Content-based Similarity Search in DNA Data Storage Systems

Callista Lavender Bee

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Luis Ceze, Chair

Karin Strauss

Georg Seelig

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Content-based Similarity Search in DNA Data Storage Systems

Callista Lavender Bee

Chair of the Supervisory Committee:
Professor Luis Ceze
Computer Science & Engineering

As global demand for digital storage capacity grows, storage technologies based on synthetic DNA have emerged as a dense and durable alternative to traditional media. Existing approaches leverage robust error correcting codes and precise molecular mechanisms to reliably retrieve specific files from large databases. Typically, files are retrieved using a pre-specified key, analogous to a filename. However, these approaches lack the ability to perform more complex computations over the stored data, such as content-based search.

Here, we demonstrate the design, implementation, and evaluation of techniques for executing similarity search in DNA-based databases. By using machine learning to build a predictor of DNA hybridization reactions, we are able to create an encoding from images to DNA sequences that is optimized for similarity search. With this encoding, an encoded query image is most likely to hybridize with targets that are encoded from images visually similar to the query. This allows a query molecule to act as a molecular filter, which can select relevant results from a large database.

We perform wetlab experiments with a database of 1.6 million images encoded and synthesized as DNA molecules, and show that our technique produces results which are comparable to those of state-of-the-art electronic implementations of similarity search. By demonstrating that DNA-based systems are capable of both storage and computation, we believe this work will encourage further development of this emerging technology.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

Dear reader, you may have noticed lately that the world is a tumultuous place. As a queer, neurodiverse transgender woman, I have experienced my share of that tumult. I yearn for a world that is more kind, more open, and more loving.

I am also a curious, nomadic person. I flit between interests and rarely stay in one place for too long. At some point on my journey, I became curious about computers. Could humans use these fascinating mathematical objects, these powerful machines, to aid in creating that kinder world?

The ways in which I have been proven wrong have been heartbreaking. Computers are used by oppressors to facilitate mass surveillance. They are used by profiteers to "solve" problems that nobody asked them to. Computers are used as dangerous weapons.

I still believe that computers can be used as tools for justice. I believe that the difference between a weapon and a tool is a deft and caring hand. That does not come from computers, and it never will. It comes from ourselves and each other. It comes from listening, struggling, vulnerability, and growth.

I am passionate about justice, but this dissertation was not created from that. It was created from my fascination with computers, mathematics, chemistry, and life. It is a new piece of technology, and a new type of computer. It could be a tool of curiosity and discovery, or a weapon of surveillance and control.

Dear reader, whatever you make of this technology, please choose love.

ACKNOWLEDGMENTS

This work, and my growth as a researcher and human during the course of my PhD, would not have been possible without the personal and professional support of a huge number of friends and colleagues.

My co-authors on the main works that make up this dissertation deserve a huge share of credit. Yuan-Jyue Chen helped to guide the design of the experimental work, which was carried out in large part by him, David Ward and Xiaomeng Liu. My advisors, Luis Ceze, Karin Strauss, and Georg Seelig have all provided excellent guidance and advice during my career. I am indebted to Luis for taking me on back when I was a curious computer architecture student. His knack for bringing people and ideas together catalyzed this work. Karin's keen insights often cut to the core of difficult issues I was facing. Georg brought a wealth of experience in DNA computing, and a well-grounded perspective whenever I backed myself into a corner.

My colleagues in the Molecular Information Systems Lab have been inspiring and wonderful. Thank you to fellow students Katie Doroschak, Phil Leung, Johannes Linder, Aishwarya Mandyam, Lee Organick, Melissa Queen, Ashley Stephenson, and Max Willsey, and to researchers Rob Carlson, Sifang Chen, Jeff Nivala, Bichlien Nguyen, and Chris Takahashi. Special thanks to Lee and Melissa, who have been stellar friends during difficult times.

I have met many incredible people during my time at UW, but special thanks to my former officemates Eunice Jun and James Bornholt, and to all my former colleagues in the Sampa Lab for helping me get (and stay) on my feet during my first years at UW. Thank you to my friend Leah Perlmutter for being there for me during some critically important moments.

I would also like to thank the people at Portland State University that helped me start my journey as a researcher. My master's thesis advisor Professor Melanie Mitchell, as well as Professors Mark Jones, Bart Massey, Jonathan Walpole, Jim Hook, and Bryant York, all helped shape my intuition as a researcher and encouraged me to pursue a PhD. My fellow students there inspired me in ways that I am still grateful for today. Special thanks to Ted Cooper, Naomi Dickerson, Finn Ellis, Ben Hamlin, Sheng Lundquist, and Andy Wood.

It has been a long journey to this point, and there are so many more friends that I want to mention and thank by name. To my family members Virginia, Dot, and Luna, and to everyone who has been a part of my life these past years, big or small, online or in-person: thank you.

DEDICATION

To all survivors and searchers.

Chapter 1

INTRODUCTION

Interacting with digital documents is a staple of modern society. Since the dawn of online social media, the amount of text, images, and videos that the world produces has been growing rapidly in an exponential fashion, outpacing the capacity growth of traditional storage media, such as solid state, magnetic, and optical media. To address this gap and store this ever-expanding library of data for future generations, researchers are developing practical systems using synthetic DNA as a high density and durable storage medium [1–7], with several orders of magnitude higher density and durability than current storage technologies. An important feature of these systems is the ability to retrieve single documents without having to sequence a large, dense pool of data in molecular form. To accomplish this, these systems leverage the specificity of DNA hybridization to implement key-based retrieval, where a short, predetermined sequence is used to retrieve the document associated with that sequence [5, 6].

Although key-based retrieval might be sufficient for a well-maintained library or archive, modern search and recommendation systems do not assume users know the exact key of the document they are looking for, and thus make heavy use of content-based retrieval.

Using DNA hybridization for key-based retrieval was first proposed by Baum [8], and in the same paper he also proposed using it for content-based retrieval. However, his design generally assumed that the data being stored is uncompressed. More recent work has shown that robust and efficient encoding of arbitrary digital data in DNA requires techniques that apply pseudo-random processes to the document content [9], which would impede direct content-based search.

Electronic database systems separate the concerns of search and storage by maintaining an index structure that is designed to facilitate content-based retrieval. This dissertation work that principle: rather than focusing on storing and retrieving full files, we present designs for DNA-based indices that are optimized for facilitating content-based retrieval.

Specifically, we focus here on *similarity search*, where an example document is used to retrieve similar items from the database. Document similarity is typically formulated as a geometric problem [10] where each document is converted to a vector in a high-dimensional *feature space*, with the property that neighboring feature vectors represent subjectively similar documents. This conversion process is called *feature extraction*, and there are a variety of methods that work well, depending on the type of document and the goals of the application. For documents like images, the intermediate layer activations from neural networks trained on image classification problems (such as VGG16 [11] or AlexNet [12]) tend to perform well as feature vectors for similarity search tasks [13]. Figure 1.1A illustrates this with a two-dimensional t-SNE (t-distributed Stochastic Neighbor Embedding) projection of 4096-dimensional feature vectors extracted using the FC2 layer of VGG16.

Searching through such a high-dimensional space is challenging for electronic systems. The "curse of dimensionality" [10] means that exact indexing schemes in high-dimensional spaces are no better than a costly linear or "brute force" search, which is infeasible for large databases. Instead, efficient similarity search algorithms perform an approximate search that allows for errors in the results.

We adopt that relaxation for our DNA-based system as well. Rather than finding exact nearest neighbors, our goal is to maximize the number of near neighbors retrieved while minimizing the number of irrelevant results. To accomplish this, we want to encode feature vectors as DNA sequences such that single stranded molecules created from an encoded target and the reverse-complement of an encoded query are likely to form stable hybridized structures when the query and target feature vectors are neighboring, but not when they are distant (Figure 1.1B).

Given such an encoding, we can construct an index where each document is associated

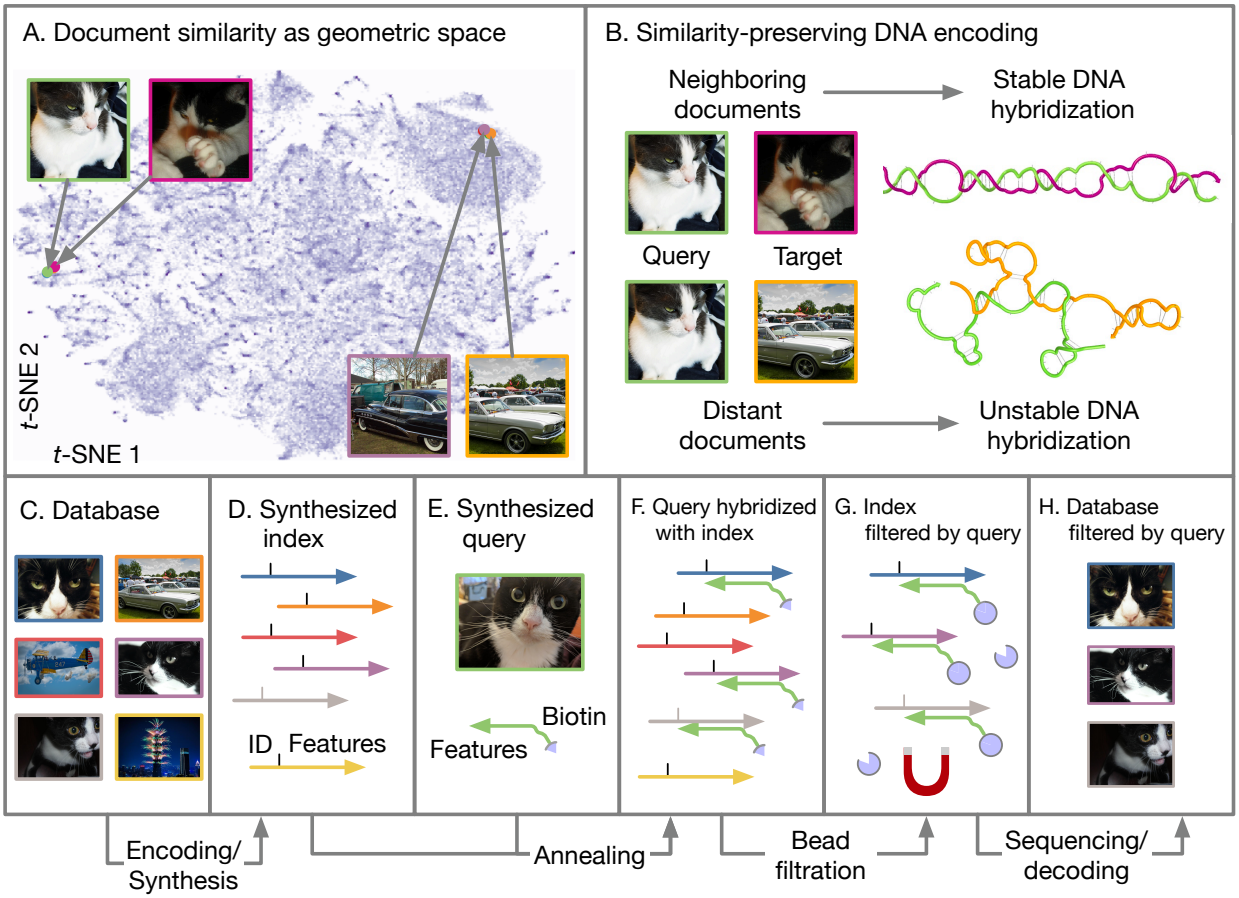


Figure 1.1: Overview of DNA-based similarity search. (A) Illustration of a feature space where neighboring documents are subjectively similar. (B) A similarity-preserving DNA encoding is one where the reverse complement of a query document's sequence hybridizes with a neighboring target document's sequence, but not with a distant target's. Note that the query is color-coded green and the targets with other colors. (C-H) The retrieval process. A database (C) is encoded and synthesized to produce an DNA-based index (D). Arrowheads indicate 3' ends of DNA. An encoded query (E) is annealed with a sample of the database (F), which is filtered with magnetic beads (G). The filtered database is sequenced to reveal the IDs of retrieved documents, which are used to look them up in the original database (H).

with a single strand of DNA that contains the document's ID alongside its encoded feature vector (Figure 1.1C-D). An encoded query (Figure 1.1E) can then be used as a hybridization probe to filter out similar documents from the index (Figure 1.1F-G). The filtered index can be sequenced and decoded to recover the IDs of documents that are similar to the query. These documents can then be retrieved from a key-based database and displayed to the user (Figure 1.1H).

A similar index construction was described by Reif et al. [14]. However, rather than leveraging inexact matching between target and query strands of DNA, their solution reduced the problem to exact key-based retrieval by pre-clustering the database and then assigning each cluster a unique DNA codeword. However, this quantized approach may cause queries to miss nearby neighbors that fall across a cluster boundary. We propose using inexact matching because it can provide a better guarantee that documents within a certain radius of the query will be retrieved.

Prior work by Tsaftaris et al. [15, 16] also proposed using inexact DNA hybridization for similarity search. However, their technique focused on one-dimensional data, and it is unclear how it would scale to encoding multidimensional vectors.

Chapter 2 describes a technique for implementing an encoding that translates multi-dimensional feature vectors into DNA sequences in a way that leverages the flexibility of inexact matching, and also presents experimental results from the construction of a small-scale prototype with 100 images. Chapter 3 presents a refined encoding, and a larger-scale implementation tested with 1.6 million images. Chapter 4 concludes with thoughts about the scalability of this technique and potential directions for future work.

Chapter 2

A SMALL-SCALE PROTOTYPE[1]

2.1 Introduction

To investigate the design proposed in Figure 1.1, we first designed a prototype that could be constructed given certain constraints on the length and number of sequences we could synthesize.

This prototype contributes two advances to the field of DNA storage: first, a strand design optimized for associative search. Second, a sequence encoder capable of preserving similarity between documents, such that a query sequence generated from a given document will retrieve similar documents from the database. We validate our designs with wetlab experiments.

While our methods should generalize to databases comprising any type of media, we focus on images in this work, as there is a rich body of prior work in content-based image retrieval to draw on.

The rest of this chapter is laid out as follows: Section 2 covers background on similarity search and DNA-based parallel search. Section 3 details our strand designs. Section 4 describes our methodology for mapping images to DNA sequences. Section 5 outlines our experimental protocol and the results of our experiments. Section 6 discusses the results and proposes future work. Section 7 addresses related work, and Section 8 concludes the chapter.

---

[1]This chapter was originally published as "A Content-Addressable DNA Database with Learned Sequence Encodings" in *DNA Computing and Molecular Programming*, 2018.

2.2 Background

2.2.1 Similarity Search

The problem of *similarity search* is to retrieve documents from a database that are similar in *content* to a given query. For media such as text, images and video, this can be a difficult task. Most state-of-the-art systems convert each document into a vector-space representation using either a hand-crafted embedding, or one learned via a neural network. These *feature vectors* can then be compared with metrics like Euclidean distance, where similar documents will tend to be close together in feature-space. Therefore, a similarity search can be reduced to a k-nearest-neighbor or R-near-neighbor search.

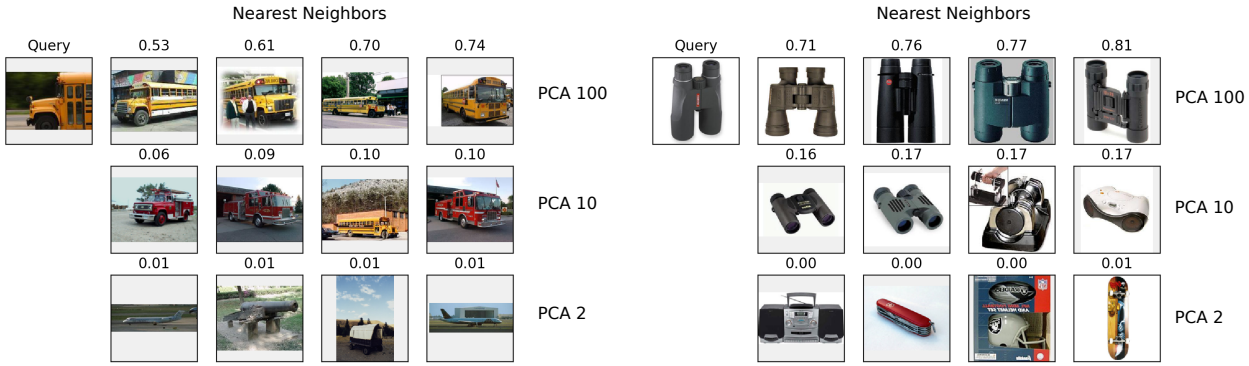


Figure 2.1: A pair of sample queries from the Caltech-256 dataset, showing the four nearest neighbors in three different feature spaces. Each neighbor is annotated with its Euclidean distance to the query in that space.

Feature vectors that are effective for similarity search tend to be high dimensional. To illustrate this, Figure 2.1 shows two queries using the Caltech-256 image dataset [17]. The visual features of each image in the dataset were extracted using VGG16, a publicly available convolutional neural network trained on an image classification task. We used the 4096-dimensional activations from the FC2 layer, as intermediate layers in deep neural networks have shown to be effective in content-based image retrieval tasks [13]. These features were reduced down to 100, 10, and 2 dimensions using principal component analysis (PCA). The

nearest neighbors in each of these subspaces (with respect to Euclidean distance) are shown to the right of each query. Qualitatively, the nearest neighbors higher-dimensional spaces appear more similar to the query than the nearest neighbors in lower-dimensional spaces.

When feature vectors have hundreds of dimensions, the well-known "curse of dimensionality" defeats efficient indexing schemes [10]. In the worst case, every item in the database must be examined to find all images within a certain distance threshold. Relaxations of the search problem that allow for errors or omissions result in much faster lookups, using algorithms such as locality-sensitive hashing (LSH) [18].

Looking toward a future where zettabytes of data are generated every year [19], even techniques such as LSH that reduce the amount of data that needs to be inspected by orders of magnitude will still burden traditional storage with a tremendous number of IO requests to a massive storage infrastructure, outstripping the time and energy cost of the feature vector distance computation itself.

Computer architects have noticed that the power required to move data from the storage device to the compute unit can be reduced by moving the compute substrate closer to the storage substrate. This class of techniques is broadly called "near-data" processing [20].

2.2.2 DNA-based Parallel Search

"Adleman-style" DNA computing [21] can be thought of as an extreme version of near-data processing: each DNA strand is designed to both store and process information — the compute and storage substrates are the same.

Like Adleman's original solution to the Hamiltonian Path problem, this style of parallel processing requires exponential amounts of DNA to solve combinatorial problems. However, for less computationally intense problems like similarity search, the amount of DNA required is much less: if each of N items in the database is mapped to a single "target" molecule, then N identical copies of a "query" molecule are sufficient to react with every item in the database. If the query is equipped with a biotin tail and designed to hybridize only with relevant data, then relevant items can be "fished out" of the database using streptavidin-

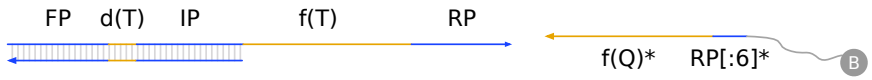
coated magnetic beads.

This amounts to an extremely high-bandwidth parallel search, in the vein of near-data processing techniques. Furthermore, because PCR can make exponentially many copies of the query molecule, the amount of DNA that needs to be directly synthesized is minimal. This makes DNA-based search especially appealing in the zettabyte-yottabyte future.

2.3 Database Design

To take advantage of the near-data processing capabilities of DNA, we need a database design that allows each element in the database to both store and process data. We choose to separate these two concerns by associating each database element with two sequences: one that stores an ID unique to that datum, and one that is generated from the semantic features of that datum, designed as a locus for a hybridization probe. The ID is not an "active" site, but rather the information to be retrieved by the search — for instance, it could be the address of the datum in another database that stores the document's complete data.

The simplest way to retain the association between the ID sequence and the feature sequence in a DNA database is to place them on the same strand of DNA. However, this association can cause unwanted secondary structures on longer strands, and can result in cross-talk if a query reacts with a potential target's ID sequence instead of its feature sequence.



(a) Database Element Design (b) Query Design

Figure 2.2: Strand Designs. Blue indicates a conserved region, orange indicates a region specific to that data item. Arrow indicates the 3' end. Star (*) indicates reverse complement. RP[:6] indicates the first six bases of domain RP.

Our strand designs address this issue, and are shown in Figure 2.2. The database entries

(Figure 2.2a) are synthesized single-stranded, but are made partially double stranded using a single-step PCR reaction starting from IP (the "internal primer"), which is conserved across all elements in the database.

This process covers up the IP region, the ID sequence associated with the data ($d(T)$), and the forward primer (FP) region, which is another conserved region used to prepare samples for sequencing. This leaves the feature sequence ($f(T)$) and the conserved reverse sequencing primer (RP) available to interact with the query.

To execute a query Q, a biotinylated query strand (Figure 2.2b) is mixed with the prepared targets. Because the query and target feature sequences are allowed to be imperfect matches, the query strand also includes the reverse complement of first six bases of RP (denoted RP[:6]*) — this exact match is designed to prevent misalignments and ensure that hybridization only depends on the interaction between $f(T)$ and $f(Q)$. The query and targets are annealed, and then streptavidin-coated magnetic beads are added to pull down targets that have hybridized with the queries.

The resulting filtered targets are amplified using FP and RP, then sequenced to retrieve the data region associated with each target.

2.4 Learned Sequence Encodings

To take advantage of the strand designs described above, we need to design a mapping from images to feature domains such that a query molecule will retrieve relevant targets from the database. To simplify our task, we pre-process all images by transforming them into the 10-dimensional subspace shown in Figure 2.1, and choose our feature domains to be 30 nucleotides in length.

Our general feature encoding strategy is inspired by semantic hashing [22], where a deep neural network transforms an input feature space into an output address space where similar items are "close" together. Our goal is to design a neural network sequence encoder that takes the 10-dimensional image feature vectors from the VGG16+PCA extraction process described in Section 2.2.1, and outputs DNA sequences that are close together if and only if

the feature vectors are close together. Following Tsaftaris et al. [16], we define a pair of query and target sequences as "close" if their hybridization reaction has a high thermodynamic yield: the proportion of target molecules that are converted into a query-target duplex.

To train the neural network, we want a loss function that will push the encoder's parameters to generate output sequences where a query retrieves a target if and only if the target and query represent similar images. The most appropriate choice for this is the cross-entropy loss[2], where the labels are binary similarity labels (similar vs. not similar) for each pair of query and target images, and the retrieval probabilities are the thermodynamic yields of each query-target hybridization reaction.

Using the cross-entropy loss requires us to define a binary notion of image similarity, and to define thermodynamic yield as a differentiable function of two DNA sequences. The function must be differentiable because neural networks are efficiently trained using gradient descent, which requires taking the derivative of the loss with respect to the encoder parameters.

In the sections below, we present a definition of binary image similarity, followed by an approximation for thermodynamic yield using Hamming distance, and an approximation for Hamming distance using the cosine distance between "one-hot" encodings of DNA bases. Finally, we present the results of using these approximations to train a neural network on a large image dataset.

2.4.1 Binary Image Similarity

As described in Section 2.2.1, a semantic notion of image "similarity" can be mapped to a real-valued number by computing the Euclidean distance between two image feature vectors. However, to use the efficient cross-entropy loss function defined above, we must label image

[2]Given a set of n pairs of binary labels $y \in \{0, 1\}$ and retrieval probabilities p, the cross-entropy loss is:

$$l(y, p) = -\frac{1}{n} \sum_{i=1}^{n} y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)$$

pairs with a binary label: "similar" or "not similar". The simplest way to do this is to apply a threshold to the Euclidean distance.

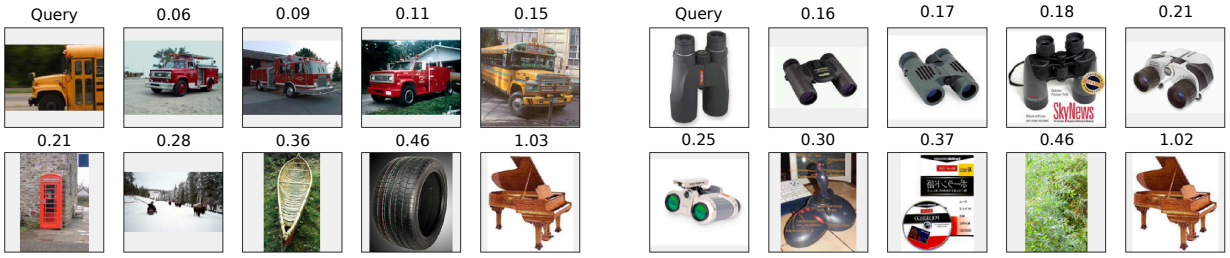


Figure 2.3: Sample queries demonstrating the relationship between image similarity and distance in the 10-dimensional PCA subspace shown in Figure 2.1. Distances less than 0.2 **usually correspond to similar images, while those greater than** 0.2 **do not.**

Because the definition of similarity is ultimately up to a human observer, we must determine this threshold by inspection. For the feature extraction method we used, we found a threshold of 0.2 to be fairly reliable across the Caltech-256 dataset. Figure 2.3 demonstrates this for a pair of sample queries.

2.4.2 Approximating Thermodynamic Yield

Thermodynamic yield can be calculated accurately by using the multi-stranded partition function [23], which is used by tools such as NUPACK [24]. Unfortunately, this calculation is expensive and not differentiable, and thus cannot be used directly to train a neural network.

However, Figure 2.4 shows that the query-target yield and the query-target Hamming distance have a noisy sigmoid[3] relationship. The best fit line provides us with a simple approximation of thermodynamic yield in terms of the Hamming distance. A drawback is that this approximation is less accurate for higher Hamming distances.

---

[3]Functions of the type:

$$f(x) = \frac{1}{1 + \exp(ax - b)}$$

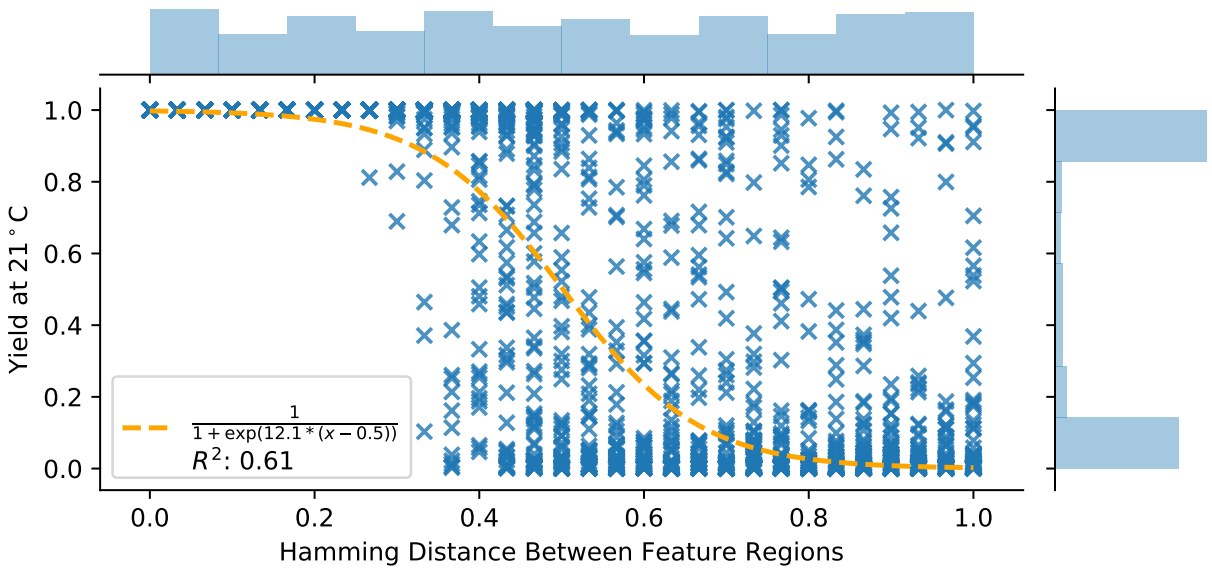


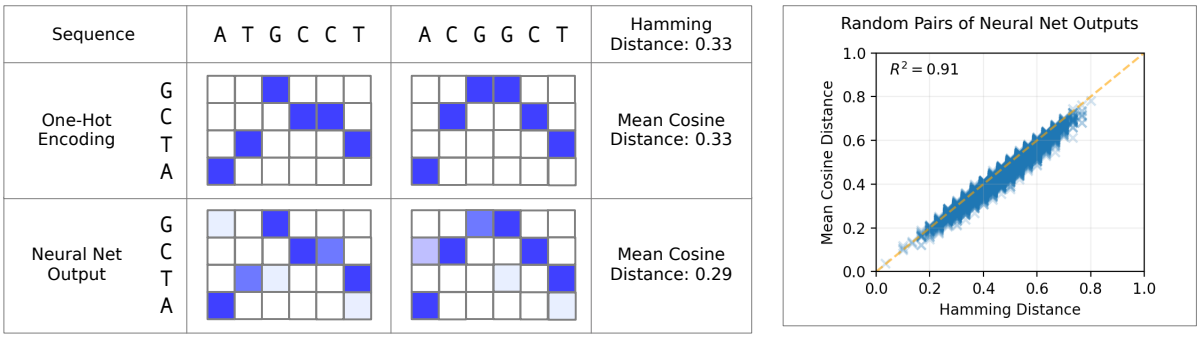
Figure 2.4: Yield vs. Hamming distance for 2000 pairs of targets and queries with feature regions of length 30, as calculated by NUPACK. The dashed line shows the best sigmoid fit to the simulations.

2.4.3 Approximating Hamming Distance

While we can use the Hamming distance to approximate thermodynamic yield, computing the Hamming distance requires discrete operations and is also not differentiable. Below, we define an alternative representation of DNA sequences, and a continuous approximation of Hamming distance that can be used with a neural network.

DNA sequences can be represented with a "one-hot" encoding, where each position is represented by a four-channel vector, and each channel corresponds to a base. For instance, if that base is an `A`, then the channel corresponding to `A` will have a value of one, and the other channels will be zero.

Figure 2.5a shows one-hot encodings of two sequences. At each position, the one-hot



(a) Comparison of sequence representations and associated distance metrics.

(b) Effectiveness of neural network output.

Figure 2.5: One-hot sequence encodings and their properties.

encodings can be compared by computing the cosine distance [4] between them. If they represent different bases, the representations will be orthogonal, and the cosine distance will be one. If they represent the same base, the cosine distance will be zero. Therefore the mean cosine distance across positions will be equal to the mean number of mismatches, which is equivalent to the Hamming distance.

A neural network cannot output differentiable representations that are exactly one-hot, because this would require discretization. However, if the channel values at each position are sufficiently far apart, we can approximate a one-hot encoding by normalizing them with a softmax function[5], which pushes the maximum value towards one while pushing the other values towards zero. Furthermore, we can encourage the channel values to be far apart by using a hidden-layer activation function with a large output range, such as the rectified

[4]Given two vectors $\mathbf{u}$ and $\mathbf{v}$, the cosine distance is:

$$d(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\ ||\mathbf{v}||}$$

[5]Given an N-dimensional vector $\mathbf{u}$, the softmax function is defined element-wise as follows:

$$\text{softmax}(\mathbf{u})_i = \frac{e^{u_i}}{\sum_{j=1}^{N} e^{u_j}}$$

linear unit (ReLU) function[6].

Figure 2.5b shows the relationship between the mean cosine distance and Hamming distance of pairs of outputs, for 10,000 pairs of random inputs to a randomly initialized neural network with 10 input units, two ReLU hidden layers of 128 units each, and 30 four-channel softmax output units. The mean cosine distance between the neural network outputs closely follows the Hamming distance between their discretized counterparts, validating our approximation.

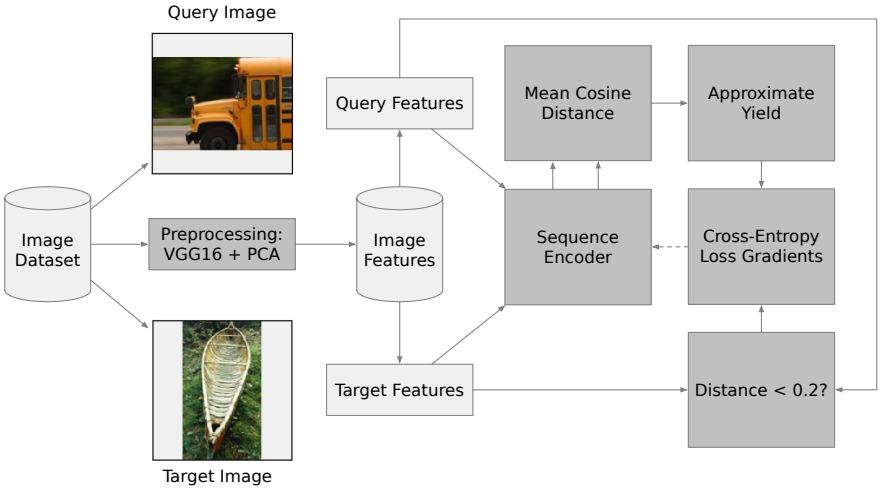


Figure 2.6: The training loop, illustrating how a pair of images is used to calculate gradients for the sequence encoder. Data is in light gray, and operations are in dark gray.

2.4.4 Neural Network Architecture

Composing the yield approximation with the Hamming distance approximation allows us to use gradient descent to train any kind of neural-network-based sequence encoder to generate good encodings for similarity search, given a suitable dataset. This process is depicted in

[6]The ReLU function is defined as:

$$\mathrm{ReLU}(x) = \max(x, 0)$$

Figure 2.6. On each iteration, a pair of images is encoded, and then the mean cosine distance between the outputs is used to calculate the approximate thermodynamic yield. Combined with the actual similarity between the feature vectors, the parameters of the neural network are updated using the gradient of the cross-entropy loss with respect to the parameters.

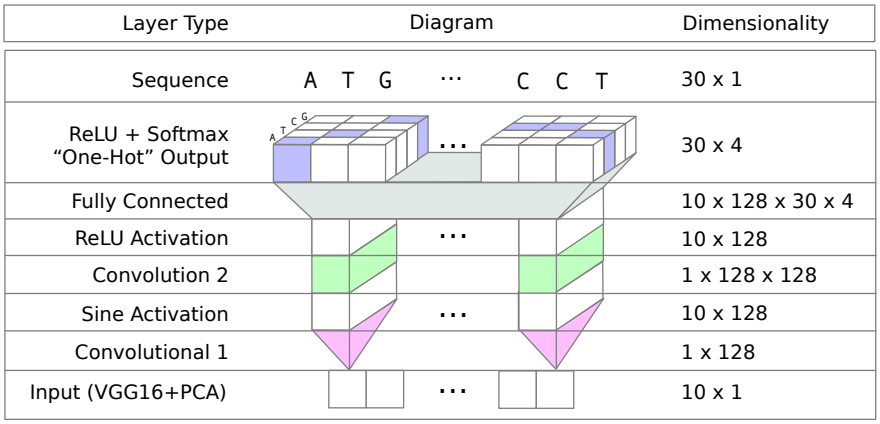


Figure 2.7: The neural network architecture for the sequence encoder.

A full exploration of the design space of neural-network-based sequence encoders is outside the scope of this work. We conducted a small-scale exploration and arrived at the architecture depicted in Figure 2.7, but this is not necessarily the best or only neural network for this task.

The network begins with two convolutional layers, where each input dimension is processed independently with a shared set of weights. This was done to preserve some of the "element-wise" structure of the Euclidean distance used to calculate the similarity label. The first convolutional layer has a sine-function activation, inspired by spectral hashing [25], a method for transforming an input feature space into a binary address space. The second convolutional layer uses the ReLU function to allow the outputs to be further apart.

Since the input dimensions do not have a spatial interpretation, we cap the convolutional layers with a set of fully connected weights to the four-channel sequence output, such that each input dimension's activation map is given a chance to influence each base in all positions. A ReLU activation followed by a softmax activation gives us the approximate one-hot

representation discussed above.

2.4.5 Training Results

To train the encoder, we first split the 30,607 images of the Caltech256 dataset into 24,485 training images and 6,122 test images. We extracted the VGG16 FC2 features from all 30,607 images, and then fitted a PCA transform to the FC2 vectors from the training set. The fitted transform was applied to all images.

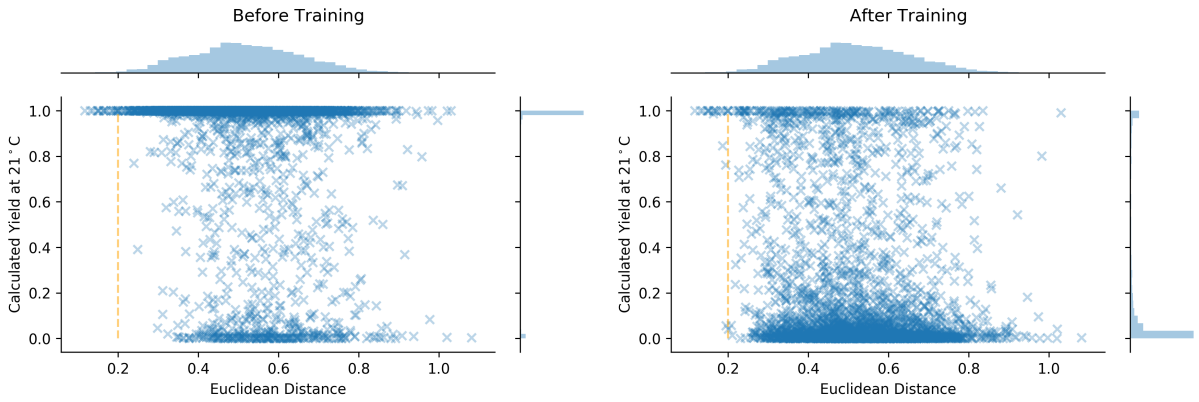


Figure 2.8: Encoder performance on 3000 pairs of images from the test set, before and after training. The x-axis is the Euclidean distance between the target and the query, and the y-axis is the thermodynamic yield (calculated with NUPACK). The orange line shows the similarity threshold of 0.2**.**

During each training iteration, a batch of random pairs of training set images was used to update the encoder weights, as depicted in Figure 2.6. The encoder was trained for 65,000 iterations using 500 random pairs of images per iteration. Figure 2.8 shows the performance of the encoder as measured by the relationship between the thermodynamic yield (calculated with NUPACK) and the Euclidean distance between the images in the pair. NUPACK was set to simulate our experimental setup, with an equal molar ratio of target to query strands, and temperature at 21°C.

The performance is shown before training (with random parameters), and again after training. Before training, nearly all pairs of images exhibit a high yield, indicating no

selectivity by distance. After training, most pairs have a low yield, but almost no pairs of images under 0.2 in Euclidean distance (which we have defined as similar) have a low yield. However, there are still non-similar images that have high yield, indicating that any successful query will also retrieve non-similar images.

2.5 Experiments

2.5.1 Dataset Construction

To test our designs in the wetlab, we constructed a subset of the test set consisting of 10 query images and 100 target images. The queries were chosen by first clustering all images in the training set into 10 groups using k-means, and then choosing a representative query image from the test set that belonged to each cluster. The k-means step ensures that none of the query images are pairwise-similar, because they all belong to different clusters in the data.

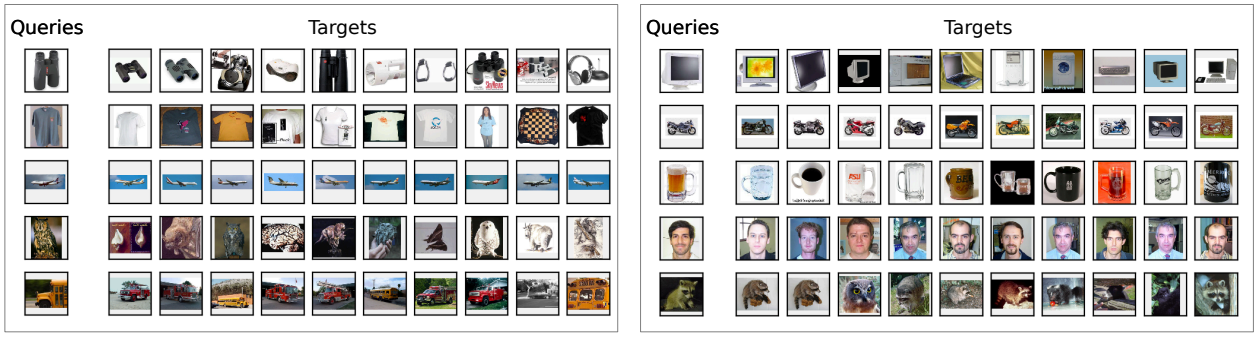


Figure 2.9: The set of query and target images used in our wetlab experiments.

For each of these 10 query images, we selected its 10 nearest neighbors in the test set. This ensures that each query image has 10 similar images and 90 dissimilar images among the 100 targets. The result of this selection process is shown in Figure 2.9.

For each image, we encoded its features as a 30-nucleotide DNA sequence using the trained encoder, as described in Section 2.4.5. For each target image, we assigned it a random 5-nt ID, and then constructed a 90-nt sequence as shown in Figure 2.2a. For each query image,

we constructed a 36-nt sequence as shown in Figure 2.2b. Target and query strands were then ordered from IDT. The query strands included the addition of a biotinylated spacer at the 5' end.

2.5.2 Target Preparation

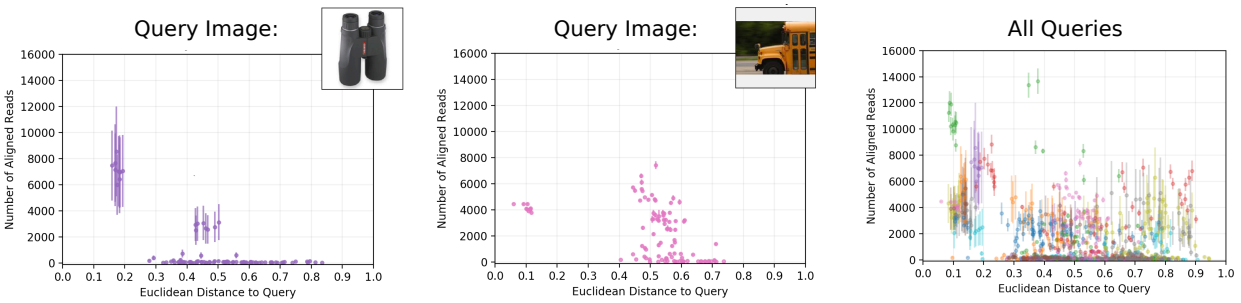
All target strands were mixed together in an equal molar ratio. The targets were then mixed with 20% excess of the primer IP* at 10μM, and 20μL of the target-primer mixture was added to 20μL of 2x KAPA HIFI PCR enzyme mix. This 40μL mixture was placed in a thermocycler with the following protocol: (1) 95°C for 3 minutes, (2) 98°C for 20 seconds, (3) 56°C for 20 seconds, (4) 72°C for 20 seconds, (5) go to step 2 one more time, and (6) 72°C for 30 seconds. This process extends the primer to cover the 5' half of each target strand.

2.5.3 Query Protocol

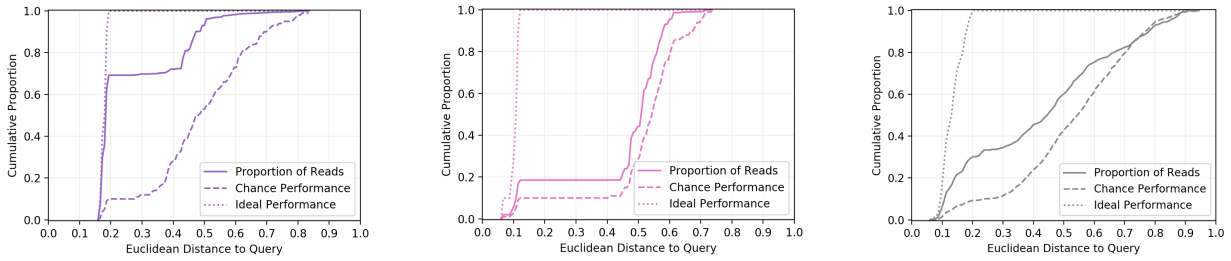
For each of the 10 query strands, a sample of the target mixture was diluted to 200 nM and mixed with an equal molar concentration of the query, then annealed in a thermocycler from 95°C to 21°C at a rate of 1°C per minute.

The annealed query-target mixture was mixed with streptavidin-coated magnetic beads, incubated at room temperature for 15 minutes, and placed on a magnetic rack. The supernatant containing non-captured DNA was removed and the beads were resuspended in elution buffer, then incubated for 5 min at 95°C and placed on a magnetic rack to separate captured DNA molecules from biotinylated query strands. The supernatant containing the captured DNA was mixed with the forward primer FP and the reverse primer RP* in a PCR reaction to amplify the captured targets. The amplified targets were ligated with Illumina sequencing adapters and then sequenced using an Illumina NextSeq.

This procedure was repeated 3 times for each of the 10 queries. Each query and replicate was given a unique sequencing index.



(a) Number of aligned reads per target vs. distance from target to query. Points indicate the mean across three replicates, and error bars indicate standard error. Different colors indicate different query images.



(b) Cumulative distribution of aligned reads as a function of increasing distance from target to query. For reference, the dashed lines show the cumulative distribution of targets by distance, and the dotted lines show an ideal where all reads are allocated to the nearest targets. Different colors indicate different query images (gray for all queries).

Figure 2.10: Selected results for two of the ten query images, and aggregated results for all queries.

2.5.4 Results

For each query and replicate, the reads were aligned with the set of all target sequences using BWA-MEM [26]. Figure 2.10a shows the number of aligned reads for each target versus the distance from that target to the query, for two sample queries, and for all queries together.

Figure 2.10b shows the cumulative distribution of aligned reads as a function of distance from the query. The dashed line is a baseline indicating the cumulative distribution of distances across the targets. The further the solid line is from the baseline, the stronger the relationship between distance and the number of reads. The dotted line shows the

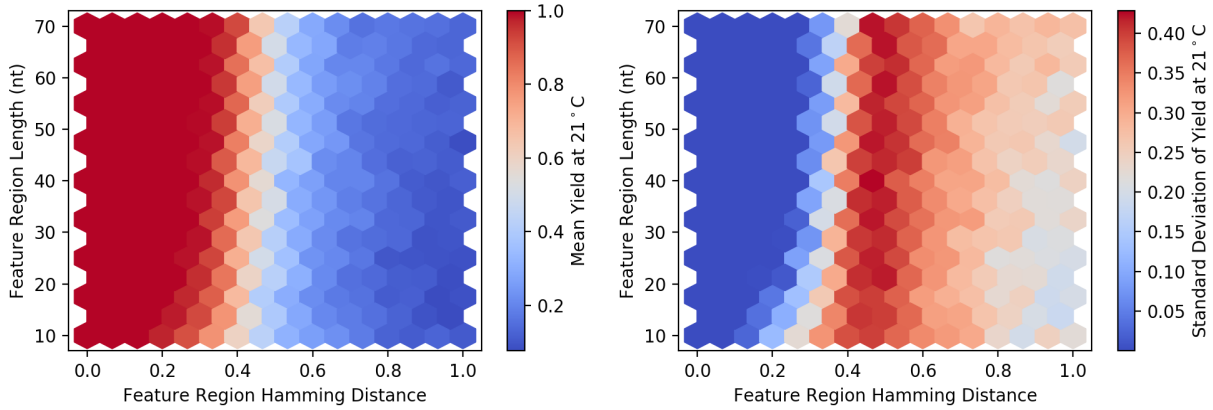


Figure 2.11: Mean and standard deviation of yield as a function of feature region length and feature region Hamming distance.

ideal result, where reads are only allocated to similar targets (those less than 0.2 Euclidean distance from the query).

The first sample query (the binoculars) shows a successful result, where most of the reads are allocated to similar targets. In contrast, the second sample query (the school bus) is less successful: the reads are distributed almost evenly across similar and non-similar images.

Across all queries, our results are moderately successful — though there are many reads going to dissimilar targets, our scheme is clearly capable of performing similarity-based enrichment: roughly 30% of the sequencing resources are being used by similar targets, which by construction make up just 10% of the database.

2.6 Discussion

In practice, the 10-dimensional image feature subspace used for our experiments is insufficiently selective. Referring back to Figure 2.1, the 100-dimensional space was more effective at relating distance to qualitative similarity. But it is difficult to train an encoder to transform this already-compressed 100-dimensional subspace into a 30-nucleotide feature sequence.

We might be tempted to try longer feature regions, but this will likely experience more

noise of the type seen in our results. Figure 2.11 illustrates this by generalizing Figure 2.4 to feature regions of different sizes. These plots bin across sequence length and target-query Hamming distance, and the color indicates either the mean (on the left) or the standard deviation (on the right) of the yield values in that bin, at our protocol temperature of 21°C. These plots tell us that selectivity decreases with increasing length, and that variance in yield for dissimilar targets increases as well.

These problems pose a difficult challenge to scaling this system. One avenue for future work is to devise a more accurate approximation for thermodynamic yield that can still be used to train a neural network. Another is to explore alternative probe designs that are meant to reduce variance, such as the toehold-exchange probes of Zhang et al. [27, 28].

2.7 Conclusion

We have presented a complete design, from encoding to sequencing, for a DNA database capable of performing content-based associative search by enriching database elements that are similar in content to a given query.

We have accomplished this by combining state-of-the-art research from the information retrieval and machine learning community with theoretical and experimental insights from the DNA computing and DNA storage communities to come up with novel encoding strategies and strand designs.

While it will be a challenge to scale this system to more complex features and larger datasets, this work is another step towards realizing the types of systems we will need to accommodate the storage demands of the future.

Chapter 3
A LARGE-SCALE IMPLEMENTATION

3.1 Introduction

Following the success of the prototype outlined in Chapter 2, we began to ask ourselves how far we could push the underlying methodology. Our goal was to scale-up the database to a much larger size (millions of images instead of hundreds). We hypothesized that a larger feature dimensionality would be required (instead of the PCA-reduced features used previously), and we re-designed the encoder to transform directly from the 4096-dimensional VGG16-FC2 space to an 80-nucleotide DNA sequence (instead of the 30-nucleotide sequences used previously).

Because we had found that the Hamming distance-based predictor became progressively less accurate with longer feature regions (see Fig 2.11), we also set out to design a more accurate hybridization predictor, which is one of the key components of our scaled-up implementation.

3.2 Encoder Design

3.2.1 Datasets and Feature Extraction

As in our prior work, we chose to focus on encoding feature vectors derived from images, as similarity between images is easy to visualize, and large datasets are readily available.

With the exception of the query images, all images were collected from Open Images V4 [29], a dataset of over 9 million URLs for images with Creative Commons licenses. Of these, approximately 1.7 million are hosted by the CVDF and available for download; the rest are raw Flickr URLs and may or may not be available. For the image database used in our experiments, we took 1.6 million images from the hosted set. For training, we took

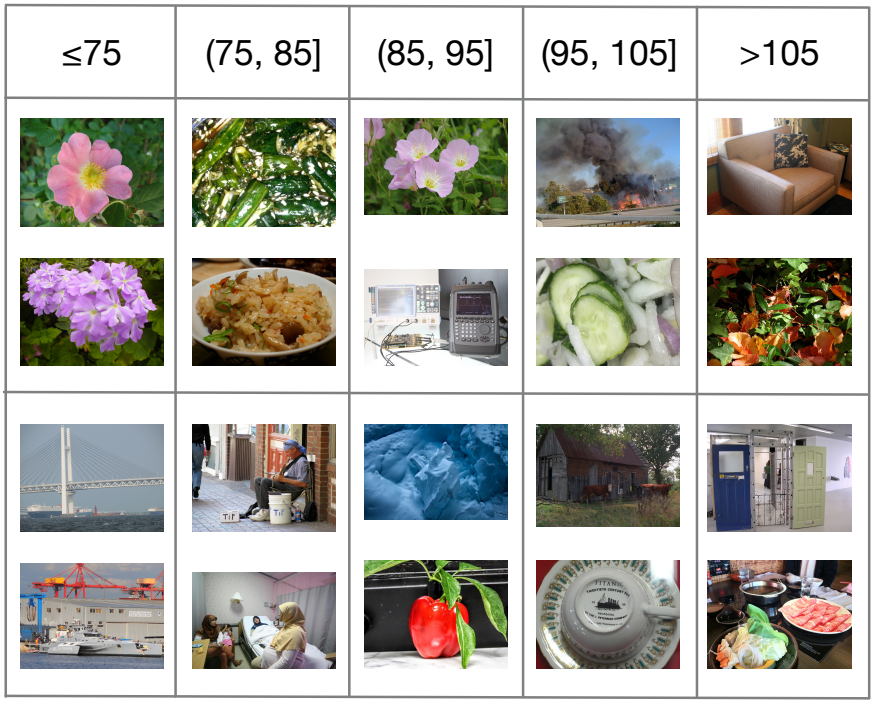


Figure 3.1: Illustration of the relationship between pairwise feature-vector Euclidean distance and pairwise subjective similarity. Each column represents a range of Euclidean distances, and each row depicts a pair of images where their feature-vector Euclidean distance falls in that range.

images from the full set of 9 million that were not used for training, testing, or experiments.

To extract image features, we processed each image with VGG16 [11], a convolutional neural network designed for image classification. The weights were loaded from the publicly available trained model and left unchanged during our processing. We used the activations of FC2 (the second fully-connected layer) as 4096-dimensional feature vectors.

As shown in Figure 3.1, pairs of images with a feature-vector Euclidean distance of 75 or less tend to be consistently similar. During training, we label these pairs as "similar" and all other pairs as "not similar".

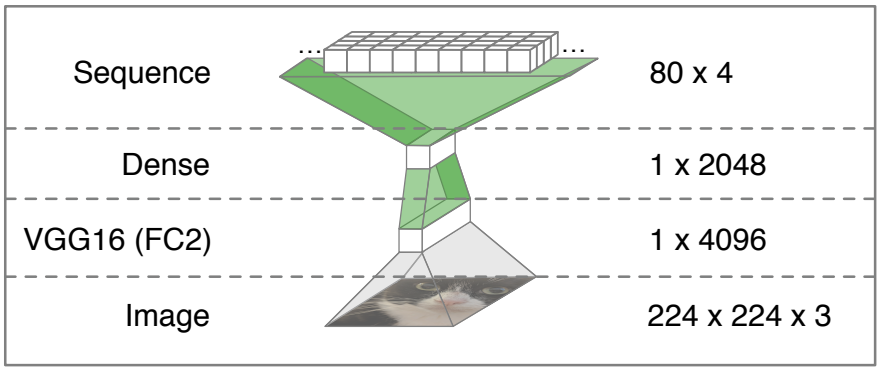


Figure 3.2: Structure of the sequence encoder network. Layers are opaque, and transformations are translucent. The dimensionality of each layer is shown on the right. Only the transformations highlighted in green have parameters that change during training.

3.2.2 Encoder Architecture

The sequence encoder is a fully-connected neural network. Its topology is depicted in Figure 3.2. The 4096-dimensional FC2 vectors are fed into a 2048-dimensional hidden layer with a rectified linear activation, followed by an output layer with a "one-hot" sequence representation that is 80 nucleotides in length. In this representation, each sequence position has four channels, one for each base. A softmax activation function is applied that forces each positions channels to sum to 1. A DNA sequence can be read off by picking the channel with the maximum activity at each position.

The yield predictor expects fully discrete one-hot sequences, but the encoder is capable of outputting a one-hot representation with indeterminate bases (for example, if all four channels at a position have a value of 0.25). Because of this, a regularization is applied during encoder training to minimize the entropy at each position. This encourages each position to have a well-defined maximum.

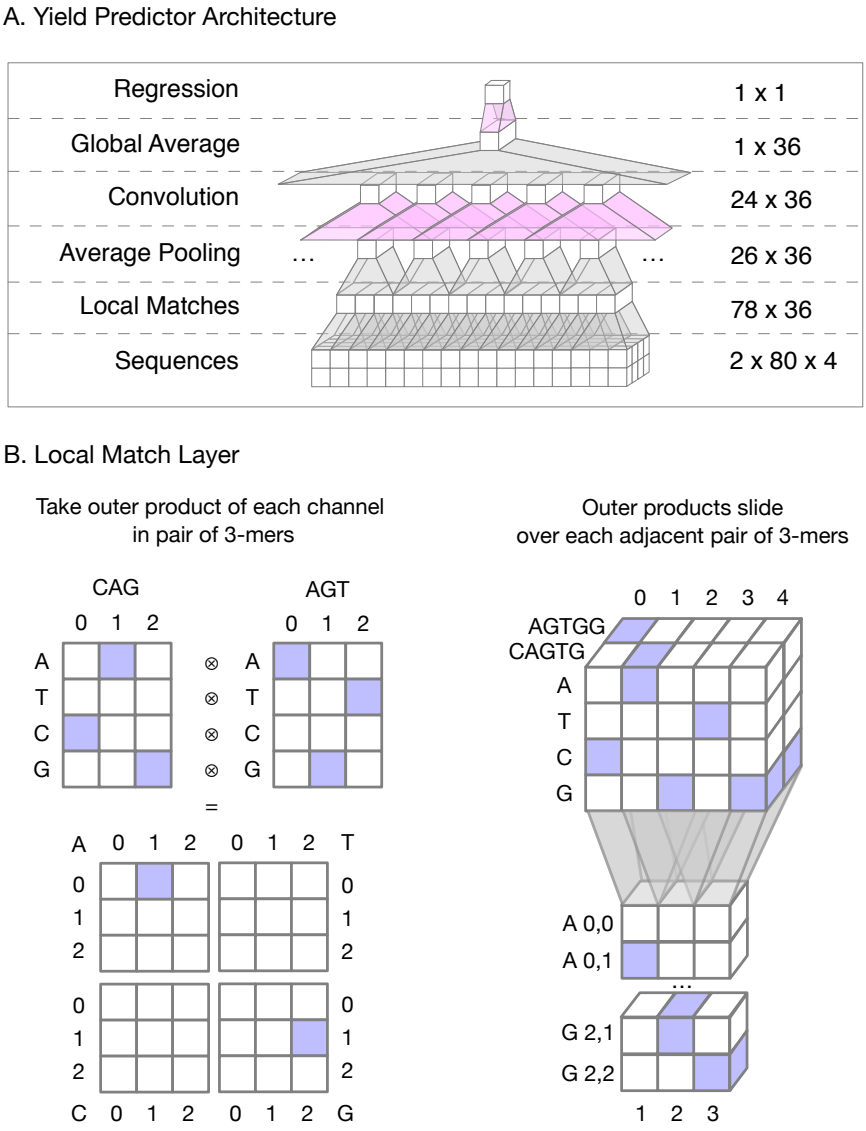


Figure 3.3: (A) Structure of the yield predictor network. Only the transformations highlighted in pink have parameters that change during training. (B) Illustration of the local match operation. Blue cells have a value of 1, and white cells have a value of 0.

3.2.3 Predictor Architecture

The hybridization predictor (Fig. 3.3A) takes a pair of one-hot sequences and predicts the yield of the hybridization reaction between the top sequence and the reverse-complement of the bottom sequence.

It first pre-processes the sequence pair with a sliding window operation that finds local matches (Fig. 3.3B). Following local average pooling, a trainable convolutional layer enables the predictor to weigh different kinds of local matches differently. Because we are not predicting where hybridization occurs, we use a global average pooling operation to remove spatial relevance. The prediction is then computed with a logistic regression (in our experience, this is more effective than a linear regression because most hybridization yields are either close to 0 or close to 1).

3.3 Oligo Design

The oligo design is similar to previous work, except that the feature region is now 80 nucleotides in length instead of 30, and the tag region is now a 30- nucleotide decodable barcode instead of a 5-nucleotide random tag.

Figure 3.4 depicts the layouts of our synthesized DNA oligomers, as well as the layouts of double-stranded complexes formed during processing. Each document in the database is associated with a single DNA oligomer (Figure 3.4A) that contains the barcode and feature regions that are unique to that document. In addition to these unique regions, each database oligo contains three conserved regions (denoted FP, RP, and IP) that are the same across all documents. PCR with FP and RP* is used to create additional copies of all database strands (Figure 3.4B), to prepare for hybridization and sequencing. Linear PCR with IP* is used to create partially-double stranded copies of each database strand that leave the feature region exposed (Figure 3.4C).

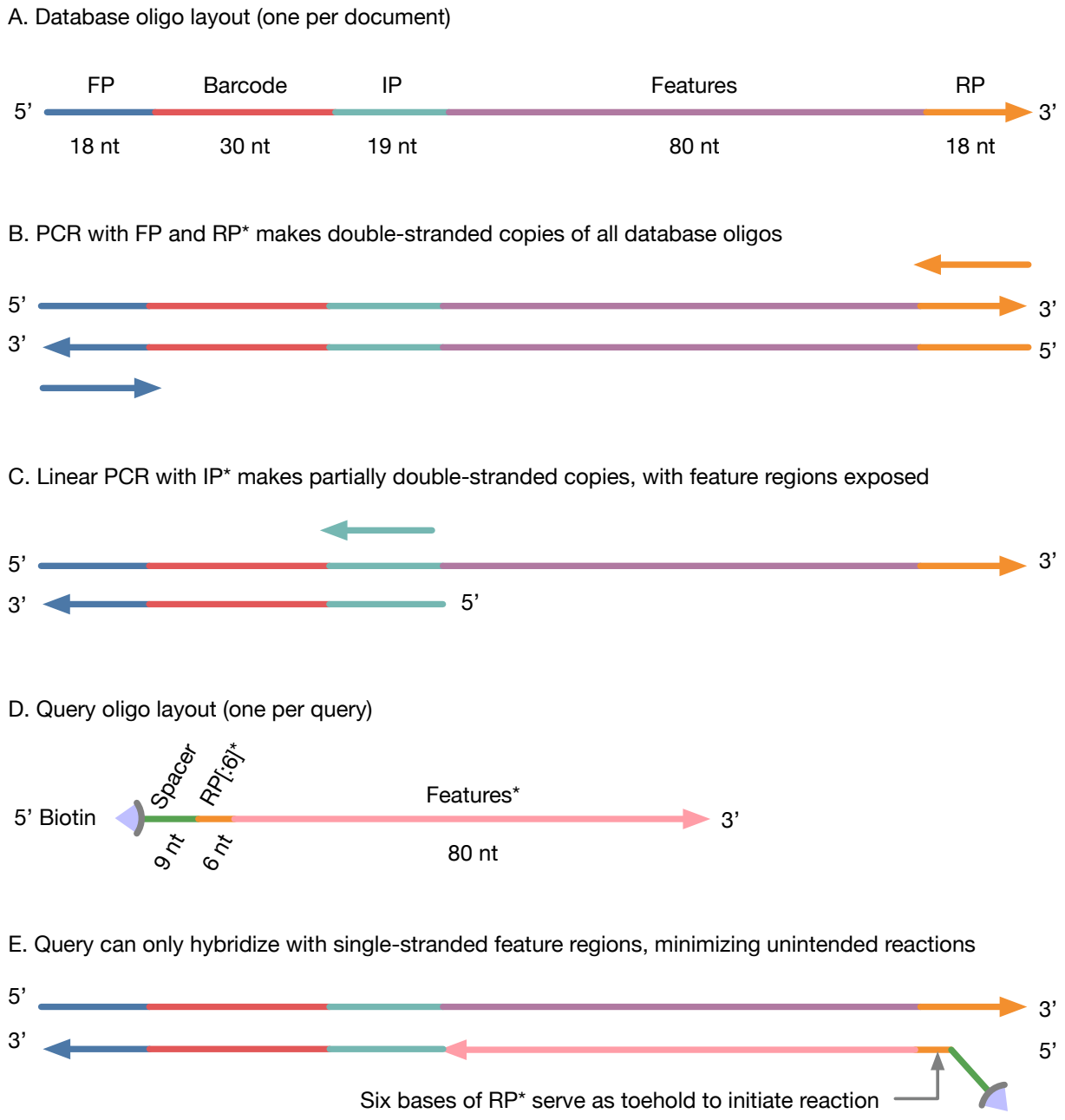


Figure 3.4: Layouts of single-stranded oligomers and intended double-stranded complexes. Arrowheads indicate 3' ends of DNA. Asterisks (*) indicate the reverse complement of a DNA sequence.

3.3.1 Barcodes

Document IDs are defined as 24-bit integers (in the range 0 to 16,777,215). To construct a DNA barcode from an ID, the ID is first mapped to another 24-bit integer using a reversible pseudo-random permutation. This step encourages sequence diversity among the barcodes. The randomized ID is then split into four six-bit symbols. These symbols are processed with a Reed-Solomon error-correcting code to produce a codeword with six symbols. Each of the six symbols is mapped to a five-nucleotide homopolymer-free DNA subsequence using a codebook with 64 entries. The final 30-nucleotide barcode is the concatenation of these six subsequences.

To decode a 30-nucleotide barcode, it is split into its six five-nucleotide subsequences, and each step of the code is reversed. Limited substitutions can be corrected, but if the sequence cannot be decoded, or it decodes to an ID that is unused, it is rejected as an invalid barcode.

3.3.2 Training Procedure

Figure 3.5A outlines the training procedure, which alternates between encoder and predictor training phases. During each round of encoder training, we draw a batch of pairs of feature vectors from the training set where half of the pairs are labeled "similar" (the Euclidean distance between the feature vectors in the pair is 75 or less). The batch of pairs is processed by the encoder, which outputs pairs of one-hot sequences. These are then processed by the yield predictor, which outputs the estimated yield of the hybridization reaction between the first sequence and the reverse complement of the second sequence. The estimated yield of each pair in the batch is used along with the similarity labels (0 for "not similar" and 1 for "similar") to compute the mean cross-entropy for the batch. We use the cross-entropy as loss function because it penalizes similar images with low estimated yield, dissimilar images with high estimated yield, and any estimated yields that are neither high nor low. The parameters of the encoder are modified (via gradient descent) to minimize the mean cross-entropy. The

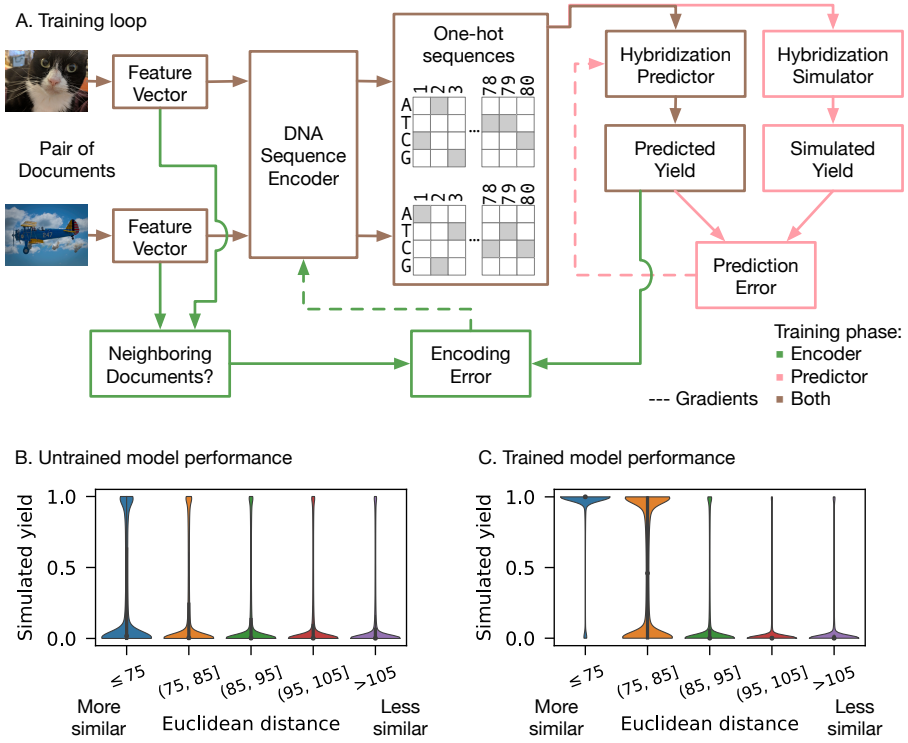


Figure 3.5: Overview of our training process. (A) The training loop for the neural networks. Lines indicate data flow; dashed lines indicate parameter gradients calculated using backpropagation. Green indicates operations only performed during encoder training, while pink indicates operations used only during yield predictor training. All other operations are used in both training phases. (B) Simulated performance of an untrained model, evaluated on 1.6 million random pairs of images. Each violin depicts the distribution of simulated hybridizations for pairs whose feature vectors' Euclidean distance lie within a certain range. (C) Simulated performance of a trained model, evaluated on the same set of random pairs.

yield predictors parameters are not changed during encoder training.

During each round of yield predictor training, we draw a random batch of pairs of feature vectors (unlike the encoder batches, these are not conditioned to have a particular distribution of similarity). These are processed by the encoder to produce one-hot sequences, which are then discretized (each position's maximum channel is set to 1, and the others to 0). The pairs of discretized one-hot sequences are fed to the yield predictor to output the estimated yield. We also simulate the reaction yields with NUPACK, as follows. The first sequence in each pair is treated as the target, and a reverse primer is appended as in Fig. 3.4A. We do not append the forward primer, barcode, or internal primer, as these regions will be double-stranded during retrieval and should not react with the query. The second sequence in each pair is treated as the query: six bases of the reverse primer are appended, and the sequence is reverse-complemented, as in Fig. 3.4D. The pairs of target and query sequences are processed with NUPACK at 21°C using default DNA parameters and an equal molar concentration of 1 nM for both query and target. The simulated reaction yields are computed by dividing the final concentration of each query-target duplex by the initial concentration of 1 nM.

We then compute the cross-entropy between NUPACK's simulated yield and the predictor's estimated yield for each pair in the batch. Cross-entropy is used instead of a standard regression loss like mean squared error, because most yields tend to be very close to either 0.0 or 1.0. The parameters of the yield predictor are modified (via gradient descent) to minimize the mean cross-entropy for the batch. The encoder's parameters are not changed during predictor training.

3.4 Experiments

3.4.1 Setup

During training, we withheld a fixed subset of 1.6 million images from OpenImages V4 to be used as our "database" for laboratory experiments. After training our encoder, we transformed each of these images into a DNA sequence using the trained encoder and synthesized

them according to the layout in Figure 3.4A.

Our query images did not come from the OpenImages dataset, and do not exist in the database. To conduct similarity search with an query image, we order a biotinylated probe oligomer that contains the reverse complement of the query's encoded feature sequence, according to the layout in Figure 3.4D. We anneal the probe with a sample of the database, and then separate the annealed target/query pairs from the database using streptavidin-conjugated magnetic beads. A full description of our laboratory protocol can be found in Appendix A. We then use high-throughput sequencing to read the filtered mixture, and decode each read's barcode according to the procedure in Section 3.3.1. By counting the occurrences of each successfully decoded document ID, we can measure how frequently each document appears in the filtered mixture.

3.4.2 Results

Figure 3.6 shows the experimental results for three different query images. If we consider images with sequencing read counts above a certain threshold to be "retrieved", we can characterize the set of retrieved images for a variety of thresholds. Figure 3.6A shows that higher read counts are associated with sets of images that are closer to the query in Euclidean distance. We can quantitatively characterize the quality of a retrieved set by its recall of the 100 nearest neighbors; that is, the number of images in the set that are among the 100 most similar images to the query in the database. Figure 3.6B shows that, as the read threshold increases, the number of total images in the retrieved set drops very low before you begin to sacrifice nearest neighbor recall. We can also visually inspect the retrieved set by sorting its contents and displaying the most similar images. Figure 3.6C shows that, even with very aggressive filtering, the retrieved set still contains images that are relevant to the query. If the read counts for each image are proportional to their concentrations in the filtered mixture, this means that the filtered mixture could be diluted about 1000x, conserving sequencing resources while still retrieving relevant images.

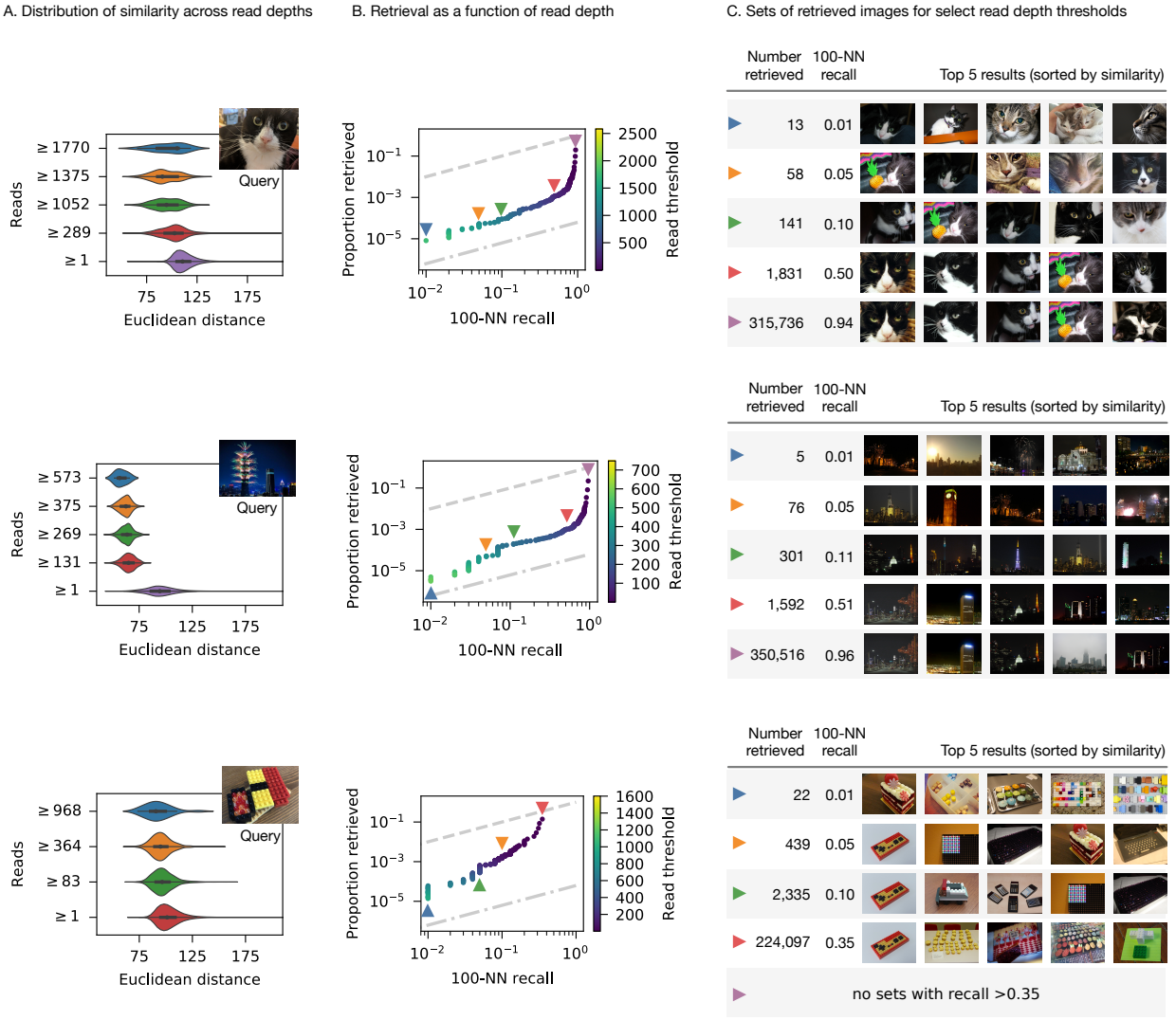


Figure 3.6: Experimental results for three different query images. Janelle, the cat (top), a building with fireworks (center) and Lego pieces assembled in the shape of sushi (bottom). (A) Distribution of Euclidean distances to the query image, among sets of images with sequencing read depth above a certain threshold. (B) The proportion of the entire dataset that must be retrieved (y-axis) to retrieve a certain proportion of the 100 most similar images (x-axis). Each point represents a threshold for which images with read depth above that threshold are considered "retrieved". The dashed line indicates chance performance, while the dashed-and-dotted line indicates perfect performance. Colored triangles indicate the thresholds depicted in the other subfigures. (C) The top 5 closest images to the query from result sets where images above a certain read depth threshold are considered "retrieved".

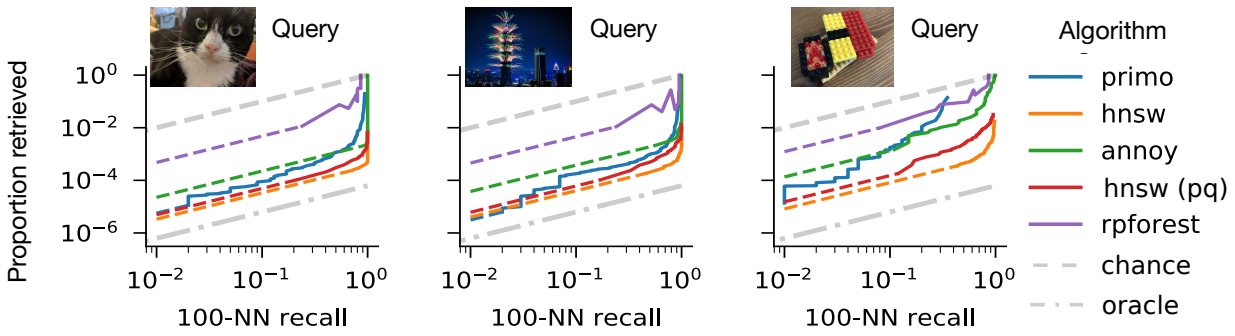


Figure 3.7: Comparison of our technique ("primo", shown in blue) with state-of-the-art algorithms for in silico similarity search. Dashed grey and dashed-and-dotted grey lines represent chance performance and perfect performance, respectively. Not all of the algorithms could produce results towards the lower-left (low recall and low proportion retrieved). We assume these algorithms could be stopped early to produce fewer results with a linear decrease in recall; dashed continuations represent these linear interpolations.

3.4.3 Analysis

The performance of a similarity search algorithm can be summarized by the curve in Figure 3.6B, which measures the proportion of the database that must be retrieved and sorted to achieve a particular 100-nearest neighbor recall. The dashed line above the curve illustrates a "naive" algorithm that randomly samples the database. To retrieve half of the hundred nearest neighbors, it must retrieve half of the database. The dashed-and-dotted line below the curve illustrates a perfect "oracle" algorithm. To retrieve half of the hundred nearest neighbors, it would retrieve exactly those 50 images from the 1.6 million in the database.

Figure 3.7 places the curve from Figure 3.6B in context alongside several state-of-the-art in silico algorithms that were benchmarked using the same query and same database for each of the queries we evaluated experimentally.

These algorithms all perform an Approximate Nearest-Neighbor (ANN) search. Given a dataset, they create an index structure that is traversed using a given querys feature vector, to retrieve the documents whose feature vectors are nearest to that query. An approximate search does not scan the entire database, but this may cause it to miss some of the nearest

neighbors. We define the *candidate set* as the subset of the database that is scanned (i.e., retrieved from memory and compared to the query). The candidate set is analogous to the set of "retrieved" documents we define by varying the read depth threshold for our lab experiments.

For each algorithm in Fig. 3.7, and for each of the three queries, we collected the candidate sets for a variety of algorithm-specific parameters that give users control over the specificity of the ANN search. The size of each candidate set (divided by the size of the full database) gives us the proportion retrieved (the y-axis of Fig. 3.7), whereas the number of true nearest neighbors in each candidate set (out of 100) gives us the 100-nearest-neighbor recall (the x-axis of Fig. 3.7).

Some algorithms could not retrieve candidate sets below a certain size for any of the attempted parameters. For these, we assume that uniform subsampling would equally limit both the size of the candidate set and the number of nearest neighbors retrieved. This assumption gives us the dashed colored lines for each algorithm in Fig. 3.7.

Implementations of HNSW (hierarchical navigable small world) graphs [30] are among the top performers on approximate nearest neighbor benchmarks [31]. HNSW requires building and storing a very large index, which may be difficult to scale to large databases. We also tested a quantized version of HNSW with lower memory utilization, developed by Facebook [32] ("faiss", shown in red), annoy [33] (shown in green), a popular algorithm developed by Spotify, and RPForest [34] (shown in purple), an algorithm designed for the lowest possible memory utilization. While there is room for improvement, our experimental performance is comparable to the state-of-the-art, indicating that DNA-based similarity search is a viable technique for searching the databases of the future.

3.5 Conclusion

This chapter detailed the first practical execution of similarity search in a DNA-based digital database, and compared its potential efficiency with electronic systems. The results suggest that, as DNA data storage becomes more practical and scales to larger data sets, similarity

search in DNA form is an attractive possibility compared to electronic systems. Combining DNA data storage with similarity search support may offer a path to viable hybrid molecular-electronic computer systems.

Chapter 4

CONCLUSION

Our results have demonstrated that DNA-based digital storage systems are capable of performing complex queries on the scale of millions of images. The question remains of how far this technique can scale, and what the major bottlenecks to scalability might be.

While we have described our technique in detail end-to-end, there are several aspects of its implementation that warrant a sensitivity analysis to understand their effects on performance. For instance, to study the limits of encoding DNA sequences with a deep neural network, we might explore the space of encoder topologies or training hyperparameters, or we could try databases composed of other types of media, which might have a different distribution of similarity or a different inherent "difficulty".

Database size is also an important factor. Larger databases might expose limitations in our encoding strategy, as well as limitations in our laboratory protocol. Along the same lines, the laboratory protocol is complex, and the space of various choices for time, temperature, and concentration have not been fully explored.

However, it is possible that using DNA hybridization as a search primitive poses a fundamental scalability bottleneck. Our technique is based on predicting and controlling the concentration of query-target duplexes at thermodynamic equilibrium. Because single-stranded DNA is prone to folding up on itself (especially at lower temperatures), reaching that equilibrium on a reasonable timescale requires annealing (heating the DNA to melt all secondary structure, then cooling it back down slowly).

Annealing single-stranded DNA incurs an unavoidable energy cost that depends on the volume of the working solution. While we may be able to push the concentration of the system to its limit to drive volume down, at a certain point, the only way to increase the

capacity of the system will be to increase its volume, and therefore also the energy required for annealing.

Fortunately, hybridization of single-stranded species is not the only way to leverage non-specific DNA binding. For instance, CRISPR nuclease enzymes, which can unwind double-stranded DNA to allow a guide sequence to bind to a target domain, have predictable off-target effects [35], and they can operate isothermally (without requiring a change in temperature). Further study of such enzymatic components may prove to be a fruitful direction for creating scalable DNA-based digital storage systems.

BIBLIOGRAPHY

[1] George M Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in DNA. *Science*, 337(6102):1628–1628, 2012.

[2] Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435):77–80, 2013.

[3] Robert N. Grass, Reinhard Heckel, Michela Puddu, Daniela Paunescu, and Wendelin J. Stark. Robust Chemical Preservation of Digital Information on DNA in Silica with Error-Correcting Codes. *Angewandte Chemie Intl. Edition*, 54(8):2552–2555, 2015.

[4] Yaniv Erlich and Dina Zielinski. DNA Fountain enables a robust and efficient storage architecture. *Science*, 355(6328):950–954, 2017.

[5] Lee Organick, Siena Dumas Ang, Yuan-Jyue Chen, Randolph Lopez, Sergey Yekhanin, Konstantin Makarychev, Miklos Z Racz, Govinda Kamath, Parikshit Gopalan, Bichlien Nguyen, Christopher N Takahashi, Sharon Newman, Hsing-Yeh Parker, Cyrus Rashtchian, Kendall Stewart, Gagan Gupta, Robert Carlson, John Mulligan, Douglas Carmean, Georg Seelig, Luis Ceze, and Karin Strauss. Random access in large-scale DNA data storage. *Nature Biotechnology*, 36(3):242–248, 2018.

[6] S M Hossein Tabatabaei Yazdi, Yongbo Yuan, Jian Ma, Huimin Zhao, and Olgica Milenkovic. A Rewritable, Random-Access DNA-Based Storage System. *Scientific Reports*, 5(1):1763, 2015.

[7] Luis Ceze, Jeff Nivala, and Karin Strauss. Molecular Digital Data Storage using DNA. *Nature Reviews Genetics*, May 2019.

[8] E B Baum. Building an associative memory vastly larger than the brain. *Science*, 268 (5210):583–585, 1995.

[9] Cyrus Rashtchian, Konstantin Makarychev, Miklos Racz, Siena Ang, Djordje Jevdjic, Sergey Yekhanin, Luis Ceze, and Karin Strauss. Clustering Billions of Reads for DNA Data Storage. In *Advances in Neural Information Processing Systems 30*, December 2017.

[10] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM. ISBN 0-89791-962-9. doi: 10.1145/276698.276876.

[11] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, cs.CV, 2014.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):8490, May 2017. ISSN 0001-0782. doi: 10.1145/3065386.

[13] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. pages 157–166, 2014. doi: 10.1145/2647868.2654948.

[14] John H. Reif and Thomas H. LaBean. Computationally inspired biotechnologies: Improved DNA synthesis and associative search using Error-Correcting Codes and Vector-Quantization. In Anne Condon and Grzegorz Rozenberg, editors, *DNA Computing*, pages 145–172, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[15] Sotirios A Tsaftaris, A K Katsaggelos, T N Pappas, and T E Papoutsakis. DNA-based matching of digital signals. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages V–581–4. IEEE, 2004.

[16] Sotirios A Tsaftaris, V Hatzimanikatis, and A K Katsaggelos. DNA hybridization as a similarity criterion for querying digital signals stored in DNA databases. In *2006 IEEE International Conference on Acoustics Speed and Signal Processing*, pages II–1084–II–1087. IEEE, 2006.

[17] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[18] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[19] IDC. Where in the world is storage, 2013. `http://www.idc.com/downloads/where_is_storage_infographic_243338.pdf`.

[20] V. T. Lee, J. Kotalik, C. C. d. Mundo, A. Alaghi, L. Ceze, and M. Oskin. Similarity search on automata processors. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 523–534, 2017.

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

[22] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[23] Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Review*, 2007.

[24] Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Marshall B Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, 2011.

[25] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Proceedings of*

the 21st International Conference on Neural Information Processing Systems, NIPS'08, pages 1753–1760, USA, 2008. Curran Associates Inc. ISBN 978-1-6056-0-949-2.

[26] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. 2013.

[27] David Yu Zhang, Sherry Xi Chen, and Peng Yin. Optimizing the specificity of nucleic acid hybridization. *Nature Chemistry*, 4(3):208–214, 2012.

[28] Lucia R Wu, Juexiao Sherry Wang, John Z Fang, Emily R Evans, Alessandro Pinto, Irena Pekker, Richard Boykin, Celine Ngouenet, Philippa J Webster, Joseph Beechem, and David Yu Zhang. Continuously tunable nucleic acid hybridization probes. *Nature Methods*, 12(12):1191–1196, 2015.

[29] OpenImagesV4. `https://storage.googleapis.com/openimages/web/download_v4.html`.

[30] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473.

[31] Aumueller, M., Bernhardsson, E., Faitfull, A. ANN Benchmarks. `http://ann-benchmarks.com`.

[32] Facebook AI Research. FAISS (Facebook AI Similarity Search). `https://github.com/facebookresearch/faiss`.

[33] Bernhardsson, E. et al. Approximate Nearest Neighbors Oh Yeah (Annoy). `https://github.com/spotify/annoy`.

[34] Kula, M., Serko, I., et al. rpforest. `https://github.com/lyst/rpforest`.

[35] Stephen K. Jones, John A. Hawkins, Nicole V. Johnson, Cheulhee Jung, Kuang Hu, James R. Rybarski, Janice S. Chen, Jennifer A. Doudna, William H. Press, and Ilya J. Finkelstein. Massively parallel kinetic profiling of natural and engineered CRISPR nucleases. *Nature Biotechnology*, Sep 2020. ISSN 1546-1696. doi: 10.1038/s41587-020-0646-5.

Appendix A

LABORATORY PROTOCOL FOR LARGE-SCALE IMPLEMENTATION[1]

The 1.6 million oligos that make up the database were ordered from Twist Bioscience. Biotinylated probe oligos were ordered from IDT. Streptavidin-coated magnetic beads (Dynabeads MyOne Streptavidin T1) was purchased from Thermo Fisher Scientific. USER enzyme was ordered from New England Lab.

The general workflow of a similarity search experiment is divided into 8 steps (Figure A.1): (1) enrichment of a synthesized oligo pool using PCR, (2) linear amplification of the pool using a forward primer, (3) linear amplification using an internal primer, (4) hybridization experiment using a query strand, (5) magnetic bead extraction, (6) releasing of bead captured strands using digestion of USER enzyme, (7) PCR enrichment of the released oligos (8) ligation to Illumina adapters for sequencing.

A DNA pool synthesized from Twist Bioscience was PCR amplified by mixing 1 μL of 1 ng/μL of the pool, 1 μL of 10 μM forward primer, 1 μL of 10 μM reverse primer, 10 μL of 2X KAPA HIFI enzyme mix, and 7 μL of molecular grade water. PCR was performed in a thermocycler with the following protocol (1) 95°C for 3 min, (2) 98°C for 20 s, (3) 56°C for 20 s, (4) 72°C for 20 s, (5) go to step 2 for about 15 cycles, (6) 72°C for 30 s. The amplified product was PCR purified using QIAGEN PCR Purification Kit (Cat No: 28104). The sample concentration was measured using Qubit 3.0 fluorometer.

This enriched Twist pool was mixed with 100 times more of the Forward Primer (e.g., [FP]/[pool]=100) at 500 nM of the pool. 20 μL of this mixture was mixed with 20 μL of 2X KAPA HIFI enzyme mix, followed by linear amplification with the following protocol:

[1]The text for this appendix was contributed by Dr. Yuan-Jyue Chen.

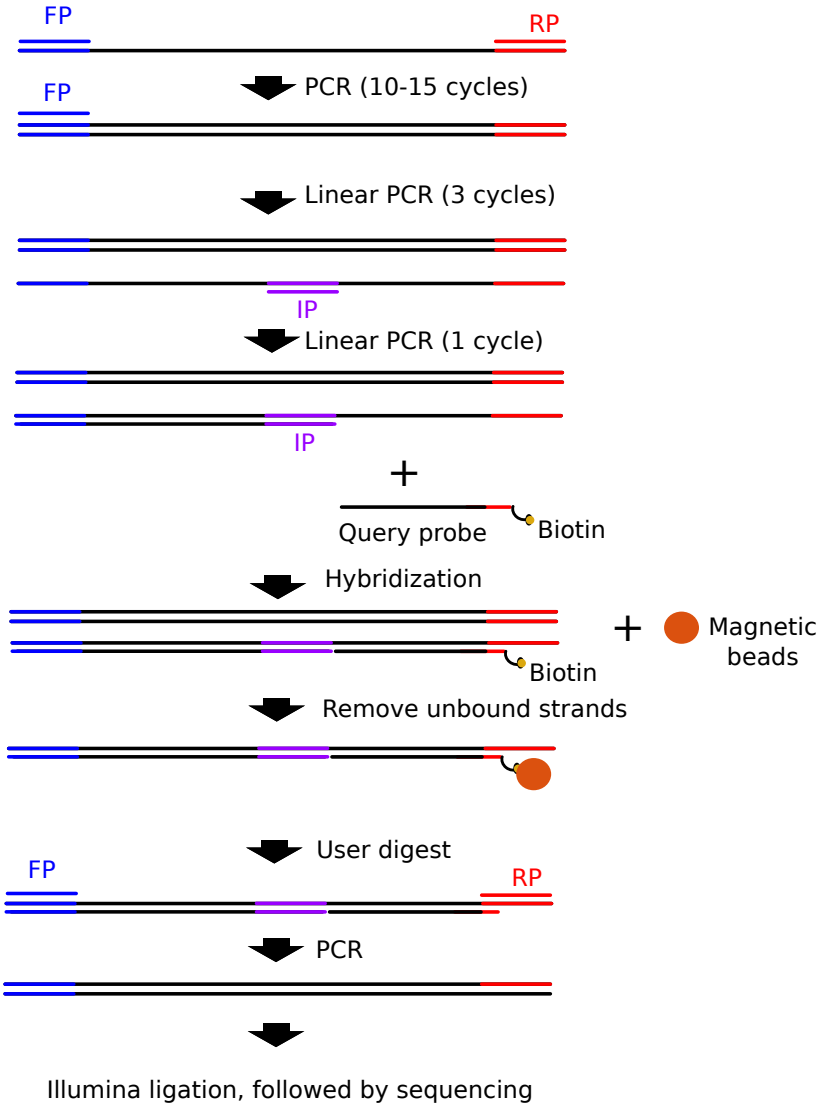


Figure A.1: Workflow of a similarity search experiment. A large DNA pool is PCR amplified using a forward primer (FP) and a reverse primer (RP). The enriched product is linearly amplified using the forward primer for 3 cycles. The sample is linearly amplified using an internal primer (IP) to make partially double-stranded copies, with feature region exposed. This mixture is then hybridized with a query strand, followed by magnetic bead extraction. The extracted strands are released from the beads using USER enzyme digestion. The released sample is PCR enriched using FP and RP. The sample is PCR again using RP and FP with a 25N overhang to create a randomized region for the diversity need of Ilumina NextSeq. The sample is ligated to Illumina adapter, followed by next-generation-sequencing.

(1) 95°C for 3 min, (2) 98°C for 20 s, (3) 62°C for 20 s, (4) 72°C for 20 s, (5) go to step 2 for 2 time, and (6) 72°C for 30 s. The mixture contains 250 nM of double-stranded DNA (dsDNA) and less than 750 nM of single-stranded DNA (ssDNA).

The sample was linearly amplified again using an Internal Primer (IP) by mixing 40 µL of the 250 nM dsDNA mixture, 12 µL of 10 µM Internal Primer (IP), and 12 µL of 2X KAPA HIFI enzyme mix. Linear amplification was performed with the following protocol: (1) 95°C for 3 min, (2) 98°C for 20 s, (3) 56°C for 20 s, (4) 72°C for 20 s, and (5) 72°C for 30 s. The mixture contains 156 nM of fully dsDNA pool and less than 468 nM of partially dsDNA with feature region exposed (feature region will hybridize to a query strand).

6.4 µL of the mixture (containing 156 nM of the fully dsDNA pool and 468 nM of partially dsDNA) was mixed with 1 µL of a query strand at 10 nM, 10 µL of 2 M sodium chloride buffer and 2.6 µL molecular grade water. This resulted in an 1:100 ratio of query to the fully dsDNA pool and a final concentration of the fully dsDNA pool at 50 nM. This mixture was annealed in a thermocycler by first heating up to 95°C for 3 mins, and then slowly cooling down to 21°C at the rate of 1°C per 20 min.

3 µg of Streptavidin-coated magnetic beads (Dynabeads MyOne Streptavidin T1, Thermo Fisher Scientific) was used for 1 pmol of a query strand. The beads were washed 3 times in binding and washing buffer (5 mM Tris-HCl (pH 7.5), 0.5 mM EDTA, 1 M NaCl), then added to the hybridization sample at room temperature. After incubating at room temperature for 15 minutes, the samples sat on a magnet rack to recover the beads and binding DNA. The supernatants were removed, and the beads were washed 3 times using 100 µL of binding and washing buffer. The beads binding DNA was resuspended in 50 µL 1X elution buffer containing 10 mM tris-Cl, at pH 8.5. The resuspended samples were digested using USER enzyme by mixing 50 µL of the sample with 2 µL of USER enzyme, and 5.8 µL of NEB 10X cut smart buffer at 37°C for 20 minutes. The sample sat on a magnetic rack for 1 minute, and the supernatants was recovered.

2 µL of the recovered solution from last step was mixed with 1 uL of 10 uM forward primer, 1 µL of 10 µM reverse primer (RP), 10 µL of 2X KAPA HIFI enzyme mix, and 6 µL

of molecular grade water for PCR. PCR was performed in a thermocycler with the following protocol (1) 95°C for 3 min, (2) 98°C for 20 s, (3) 62°C for 20 s, (4) 72°C for 20 s, (5) go to step 2 for a varying number of times depending on the recovery yield of beads extraction, and (6) 72°C for 30 s. 2 µL of the amplified sample was mixed with 1 µL of 10 µM forward primer with an overhang of a randomized region (25N), 1 µL of 10 µM reverse primer (RP), 10 µL of 2X KAPA HIFI enzyme mix, and 6 µL of molecular grade, followed by the following thermocycling protocol: (1) 95°C for 3 min, (2) 98°C for 20 s, (3) 62°C for 20 s, (4) 72°C for 20 s, (5) go to step 2 for a varying number of times depending on the recovery yield. This PCR step added a randomized region to the sample for the diversity need of Illumina NextSeq. The size of the PCR product was verified using QIAGEN bioanalyzer. The amplified product was ligated to Illumina sequencing adapters with TruSeq Nano reagents and protocol. The ligated samples were sequenced using Illumina NextSeq.