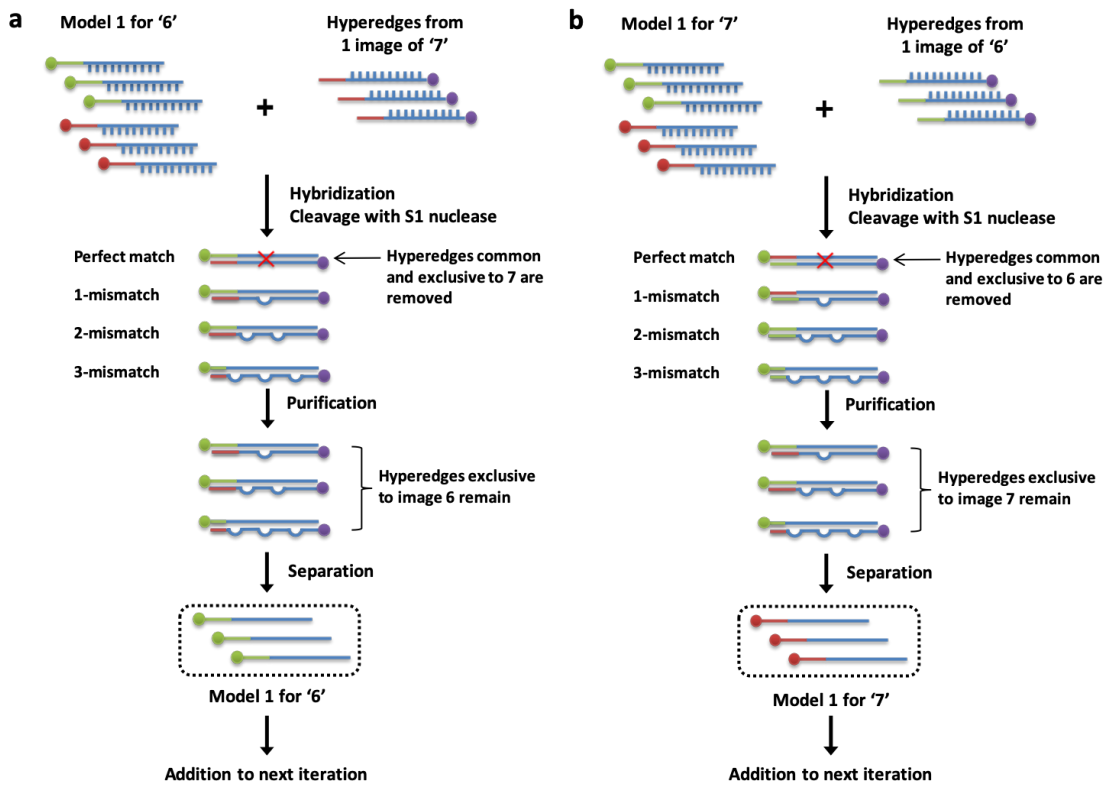


Figure 2.4: Overview of experimental scheme implementing molecular learning of hypernetworks. a. Experimental steps for training the image '6'. b. Experimental steps for training the image '7'.

2.5 In Silico Experiment of DNA Hypernetwork

As described in the Materials and Methods section, we used the MNIST dataset to measure the classifier accuracy (Sammut & Webb, 2011), which is defined as the estimation of number of correctly classified objects over the total number of objects.

To verify the learning capability of our proposed model, both incremental and online aspects, we compared our model to two existing models; the perceptron model described in (Cherry & Qian, 2018) and the conventional neural network (LeCun, Bottou, Bengio, Haffner, et al., 1998) as a representative example of non-linear classification.

In (Cherry & Qian, 2018), a basic perceptron model outputs the weighted sum for each class and selects the maximum value as their winning final output. 2-class classification between digits '6' and '7' is demonstrated and nine label 3 grouped class-classifier is described, where all methods first eliminate the outlier and the performance achieved by providing probabilistically calculated weights of the 10 most characteristic features to the designer as a prerequisite. However, in our study, we do not eliminate outliers or give prior weights and use the MNIST dataset as it is for our performance. We exploit the learning ability of a DNA computing model without the need for the designer to previously define weights. Not only do we reduce the labor required by the designer to define weights for selected features but we exploit the massively parallel processing capability of DNA computing whilst demonstrating molecular learning which improves performance with experience as our model is designed for implementation in vitro through molecular biology technique with wet DNA.

Two types of initialization of weights are introduced in our simulation results, 0 weight initialization which is easily implemented in DNA experiments, and random weight initialization which is harder to be conducted in vitro but is more conventionally used in perceptron and neural network models. The perceptron and neural networks convergence of performance are dependent on their initialized weights (Nguyen & Widrow, 1990). We conducted these two methods of initializing the weights, first starting with 0 weight, and second providing random values to the weights.

For the 2-class classification, 1127 and 11,184 images were used for the test and training data respectively. For the 10-class classification 10,000 and 60,000 images were used for the test and training data respectively. As the MNIST dataset is balanced over all classes and not skewed to any class, the accuracy measurement is sufficient to evaluate the classifier's performance (Powers, 2011). For all cases of learning, we randomly sampled five images. We did this to demonstrate our model's capacity to implement online learning in only a few iterations and more significantly for our work, for the correlation to our wet lab molecular learning protocol, where only five iterations of molecular learning experiments need to be carried out for learning to produce classification performance.

For both the perceptron and neural network model, the learning rate was set to 0.1. For the perceptron model, of the 10 output values from the given input, the output with the biggest value was selected in a winner-take-all method. As the hyperedges produced from the hypernetwork in the 10-class classification (all with weights) was 284, we chose to use 300 hidden nodes for the neu-

ral network. All the source codes and relevant results can be found on github repository (https://github.com/drafity/dnaHN).

Figure 2.5 shows the results of our in silico classification results. As the number of epochs increased, the test accuracy of the hypernetwork also increased. The accuracy of the hypernetwork was also higher than the comparative models of the perceptron and neural network. We note here the significance of having used an accuracy measure to evaluate our classifier. The DNA learning models implemented in vitro in the mentioned related works often lacked appropriate measures to evaluate the classifier's performance. Furthermore, though recognition abilities have been reported through in vitro molecular learning, classification of data through learning and testing this, to consequently group or label the unknown test data to a category by a molecular learning model is to the authors' knowledge a novelty in itself.

A key feature of our model in comparison to the perceptron or neural network models is the minimum number of iterations required to observe significant performance. Our proposed model only needs five iterations of learning to achieve significant classification performance. However, as the results show in Figures 2.5 and 2.6, initializing the weight to 0 or giving random weights to the compared models still resulted in low accuracy in small epoch sizes. The perceptron and the neural network require a much larger epoch size for significant classification performance to be achieved.

This is crucial as in vitro experiments to perform molecular learning not only require time-consuming laborious work, but issues with contamination and denaturation can affect the quality of the experimental results. It is only

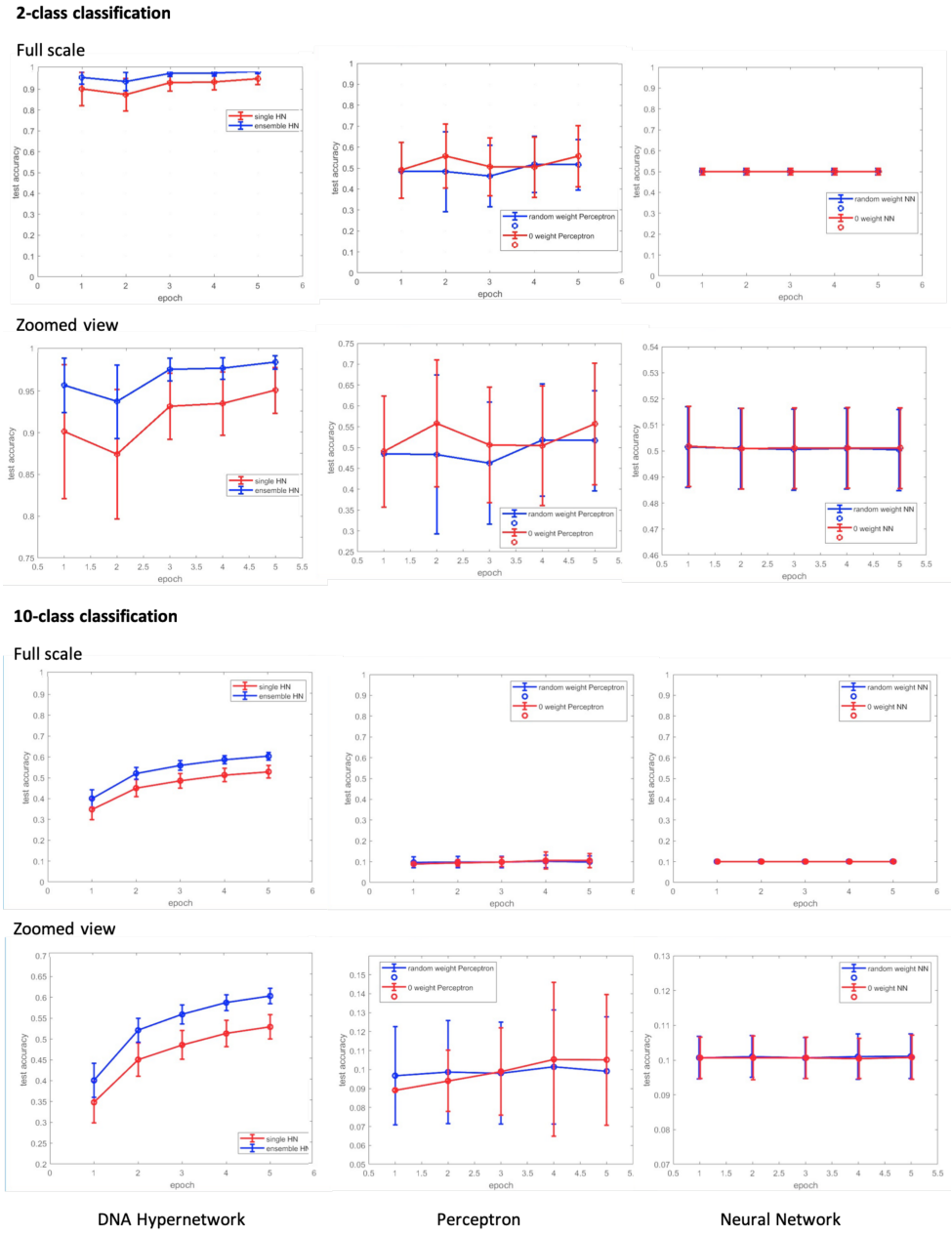


Figure 2.5: In silico experimental results. Computer simulation of 2-class and 10-class classification with the hypernetwork.

more suitable for molecular learning experiments performed in wet lab conditions to be efficient, exploiting the massively parallel computing possible with DNA but also minimizing the protocol required to perform molecular learning. Our model is designed to do this by autonomously constructing higher order representations, using massively parallel DNA processes to create and update weights in minimal iterations. Furthermore, compared to state-of-the-art studies in molecular recognition, we were able to achieve over 90% accuracy and 60% accuracy in 2-class and 10-class classification respectively, through a molecular learning algorithm in five iterations. Thus, this result present that our model is a novel molecular learning model which learns in an online manner through minimal iterations of learning, suitable for wet lab implementations using DNA.

The hypernetwork, inspired by DNA chemical reactions, when computed in silico, clearly showed the disadvantage of sequential computing in silico and the massively parallel processing advantage of DNA computing in vitro. In an instant, DNA molecules hybridize when complementary strands are added together in an appropriate buffer and thus almost immediately the computing in that tube comes to an end. However, implementation of the hypernetwork in silico is iterative, sequential. For each training and test data, the number of matches and mismatches need to be calculated sequentially, and as the order of hyperedges increases, computational time complexity increase exponentially. As a result, with our computing power, empirically, 1000 iterations require 1000×20 min, a total of approximately 10 days to complete. Therefore it is important to note that there is a sheer advantage in DNA implementation of the

hypernetwork compared to in silico. For the same reason, the neural network requires around 1000 iterations to converge and in the case of non-linearly separable data when using the perceptron model, it fails to converge. Thus, the proposed hypernetwork may also be introducing the possibility of a new computing method.

We also discuss the reasons why the perceptron does not perform as well. Due to the nature of perceptron models, as a representative linear classifier, it is difficult for it to solve linearly inseparable problems (XOR problems) without any preprocessing or adding layers to the perceptron to deal with non-linearity problems (Rosenblatt, 1957; Russell & Norvig, 2016). As illustrated in Figure 2.5, the perceptron model shows performances close to what would be achieved from random picking for both small and large number of iterations. Depending on how the data is fed, 2-class classification performance levels show major fluctuations, where up to 80% performance is achieved at times and others much lower performance. This phenomenon is typically representative of unsuccessful generalization of the data also called overfitting. For example, in the case of the perceptron, as described in the reference, the performance is achieved only for the data that can be fitted linearly. To learn linearly inseparable data, the model needs a feature reduction or extraction preprocessing methods (Liu, Nakashima, Sako, & Fujisawa, 2003) or a nonlinear kernel to model (e.g., Support Vector Machine (Scholkopf & Smola, 2001), Neural Network (Russell & Norvig, 2016) the high dimension input dataset. As this paper focuses on the implementation of a learning model in vitro only using easy, basic and fundamental learning processes, we believe this is out of the scope

of our paper and omit further discussion.

As a support to such arguments, as shown in Figures 2.5 and 2.6, both in our results and in Cherry and Qian's work, there are cases where a variety of elimination conditions and previously providing the optimal weights of batch data by the designer can achieve significant performance in 2-class classification i.e., overfitted results (the maximum value of the error bars). However, as in the case of 10-class classification tasks, where the data is not linearly separable where it exceeds the model's capacity, the range of performance levels are smaller and, as acknowledged by Cherry and Qian in their paper, it is difficult for the designer to find the optimal weights for the model.

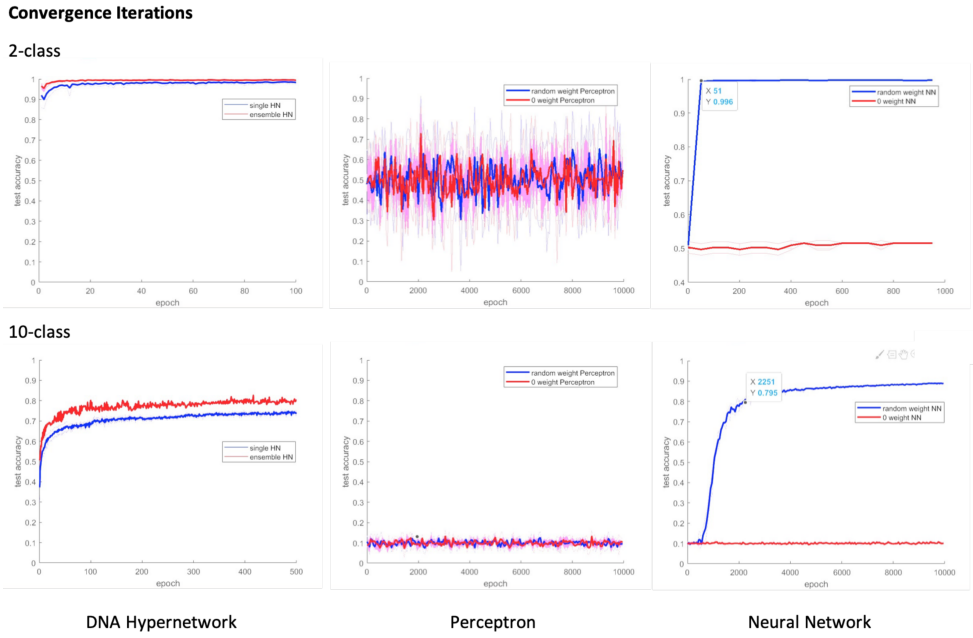


Figure 2.6: In silico experimental results. Computer simulation of 10-class classification with the hypernetwork.

Chapter 3

Experimental Design

3.1 DNA Dataset Construction

As an example of pattern classification in a test tube, we use the handwritten digit images from the MNIST database (LeCun, 1998), which is commonly used to test machine learning algorithms (Deng, 2012). In our case this is used to test a two-class classification problem with digits '6' and '7' (Figure 3.1). The dimensions of the digit images are reduced to 10×10 which are then used as the input data.

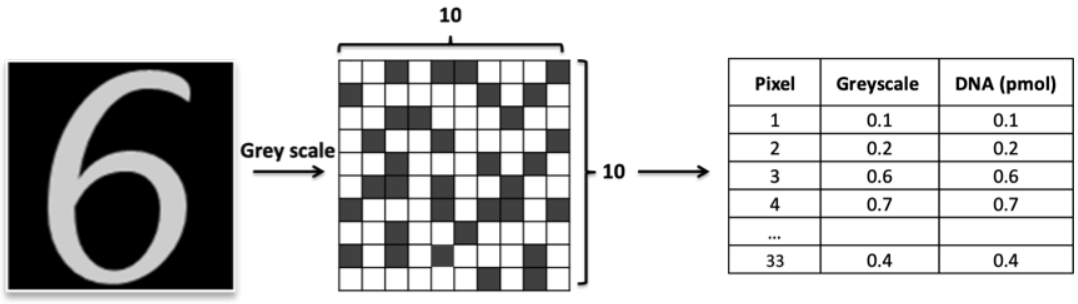


Figure 3.1: Encoding of image data to unique DNA sequences. Each greyscale value of the image corresponds to the amount of DNA added to each tube for DNA representation of the full image.

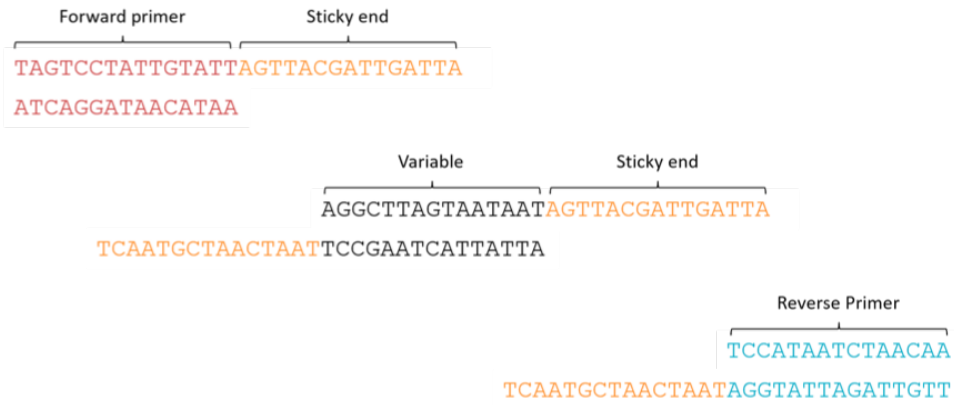
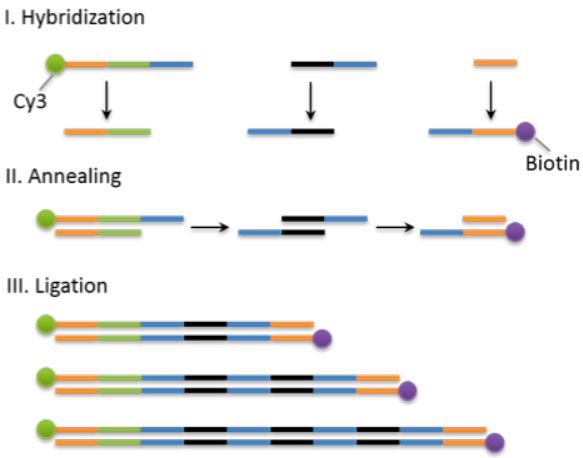


Figure 3.2: Complementary DNA sequences of primers, sticky ends and variable DNA that are ligated to form hyperedges.



Encoded DNA	Sequence (3′ to 5′)
Forward primer (FP)	TAGTCCTATTGTATT
Sticky end (T)	AGTTACGATTGATTA
Reverser primer (RP)	TCCATAATCTAACAA
Pixel 1	AGGCTTAGTAATAAT
Pixel 2	AGAATGCTATGTAAT
Pixel 3	ATCGCAGTTATTATT

Figure 3.3: Self-assembling of free-order hyperedges through three processes, hybridization, annealing, and ligation. Key: Primers (orange), class label (green), tag or sticky end (blue), pixel (black). Please note, the pixel DNA colored black represent unique DNA sequences of various pixels, but for simplicity have been colored black to group them as variable DNA. Table shows DNA sequences for encoded pixels, primers and sticky ends. The final double strands in the sample are separated to single stranded DNA for use as the random DNA library set.

From each image 33 pixels are randomly selected in a non-replacement manner to form a hypernetwork model. 33 pixels were used as we wanted to use the least amount of pixels to represent the largest search space of 10 by 10 pixel images. In other words, only 33 pixels were used in each ensemble of each iteration. So 33 unique DNA sequences could be used to represent 33×3, so 99 pixels in total for each ensemble of learning. In our experiment, we produce three ensembles for each image (Figure 3.5). The 33 unique pixels from each ensemble are encoded to DNA by allocating 33 unique DNA sequences consisting of 15 base pairs each. Unique DNA oligomers are sequenced with an exhaustive DNA sequence design algorithm, EGNAS (Kick, Bönsch, & Mertig, 2012). EGNAS, stands for Exhaustive Generation of Nucleic Acid Sequence, and is a software tool used to control both interstrand and intrastrand properties of DNA to generate sets with maximum number of sequence designs with defined properties such as the guanine-cytosine content. This tool is available online for noncommercial use at http://www.chm.tu-dresden.de/pc6/EGNAS. Once each DNA oligomer is assigned to a pixel, that is, one unique DNA to each pixel (Figure 3.2), it is the grey scale value (between 0 and 1) for each pixel that determines the amount of DNA to be added to make the DNA dataset (Figure 3.1). Each class is labeled with a different fluorescent protein to allow visualization of classes, allowing for the learning of two classes in one tube.

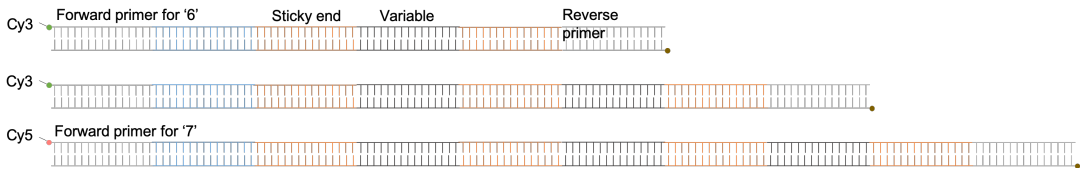


Figure 3.4: Construction of a DNA database using self-assembling DNA. DNA data features are represented as free-order hyperedges. Example of a 1, 2 and 3-order hyperedge shows the number of variables presented within the complete DNA structure.

Following the addition of relative amounts of DNA oligomers according to the pixel value, the sequences are joined together to produce free-order hyperedges for the initialized hypernetwork, and training and validation datasets (Figure 3.4). Here, free-order refers to any number of linked variables in the DNA sequence, for example, 1-order hyperedge consists of one variable, 2-order hyperedge with two variables and 3-order hyperedge with three variables and so on. Using PCR, the variable DNA, in this instance the pixel, the forward and reverse primers are hybridized to their respective complementary strands to form double stranded DNA. These three units act as building blocks for constructing free-order hyperedges as they are annealed at the tag or sticky end regions with enzyme ligase. It is worth noting that free-order hyperedges enhance the robustness of the model, and it is not only the variables that are learned through the self-organizing hypernetwork, but also the order of hyperedges.

3.2 Learning with Enzymatic Weight Update

Our main idea is that the dual hybridization-separation process with enzymatic weight update can be interpreted as an approximation of the stochastic gradient descent of hypernetworks.

In the test tube, enzymatic weight update is realized with enzymes which target specific DNA structures. Molecular recognition through hybridization of complementary base pairs allows matching of data to form symmetrical internal loops if incorrectly matched. Symmetric internal loops of DNA (Peritz, Kierzek, Sugimoto, & Turner, 1991; Zacharias & Hagerman, 1996; Zeng &

Zocchi, 2006) are used to correlate the differences in training instances. This physical DNA structure is used to determine the degree of matching between two complementary strands for pattern matching. It is these DNA structures which are cleaved by specially chosen enzymes to perform the enzymatic weight update stage of the learning process. Consequently, the cleaving results in decrease in concentration of DNA with symmetric internal loops in the test tube.

First, the training data for digits '6' and '7' is hybridized with the training data for '7' (Figure 2.4a). Then, S1 nuclease, an enzyme which cleaves the perfectly matched DNA sequences (completely hybridized hyperedges) is added. This allows for the selection of only the perfectly matched hyperedges, leaving any degree of mismatched hyperedges in the tube. This is the enzymatic weight update operation that is demonstrated in vitro. DNA is then purified, separated using biotin, and amplified with PCR resulting in a mixture of single-stranded DNA hyperedges exclusive to '6'. This is repeated for the training data for '6' to train '7' (Figure 2.4b).

With the enzymatic weight update, through the decrease weight function of S1 nuclease, we eliminate the hyperedges common to both '6' and '7' resulting in only exclusive hyperedges characterizing '6' and '7' for successful digit classification. This corresponds to the negative weight update idea. The discrepancy between the two classes of digit data is represented by the remaining pool of DNA sequences, which are then added to one tube. The addition of the remaining sequences symbolizes the positive weight function. The trained hypernetwork from mini-batch 1 is added to the next minibatch and so on. Here the concept of online learning is applied. The weak learner 1, 2, 3 for

ensembles 1, 2, and 3 respectively is added at each iteration to form the final trained weak learner after mini-batch 5 (Figure 3.5). The ensemble method is discussed in the next section. Repetition of these learning steps is predicted to construct an ensemble of three molecular classifiers which can be used for ensemble prediction in the test stage by measuring the final ratio of fluorescence for Cy3 (Label for 6) and Cy5 (Label for 7). To perform online learning, after the model is trained with each mini-batch, the DNA pool is combined to create a final hypernetwork model for given classification tasks.

The above process can be described theoretically, where the set of hyperedges is determined or the connections between the nodes are strengthened or weakened through the process of weight update or error correction during learning. Equation 3.1 is the gradient of the log-likelihood of Equation 2.1.

$$\Delta w_i = \frac{1}{N}\sum_{n=1}^{N}\phi^{(i)}(x^{(n)}, y^{(n)})\left[1 - \frac{1}{Z}\exp(\sum_{i'} w_{i'}\phi^{(i')}(x^{(n)}, y^{(n)}))\right] \quad (3.1)$$

The next step of the algorithm consists of pattern matching and weight update (Algorithm 1, line 7, 8). Equation 3.1 shows the gradient of the log-likelihood of Equation 2.1. Our algorithm illustrates our learning process of hypernetworks, which can be naturally applied to online learning. The algorithm consists of both parameter learning and structure learning. In parameter learning, Equation 3.1 is used for stochastic gradient descent. Equation 3.1 consists of the positive term $\phi^{(i)}(x, y)$ and the negative term $-\frac{1}{Z} \cdot \phi^{(i)}(x, y) \cdot exp(\sum_{i'} w_{i'}\phi^{(i')}(x, y))$.

Without the terms of matching instance $\phi^{(i)}(x, y)$, the positive term is 1 and the negative term is $\frac{1}{Z} \cdot \phi^{(i)}(x, y) \cdot exp(\sum_{i'} w_{i'}\phi^{(i')}(x, y))$. In structure learning, the feature set of hyperedges Φ is updated. The number of possible kernel

functions $\phi^{(i)}$ is exponential to the order of the hyperedge. This is required to select the subset of hyperedges, which consist of separable patterns. The candidate hyperedges are sampled from the data instance, where values of the partial input of an instance are used as the features. The hyperedges which are not important are pruned. Large absolute weight values or non-negative weight values can be used as the measure of importance; we use the latter case.

3.3 Ensemble Learning of Hypernetworks

Figure 3.5 shows that three ensembles were used to train images '6' and '7' in an online manner. We apply the ensemble method to our model for the following reasons. First, the ensemble method guarantees maximum performance within the ensemble models. When different types of models are being ensembled together, the overall performance increases as each model's characteristics of representation and search spaces are different from one another Oza, 2005. Another reason for creating a three ensemble model is due to the limitation of interpretability. Since our designed model created free-ordered hyperedges, using 100 pixel produces 100! (factorial of 100) different hyperedges which is almost impossible to visualize with current electrophoresis techniques. Moreover, the formation of the hyperedges is unpredictable since it is affected by a range of external stimuli (temperature, time, concentration etc.). Therefore, we divide the image into three sets and perform the voting method with the final produced results by each of three ensemble models.

The inference procedure is almost the same as the learning process. However, before the S1 nuclease is applied, the concentration of the perfectly matched

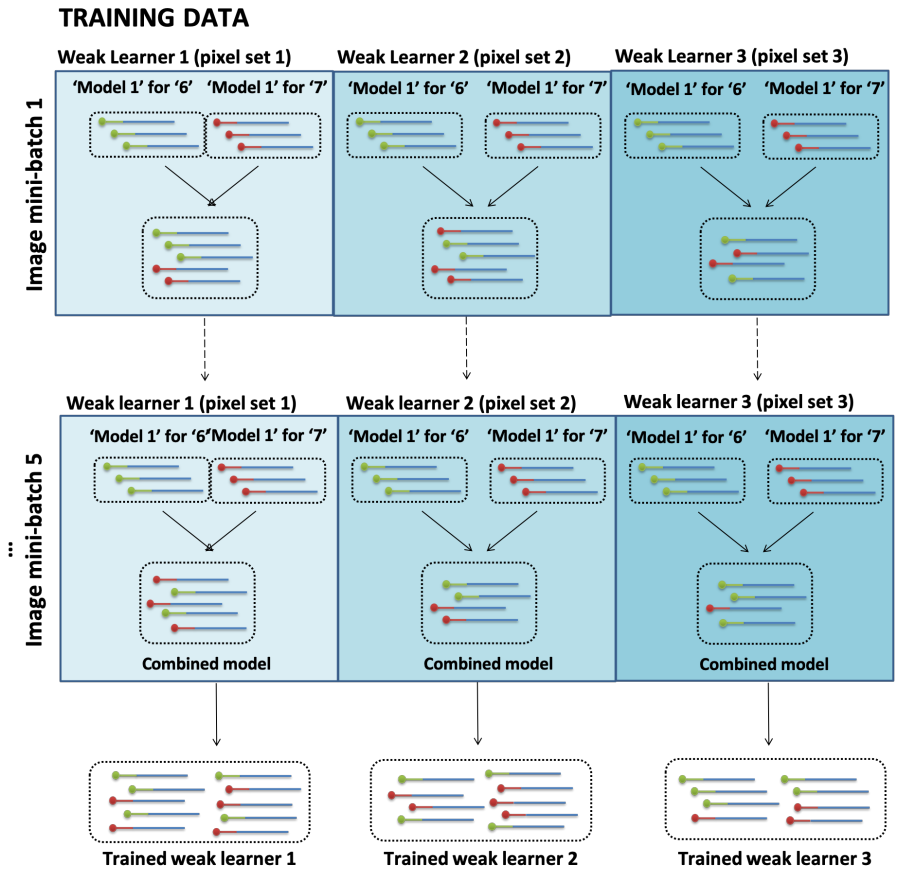


Figure 3.5: Ensemble of hypernetworks. Three ensembles of both trained models for digit images '6' and '7' are added in an online manner (combined model). The ensembles are added together for ensemble prediction in digit classification in the test stage.

DNA is measured to decide whether the test data is a digit ‘6’ or ‘7’.

Chapter 4

In Vitro Experiment Validation of DNA Hypernetwork

To ensure that the experimental protocol is implemented to the highest degree of efficiency and accuracy, a series of preliminary experiments were undertaken to validate the experimental steps involved in demonstrating the molecular hypernetwork.

4.1 DNA Quantification of 3-Order Hyperedges

The formation of a random single-stranded library was critical in verifying the success of the full experimental scheme. The experimental steps and results are as follows:

1. Hybridization of upper and bottom strands of variable units.

2. Ligation of these variables in a random fashion, all in one tube to create a

double-stranded DNA random library.

3. Purification of the sample from ligase.
4. Separation of the double-stranded library to a single-stranded random library using Streptavidin and Biotin.
5. Centrifugal filtering of DNA for concentration.
6. Verification of library formation with the use of complementary strands.

Each step listed above was carried out using wet DNA in a test tube, and at each step, the DNA concentration was measured using a NanoDrop Nucleic Acid Quantification machine, which is a common lab spectrophotometer (Figure 4.1). Hybridization was performed on the PCR machine with a decrement of 2° from 95° to 10° 100 pmol of each upper and lower strand was used.

Ligation was carried out using Thermo Fisher's T4 ligase enzyme. This enzyme joins DNA fragments with staggered end and blunt ends and repairs nicks in double stranded DNA with 3'-OH and 5'-P ends. Three units of T4 ligase was used with 1 l of ligation buffer.

Annealing of the ligated DNA strands are then put into PCR conditions with a decrement of 1 degree from 30 to 4 degrees. Purification and extraction of DNA from the ligase inclusive sample was carried out using the QIAEX II protocol for desalting and concentrating DNA solutions. The standard procedures for this were used (Hamaguchi & Geiduschek, 1962; Sambrook, Fritsch, Maniatis, et al., 1989; Vogelstein & Gillespie, 1979). This procedure is commonly used to remove impurities (phosphatases, endonucleases, dNTPs etc.)

from DNA, and to concentrate DNA. The QIAEX desalting and concentration protocol gives quite a detailed description of the procedure.

While there was a significant loss of DNA content after ligation, a sufficient concentration was recovered from the centrifugal filtering step allowing identification of the nine complementary strands possible from the combinations of the three different variables initially used. Bands at the 70 bp marker were present for all nine types of sequences which confirm that all possible sequences were successfully constructed and retrieved during the experimental process. The results shown in Figure 4.1 present DNA concentrations at various stages of the learning process, and the final confirmation of the success in making a random double-stranded library.

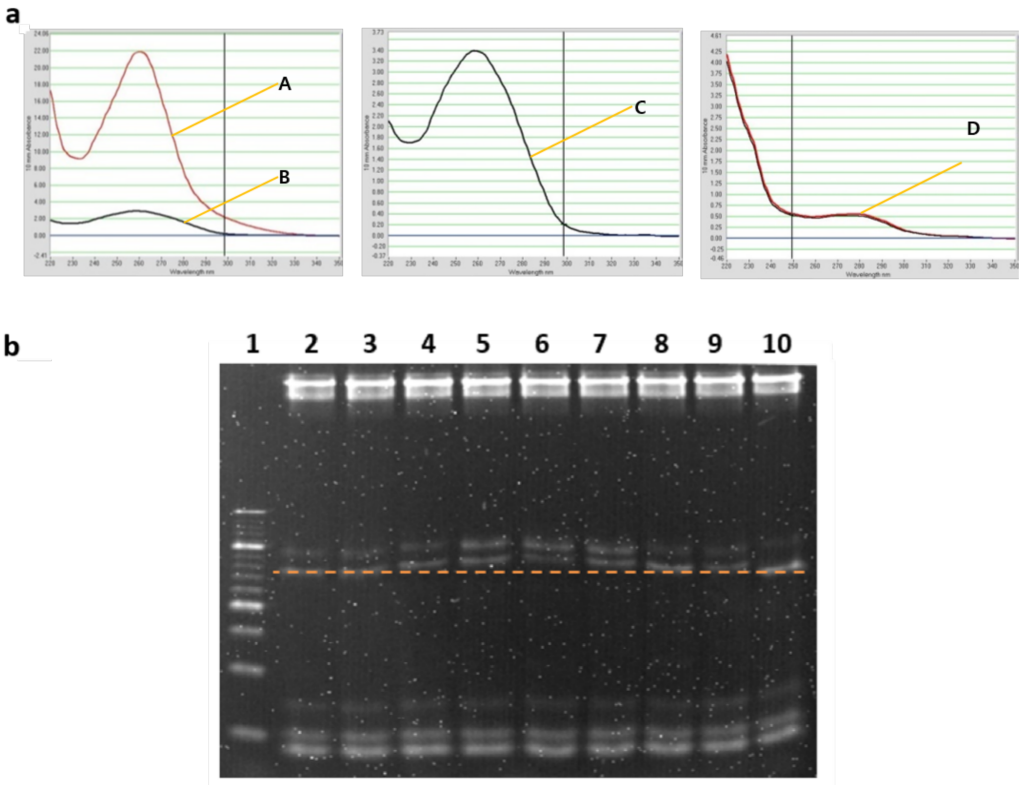


Figure 4.1: DNA quantification and hybridization of final complementary strands for verifying the presence of a 3-order double-stranded random DNA library. a. Nanodrop analysis for quantification of DNA amount at each step of the protocol; pre-hybridization of variables pre-ligase (A), post-ligase (B), post-purification (C) post-separation (D) b. gel electrophoresis of 3-order random library hybridized with each type of complementary strands (C1-C9 in lanes 2–10), so lane 2 contains the first complementary possible complementary sequence and so on to a total of nine possible hyperedges that could have been produced. Marked line shows 70-mer sequences.

4.2 Creating Free-Order DNA Hypernetworks

For the creation of the free-order hyperedges or different lengths of hyperedges, the concentrations of DNA sequences according to its corresponding pixel greyscale value, was added to a tube and ligation performed as described above. The PAGE electrophoresis gel illustrates the free-order hyperedges which were produced from the ligation procedure (Figures 3.3 and 4.2 a). Against the 10 bp ladder, many bands of varying intensities can be seen, representing different lengths of DNA present in the sample. From the original tube of 15 bp single stranded DNA following the ligation protocol, the formation of random hyperedges from 30 bp to 300 bp are visible (Figure 4.2 a). This correlates to 0-order to 15-order hyperedges. The same procedure was carried out to produce free-order hyperedges for every dataset: Training data and test data.

4.3 Weight Update Feedback Loop for DNA Hypernetwork

Figure 4.2b shows the result of enzyme treatment for S1 nuclease. DNA was incubated with the enzyme for 30 min at room temperature than enzyme inactivated with 0.2 M of EDTA at heating at 70° for 10 min. The control lane shows 4 bands, the perfectly matched strand, 1-mismatched, 2-mismatched, and 3-mismatched strands from the bottom to the top of the gel. The function of the S1 nuclease is investigated for decreasing weights or in this case DNA concentration, where perfectly matched DNA sequences are cleaved and mismatched sequences remain. In the 0.5 S1 lane, it is evident that only the perfectly matched DNA strands are cleaved and the mismatched strands remain in

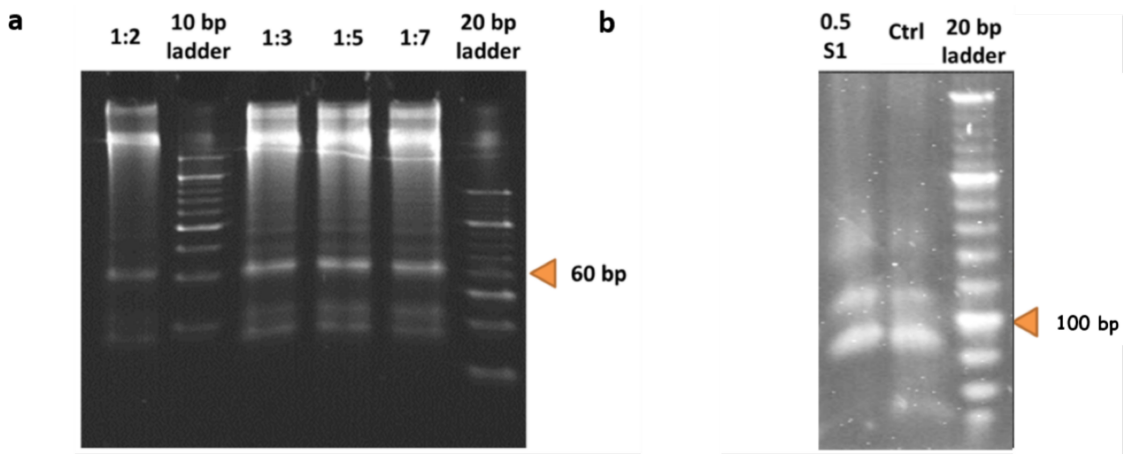


Figure 4.2: In vitro experimental results. a. Free-order hyperedge production. The 10 bp and 20 bp ladders were added in lanes 2 and 6. The ratio between primers and variable DNA sequences were varied to see if there was any effect on the length of the hyperedges produced. b. The control lane shows from 70 base pairs up, perfect, 1, 2, 3-mismatched sequences. The perfectly matched DNA at 70 bp, 1-mismatch at 90 bp, 2-mismatched at 110 bp and 3-mismatched at 130 bp. Lane 1 contains the sample with S1 nuclease treatment. The perfectly matched DNA strands in the lower most band was cleaved.

the mixture. This represents the sequences only present exclusively in the data for digits ‘6’ or ‘7’ can be reproduced with the use of S1 nuclease in the weight update algorithm.

It is interesting to note that the issue of scalability may be addressed through our design which allows 10-class digit classification within the same number of experimental steps. More classes of training data could be added in the hybridization stage to all but the one class of training data for which the label is being learned. This provides a larger scale of digit classification without drastically increasing the workload, time or the need to order new sequences. This novel method of implementing digit classification and experimental results demonstrate the enzymatic reactions which is prerequisite to making this experimentally plausible.

Chapter 5

Implementation of Molecular Machine Learning In Vitro

The full implementation of molecular machine is discussed in detail in the last chapter of the dissertation. Here, the full implementation refers to the five iterations of molecular learning carried out in vitro, to classify digits '6' and '7'. We put into practice, the theory of a DNA computing model capable of directing molecular machine learning through the proposed algorithm in our study, the hypernetwork.

In this chapter, we discuss in detail the steps required in a sequential manner to demonstrate not only the efforts of devising sound experimental design for actuating the theory to real experimental, but to give an insight in how DNA computing is realized in the laboratory, together with overcoming some of the the limitations discussed in Chapter 1.1.4.

Please note, some sections may be redundant to some materials covered

earlier in the dissertation. However, in this chapter, unlike the proof-of-concept experimental results presented earlier to ensure the critical components of this molecular machine learning system were functional, here the purpose is covering the complete scheme of practice molecular machine learning. How the theory can be put into practice and specific examples and protocols are illustrated for understanding our model put into practice in laboratory conditions.

5.1 Construction of the DNA Dataset

A crucial part of this dissertation was the novel method in which DNA datasets were constructed. To fulfil the purpose of a DNA dataset, witholding vast amounts of information in a large number of hyperedges, we provide the means of using the ligation technique to assemble multiple variables or pixel data in varying lengths of hyperedges. Figure 3.3 presents the sequential process of hybridizaton, annealing and ligation.

First, the DNA oligomers designed for each forward and reverse primer with labels ‘6’ and ‘7’ along with unique DNA sequences for pixels 1 to 25 were ordered as listed in Table 5.1. All DNA strands were purchased from Integrated DNA Technologies (IDT). Unique DNA sequences were designed with the use of the EGNAS program as discussed in Chapter 3. Figure 5.1 illustrates the predicted structures of the DNA designed using the mFold webserver. DNA oligomers were obtained in tubes which needed to be diluted by adding distilled water. 20 pmol/ul stock samples were made. Each diluted stock of DNA tubes were then carefully labelled for use in making the DNA dataset. Figure 5.2 illustrates the structures of DNA building blocks ordered as per the table in

Table 5.1.

Table 5.1: DNA sequences order list. Primers and variable were designed with relevant modifications. Key: FP = For- ward primer, RP = Reverse primer, L6 = Label for 6, L7 = Label for 7, T = Tag sequence (Sticky end), V = Variable (Pixel).

Label	5' Modifications	DNA Sequence	3' Modifications
Cy3-FP-L6-T	5' Cy3	TCGTAGAGTGTTATTTACAGTTGGAAGATATATTCGGAGATGTTA	
Cy5-FP-L7-T	5' Cy5	TCGTAGAGTGTTATTAATCTGAGTCCTTATTATTCGGAGATGTTA	
FP-L6-T		TCGTAGAGTGTTATTTACAGTTGGAAGATATATTCGGAGATGTTA	
FP-L7-T		TCGTAGAGTGTTATTAATCTGAGTCCTTATTATTCGGAGATGTTA	
FP'-L6'	5' Phosphate	TATCTTCCAACTGTAAATAACACTCTACGA	
FP'-L7'	5' Phosphate	ATAAGGACTCAGATTAATAACACTCTACGA	
RP-Biotin	5' Phosphate	ATAACCTGGATGAAT	3' Biotin
T'-RP'		ATTCATCCAGGTTATTAACATCTCCGAATA	
V1-T	5' Phosphate	TATCTCGCAAGTAATTATTCGGAGATGTTA	
V1'-T'	5' Phosphate	ATTACTTGCGAGATATAACATCTCCGAATA	
V2-T	5' Phosphate	AGGAATGTTGATACTTATTCGGAGATGTTA	
V2'-T'	5' Phosphate	AGTATCAACATTCCTTAACATCTCCGAATA	
V3-T	5' Phosphate	ACTGTGCTAATGATATATTCGGAGATGTTA	
V3'-T'	5' Phosphate	TATCATTAGCACAGTTAACATCTCCGAATA	
V4-T	5' Phosphate	TTGCCTCTATGATTATATTCGGAGATGTTA	
V4'-T'	5' Phosphate	TAATCATAGAGGCAATAACATCTCCGAATA	
V5-T	5' Phosphate	GATAACTACAAGGAATATTCGGAGATGTTA	
V5'-T'	5' Phosphate	TTCCTTGTAGTTATCTAACATCTCCGAATA	
V6-T	5' Phosphate	TACTAATCGTGAGAATATTCGGAGATGTTA	
V6'-T'	5' Phosphate	TTCTCACGATTAGTATAACATCTCCGAATA	
V7-T	5' Phosphate	TAGTCGTTGTAAGATTATTCGGAGATGTTA	
V7'-T'	5' Phosphate	ATCTTACAACGACTATAACATCTCCGAATA	
V8-T	5' Phosphate	GTTAGTCATACAGAATATTCGGAGATGTTA	
V8'-T'	5' Phosphate	TTCTGTATGACTAACTAACATCTCCGAATA	
V9-T	5' Phosphate	TATGTCTGAATCCAATATTCGGAGATGTTA	
V9'-T'	5' Phosphate	TTGGATTCAGACATATAACATCTCCGAATA	
V10-T	5' Phosphate	TACCGAAGTCAATAATATTCGGAGATGTTA	
V10'-T'	5' Phosphate	TTATTGACTTCGGTATAACATCTCCGAATA	
V11-T	5' Phosphate	AGTGAGGTAGATAATTATTCGGAGATGTTA	
V11'-T'	5' Phosphate	ATTATCTACCTCACTTAACATCTCCGAATA	
V12-T	5' Phosphate	TACACCTAAGCAATATATTCGGAGATGTTA	
V12'-T'	5' Phosphate	TATTGCTTAGGTGTATAACATCTCCGAATA	
V13-T	5' Phosphate	ATCTTGTCATTCGTATATTCGGAGATGTTA	
V13'-T'	5' Phosphate	TACGAATGACAAGATTAACATCTCCGAATA	
V14-T	5' Phosphate	CTGGTTAGAATACAATATTCGGAGATGTTA	
Continued on next page...			

Table 5.1 – continued from previous page.

Label	5' Modifications	DNA Sequence	3' Modifications
V14'-T'	5' Phosphate	TTGTATTCTAACCAGTAACATCTCCGAATA	
V15-T	5' Phosphate	CAGTGATTGTTCTATTATTCGGAGATGTTA	
V15'-T'	5' Phosphate	ATAGAACAATCACTGTAACATCTCCGAATA	
V16-T	5' Phosphate	TCAGAACTTACGATATATTCGGAGATGTTA	
V16'-T'	5' Phosphate	TATCGTAAGTTCTGATAACATCTCCGAATA	
V17-T	5' Phosphate	ATCCTGTAACTCATATATTCGGAGATGTTA	
V17'-T'	5' Phosphate	TATGAGTTACAGGATTAACATCTCCGAATA	
V18-T	5' Phosphate	ACTGCTTGACATTATTATTCGGAGATGTTA	
V18'-T'	5' Phosphate	ATAATGTCAAGCAGTTAACATCTCCGAATA	
V19-T	5' Phosphate	TACCTATTGGACTTATATTCGGAGATGTTA	
V19'-T'	5' Phosphate	TAAGTCCAATAGGTATAACATCTCCGAATA	
V20-T	5' Phosphate	ATACCGTTGAGATTATATTCGGAGATGTTA	
V20'-T'	5' Phosphate	TAATCTCAACGGTATTAACATCTCCGAATA	
V21-T	5' Phosphate	CTTACTGAGACAATATATTCGGAGATGTTA	
V21'-T'	5' Phosphate	TATTGTCTCAGTAAGTAACATCTCCGAATA	
V22-T	5' Phosphate	ATCGTTATGGAGAATTATTCGGAGATGTTA	
V22'-T'	5' Phosphate	ATTCTCCATAACGATTAACATCTCCGAATA	
V23-T	5' Phosphate	ATAGACTGAAGGATTTATTCGGAGATGTTA	
V23'-T'	5' Phosphate	AATCCTTCAGTCTATTAACATCTCCGAATA	
V24-T	5' Phosphate	TAACTGCGATACATTTATTCGGAGATGTTA	
V24'-T'	5' Phosphate	AATGTATCGCAGTTATAACATCTCCGAATA	
V25-T	5' Phosphate	TAATGCGTTCACTATTATTCGGAGATGTTA	
V25'-T'	5' Phosphate	ATAGTGAACGCATTATAACATCTCCGAATA	

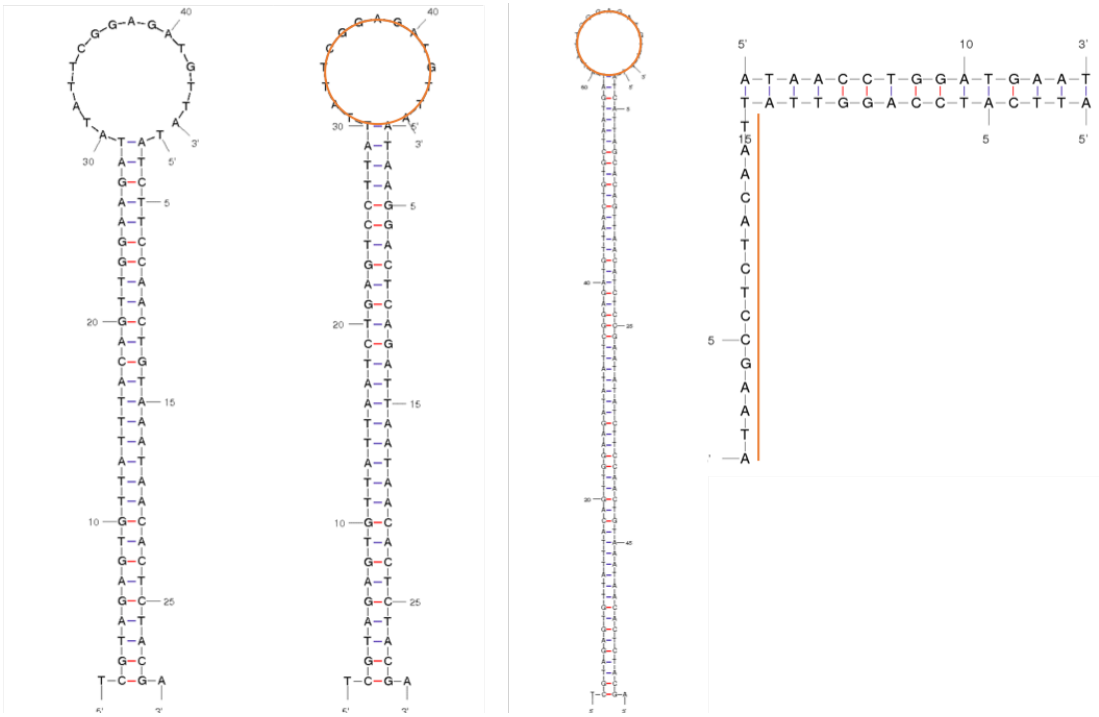


Figure 5.1: Prediction of DNA structures using the mFold webserver shows free sticky ends highlighted in color and perfectly matched DNA. From left, the forward primer units for digits '6' and '7 are shown. The third structure shows a variable with two sticky ends on either side, and the reverse primer unit with one single stranded region connected to a double stranded section of DNA.

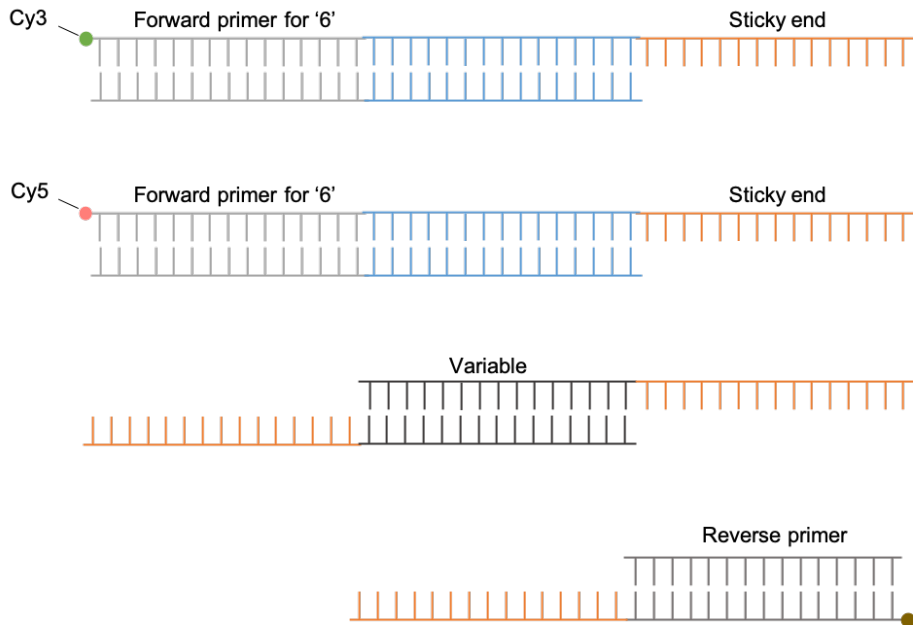


Figure 5.2: DNA primers and variable units with sticky ends for ligation of separate DNA oligomers in a random manner. From top, forward primer with Cy3 or Cy5 fluorescence tags act as the first unit of the DNA construct for digits '6' and '7' respectively. They both have sticky ends for ligation of the variable units. The variable unit is designed with sticky ends on either side allowing for more than one variable to be bound in the self-assembling process. The reverse primer acts as the third right most unit containing a biotin modification for separation of double stranded DNA strands to single stranded DNA during the training and test process.

As the ordered DNA oligomers are received as single stranded DNA, the complementary strands for each designed strand was also ordered. The next step was to put these single stranded DNA with their complementary base pairing sequences for hybridization to form double stranded DNA. For example, V1-T sequence with V1' - T' in one tube in equal amounts. 5 ul of each DNA, top and bottom strands, 10 X SSC and distilled water were added to make 20 ul of 5 pmol/ul of variables and primers in their respective tubes. These were put into a PCR machine which heated the DNA for two minutes at 95° then decreased to 10° by 1° every two minutes.

The final double stranded DNA sequences were diluted with distilled water to 200 ul of 0.5 pmol/ul to be used in making datasets for each MNIST image.

Table 5.2 illustrates how the pixels encoded in DNA are combined to represent an image in a test tube. Each value of the pixel displayed are 10 times the value of their actual grey scale value (scaled from 0 to 1). As pipettes are accurate enough to measure and expel DNA volumes from 1 ul and up, sufficient accuracy of each DNA for each pixel added into the tube representing the image was demonstrated.

Table 5.2: Data representation with DNA sequences representing 25 randomly chosen pixels of a 10 x 10 MNIST image. Blue = Cy5 label for '6', Red = Cy3 label for '7'.

	Minibatch 1						**Minibatch 2**					
	Ensemble 1		**Ensemble 2**		**Ensemble 3**		**Ensemble 1**		**Ensemble 2**		**Ensemble 3**	
Pixel	1.1	1.1	2.1	2.1	3.1	3.1	1.2	1.2	2.2	2.2	3.2	3.2
Pixel 1	1.45	0.00	0.00	0.57	3.11	6.33	5.75	0.00	0.00	6.40	0.00	0.36
Pixel 2	0.00	0.00	9.36	0.00	5.64	4.43	0.00	0.00	3.73	0.00	0.07	0.00
Pixel 3	0.00	0.00	0.26	8.59	7.20	2.04	0.00	0.00	0.27	3.63	0.00	0.25
Pixel 4	7.43	0.00	0.00	0.00	9.11	0.00	0.25	5.52	4.83	0.00	1.32	0.28
Pixel 5	5.41	9.84	1.37	0.00	0.00	0.00	0.98	7.57	0.00	0.00	0.00	0.00
Pixel 6	0.00	0.00	6.54	1.10	0.00	0.00	0.00	0.00	1.41	0.00	0.00	0.00
Pixel 7	0.00	3.44	0.00	0.00	1.09	2.49	4.89	0.00	0.00	0.00	0.00	4.16
Pixel 8	0.00	0.00	0.00	2.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pixel 9	0.00	0.00	0.00	0.00	6.75	6.50	0.00	0.00	0.00	0.00	7.84	0.96
Pixel 10	0.00	0.00	0.00	0.00	9.90	4.65	0.00	0.00	0.00	0.00	6.74	7.71
Pixel 11	0.00	0.00	0.00	0.00	0.00	6.62	0.00	0.00	0.00	0.00	0.00	3.90
Pixel 12	0.00	0.00	0.00	0.00	3.64	3.32	0.00	0.00	1.33	0.00	6.45	0.36
Pixel 13	8.95	9.54	0.50	0.00	0.47	0.00	1.17	0.62	0.00	0.00	5.51	0.74
Pixel 14	8.23	4.98	0.00	0.00	2.34	0.00	5.96	8.45	0.00	0.00	0.00	0.00
Pixel 15	0.00	2.09	8.48	2.10	7.27	1.23	0.00	0.00	0.42	0.00	0.05	0.00
Pixel 16	0.00	9.88	9.90	9.81	8.21	1.13	0.00	5.71	3.14	7.16	6.30	0.00
Pixel 17	0.00	0.00	0.00	0.00	0.00	2.13	0.00	0.00	0.00	0.00	0.00	0.00
Pixel 18	9.43	9.31	6.29	0.00	2.05	0.10	1.05	7.16	6.04	0.00	5.25	7.67
Pixel 19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pixel 20	0.91	0.00	2.49	0.00	0.00	0.00	0.47	1.04	0.00	0.00	0.00	0.00
Pixel 21	4.75	0.00	0.38	0.00	2.30	9.60	6.55	0.00	6.32	0.00	0.00	2.31
Pixel 22	9.13	0.00	8.72	9.03	0.00	0.00	1.31	0.00	0.00	0.40	0.00	0.00
Pixel 23	0.00	0.00	8.92	4.36	2.75	9.61	0.00	0.00	4.49	8.46	0.00	9.97
Pixel 24	8.52	0.00	0.00	0.00	0.00	0.00	5.51	0.00	0.00	0.00	0.00	0.00
Pixel 25	0.00	1.00	2.61	4.66	0.44	1.39	0.00	3.15	0.00	2.79	0.02	0.00
RP	12.84	10.02	13.17	8.54	14.45	12.31	6.78	7.84	6.40	5.77	7.91	7.73
Cy5-FP-6	12.84		13.17		14.45		6.78		6.40		7.91	
Cy3-FB-7		10.02		8.54		12.31	6.78	7.84		5.77		7.73

A key point to this research is the use of ensemble data to use only 25 pixels to represent 100 pixels (of a 10 x 10 image) which allows more data to be represented with fewer number of unique DNA sequences. The five iterations of learning that will be performed means five difference images of the MNIST dataset are going to be 'seen' or experienced by the learning model. The five images used were randomly selected and allocated to five minibatches. Each minibatch contains three ensemble groups, that is the three different sets of randomly chosen 25 pixels for each minibatch (image). Three ensembles for five minibatches are constructed through a DNA dataset and this data creation is carried our for both '6' and '7' images. In other words, five '6' images are selected to be in each minibatch and within a minibatch, of that very '6' image, 25 pixel DNA are chosen to represent the ensemble. Three ensembles are made for each minibatch. The same process is applied to '7' images. Figure 2.6 illustrates how the minibatch and ensemble concept allows for molecular learning through only five iterations and using 25 pixels to represent 10 by 10 images.

The process of hybridization where single stranded DNA oligomers were made to match with complementary base pairs to form double stranded DNA oligomers for the DNA dataset is depicted in Figure 3.3 I.

The next step is to anneal the sticky ends or tag sequences at the ends of the variable DNA to form random combinations of variable DNA or hyperedges (Figure 3.3 II). To each tube containing an ensemble of DNA for each minibatch, 10X SSC and distilled water was added to undergo PCR at 30° to 4° at 5 minute intervals.

Finally, the annealed variables are treated with T4 DNA ligase to firmly ligate or bind the sticky ends of DNA together to form stable hyperedges. Table 5.3 shows the DNA concentrations obtained from measuring each tube of sample in the nanodrop machine. According to the detected level of DNA, 5 ul of T4-ligase was added to each tube and mixed gently by pipetting up and down. The samples were incubated in room temperature for two hours then inactived by heat at 65° for 10 minutes and chilled on ice.

The final section to creating DNA database, is the separation double stranded hyperedges two single-stranded hyperedges. For this process we use the biotin streptavidin technique, Within the beats are used to separate the two strands update double stranded helix. Table 7.4 shows the amount of DNA left and each some pool after the litigation process. Unfortunately the yield of DNA off the such enzymatic processes is quite low and is an important point of discussion when taking into account the practical difficulties of using enzymes in DNA computing settings.

After taking into account the reduced yield in final concentration of DNA after the ligation process, a percentage of the sample is transferred two new tubes to undergo the bead separation process. According to the amount of DNA in the sample, the amounts of beads needed is carefully calculated and prepared for the full separation.

Table 5.3: Ligation of variable DNA to form hyperedges. Blue = Cy5 label for ‘6’, Red = Cy3 label for ‘7’.

	Minibatch 1						**Minibatch 2**					
	Ensemble 1		**Ensemble 2**		**Ensemble 3**		**Ensemble 1**		**Ensemble 2**		**Ensemble 3**	
Experiment	1.1	1.1	2.1	2.1	3.1	3.1	1.2	1.2	2.2	2.2	3.2	3.2
C from nanodrop (ng/ul)	181.13	138.43	213.9	190.19	223.5	176.92	228.51	149	186.99	153.13	153.8	146.86
DNA from nanodrop (umol)	263.06	235.28	421.95	275.58	474.85	369.49	300.31	198.20	202.96	149.86	206.38	192.67
V to ligate (ul)	26.97	31.56	41.48	26.90	45.52	38.78	21.35	24.70	20.15	18.17	24.92	24.36
T4-ligase (ul)	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
10X ligation buffer (ul)	3.60	2.37	3.11	2.02	3.41	2.91	1.60	1.85	1.51	1.36	1.87	1.83
DW (ul)	0.43	16.30	23.00	13.16	25.72	21.18	9.41	11.67	8.60	7.27	11.82	11.44
Total volume (ul)	36.00	23.7	31.1	20.2	34.1	29.1	16.0	18.5	15.1	13.6	18.7	18.3

Table 5.4: Biotin separation for making a single stranded DNA dataset.

	Minibatch 1						**Minibatch 2**					
	Ensemble 1		**Ensemble 2**		**Ensemble 3**		**Ensemble 1**		**Ensemble 2**		**Ensemble 3**	
Experiment	1.1	1.1	2.1	2.1	3.1	3.1	1.2	1.2	2.2	2.2	3.2	3.2
Amount in sample (pmol)	26.31	23.53	47.77	27.55	54.78	36.94	26.27	19.81	20.29	14.98	20.63	19.26
C in sample (pmol/ul)	9.75	7.45	11.52	10.24	12.03	9.53	12.30	8.02	10.07	8.24	8.28	7.90
Volume left to separate (ul)	8.09	9.47	12.44	8.07	13.66	11.63	6.41	7.41	6.05	5.45	7.47	7.31
Amount in sample (pmol)	78.92	70.58	143.32	82.66	164.33	110.82	78.81	59.44	60.86	44.94	61.88	57.77
C in sample (pmol/ul)	9.75	7.45	11.52	10.24	12.03	9.53	12.30	8.02	10.07	8.24	8.28	7.90
Amount in sample (ng)	1465	1310	2661	1535	3052	2058	1464	1104	1131	834.8	1150	1073
Amount in sample (ug)	1.47	1.31	2.67	1.54	3.05	2.06	1.46	1.10	1.13	0.83	1.15	1.07
Bead needed (ul)	15	13	27	15	31	21	15	11	11	8	11	11
Bead V with 2X B&W (ul)	29.3	26.2	53.2	30.7	61.0	41.2	29.3	22.1	22.6	16.7	23.0	21.5
Volume of DNA (ul)	15	13	27	15	31	21	15	11	11	8	11	11

The separation process is divided into two parts. First the washing of the beads, with the calculated amount of being transferred into a test tube and washed with washing buffer, resuspended, placed in the magnet for three minutes and it's supernatant and discarded. This washing process is repeated for total of three washes. Next the immobilization process uses that 2X BW buffer, which is added and twice the original volume. Here, the calculated volume of biotinylated DNA is added to the mixture. The sample is incubated for 15 minutes at room temperature using gentle rotation. After this, the samples are placed back into the magnetic rack And left for three minutes so that the biotinylated DNA can be separated from it's complimentary strand. The supernatant removed from the tube is then washed and resuspended to the desired concentration and the DNA left In the tube is immobilized from the beads in a low salt concentration of 10X SSC.

5.2 Training Iterations

The first iteration starts with hybridizing the minibatch one data for '6' and '7', for each of the three ensembles independently. This is depicted in figure 5.3, with the venn diagram and matching of upper and lower strands of the '6' and '7' training data, respectively to form a common and uncommon area of matching.

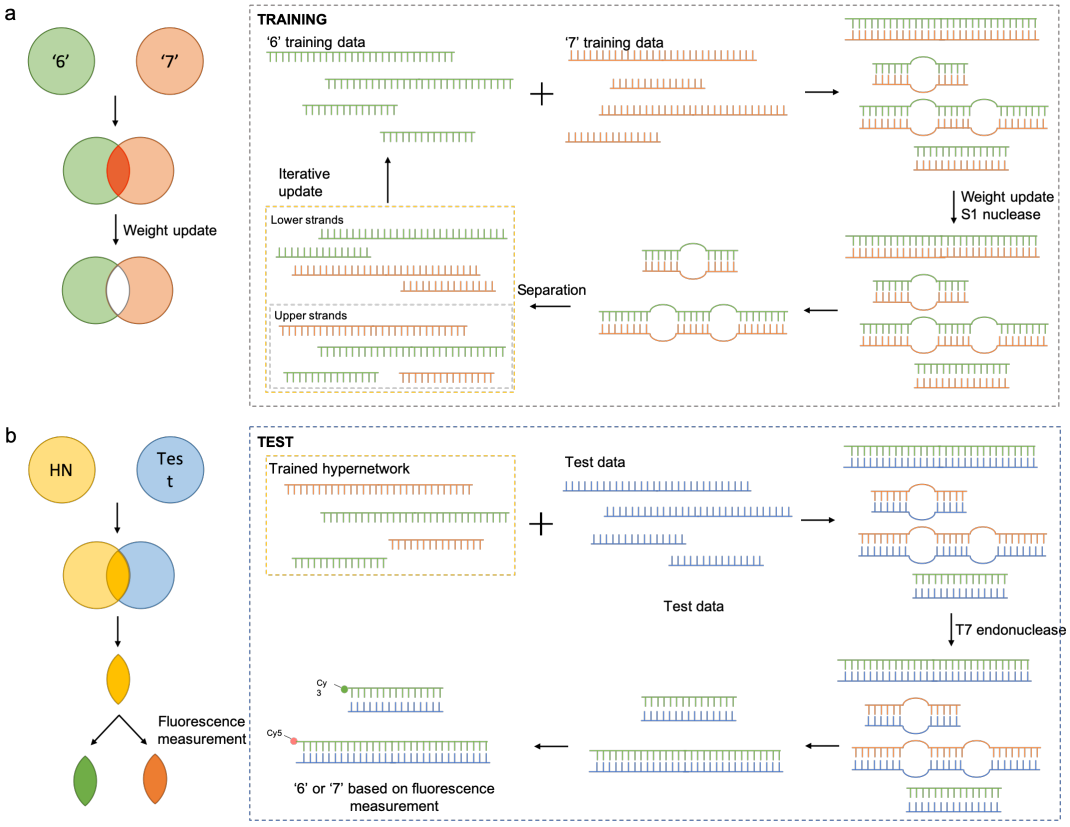


Figure 5.3: a. Venn diagram and experimental scheme for training the molecular hypernetwork model. b. Venn diagram and experimental scheme for testing the molecular hypernetwork model.

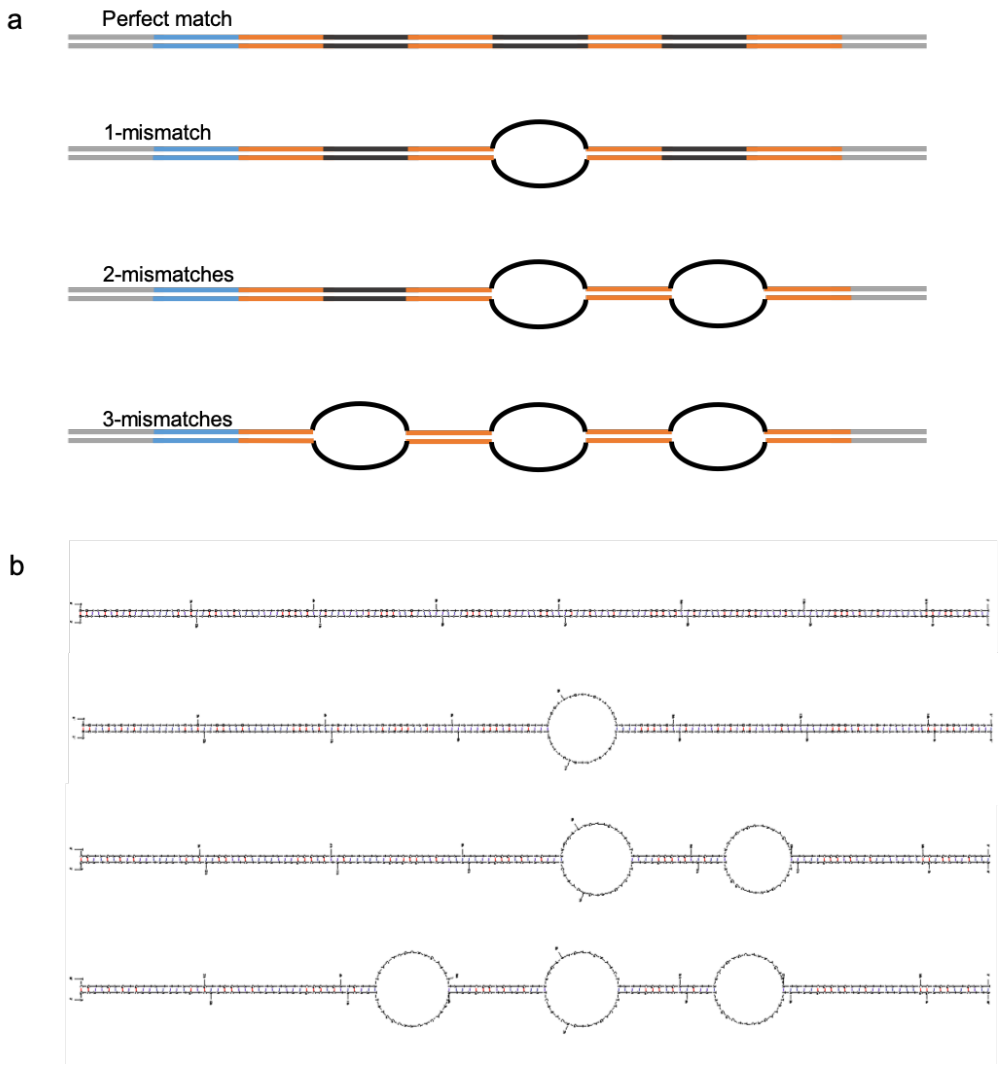


Figure 5.4: a. DNA internal loops are formed when mismatches in DNA hybridization occur. These loops are used as target sites of S1 nuclease which is used in the enzymatic weight update process. b. 3-order hyperedges from perfect match to 1,2,3-mismatched double stranded DNA are predicted using the mFold webserver.

Figure 5.4 gives a visualization of the internal DNA loops which are formed from the matches and mismatches of complementary and non-complementary matches between the DNA variables. If the pixel was not matching on the hyperedges formed in the model for ‘6’ and ‘7’, the internal loops form. This allows a method of measuring the discrepancy between learning the digits by updating the hyperedge pool or hyoernetwork model for ‘6’ and ‘7’. A perfectly matched strand will travel futher on the electrophloresis gel compared to the DNA strands with internal loop structures, with the DNA with 3-mismatches travelling the slowest. The mFold webserver can be used to visualize the hybridized DNA hyperedges which predicts the internal loops according to the number of mismatches present.

The first training iteration initiates by hybridizing the training data for ‘6’ and ‘7’ for each ensemble in the first minibatch. Table 5.5 shows the volumes mixed for each ensemble in iteration 1. 1.1 signifying the first ensemble of the first minibatch and 2.1, the second ensemble of the first minibatch and so on. Red colored tubes are the top strands of the hyperedges from the image ‘6’ and blue for ‘7’. As shown by the venn diagram earlier, this is the combination of the training data for ‘6’ and ‘7’. With 10X SSC, the tubes are prepared to undergo a PCR hybridization reaction.

The same procedure is taken for training the model ‘7’, where in this case, the top strand is ‘7’ image hyperedges and bottom is ‘6’ image hyperedges (Table 5.6).

Table 5.5: Hybridization of ‘6’ and ‘7’ for pattern matching between hyperedges. For testing ‘6’.

TOP	**70% of V**	**BOTTOM**	**70% of V**	**Tube label**	**Sum**	**10X SSC**	**DW**	**Total**
1.1	12.7	1.1	13.8	6-1.1	26.43	3	0.57	30
2.1	14.2	2.1	22.1	6-2.1	36.38	4.1	0.52	41
3.1	17.9	3.1	24.9	6-3.1	42.77	4.8	0.43	48

Table 5.6: Hybridization of ‘6’ and ‘7’ for pattern matching between hyperedges. Model for ‘7’.

TOP	**70% for test**	**BOTTOM**	**70% for test**		**Sum**	**10X SSC**	**DW**	**Total**
1.1	13.8	1.1	12.7	7-1.1	26.43	2.4	0.45	24
2.1	22.1	2.1	14.2	7-2.1	36.38	3.3	0.60	33
3.1	24.9	3.1	17.9	7-3.1	42.77	4	1.78	

The next step is the enzymatic weight update. S1 nuclease enzyme as shown in previous results, cleave the single stranded regions of double stranded DNA, or in this case, the mismatched regions which form internal loops of DNA. Table 5.7 this step in detail. First, 80% of the volume from the hybridized hyperedges is taken and to this, the relevant amount of S1 nuclease, 5X reaction buffer and nuclease free water. The samples are incubated in room temperature for 30 minutes for the nucleaes reaction to take place, then inactivate by adding 2 ul of 0.5 M EDTA and heating it in a heat block for 10 minutes at 70° .

This process is again repeated in the same way for the training of the hyper-network model for '7' as shown in Table 5.8.

The training process proceeds with the separation of the double stranded DNA hyperedges to single stranded DNA for use in the updating process of the next iteration of learning (Figure 5.3 a). It is important to note here, the information learned in iteration one is transferred to the next iteration of learning. This is the concept of online learning. Table 5.9 shows the calculations for the separation process using streptavidin and biotin in the protocol described in the previous section (Chapter 5.1). The amount of bead needed for the sample for each ensemble for both the models for '6' and '7' are calculated and carried out carefully to maximize the yield.

Table 5.7: Enzymatic weight update for iteration one using S1 nuclease to cleave. Model for '6'.

							Enzyme S1 nuclease (20 U/ul)			
	Total	**80% for training**	**C (ng/ul)**	**w (ng)**	**w (ug)**	**n (pmol)**	**5X Reaction buffer**	**S1 nuc (20U/ul)**	**Nuclease free W**	**Total**
6-1.1	30	24	122.53	2941	2.941	158.4	6.4	1	0.6	32
6-2.1	41	32.8	62.1	2037	2.037	109.7	8.6	1	0.6	43
6-3.1	48	38.4	95.52	3668	3.668	197.5	10	1	0.6	50
6-1.2	29	23.2								
6-2.2	24	19.2								
6-3.2	26	20.8								
6-1.3	27	21.6								
6-2.3	34	27.2								
6-3.3	40	32								
6-1.4	33	26.4								
6-2.4	41	32.8								
6-3.4	40	32								
6-1.5	31	24.8								
6-2.5	41	32.8								
6-3.5	51	40.8								

Table 5.8: Enzymatic weight update for iteration one using S1 nuclease to cleave. Model for '7'.

							Enzyme S1 nuclease (20 U/ul)			
	Total	**80% for training**	**C (ng/ul)**	**w (ng)**	**w (ug)**	**n (pmol)**	**5X Reaction buffer**	**S1 nuc (20U/ul)**	**Nuclease free W**	**Total**
6-1.1	30	24	122.53	2941	2.941	158.4	6.4	1	0.6	32
6-2.1	41	32.8	62.1	2037	2.037	109.7	8.6	1	0.6	43
6-3.1	48	38.4	95.52	3668	3.668	197.5	10	1	0.6	50
6-1.2	29	23.2								
6-2.2	24	19.2								
6-3.2	26	20.8								
6-1.3	27	21.6								
6-2.3	34	27.2								
6-3.3	40	32								
6-1.4	33	26.4								
6-2.4	41	32.8								
6-3.4	40	32								
6-1.5	31	24.8								
6-2.5	41	32.8								
6-3.5	51	40.8								

Table 5.9: Bead separation for iteration 1 after nuclease weight update for the next iteration and use in the test stage.

	6-1.1	**6-2.1**	**6-3.1**	**7-1.1**	**7-2.1**	**7-3.1**
V after nuclease (ul)	32	43	50	26	35	42
V left for separation (80%) (ul)	25.6	34.4	40	20.8	28	33.6
C (ng/ul)	24.90	70.07	16.77	18.55	103.8	8.18
n (ng)	637.4	2410	670.8	385.8	2907	274.8
n (pmol)	34.33	129.8	36.13	20.78	156.5	14.80
Amount in sample (ng)	637.4	2410	670.8	385.8	2907	274.8
Amount in sample (ug)	0.637	2.410	0.671	0.386	2.907	0.275
Bead needed (mg)	0.0319	0.121	0.034	0.019	0.145	0.014
Bead needed (mg) x2	0.064	0.241	0.067	0.039	0.291	0.027
bead needed (ul)	6.374	24.10	6.708	3.858	29.07	2.748
Bead V with 2X B&W (ul)	6.4	24.1	6.7	3.9	29.1	2.7
Minimum beads, 5 ul	10.0	24.1	10.0	10.0	29.1	10.0
Total bead needed (mg)	0.729					
Total bead needed (ul)	73					

The final stage of the training process is the addition of the trained hyperedges to the next iteration of learning. For the second iteration, the top strands of the ‘6’ model and bottom strands of the ‘7’ model will be hybridized again to measure the degree of matching using weight update. Before this, the top strands for the model learning ‘6’ is added to their respective ensembles. 6-1.1 - TOP as shown in red and 7-1.1 - BOT in Table 5.10 is added to the tube with the top strands of the first ensemble of the second iteration or image representation (1.2 in red for the digit ‘6’) and bottom strands in the same manner (1.2 in blue for the digit ‘7’).

Table 5.10: Combining hyperedges from iteration one to minibatch 2 data (1.2 - 3.2) for iteration 2.

TOP	70% for test	BOTTOM	70% for test	Label	sum	TOP	V (ul)	BOTTOM	V (ul)	Total DNA (ul)	10X SSC	DW	Total
1.1	12.7	1.1	13.8	6-1.1	26.43						3	0.57	30
2.1	14.2	2.1	22.1	6-2.1	36.38						4.1	0.52	41
3.1	17.9	3.1	24.9	6-3.1	42.77						4.8	0.43	48
1.2	11.2	1.2	13.7	6-1.2	24.97	6-1.1 - TOP	19.2	7-1.1 - BOT	15.6	44.17	5	0.83	50
2.2	9.3	2.2	11.4	6-2.2	20.76	6-2.1 - TOP	25.8	7-2.1 - BOT	21	46.56	5.3	1.14	53
3.2	11.0	3.2	11.5	6-3.2	22.56	6-3.1 - TOP	30	7-3.1 - BOT	25.2	52.56	5.9	0.54	59

5.3 Testing the Model

After each trained hypernetwork is formed, a test stage is paramount to assessing the success of the model in classifying the digits '6' and '7'. Figure 5.3 b shows how this is designed. The top strands of the hypernetwork trained from each iteration is hybridized with the bottom strands of the test images encoded in the DNA data construction process. This is illustrated in Figure 5.3 b, the scheme for testing the molecular hypernetwork. The hybridization process is the same as described in Chapter 5.2 for the first step of the training process using PCR. In the same way, internal DNA loops are formed where there are mismatches of variables in the hyperedges. However, in this case, as shown in the venn diagram, it is the mutually relevant hyperedges which need to be conserved. Intuitively, the perfectly matched DNA hyperedges are the hyperedges from the trained hypernetwork and the test images which matched better, which could be either the hyperedges from the '6' or '7' trained model are retrieved from the test hybridization.

To preserve the perfectly matched double stranded DNA hyperedges as opposed to the mismatched hyperedges as in the training process, a different enzyme is used. The T7 endonuclease I cleaves the mismatched DNA leaving the perfectly matched DNA. The protocol used for this enzyme is shown in Table 5.11, where the hybridized trained model and test data DNA has added the reaction buffer and 20 U/ul of T7 endonuclease I and is incubated for 30 minutes in room temperature. After this, the sample is inactivated by heat for 20 minutes at 65°.

Finally, the sample is ready for measuring the fluorescence levels of Cy3

and Cy5 which were used to label ‘6’ and ‘7’ models in the data construction process. This determines the majority of the hyperedges to be either from the hyperedges trained for the model for ‘6’ or for ‘7’.

Table 5.11: Test for iteration 1 for ‘6’ using bottom strands from the test image, for each ensemble independently.

									Enzyme T7 endonuclease (20 U/ul)			
Label	**30% for test**	**Divde to two**	**V (ul)**	**Test DNA**	**V (ul)**	**10X SSC**	**DW**	**Total**	**V to cleave**	**10X Reaction buffer**	**T7 (20U/ul)**	**Total**
6-1.1 - TOP	6.4	6-1.1 - TOP	3.2	1T - BOT	2	1	3.80	10	8	1	1	10
6-2.1 - TOP	8.6	6-2.1 - TOP	4.3	2T - BOT	2	1	2.70	10	8	1	1	10
6-3.1 - TOP	10	6-3.1 - TOP	5.0	3T - BOT	2	1	2.00	10	8	1	1	10
		7-1.1 - TOP	2.6	1T - BOT	2	1	4.40	10	8	1	1	10
		7-2.1 - TOP	3.5	2T - BOT	2	1	3.50	10	8	1	1	10
		7-3.1 - TOP	4.2	3T - BOT	2	1	2.80	10	8	1	1	10

5.4 Classification Results

In this section, the implementation of five learning iterations concludes with a learning curve which shows gradual learning of the digits ‘6’ and ‘7’ through molecular learning. The results were measured using modified DNA oligomers with Cy3 (red) and Cy5 (blue) dyes for ‘6’ and ‘7’, respectively.

Fluorescence measurement allows the relative comparison of the presence of Cy3 and Cy5 dyes which are linked to ‘6’ and ‘7’ image DNA data respectively. Cy3 and Cy5 have an excitation/emission spectra of 530-20/580-30 and 610-30/675-60 respectively. The fluorescence was measured using the CLARIOstar multichromator microplate reader from BMG Labtech.

Our results from operating five iterations of learning, in a massively parallel manner, with the sequential update after each iteration as shown in Figure 5.5. From each ensemble, or weak learner model trained from the first image minibatch, names the trained HN1 is added to the next iteration trained by image minibatch number two and so on.

For the test sets, each ensemble of each iteration is tested against the correlating ensemble of the test image. From the three ensemble test classification results, the voting system is used to produce the final classification result of either ‘6’ or ‘7’.

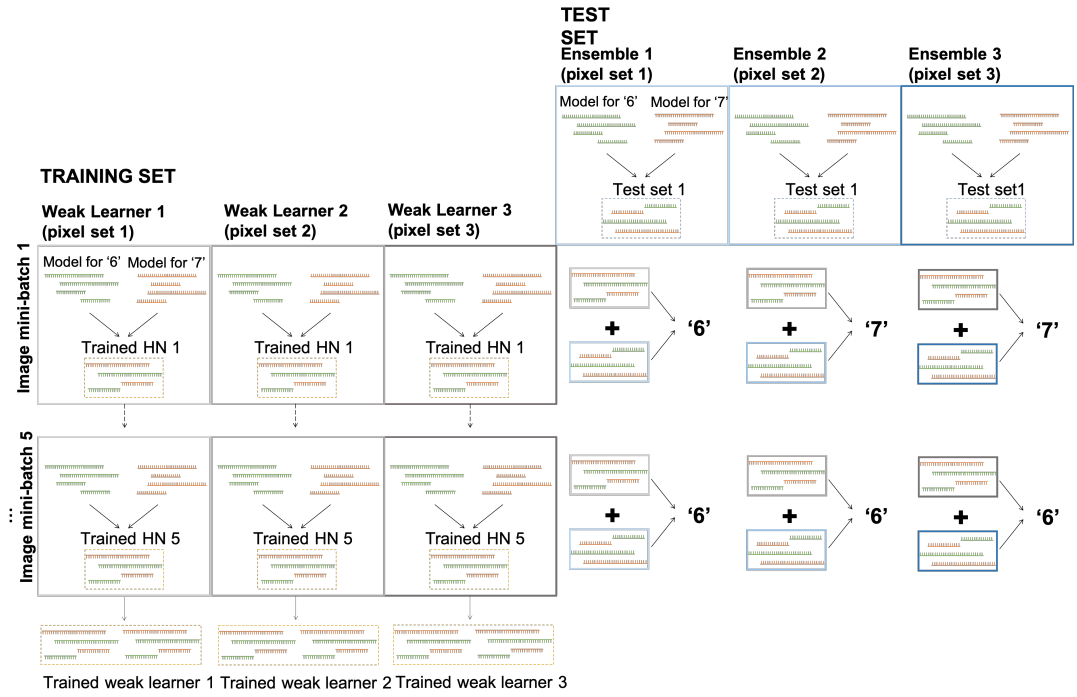


Figure 5.5: Ensemble learning process for the training and test over five iterations.

It is important to note here again, that we use ensembles to cover a much larger search space with the same number of DNA oligomers through the construction of free-order hyperedges. Each ensemble consists of a pixel set with 25 randomly selected pixels, chosen in a non-replaceable manner. In Figure 5.5, the overall training and test process is illustrated. For each ensemble, the model for '6' and '7' are combined. Weight update is performed as described in the methods section. Then a subset of DNA data is used to test the current ability to classify the digit and the rest is added to the next iteration for continue online learning.

Figure 5.6 and 5.7 shows the fluorescent measurements from the hypernetwork models tested at every iteration, for each ensemble. By seeing which fluorescent level was higher for each ensemble test, Cy3 or Cy5, the output is '6' or '7' respectively. For testing the HN6 or trained hypernetwork for the digit '6', the outs are, '7', '7', '6', for the first ensemble of the first iteration (HN6_1.1 - HN6_1.3). From these three outputs, through the voting method, we would chose '7' as the final output. The results demonstrate in increase in correctly answered trained models when tested for the model for '6', however, for '7' it was less successful. The voting system shows two incorrect, correct, then two incorrect outputs as the final answer for test with '7' images.

As this was the first implementation, to analyze the results further, the fluorescent level for all the ensembles in each iteration of the trained models were averaged for each iteration. This was used to compare and infer the changes in the pool of hyperedges with labels for '6' and '7' in each of the tested models to see the accuracy produced, disregarding the voting method. Figure 5.8

shows an increasing accuracy trend of the classification task with the increase in the number of iterations the training the models had gone through. Thus, the idea of generalization in machine learning, where with more experience or training iterations where the model is exposed to training data, an increase in the accuracy of the classification task can be observed.

Finally, the first result of an increase in learning accuracy achieved from the implementation of molecular machine learning for a 2-class classification problem. The test accuracy was calculated from averaging the fluorescent levels measured from each sample during the testing of the model. For each ensemble, the higher level of fluorescence of Cy3 or Cy5 was deemed the 'answer' given by the model in the classification task. For the three ensembles of each iteration, the voting system selected the final output as to a classification result of either '6' or '7'.

It is interesting to note the difference in the learning rate with the learning curves shown in the computer models in Chapter 2.5. The learning curve from molecular machine learning did not peak as quickly as the computer models in its entirety, however, it only took 5 iterations to reach an accuracy of over 90%. This highlights the capability of massively parallel processing when using DNA computing methods to solve problems, when a normal computer would require thousands of images of training iterations to achieve high classification accuracy results. Ofcourse, this was only the result of only one trial, and is the implementation of molecular machine learning in it's infancy. For future work, more trials would need to be carried out.

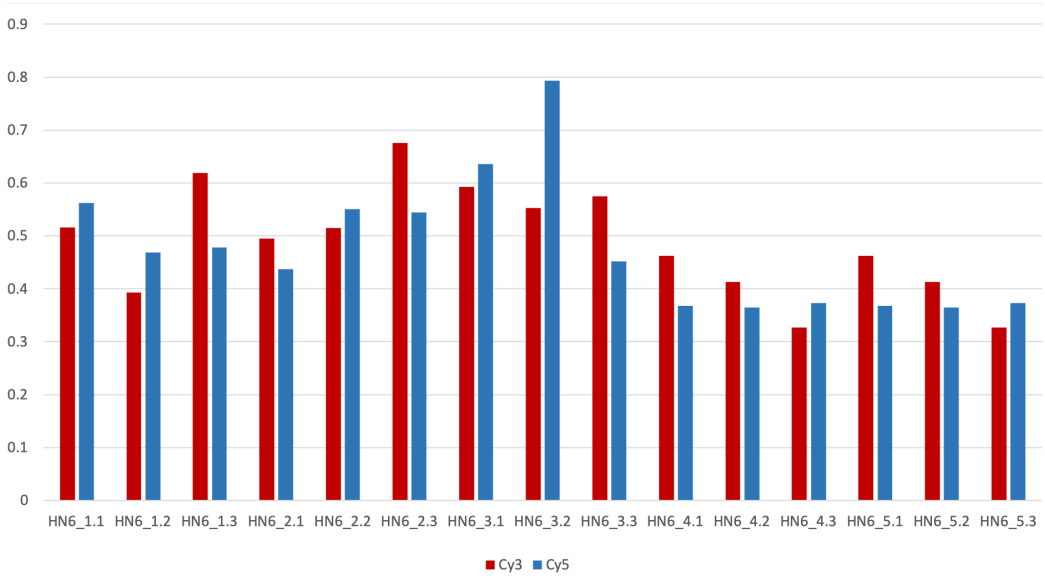


Figure 5.6: Fluorescent measurement results of the trained DNA hypernetwork model for '6' tested with test images (Scaled from 0 to 1).

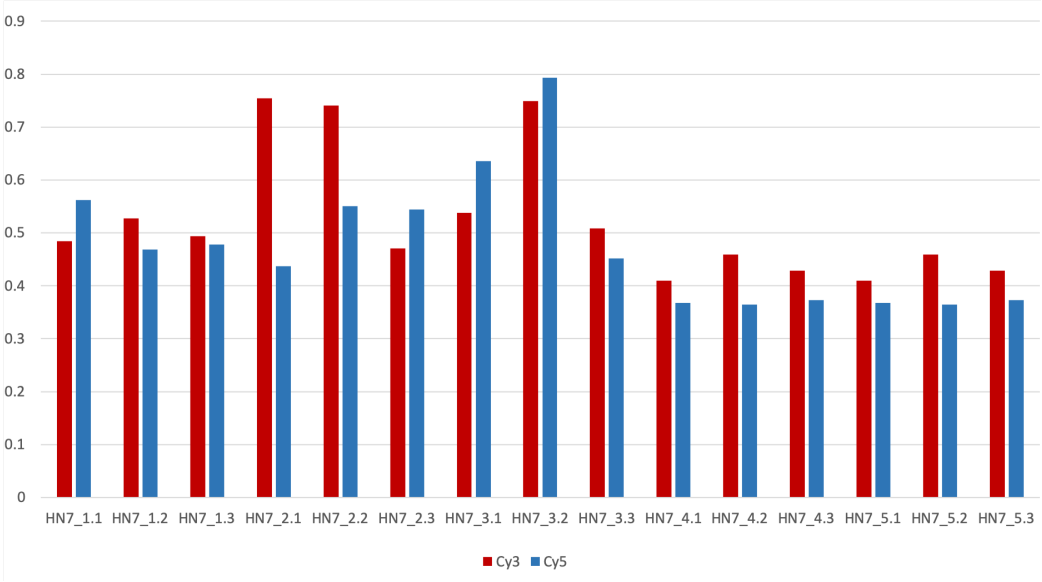


Figure 5.7: Fluorescent measurement results of the trained DNA hypernetwork model for '7' tested with test images (Scaled from 0 to 1).

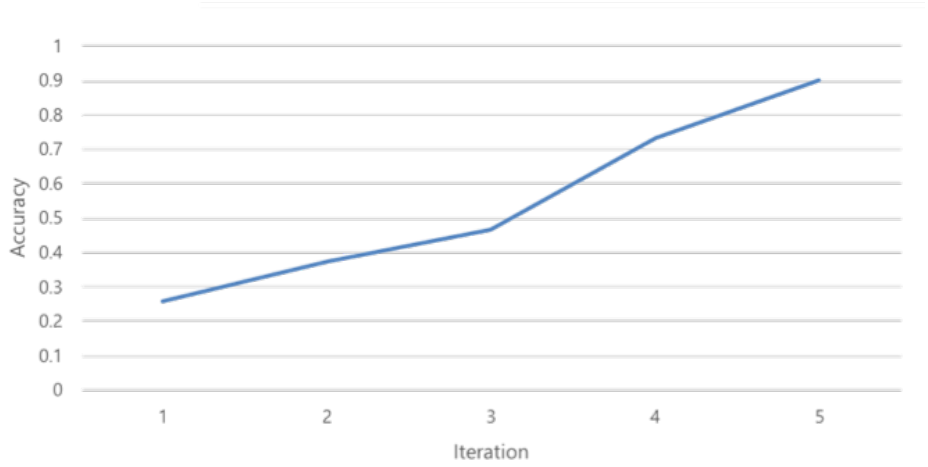


Figure 5.8: Experimental results of trained DNA model and test results. Test accuracy after every iteration of training shows an increasing learning accuracy.

Chapter 6

Conclusion

In this dissertation, we have proposed a novel molecular machine learning algorithm with a validated experimental scheme for in vitro demonstration of molecular learning with enzymatic weight update. The experiments are designed for plausible pattern recognition with DNA through iterative processes of self-organization, hybridization, and enzymatic weight update through the hypernetwork algorithm. Natural DNA processes act in unison with the proposed molecular learning algorithm using appropriate enzymes which allowed updating of weights to be realized in vitro. Unlike in previous studies, a molecular learning algorithm with enzymatic weight update is proposed, where the positive and negative terms of weight update are considered in the model for learning. Using the validated experimental steps, the model can be used for repeated learning iterations for the selection of relevant DNA to cause the DNA pool to continuously change and optimize, allowing large instance spaces to reveal a mixture of molecules most optimized to function as a DNA pattern

recognition classifier.

Our experiments showed a higher order feature extraction method was possible in vitro using higher-order DNA hyperedges which was demonstrated by constructing longer DNA sequence datasets. This method of DNA data construction dramatically reduced the number of unique DNA sequences required to cover the large search space of image feature sets. Finally, DNA ensemble learning is introduced for use of the full feature space in the classification tasks.

Although the complete iterations of learning still need to be improved, the aim of this dissertation was to provide a framework, results of fundamental operations required for realizing this in vitro and present preliminary results of carrying out the five iterations of learning to demonstrate a synergistic approach between theoretical and experimental designs of molecular learning algorithm. In future experiments we will continue to carry out the iterative molecular learning scheme in wet laboratory conditions to address the practical limitations of making molecular machine learning a reality.

By harnessing the strength of using biomolecules as building blocks for basic computations, new and exciting concepts of information processing have the potential to be discovered through more molecular computing methods. In turn, the implementation of machine learning algorithms through DNA could also act as a starting point for emerging technologies of computational molecular devices, implicated in a diverse range of fields such as intelligent medical diagnostics or therapeutics, drug delivery, tissue engineering, and assembly of nanodevices. As more advanced applications are explored, more intelligent molecular computing systems, with suitable intelligence to navigate and func-

tion in dynamic in vivo environments, may bridge gaps in current molecular computing technologies, so that DNA systems can function in uncontrolled, natural environments containing countless unforeseeable variables.

Bibliography

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, *266*(5187), 1021–1024.

Adleman, L. M. (1998). Computing with dna. *Scientific american*, *279*(2), 54–61.

Amir, Y., Ben-Ishay, E., Levner, D., Ittah, S., Abu-Horowitz, A., & Bachelet, I. (2014). Universal computing by dna origami robots in a living animal. *Nature nanotechnology*, *9*(5), 353–357.

Arkin, A. (2008). Setting the standard in synthetic biology. *Nature biotechnology*, *26*(7), 771.

Baek, C., Lee, S.-W., Lee, B.-J., Kwak, D.-H., & Zhang, B.-T. (2019). Enzymatic weight update algorithm for dna-based molecular learning. *Molecules*, *24*(7), 1409.

Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., & Shapiro, E. (2001). Programmable and autonomous computing machine made of biomolecules. *Nature*, *414*(6862), 430–434.

Bishop, C. M. (2006). Machine learning and pattern recognition. *Information Science and Statistics. Springer, Heidelberg*.

Brown, C. W., Lakin, M. R., Stefanovic, D., & Graves, S. W. (2014). Catalytic molecular logic devices by dnazyme displacement. *ChemBioChem*, *15*(7), 950–954.

Carmean, D., Ceze, L., Seelig, G., Stewart, K., Strauss, K., & Willsey, M. (2018). Dna data storage and hybrid molecular–electronic computing. *Proceedings of the IEEE*, *107*(1), 63–72.

Chang, M., Yang, C.-S., & Huang, D.-M. (2011). Aptamer-conjugated DNA icosahedral nanoparticles as a carrier of doxorubicin for cancer therapy. *ACS nano*, *5*(8), 6156–6163.

Chen, J., Deaton, R., & Wang, Y.-Z. (2003). A DNA-based memory with in vitro learning and associative recall. In *Dna* (pp. 145–156). Springer.

Cherry, K. M., & Qian, L. (2018). Scaling up molecular pattern recognition with dna-based winner-take-all neural networks. *Nature*, *559*(7714), 370.

Church, G. M., Gao, Y., & Kosuri, S. (2012). Next-generation digital information storage in dna. *Science*, *337*(6102), 1628–1628.

Deaton, R., Murphy, R. C., Rose, J. A., Garzon, M., Franceschetti, D. R., & Stevens, S. (1997). A dna based implementation of an evolutionary search for good encodings for dna computation. In *Proceedings of 1997 ieee international conference on evolutionary computation (icec'97)* (pp. 267–271). IEEE.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

Douglas, S. M., Bachelet, I., & Church, G. M. (2012). A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, *335*(6070), 831–834.

Farzadfard, F., & Lu, T. K. (2018). Emerging applications for dna writers and molecular recorders. *Science*, *361*(6405), 870–875.

Fu, T., Lyu, Y., Liu, H., Peng, R., Zhang, X., Ye, M., & Tan, W. (2018). Dna-based dynamic reaction networks. *Trends in biochemical sciences*.

Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E. M., Sipos, B., & Birney, E. (2013). Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*, *494*(7435), 77.

Greengard, S. (2017). Cracking the code on dna storage. *Commun. ACM*, *60*(7), 16–18.

Hamaguchi, K., & Geiduschek, E. P. (1962). The effect of electrolytes on the stability of the deoxyribonucleate helix. *Journal of the American Chemical Society*, *84*(8), 1329–1338.

Heckel, R., Shomorony, I., Ramchandran, K., & David, N. (2017). Fundamental limits of dna storage systems. In *2017 ieee international symposium on information theory (isit)* (pp. 3130–3134). IEEE.

Heo, M.-O., Lee, S.-W., Lee, J., & Zhang, B.-T. (2013). Learning global-to-local discrete components with nonparametric bayesian feature construction. *NIPS workshop on Constructive Machine Learning*.

Huang, Y., & He, L. (2011). Dna computing research progress and application. In *2011 6th international conference on computer science & education (iccse)* (pp. 232–235). IEEE.

Jones, M. R., Seeman, N. C., & Mirkin, C. A. (2015). Programmable materials and the nature of the dna bond. *Science*, *347*(6224), 1260901.

Junnarkar, S. (n.d.). In just a few drops, a breakthrough in computing. *The New York Times*. Retrieved from https://www.nytimes.com/library/cyber/week/052197dna.html

Kaplan, P., Cecchi, G., & Libchaber, A. (1999). Dna-based molecular computation: Template-template interactions in pcr. *DNA Based Computers II*, *44*, 97–104.

Katz, E., & Privman, V. (2010). Enzyme-based logic systems for information processing. *Chemical Society Reviews*, *39*(5), 1835–1857.

Kick, A., Bönsch, M., & Mertig, M. (2012). Egnas: An exhaustive DNA sequence design algorithm. *BMC bioinformatics*, *13*(1), 138.

Kim, J., & Winfree, E. (2011). Synthetic in vitro transcriptional oscillators. *Molecular systems biology*, *7*(1), 465.

Kumar, S. N. (2015). A proper approach on dna based computer. *American Journal of Nanomaterials*, *3*(1), 1–14.

Lakin, M. R., & Stefanovic, D. (2016). Supervised learning in adaptive DNA strand displacement networks. *ACS Synthetic Biology*, *5*(8), 885–897.

Lakin, M., Minnich, A., Lane, T., & Stefanovic, D. (2012). Towards a biomolecular learning machine. *Unconventional computation and natural computation*, 152–163.

LeCun, Y. (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, J.-H., Lee, B., Kim, J. S., Deaton, R., & Zhang, B.-T. (2011). A molecular evolutionary algorithm for learning hypernetworks on simulated DNA computers. In *Evolutionary computation (cec), 2011 ieee congress on* (pp. 2735–2742). IEEE.

Lee, J.-H., Lee, S. H., Baek, C., Chun, H., Ryu, J.-h., Kim, J.-W., ... Zhang, B.-T. (2017). *in vitro* molecular machine learning algorithm via symmetric internal loops of DNA. *Biosystems*, *158*, 1–9.

Lim, H.-W., Lee, S. H., Yang, K.-A., Lee, J. Y., Yoo, S.-I., Park, T. H., & Zhang, B.-T. (2010). *In vitro* molecular pattern classification via DNA-based weighted-sum operation. *Biosystems*, *100*(1), 1–7.

Lim, H.-W., Yun, J.-E., Jang, H.-M., Chai, Y.-G., Yoo, S.-I., & Zhang, B.-T. (2002). Version space learning with DNA molecules. In *International workshop on dna-based computers* (pp. 143–155). Springer.

Liu, C.-L., Nakashima, K., Sako, H., & Fujisawa, H. (2003). Handwritten digit recognition: Benchmarking of state-of-the-art techniques. *Pattern recognition*, *36*(10), 2271–2285.

Mao, C., LaBean, T. H., Reif, J. H., & Seeman, N. C. (2000). Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, *407*(6803), 493–496.

Mills Jr, A., Turberfield, M., Turberfield, A. J., Yurke, B., & Platzman, P. M. (2001). Experimental aspects of DNA neural network computation. *Soft Computing*, *5*(1), 10–18.

Montagne, K., Plasson, R., Sakai, Y., Fujii, T., & Rondelez, Y. (2011). Programming an in vitro dna oscillator using a molecular networking strategy. *Molecular systems biology*, *7*(1), 466.

Nguyen, D., & Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *1990 ijcnn international joint conference on neural networks* (pp. 21–26). IEEE.

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. In *Ijcai-99 workshop on machine learning for information filtering* (Vol. 1, pp. 61–67).

Ogihara, M., & Ray, A. (1997). Dna-based parallel computation by" counting". *DNA Based Computers*, *3*, 255–264.

Oza, N. C. (2005). Online bagging and boosting. In *Systems, man and cybernetics, 2005 ieee international conference on* (Vol. 3, pp. 2340–2345). Ieee.

Parker, J. (2003). Computing with dna. *EMBO reports*, *4*(1), 7–10.

Pei, R., Matamoros, E., Liu, M., Stefanovic, D., & Stojanovic, M. N. (2010). Training a molecular automaton to play a game. *Nature nanotechnology*, *5*(11), 773–777.

Peritz, A. E., Kierzek, R., Sugimoto, N., & Turner, D. H. (1991). Thermodynamic study of internal loops in oligoribonucleotides: Symmetric loops are more stable than asymmetric loops. *Biochemistry*, *30*(26), 6428–6436.

Powers, D. M. (2011). Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation.

Qian, J., Ferguson, T. M., Shinde, D. N., Ramırez-Borrero, A. J., Hintze, A., Adami, C., & Niemz, A. (2012). Sequence dependence of isothermal dna amplification via expar. *Nucleic acids research*, *40*(11), e87–e87.

Qian, L., Winfree, E., & Bruck, J. (2011). Neural network computation with DNA strand displacement cascades. *Nature*, *475*(7356), 368–372.

Reif, J. H. (1995). Parallel molecular computation. In *Proceedings of the seventh annual acm symposium on parallel algorithms and architectures* (pp. 213–223). ACM.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton project para*. Cornell Aeronautical Laboratory.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A modern approach*. Malaysia; Pearson Education Limited,

Sakellariou, J., Tria, F., Loreto, V., & Pachet, F. (2015). Maximum entropy model for melodic patterns. In *Icml workshop on constructive machine learning*.

Salehi, S. A., Liu, X., Riedel, M. D., & Parhi, K. K. (2018). Computing mathematical functions using dna via fractional coding. *Scientific reports*, *8*(1), 8312.

Sambrook, J., Fritsch, E. F., Maniatis, T., et al. (1989). *Molecular cloning: A laboratory manual.* Cold spring harbor laboratory press.

Sammut, C., & Webb, G. I. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

Scholkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press.

Seelig, G., Soloveichik, D., Zhang, D. Y., & Winfree, E. (2006). Enzyme-free nucleic acid logic circuits. *science*, *314*(5805), 1585–1588.

Seeman, N. C. (2003). Dna in a material world. *Nature*, *421*(6921), 427.

Seeman, N., Wang, H., Liu, B., Qi, J., Li, X., Yang, X., ... Wang, Y., et al. (1998). The perils of polynucleotides: The experimental gap between the design and assembly of unusual dna structures. In *Dna based computers ii*. American Mathematical Society.

Selvam, M. A., & Suganya, S. (2014). Recent advancements in dna computing. *International Journal of Computing Algorithm*, *3*(3), 269–270.

El-Seoud, S. A., Mohamed, R., & Ghoneimy, S. (2017). Dna computing: Challenges and application. *International Journal of Interactive Mobile Technologies*, *11*(2).

Sheth, R. U., & Wang, H. H. (2018). Dna-based memory devices for recording cellular events. *Nature Reviews Genetics*, *19*(11), 718–732.

Song, T., Garg, S., Mokhtar, R., Bui, H., & Reif, J. (2017). Design and analysis of compact dna strand displacement circuits for analog computation using autocatalytic amplifiers. *ACS synthetic biology*, *7*(1), 46–53.

Song, X., & Reif, J. (2019). Nucleic acid databases and molecular-scale computing. *ACS nano*, *13*(6), 6256–6268.

Steele, G., & Stojkovic, V. (2004). Agent-oriented approach to dna computing. In *Proceedings. 2004 ieee computational systems bioinformatics conference, 2004. csb 2004.* (pp. 546–551). IEEE.

Stojanovic, M. N., & Stefanovic, D. (2003). A deoxyribozyme-based molecular automaton. *Nature biotechnology*, *21*(9), 1069–1074.

Stryer, L. (1995). Biochemistry.

Vogelstein, B., & Gillespie, D. (1979). Preparative and analytical purification of dna from agarose. *Proceedings of the National Academy of Sciences*, *76*(2), 615–619.

Weiner, A. M., & Watson, J. (1987). Molecular biology of the gene. The Benjamin/Cummings Publishing Co. Inc., Menlo Park, California.

Yazdi, S. H. T., Yuan, Y., Ma, J., Zhao, H., & Milenkovic, O. (2015). A rewritable, random-access dna-based storage system. *Scientific reports*, *5*, 14138.

Yurke, B., Turberfield, A. J., Mills, A. P., Simmel, F. C., & Neumann, J. L. (2000). A DNA-fuelled molecular machine made of dna. *Nature*, *406*(6796), 605–608.

Zacharias, M., & Hagerman, P. J. (1996). The influence of symmetric internal loops on the flexibility of RNA. *Journal of molecular biology*, *257*(2), 276–289.

Zeng, Y., & Zocchi, G. (2006). Mismatches and bubbles in DNA. *Biophysical journal*, *90*(12), 4522–4529.

Zhang, B.-T. (2008). Hypernetworks: A molecular evolutionary architecture for cognitive learning and memory. *IEEE computational intelligence magazine*, *3*(3), 49–63.

Zhang, B.-T., Ha, J.-W., & Kang, M. (2012). Sparse population code models of word learning in concept drift. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 34, *34*).

Zhang, B.-T., & Jang, H.-Y. (2005). Molecular programming: Evolving genetic programs in a test tube. In *Proceedings of the 7th annual conference on genetic and evolutionary computation* (pp. 1761–1768). ACM.

Zhang, B.-T., & Kim, J.-K. (2006). DNA hypernetworks for information storage and retrieval. *Lecture Notes in Computer Science*, *4287*, 298.

Zhang, Q., Jiang, Q., Li, N., Dai, L., Liu, Q., Song, L., ... Ding, B., et al. (2014). DNA origami as an *in vivo* drug delivery vehicle for cancer therapy. *Acs Nano*, *8*(7), 6633–6643.

Zhirnov, V., Zadegan, R. M., Sandhu, G. S., Church, G. M., & Hughes, W. L. (2016). Nucleic acid memory. *Nature materials*, *15*(4), 366.

Zhou, G., Sohn, K., & Lee, H. (2012). Online incremental feature learning with denoising autoencoders. In *Artificial intelligence and statistics* (pp. 1453–1461).

초 록

DNA 컴퓨팅은 분자 수준의 생물학적 기질을 기반으로 하는 새로운 컴퓨팅 방식을 제시한다. 그리고 최근 해당 연구 분야에서는 종래의 컴퓨팅 방식의 한계를 극복하는 다양한 결과를 보고하고 있다. 하지만 보고되고 있는 대부분의 방법은 변화하는 환경에 적응이 불가능한 사전 프로그램 정의 방법 또는 규칙 기반의 방법으로 실효성과 확장성 측면에 의문이 제기되고 있다. 따라서 본 논문에서는 학습의 개념을 이용하여 확장성 및 적응성을 갖는 in vitro의 새로운 분자 알고리즘을 제안한다.

제안하는 방법은 습식의 분자 시스템이 훈련 데이터를 이용하여 분자 모델을 학습시키는 방식으로, 기계학습 알고리즘 기반의 하이퍼네트워크 모델을 기반으로 설계되었다. 본 논문에서 제안하는 새로운 접근 방식은 대량의 병렬 처리가 가능한 DNA의 특성과 효소의 특정 구조적 선택 방식의 결합을 통한 방식으로, 기존의 시간 소모가 큰 in silico 컴퓨팅 문제를 해결하고 있으며, 또한 반복이 가능한 분자 학습 방식을 채택함으로써 변화하는 환경에도 적응이 가능한 특징을 지니고 있다.

해당 모델의 학습 알고리즘은 DNA의 특성을 고려한 앙상블 방식을 채택하여 각 효소의 가중치를 업데이트하는 내부 루프 구조로 구성되어있으며, 본 논문에서는 해당 구조를 최대한 활용할 수 있는 방법의 설계 그리고 이에

대한 실험적 증명 과정을 포함하고 있다.

해당 내용으로는 실험 튜브에서의 분자 알고리즘 구현 방법에서부터, 해당 방법의 최적 구현을 위한 DNA 데이터 셋 구성 방법 그리고 넓은 탐색 공간의 효율적인 탐색을 통한 최적의 학습이 가능하게 하는 특징점 선정 방법, 효소를 이용한 학습 가중치 갱신 방법을 서술하고 있으며, 마지막으로 순차적으로 in vitro에서 다섯 번의 반복 학습으로 통해 도출된 결과를 수록하고 있다.

이를 통해 우리가 본 논문에서 제시하는 방법이 차세대의 지능형 분자시스템을 위한 새로운 DNA 컴퓨팅 방식의 기초 연구가 될 수 있음을 기대하며, 분자 컴퓨팅 기술 개발에 한 걸음 더 다가갈 수 있는 가능성을 확인한다.

주요어: DNA 컴퓨팅, 분자컴퓨팅, 패턴인식, 분자적기계학습

학번: 2014-31418