Recursive Sketches for Modular Deep Learning

Badih Ghazi [*1] **Rina Panigrahy** [*1] **Joshua R. Wang** [*1]

Abstract

We present a mechanism to compute a sketch (succinct summary) of how a complex modular deep network processes its inputs. The sketch summarizes essential information about the inputs and outputs of the network and can be used to quickly identify key components and summary statistics of the inputs. Furthermore, the sketch is recursive and can be unrolled to identify sub-components of these components and so forth, capturing a potentially complicated DAG structure. These sketches erase gracefully; even if we erase a fraction of the sketch at random, the remainder still retains the "high-weight" information present in the original sketch. The sketches can also be organized in a repository to implicitly form a "knowledge graph"; it is possible to quickly retrieve sketches in the repository that are related to a sketch of interest; arranged in this fashion, the sketches can also be used to learn emerging concepts by looking for new clusters in sketch space. Finally, in the scenario where we want to learn a ground truth deep network, we show that augmenting input/output pairs with these sketches can theoretically make it easier to do so.

1. Introduction

Machine learning has leveraged our understanding of how the brain functions to devise better algorithms. Much of classical machine learning focuses on how to *correctly* compute a function; we utilize the available data to make more accurate predictions. More recently, lines of work have considered other important objectives as well: we might like our algorithms to be small, efficient, and robust. This work aims to further explore one such sub-question: can we design a system on top of neural nets that efficiently stores information?

Our motivating example is the following everyday situation. Imagine stepping into a room and briefly viewing the objects within. Modern machine learning is excellent at answering immediate questions about this scene: "Is there a cat? How big is said cat?" Now, suppose we view this room every day over the course of a year. Humans can reminisce about the times they saw the room: "How often did the room contain a cat? Was it usually morning or night when we saw the room?"; can we design systems that are also capable of efficiently answering such memory-based questions?

Our proposed solution works by leveraging an existing (already trained) machine learning model to understand individual inputs. For the sake of clarity of presentation, this base machine learning model will be a modular deep network.[1] We then augment this model with sketches of its computation. We show how these sketches can be used to efficiently answer memory-based questions, despite the fact that they take up much less memory than storing the entire original computation.

A modular deep network consists of several independent neural networks (modules) which only communicate via one's output serving as another's input. Figure 1 presents a cartoon depiction of a modular network for our example. Modular networks have both a biological basis (Azam, 2000) and evolutionary justification (Clune et al., 2013). They have inspired several practical architectures such as neural module networks (Andreas et al., 2016; Hu et al., 2017), capsule neural networks (Hinton et al., 2000; 2011; Sabour et al., 2017), and PathNet (Fernando et al., 2017) and have connections to suggested biological models of intelligence such as hierarchical temporal memory (Hawkins & Blakeslee, 2007). We choose them as a useful abstraction to avoid discussing specific network architectures.

[*]Equal contribution [1]Google Research, Mountain View, CA, USA.. Correspondence to: Rina Panigrahy <rinap@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]Of course, it is possible to cast many models as deep modular networks by appropriately separating them into modules.

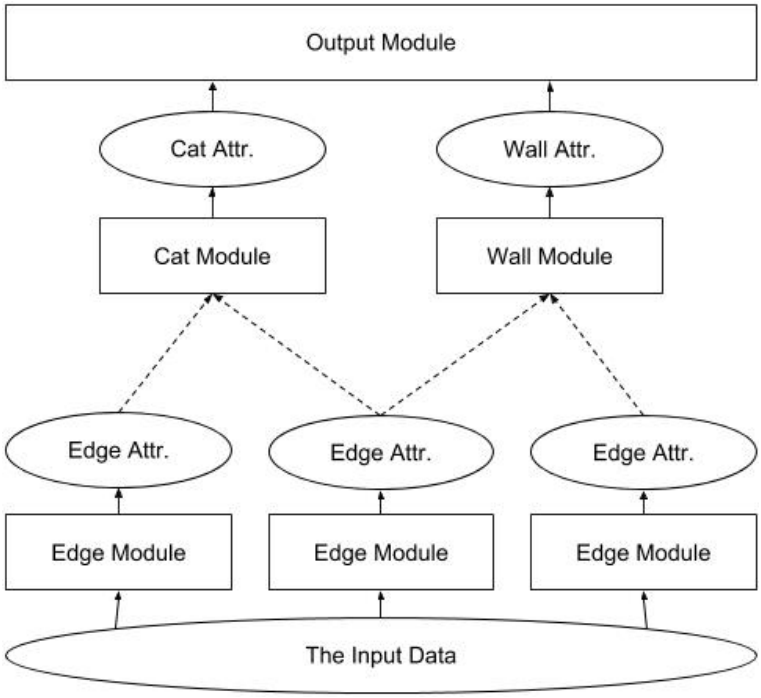

Figure 1. Cartoon depiction of a modular network processing an image of a room. Modules are drawn as rectangles and their inputs/outputs (which we refer to as object attributes) are drawn as ovals. The arrows run from input vector to module and module to output vector. There may be additional layers between low level and high level modules, indicated by the dashed arrows. The output module here is a dummy module which groups together top-level objects.

What do these modules represent in the context of our room task? Since we are searching for objects in the room, we think of each module as attempting to identify a particular type of object, from the low level edge to the high level cat. For the reader familiar with convolutional neural networks, it may help to think of each module as a filter or kernel. We denote the event where a module produces output as an object, the produced output vector as an attribute vector, and all objects produced by a module as a class. In fact, our sketching mechanism will detail how to sketch each object, and these sketches will be *recursive*, taking into account the sketches corresponding to the inputs of the module.

Armed with this view of modular networks for image processing, we now entertain some possible sketch properties. One basic use case is that from the output sketch, we should be able to say something about the attribute vectors of the objects that went into it. This encompasses a question like "How big was the cat?" Note that we need to assume that our base network is capable of answering such questions, and the primary difficulty stems from our desire to maintain this capability despite only keeping a small sketch. Obviously, we should not be able to answer such questions as precisely as we could with the original input. Hence a reasonable *attribute recovery* goal is to be able to approximately recover an object's attribute vector from its sketch. Concretely, we should be able to recover cat attributes from our cat sketch.

Since we plan to make our sketches recursive, we would actually like to recover the cat attributes from the final output sketch. Can we recover the cat's sketch from final sketch? Again, we have the same tug-of-war with space and should expect some loss from incorporating information from many sketches into one. Our *recursive sketch* property is that we can recover (a functional approximation of) a sketch from a later sketch which incorporated it.

Zooming out to the entire scene, one fundamental question is "Have I been in this room before?" In other words, comparing sketches should give some information about how similar the modules and inputs that generated them. In the language of sketches, our desired *sketch-to-sketch similarity* property is that two completely unrelated sketches will be dissimilar while two sketches that share common modules and similar objects will be similar.

Humans also have the impressive ability to recall approximate counting information pertaining to previous encounters (Brannon & Roitman, 2003). The *summary statistics* property states that from a sketch, we can approximately recover the number of objects produced by a particular module, as well as their mean attributes.

Finally, we would like the nuance of being able to forget information gradually, remembering more important facts for longer. The *graceful erasure* property expresses this idea: if (a known) portion of our sketch is erased, then the previous properties continue to hold, but with additional noise depending on the amount erased.

We present a sketching mechanism which simultaneously satisfies all these desiderata: *attribute recovery*, *recursive sketch*, *sketch-to-sketch similarity*, *summary statistics*, and *graceful erasure*.

1.1. Potential Applications

We justify our proposed properties by describing several potential applications for a sketching mechanism satisfying them.

Sketch Repository. The most obvious use case suggested by our motivating example is to maintain a repository of sketches produced by the (augmented) network. By the *recursive sketch* and *attribute recovery* properties, we can query our repository for sketches with objects similar to a particular object. From the *sketch-to-sketch similarity* property, we can think of the sketches as forming a knowledge graph; we could cluster them to look for patterns in the data or use locality-sensitive hashing to quickly fish out related sketches. We can combine these overall sketches once more and use the *summary statistics* property to get rough statistics of how common certain types of objects are, or what a typical object of a class looks like. Our repository may follow a generalization of a "least recently used" policy, taken advantage of the *graceful erasure* property to only partially forget old sketches.

Since our sketches are recursively defined, such a repository is not limited to only keeping overall sketches of inputs; we can also store lower-level object sketches. When we perform object attribute recovery using the *attribute recovery* property, we can treat the result as a fingerprint to search for a more accurate sketch of that object in our repository.

Learning New Modules. This is an extension of the sketch repository application above. Rather than assuming we start with a pre-trained modular network, we start with a bare-bones modular network and want to grow it. We feed it inputs and examine the resulting sketches for emerging clusters. When we find such a cluster, we use its points to train a corresponding (reversible) module. In fact, this clustering may be done in a hierarchical fashion to get finer classes. For example, we may develop a module corresponding to "cat" based on a coarse cluster of sketches which involve cats. We may get sub-clusters based on cat species or even finer sub-clusters based on individual cats repeatedly appearing.

The modules produced by this procedure can capture important primitives. For example, the *summary statistics* property implies that a single-layer module could count the number of objects produced by another module. If the modules available are complex enough to capture curves, then this process could potentially allow us to identify different types of motion.

Interpretability. Since our sketches store information regarding how the network processed an input, they can help explain how that network came to its final decision. The *sketch-to-sketch similarity* property tells us when the network thinks two rooms are similar, but what exactly did the network find similar? Using the *recursive sketch* property, we can drill down into two top level sketches, looking for the pairs of objects that the network found similar.

Model Distillation. Model distillation (Buciluǎ et al., 2006; Hinton et al., 2015) is a popular technique for improving machine learning models. The use case of model distillation is as follows. Imagine that we train a model on the original data set, but it is unsatisfactory in some regard (e.g. too large or ensemble). In model distillation, we take this first model and use its features to train a second model. These stepping stones allows the second model to be smaller and less complex. The textbook example of feature choice is to use class probabilities (e.g. the object is 90% A and 10% B) versus the exact class found in the original input (e.g. the object is of class A). We can think of this as an intermediate option between the two extremes of (i) providing just the original data set again and (ii) providing all activations in the first model. Our sketches are an alternative option. To showcase the utility of our sketches in the context of learning from a teacher network, we prove that (under certain assumptions) augmenting the teacher network with our sketches allows us to learn the network. Note that in general, it is not known how the student can learn solely from many input/output pairs.

1.2. Techniques

The building block of our sketching mechanism is applying a random matrix to a vector. We apply such transformations to object attribute vectors to produce sketches, and then recursively to merge sketches. Consequently, most of our analysis revolves around proving and utilizing properties of random matrices. In the case where we know the matrices used in the sketch, then we can use an approximate version of isometry to argue about how noise propagates through our sketches. On the other hand, if we do not know the matrices involved, then we use sparse recovery/dictionary learning techniques to recover

them. Our recovery procedure is at partially inspired by several works on sparse coding and dictionary learning (Spielman et al., 2012; Arora et al., 2014c), (Arora et al., 2014a) and (Arora et al., 2015) as well as their known theoretical connections to neural networks (Arora et al., 2014b; Papyan et al., 2018). We have more requirements on our procedure than previous literature, so we custom-design a distribution of block-random matrices reminiscent of the sparse Johnson-Lindenstrauss transform. Proving procedure correctness is done via probabilistic inequalities and analysis tools, including the Khintchine inequality and the Hanson-Wright inequality.

One could consider an alternate approach based on the serialization of structured data (for example, protocol buffers). Compared to this approach, ours provides more fine-grained control over the accuracy versus space trade-off and aligns more closely with machine learning models (it operates on real vectors).

1.3. Related Work

There have been several previous attempts at adding a notion of memory to neural networks. There is a line of work which considers augmenting recurrent neural networks with external memories (Graves et al., 2014; Sukhbaatar et al., 2015; Joulin & Mikolov, 2015; Grefenstette et al., 2015; Zaremba & Sutskever, 2015; Danihelka et al., 2016; Graves et al., 2016). In this line of work, the onus is on the neural network to learn how to write information to and read information from the memory. In contrast to this line of work, our approach does not attempt to use machine learning to manage memory, but rather demonstrates that proper use of random matrices suffices. Our results can hence be taken as theoretical evidence that this learning task is relatively easy.

Vector Symbolic Architectures (VSAs) (Smolensky, 1990; Gayler, 2004; Levy & Gayler, 2008) are a class of memory models which use vectors to represent both objects and the relationships between them. There is some conceptual overlap between our sketching mechanism and VSAs. For example, in Gayler's MAP (Multiply, Add, Permute) scheme (Gayler, 2004), vector addition is used to represent superposition and permutation is used to encode quotations. This is comparable to how we recursively encode objects; we use addition to combine multiple input objects after applying random matrices to them. One key difference in our model is that the object attribute vectors we want to recover are provided by some pre-trained model and we make no assumptions on their distribution. In particular, their possible correlation makes it necessary to first apply a random transformation before combining them. In problems where data is generated by simple programs, several papers (Wu et al., 2017; Yi et al., 2018; Ellis et al., 2018; Andreas et al., 2016; Oh et al., 2017) attempt to infer programs from the generated training data possibly in a modular fashion.

Our result on the learnability of a ground truth network is related to the recent line of work on learning small-depth neural networks (Du et al., 2017b;a; Li & Yuan, 2017; Zhong et al., 2017a; Zhang et al., 2017; Zhong et al., 2017b). Recent research on neural networks has considered evolving neural network architectures for image classification (e.g., (Real et al., 2018)), which is conceptually related to our suggested Learning New Models application.

1.4. Organization

We review some basic definitions and assumptions in Section 2. An overview of our final sketching mechanism is given in Section 3 (Theorems 1, 2, 3). Section 4 gives a sequence of sketch prototypes leading up to our final sketching mechanism. In Section 5, we explain how to deduce the random parameters of our sketching mechanism from multiple sketches via dictionary learning. Further discussion and detailed proofs are provided in the appendices.

2. Preliminaries

In this section, we cover some preliminaries that we use throughout the remainder of the paper. Throughout the paper, we denote by $[n]$ the set $\{1, \ldots, n\}$. We condense large matrix products with the following notation.

$$\prod_{i=1}^{\curvearrowright n} R_i \coloneqq R_1 R_2 \cdots R_n$$

$$\prod_{i=1}^{\curvearrowleft n} R_i \coloneqq R_n \cdots R_2 R_1$$

2.1. Modular Deep Networks for Object Detection

For this paper, a modular deep network consists of a collection of modules $\{M\}$. Each module is responsible for detecting a particular class of object, e.g. a cat module may detect cats. These modules use the outputs of other modules, e.g. the cat module may look at outputs of the edge detection module rather than raw input pixels. When the network processes a single input, we can depict which objects feed into which other objects as a (weighted) communication graph. This graph may be data-dependent. For example, in the original Neural Module Networks of Andreas et al. (Andreas et al., 2016), the communication graph is derived by taking questions, feeding them through a parser, and then converting parse trees into structured queries. Note that this pipeline is disjoint from the modular network itself, which operates on images. In this paper, we will not make assumptions about or try to learn the process producing these communication graphs. When a module does appear in the communication graph, we say that it detected an object θ and refer to its output as x_θ, the attribute vector of object θ. We assume that these attribute vectors x_θ are nonnegative and normalized. As a consequence of this view of modular networks, a communication graph may have far more objects than modules. We let our input size parameter N denote twice the maximum between the number of modules and the number of objects in any communication graph.

We assume we have access to this communication graph, and we use $\theta_1, ..., \theta_k$ to denote the k input objects to object θ. We assume we have a notion of how important each input object was; for each input θ_i, there is a nonnegative weight w_i such that $\sum_{i=1}^{k} w_i = 1$. It is possible naively choose $w_i = 1/k$, with the understanding that weights play a role in the guarantees of our sketching mechanism (it does a better job of storing information about high-weight inputs).

We handle the network-level input and output as follows. The input data can be used by modules, but it itself is not produced by a module and has no associated sketch. We assume the network has a special output module, which appears exactly once in the communication graph as its sink. It produces the output (pseudo-)object, which has associated sketches like other objects but does not count as an object when discussing sketching properties (e.g. two sketches are not similar just because they must have output pseudo-objects). This output module and pseudo-object only matter for our overall sketch insofar as they group together high-level objects.

2.2. Additional Notation for Sketching

We will (recursively) define a sketch for every object, but we consider our sketch for the *output object* of the network to be the overall sketch. This is the sketch that we want to capture how the network processed a particular input. One consequence is that if an object does not have a path to the output module in the communication graph, it will be omitted from the overall sketch.

Our theoretical results will primarily focus on a single path from an object θ to the output object. We define the "effective weight" of an object, denoted w_θ, to be the product of weights along this path. For example, if an edge was 10% responsible for a cat and the cat was 50% of the output, then this edge has an effective weight of 5% (for this path through the cat object). We define the "depth" of an object, denoted $h(\theta)$, to be the number of objects along this path. In the preceding example, the output object has a depth of one, the cat object has a depth of two, and the edge object has a depth of three. We make the technical assumption that all objects produced by a module must be at the same depth. As a consequence, modules M also have a depth, denoted $h(M)$. To continue the example, the output *module* has depth one, the cat *module* has depth two, and the edge *module* has depth three. Furthermore, this implies that each object and module can be assigned a unique depth (i.e. all paths from any object produced by a module to the output object have the same number of objects).

Our recursive object sketches are all d_{sketch} dimensional vectors. We assume that d_{sketch} is at least the size of any object attribute vector (i.e. the output of any module). In general, all of our vectors are d_{sketch}-dimensional; we assume that shorter object attribute vectors are zero-padded (and normalized). For example, imagine that the cat object θ is output with the three-dimensional attribute vector $(0.6, 0.0, 0.8)$, but d_{sketch} is five. Then we consider its attribute vector to be $x_\theta = (0.6, 0.0, 0.8, 0.0, 0.0)$.

3. Overview of Sketching Mechanism

In this section, we present our sketching mechanism, which attempts to summarize how a modular deep network understood an input. The mechanism is presented at a high level; see Section 4 for how we arrived at this mechanism and Appendix D for proofs of the results stated in this section.

Symbol	Meaning
b	a bit in $\{\pm 1\}$
d	dimension size
$\mathcal{D}$	distribution over random matrices
f	a coin flip in $\{\pm 1\}$
h	depth (one-indexed)
H	$3 \cdot$ maximum depth
I	identity matrix
k	#inputs
ℓ	ℓ-th moment of a random variable
M	module
N	$3 \cdot \max(\#\text{modules}, \#\text{objects})$
q	block non-zero probability
R	random matrix
s	sketch
S	#samples
w	importance weight
x	attribute vector / unknown dictionary learning vector
y	dictionary learning sample vector
z	dictionary learning noise
δ	error bound, isometry and desynchronization
ϵ	error bound, dictionary learning
θ	object

Table 1. Symbols used in this paper.

3.1. Desiderata

As discussed in the introduction, we want our sketching mechanism to satisfy several properties, listed below.

Attribute Recovery. Object attribute vectors can be approximately recovered from the overall sketch, with additive error that decreases with the effective weight of the object.

Sketch-to-Sketch Similarity. With high probability, two completely unrelated sketches (involving disjoint sets of modules) will have a small inner product. With high probability, two sketches that share common modules and similar objects will have a large inner product (increasing with the effective weights of the objects).

Summary Statistics. If a single module detects several objects, then we can approximately recover summary statistics about them, such as the number of such objects or their mean attribute vector, with additive error that decreases with the effective weight of the objects.

Graceful Erasure. Erasing everything but d'_{sketch}-prefix of the overall sketch yields an overall sketch with the same properties (albeit with larger error).

3.2. Random Matrices

The workhorse of our recursive sketching mechanism is random (square) matrices. We apply these to transform input sketches before incorporating them into the sketch of the current object. We will defer the exact specification of how to generate our random matrices to Section 5, but while reading this section it may be convenient for the reader to think of our random matrices as uniform random *orthonormal matrices*. Our actual distribution choice is similar with respect to the properties needed for the results in this section. One notable parameter for our family of random matrices is $\delta \geq 0$, which expresses how well they allow us to estimate particular quantities with high probability. For both our random matrices and uniform random orthonormal matrices, δ is $\tilde{O}(1/\sqrt{d_{\text{sketch}}})$. We prove various results about the properties of our random matrices in Appendix B.

We will use R with a subscript to denote such a random matrix. Each subscript indicates a fresh draw of a matrix from our

distribution, so every R_1 is always equal to every other R_1 but R_1 and $R_{\text{cat},0}$ are independent. Throughout this section, we will assume that we have access to these matrices when we are operating on a sketch to retrieve information. In Section 5, we show how to recover these matrices given enough samples.

3.3. The Sketching Mechanism

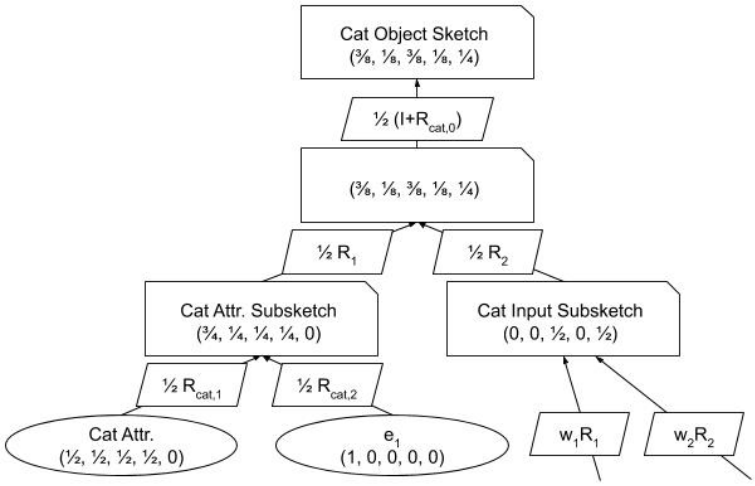

Figure 2. Illustration of how to compute the sketch for a cat object from Figure 1. Attribute vectors (and e_1) are drawn as ovals; note how they have been normalized and padded to $d_{\text{sketch}} = 5$ dimensions. Sketches are drawn as rectangles with a cut corner. The arrows run from vectors used to compute other vectors, and are labeled with the matrix that is applied in the process. For the purposes of this diagram, all random matrices happen to be the identity matrix.

The basic building block of our sketching mechanism is the tuple sketch, which we denote s_{tuple}. As the name might suggest, its purpose is to combine k sketches $s_1, ..., s_k$ with respective weights $w_1, ..., w_k \geq 0$ into a single sketch (for proofs, we will assume that these weights sum to at most one). In the case where the k sketches are input sketches, these weights will be the importance weights from our network. Otherwise, they will all be $1/k$. Naturally, sketches with higher weight can be recovered more precisely. The tuple sketch is formally defined as follows. If their values are obvious from context, we will omit $w_1, ..., w_k$ from the arguments to the tuple sketch. The tuple sketch is computed as follows.[2]

$$s_{\text{tuple}}(s_1, ..., s_k, w_1, ..., w_k) \coloneqq \sum_{i=1}^{k} w_i \left(\tfrac{I+R_i}{2}\right) s_i$$

Note that I is the identity matrix, and we will define a tuple sketch of zero things to be the zero vector.

Each object θ is represented by a sketch, which naturally we will refer to as an object sketch. We denote such a sketch as s_{object}. We want the sketch of object θ to incorporate information about the attributes of θ itself as well as information about the inputs that produced it. Hence we also define two subsketches for object θ. The attribute subsketch, denoted s_{attr}, is a sketch representation of object θ's attributes. The input subsketch, denoted s_{input}, is a sketch representation of the inputs that produced object θ. These three sketches are computed as follows.

$$\begin{aligned}
s_{\text{object}}(\theta) &\coloneqq \left(\tfrac{I+R_{M(\theta),0}}{2}\right) s_{\text{tuple}}\left(s_{\text{attr}}(\theta), s_{\text{input}}(\theta), \tfrac{1}{2}, \tfrac{1}{2}\right) \\
s_{\text{attr}}(\theta) &\coloneqq \tfrac{1}{2} R_{M(\theta),1} x_\theta + \tfrac{1}{2} R_{M(\theta),2} e_1 \\
s_{\text{input}}(\theta) &\coloneqq s_{\text{tuple}}\left(s_{\text{object}}(\theta_1), ..., s_{\text{object}}(\theta_k), w_1, ..., w_k\right)
\end{aligned}$$

Note that e_1 in the attribute subsketch is the first standard basis vector; we will use it for frequency estimation as part of our **Summary Statistics** property. Additionally, $\theta_1, ..., \theta_k$ in the input sketch are the input objects for θ and $w_1, ..., w_k$ are their importance weights from the network.

[2] Rather than taking a convex combination of $\left(\frac{I+R_i}{2}\right) s_i$, one might instead sample a few of them. Doing so would have repercussions on the results in this section, swapping many bounds from high probability to conditioning on their objects getting through. However, it also makes it easier to do the dictionary learning in Section 5.

The overall sketch is just the output psuedo-object's *input subsketch*. It is worth noting that we do not choose to use its object sketch.

$$s_{\text{overall}} \coloneqq s_{\text{input}}(\text{output pseudo-object})$$

We want to use this to (noisily) retrieve the information that originally went into these sketches. We are now ready to present our results for this sketching mechanism. Note that we provide the technical machinery for proofs of the following results in Appendix A and we provide the proofs themselves in Appendix D. Our techniques involve using an approximate version of isometry and reasoning about the repeated application of matrices which satisfy our approximate isometry.

Our first result concerns **Attribute Recovery** (example use case: roughly describe the cat that was in this room).

Theorem 1. *[Simplification of Theorem 9] Our sketch has the simplified **Attribute Recovery** property, which is as follows. Consider an object $\theta^\star$ at constant depth $h(\theta^\star)$ with effective weight $w_{\theta^\star}$.*

(i) Suppose no other objects in the overall sketch are also produced by $M(\theta^\star)$. We can produce a vector estimate of the attribute vector $x_{\theta^\star}$, which with high probability has at most $O(\delta/w_{\theta^\star})$ ℓ_∞-error.
(ii) Suppose we know the sequence of input indices to get to $\theta^\star$ in the sketch. Then even if other objects in the overall sketch are produced by $M(\theta^\star)$, we can still produce a vector estimate of the attribute vector $x_{\theta^\star}$, which with high probability has at most $O(\delta/w_{\theta^\star})$ ℓ_∞-error.

As a reminder, ℓ_∞ error is just the maximum error over all coordinates. Also, note that if we are trying to recover a quantized attribute vector, then we may be able to recover our attribute vector *exactly* when the quantization is larger than our additive error. This next result concerns **Sketch-to-Sketch Similarity** (example use case: judge how similar two rooms are). We would like to stress that the output psuedo-object does not count as an object for the purposes of (i) or (ii) in the next result.

Theorem 2. *[Simplification of Theorem 10] Our sketch has the **Sketch-to-Sketch Similarity** property, which is as follows. Suppose we have two overall sketches s and $\bar{s}$.*

(i) If the two sketches share no modules, then with high probability they have at most $O(\delta)$ dot-product.
(ii) If s has an object $\theta^\star$ of weight $w_{\theta^\star}$ and $\bar{s}$ has an object $\bar{\theta}^\star$ of weight $\bar{w}_{\theta^\star}$, both objects are produced by the same module, and both objects are at constant depth $h(\theta^\star)$ then with high probability they have at least $\Omega(w_{\theta^\star}\bar{w}_{\bar{\theta}^\star}) - O(\delta)$ dot-product.
(iii) If the two sketches are identical except that the attributes of any object θ differ by at most ϵ in ℓ_2 distance, then with probability one the two sketches are at most $\epsilon/2$ from each other in ℓ_2 distance.

Although our **Sketch-to-Sketch Similarity** result is framed in terms of two overall sketches, the recursive nature of our sketches makes it not too hard to see how it also applies to any pair of sketches computed along the way. This is depicted in Figure 3.

This next result concerns **Summary Statistics** (example use case: how many walls are in this room).

Theorem 3. *[Simplification of Theorem 11] Our sketch has the simplified **Summary Statistics** property, which is as follows. Consider a module $M^\star$ which lies at constant depth $h(M^\star)$, and suppose all the objects produced by $M^\star$ has the same effective weight $w^\star$.*

*(i) **Frequency Recovery**. We can produce an estimate of the number of objects produced by $M^\star$. With high probability, our estimate has an additive error of at most $O(\delta/w^\star)$.*
*(ii) **Summed Attribute Recovery**. We can produce a vector estimate of the summed attribute vectors of the objects produced by $M^\star$. With high probability, our estimate has at most $O(\delta/w^\star)$ ℓ_∞-error.*

Again, note that quantization may allow for exact recovery. In this case, the number of objects is an integer and hence can be recovered exactly when the additive error is less than $1/2$. If we can recover the exact number of objects, we can also estimate the mean attribute vector.

Appendix Subsection D.4 explains how to generalize our sketch to incorporate signatures for the **Object Signature Recovery** property and Appendix Subsection D.5 explains how to generalize these results to hold when part of the sketch is erased for the **Graceful Erasure** property.

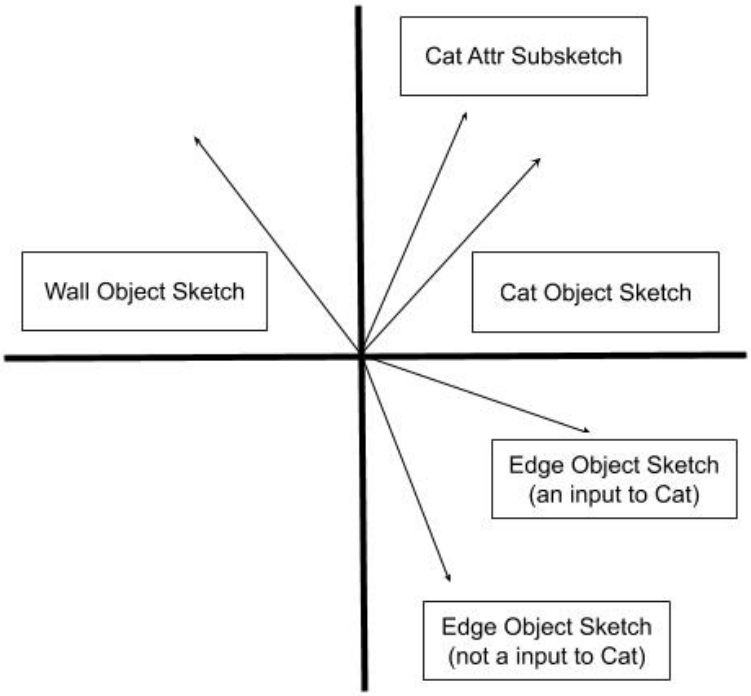

Figure 3. A two-dimensional cartoon depiction of the **Sketch-to-Sketch Similarity** property with respect to the various sketches we produce from a single input. Each vector represents a sketch, and related sketches are more likely to cluster together.

4. The Road to a Sketch

In this section, we present a series of sketch prototypes which gradually grow more complex, capturing more of our desiderata while also considering more complex network possibilities. The final prototype is our actual sketch.

We begin by making the following simplifying assumptions: (i) each object is produced by a unique module, (ii) our network is of depth one (i.e. all objects are direct inputs to the output pseudo-object), and (iii) we can generate random orthonormal matrices R to use in our sketches. In this section, we eventually relax all three assumptions to arrive at our final sketch, which was presented in Section 3. Orthonormal matrices are convenient for us due to the following two properties.

Isometry: If x is a unit vector and R is an orthonormal matrix, then $\left|||Rx||_2^2 - ||x||_2^2\right| = 0$. One nice consequence of this fact is that R^T is the inverse of R.

Desynchronization: If x and y are unit vectors and then R is drawn to be a random orthonormal matrix, then $|\langle Rx, y\rangle| \leq O(\sqrt{\log N}/\sqrt{d_{\text{sketch}}})$ with probability at least $1 - 1/N^{\Omega(1)}$ (i.e. with high probability).

When we later use block sparse random matrices, these properties are noisier and we need to carefully control the magnitude of the resulting noise. Roughly speaking, the absolute values in both definitions will be bounded by a parameter δ (the same δ as in Section 3). In Appendix A, we establish the technical machinery to control this noise.

Proof Ideas. Appendix C contains proof sketches of the claims in this section. At an even higher level, the key idea is as follows. Our sketch properties revolve around taking products of our sketches and random matrices. These products are made up of terms which in turn contain matrix products of the form $R_2^T R_1$. If these two random matrices are the same, then this is the identity matrix by Isometry. If they are independent, then this is a uniform random orthonormal matrix and by desynchronization everything involved in the product will turn into bounded noise. The sharp distinction between these two cases means that Rx encodes the information of x in a way that only applying R^T will retrieve.

Prototype A: A Basic Sketch. Here is our first attempt at a sketch, which we will refer to as prototype A. We use A superscripts to distinguish the definitions for this prototype. The fundamental building block of all our sketches is the tuple sketch, which combines k (weighted) sketches into a single sketch. Its input is k sketches $s_1, ..., s_k$ along with their

nonnegative weights $w_1, ..., w_k \geq 0$ which sum to one. For now, we just take the convex combination of the sketches:

$$s^A_{\text{tuple}}(s_1, ..., s_k, w_1, ..., w_k) \coloneqq \sum_{i=1}^{k} w_i s_i$$

Next, we describe how to produce the sketch of an object θ. Due to assumption (ii), we don't need to incorporate the sketches of input objects. We only need to take into account the object attribute vector. We will keep a random matrix R_M for every module M and apply it when computing object sketches:

$$s^A_{\text{object}}(\theta) \coloneqq R_{M(\theta)} x_\theta$$

Finally, we will specially handle the sketch of our top level object so that it can incorporate its direct inputs (we will not bother with its attributes for now):

$$s^A_{\text{overall}} \coloneqq s^A_{\text{tuple}}(s^A_{\text{object}}(\theta_1), ..., s^A_{\text{object}}(\theta_k))$$

where $\theta_1, ..., \theta_k$ are the direct inputs to the output pseudo-object and $w_1, ..., w_k$ are their importance weights from the network.

Our first result is that if we have the overall sketch s^A_{overall}, we can (approximately) recover the attribute vector for object θ by computing $R^T_{M(\theta)} s^A_{\text{overall}}$.

Claim 1. *Prototype A has the **Attribute Recovery** property: for any object $\theta^\star$, with high probability (over the random matrices), its attribute vector $x_{\theta^\star}$ can be recovered up to at most $O(\log N/(w_{\theta^\star}\sqrt{d_{sketch}}))$ ℓ_∞-error.*

Our next result is that overall sketches which have similar direct input objects will be similar:

Claim 2. *Prototype A has the **Sketch-to-Sketch Similarity** property: (i) if two sketches share no modules, then with high probability they have at most $O(\sqrt{\log N}/\sqrt{d_{sketch}})$ dot-product, (ii) if two sketches share an object $\theta^\star$, which has $w_{\theta^\star}$ in the one sketch and $\bar{w}_{\theta^\star}$ in the other, then with high probability they have at least $w_{\theta^\star}\bar{w}_{\theta^\star} - O(\sqrt{\log N}/\sqrt{d_{sketch}})$ dot-product, (iii) if two sketches have objects produced by identical modules and each pair of objects produced by the same module differ by at most ϵ in ℓ_2 distance and are given equal weights, then they are guaranteed to be at most ϵ from each other in ℓ_2 distance.*

Prototype B: Handling Module Multiplicity. We will now relax assumption (i); a single module may appear multiple times in the communication graph, producing multiple objects. This assumption previously prevented the **Summary Statistics** property from making sense, but it is now up for grabs. Observe that with this assumption removed, computing $R^T_M s^A_{\text{overall}}$ now sums the attribute vectors of *all objects produced by module M*. As a consequence, we will now need a more nuanced version of our **Attribute Recovery** property, but we can leverage this fact for the **Summary Statistics** property. In order to estimate object frequencies, we will (in spirit) augment our object attribute vectors with a entry which is always one. In reality, we draw another random matrix for the module and use its first column.

$$s^B_{\text{object}}(\theta) \coloneqq \tfrac{1}{2} R_{M(\theta),1} x_\theta + \tfrac{1}{2} R_{M(\theta),2} e_1$$

where e_1 is the first standard basis vector in d_{sketch} dimensions.

We still need to fix attribute recovery; how can we recover the attributes of an object that shares a module with another object? We could apply another random matrix to every input in a tuple sketch, but then we would not be able to get summed statistics. The solution is to both apply another random matrix and apply the identity mapping. We will informally refer to $\left(\frac{I+R}{2}\right)$ as the transparent version of matrix R. The point is that when applied to a vector x, it both lets through information about x and sets things up so that R^T will be able to retrieve the same information about x. Our new tuple sketch is:

$$s^B_{\text{tuple}}(s_1, \ldots, s_k, w_1, \ldots, w_k) \coloneqq \sum_{i=1}^{k} w_i \left(\tfrac{I+R_i}{2}\right) s_i$$

Our special overall sketch is computed as before.

$$s^B_{\text{overall}} \coloneqq s^B_{\text{tuple}}(s^B_{\text{object}}(\theta_1), ..., s^B_{\text{object}}(\theta_k))$$

Here is our more nuanced attribute recovery claim.

Claim 3. *Prototype B has the **Attribute Recovery** property, which is as follows. Consider an object $\theta^\star$ with weight $w_{\theta^\star}$. Then (i) if no other objects are produced by the same module in the sketch, with high probability we can recover w_θ up to $O(\frac{\sqrt{\log N}}{w_{\theta^\star}\sqrt{d_{sketch}}})$ ℓ_∞-error and (ii) even if other objects are output by the same module, if we know $\theta^\star$'s index in the overall sketch, with high probability we can recover w_θ up to $O(\frac{\sqrt{\log N}}{w_{\theta^\star}\sqrt{d_{sketch}}})$ ℓ_∞-error.*

Sketches are still similar, and due to the e_1 we inserted into the object sketch, two objects produced by the same module now always have a base amount of similarity.

Claim 4. *Prototype B has the **Sketch-to-Sketch Similarity** property, just as in Claim 2, except in case (ii) the dot-product is at least $\frac{1}{4}w_{\theta^\star}\bar{w}_{\theta^\star} - O(\sqrt{\log N}/\sqrt{d_{sketch}})$, in case (iii) the sketches are at most $\epsilon/2$ from each other in ℓ_2 distance, and we have an additional case (iv) if one sketch has an object $\theta^\star$ of weight $w_{\theta^\star}$ and the other sketch has an object $\bar{\theta}^\star$ of weight $\bar{w}_{\bar{\theta}^\star}$ and both objects are produced by the same module, then with high probability they have at least $\frac{1}{16}w_{\theta^\star}\bar{w}_{\theta^\star} - O(\sqrt{\log N}/\sqrt{d_{sketch}})$.*

Finally, we can recover some summary statistics.

Claim 5. *Prototype B has the **Summary Statistics** property, which is as follows. Consider a module $M^\star$, and suppose all objects produced by $M^\star$ have the same weight in the overall sketch. Then (i) we can estimate the number of objects produced by $M^\star$ up to an additive $O(\frac{\sqrt{\log N}}{w^\star\sqrt{d_{sketch}}})$ and (ii) we can estimate the summed attribute vector of objects produced by $M^\star$ up to $O(\frac{\sqrt{\log N}}{w^\star\sqrt{d_{sketch}}})$ ℓ_∞-error.*

Last Steps: Handling Deep Networks. We conclude by describing how to get from prototype B to the sketch we presented in Section 3. We have two assumptions left to relax: (ii) and (iii). We gloss over how to relax assumption (iii) here, since the push back comes from the requirements of dictionary learning in Section 5, and the exact choice of random matrices is not so illuminating for this set of results. Instead, we focus on how to relax assumption (ii) and make our sketches recursive. As a first point, we'll be keeping the tuple sketch from prototype B:

$$s_{\text{tuple}}(s_1, ..., s_k, w_1, ..., w_k) \coloneqq \sum_{i=1}^{k} w_i \left(\tfrac{I+R_i}{2}\right) s_i \qquad (= s^B_{\text{tuple}}(s_1, ..., s_k, w_1, ..., w_k))$$

The main difference is that we now want object sketches to additionally capture information about their input objects. We define two important subsketches for each object, one to capture its attributes and one to capture its inputs. The attribute subsketch of an object what was the object sketch in prototype B:

$$s_{\text{attr}}(\theta) \coloneqq \tfrac{1}{2}R_{M(\theta),1}x_\theta + \tfrac{1}{2}R_{M(\theta),2}e_1 \qquad (= s^B_{\text{object}}(\theta))$$

The input subsketch is just a tuple sketch:

$$s_{\text{input}}(\theta) \coloneqq s_{\text{tuple}}\left(s_{\text{object}}(\theta_1), ..., s_{\text{object}}(\theta_k), w_1, ..., w_k\right)$$

The object sketch just tuple sketches together the two subsketches and applies a transparent matrix. The idea is that we might both want to query over the input object's module, or both the current object's module and the input object's module (e.g. summary statistics about all edges versus just edges that wound up in a cat). The transparent matrix means the information gets encoded both ways.

$$s_{\text{object}}(\theta) \coloneqq \left(\tfrac{I+R_{M(\theta),0}}{2}\right) s_{\text{tuple}}\left(s_{\text{attr}}(\theta), s_{\text{input}}(\theta), \tfrac{1}{2}, \tfrac{1}{2}\right)$$

One nice consequence of all this is that we can now stop special-casing the overall sketch, since we have defined the notion of input subsketches. Note that we *do not want* the overall sketch to be the output pseudo-object's object sketch, because that would incorporate a $R_{\text{output},2}e_1$, which in turn gives all sketches a base amount of dot-product similarity that breaks the current **Sketch-to-Sketch Similarity** property.

$$s_{\text{overall}} \coloneqq s_{\text{input}}(\text{output pseudo-object})$$

5. Learning Modular Deep Networks via Dictionary Learning

In this section, we demonstrate that our sketches carry enough information to learn the network used to produce them. Specifically, we develop a method for training a new network based on (input, output, sketch) triples obtained from a teacher modular deep network. Our method is powered by a novel *dictionary learning* procedure. Loosely speaking, dictionary learning tries to solve the following problem. There is an unknown dictionary matrix R, whose columns are typically referred to as atoms. There is also a sequence of unknown sparse vectors $x^{(k)}$; we only observe how they combine the atoms, i.e., $\{y^{(k)} = Rx^{(k)}\}$. We want to use these observations $y^{(k)}$ to recover the dictionary matrix R and the original sparse vectors $x^{(k)}$.

While dictionary learning has been well-studied in the literature from both an applied and a theoretical standpoint, our setup differs from known theoretical results in several key aspects. The main complication is that since we want to apply our dictionary learning procedure recursively, our error in recovering the unknown vectors $x^{(k)}$ will become noise on the observations $y^{(k)}$ in the very next step. Note that classical dictionary learning can only recover the unknown vectors $x^{(k)}$ up to permutation and coordinate-wise sign. To do better, we will carefully engineer our distribution of dictionary matrices R to allow us to infer the permutation (between the columns of a matrix) and signs, which is needed to recurse. Additionally, we want to allow very general unknown vectors $x^{(k)}$. Rather than assuming sparsity, we instead make the weaker assumption that they have bounded ℓ_2 norm. We also allow for distributions of $x^{(k)}$ which make it impossible to recover all columns of R; we simply recover a subset of essential columns.

With this in mind, our desired dictionary learning result is Theorem 4. We plan to apply this theorem with N as thrice the maximum between the number of modules and the number of objects in any communication graph, $\underset{\sim}{S}$ as the number of samples necessary to learn a module, and H as three times the depth of our modular network. Additionally, we think of ϵ_1 as the tolerable input ℓ_∞ error while ϵ_H is the desired output ℓ_∞ error after recursing H times.

Theorem 4. *[Recursable Dictionary Learning] There exists a family of distributions $\{\mathcal{D}(b, q, d_{sketch})\}$ which produce $d_{sketch} \times d_{sketch}$ matrices satisfying the following. For any positive integers $N, \underset{\sim}{S}$, positive constant H, positive real ϵ_H, block size $b \geq poly(\log N, \log d_{sketch}, 1/\epsilon_H)$, nonzero block probability $q \geq poly(\log N, \log d_{sketch}, 1/\epsilon_H)/\sqrt{d_{sketch}}$, and dimension $d_{sketch} \geq poly(1/\epsilon_H, \log N, \log \underset{\sim}{S})$, there exists:*

- *a base number of samples S where $\underset{\sim}{S} \leq S \leq \underset{\sim}{S} \cdot poly(N)$,*
- *and a sequence of ℓ_∞ errors $(0 <)\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_{H-1}(\leq \epsilon_H)$ with $\epsilon_1 \geq poly(\epsilon_H)$,*

such that the following is true. For any $h \in [H-1]$, let $S_h = SN^{h-1}$. For any unknown vectors $x^{(1)}, ..., x^{(S_h)} \in \mathbb{R}^{d_{sketch} \times N}$ with ℓ_2 at most $O(1)$, if we draw $R_1, ..., R_N \sim \mathcal{D}(d_{sketch})$ and receive S_h noisy samples $y^{(k)} := [R_1 R_2 \cdots R_N]x^{(k)} + z_1^{(k)} + z_\infty^{(k)}$ where each $z_1^{(k)} \in \mathbb{R}^{d_{sketch}}$ is noise with $\left|\left|z_1^{(k)}\right|\right|_1 \leq O(\sqrt{d_{sketch}})$ (independent of our random matrices) and each $z_\infty^{(k)} \in \mathbb{R}^{d_{sketch}}$ is noise with $\left|\left|z_\infty^{(k)}\right|\right|_\infty \leq \epsilon_h$ (also independent of our random matrices), then there is an algorithm which takes as input h, $y^{(1)}$, ..., $y^{(S_h)}$, runs in time $poly(S_h, d_{sketch})$, and with high probability outputs $\hat{R}_1, ..., \hat{R}_N, \hat{x}^{(1)}, ..., \hat{x}^{(S_h)}$ satisfying the following for some permutation π of $[N]$:

- *for every $i \in [N]$ and $j \in [d_{sketch}]$, if there exists $k^\star \in [S_h]$ such that $\left|x^{(k^\star)}_{(i-1)d_{sketch}+j}\right| \geq \epsilon_{h+1}$ then the j^{th} column of $\hat{R}_{\pi(i)}$ is $0.2d_{sketch}$ in Hamming distance from the j^{th} column of R_i.*
- *for every $k \in [S_h], i \in [N], j \in [d_{sketch}]$, $\left|x^{(k)}_{(\pi(i)-1)d_{sketch}+j} - x^{(k)}_{(i-1)d_{sketch}+j}\right| \leq \epsilon_{h+1}$.*

5.1. Recursable Dictionary Learning Implies Network Learnability

We want to use Theorem 4 to learn a teacher modular deep network, but we need to first address an important issue. So far, we have not specified how the deep modular network decides upon its input-dependent communication graph. As a result, the derivation of the communication graph cannot be learned (it's possibly an uncomputable function!). When the communication graph is a fixed tree (always the same arrangement of objects but with different attributes), we can learn it. Note that any fixed communication graph can be expanded into a tree; doing so is equivalent to not re-using computation. Regardless of the communication graph, we can learn the input/output behavior of each module regardless of whether the communication graph is fixed.

Theorem 5. *If the teacher deep modular network has constant depth, then any module $M^\star$ which satisfies the following two properties:*

- *(Robust Module Learnability) The module is learnable from $(\alpha = poly(N))$ input/output training pairs which have been corrupted by ℓ_∞ error at most a constant $\epsilon > 0$.*
- *(Sufficient Weight) In a $\left(\beta = \frac{1}{poly(N)}\right)$-fraction of the inputs, the module produces an object and all of the input objects to that object have effective weight at least w.*

can, with high probability, be learned from $poly(N)$ overall sketches of dimension $poly(1/w, 1/\epsilon) \log^2 N$.

Suppose we additionally know that the communication graph is a fixed tree. We can identify the sub-graph of objects which each have effective weight w in a $\left(\beta = \frac{1}{poly(N)}\right)$-fraction of the inputs.

Theorem 5 is proved in Appendix E. The main idea is to just repeatedly apply our recursable dictionary learning algorithm and keep track of which vectors correspond to sketches and which vectors are garbage.

5.2. Recursable Dictionary Learning: Proof Outline

The main idea of our recursable dictionary learning algorithm is the following. The algorithm proceeds by iteratively examining each of the d_{sketch}/b blocks of each of the S_h received samples. For the ℓth block of the ith sample, denoted by $y^{(i)}[(\ell-1)b+1:\ \ell b+1]$, it tries to determine whether it is dominated by the contribution of a single $(\sigma_s, \sigma_c, \sigma_m)$ (and is not composed of the superposition of more than one of these). To do so, it first normalizes this block by its ℓ_1-norm and checks if the result is close to a point on the Boolean hypercube in ℓ_∞ distance. If so, it rounds (the ℓ_1-normalized version of) this block to the hypercube; we here denote the resulting rounded vector by $\overline{y}^{(i)}[(\ell-1)b+1:\ \ell b+1]$. We then count the number of blocks $\ell' \in [d_{\text{sketch}}/b]$ whose rounding $\overline{y}^{(i)}[(\ell'-1)b+1:\ \ell' b+1]$ is close in Hamming distance to $\overline{y}^{(i)}[(\ell-1)b+1:\ \ell b+1]$. If there are at least $0.8qd_{\text{sketch}}$ of these, we add the rounded block $\overline{y}^{(i)}[(\ell-1)b+1:\ \ell b+1]$ to our collection (if it's not close to an already added vector). We also cluster the added blocks in terms of their matrix signatures σ_m in order to associate each of them to one of the matrices $R_1, \ldots, R_N$. Using the matrix signatures as well as the column signatures $\{\sigma_c\}$ allows us to recover all the essential columns of the original matrices. The absolute values of the x coordinates can also be inferred by adding the ℓ_1-norm of a block when adding its rounding to the collection. The signs of the x coordinates can also be inferred by first inferring the true sign of the block and comparing it to the sign of the vector whose rounding was added to the collection.

5.3. The Block-Sparse Distribution $\mathcal{D}$

We are now ready to define our distribution $\mathcal{D}$ on random (square) matrices R. It has the following parameters:

- The block size $b \in \mathbb{Z}^+$ which must be a multiple of 3 and at least $3\max(\lceil \log_2 N \rceil, \lceil \log_2 d_{\text{sketch}} \rceil + 3)$.
- The block sampling probability $q \in [0, 1]$; this is the probability that a block is nonzero.
- The number of rows/columns $d_{\text{sketch}} \in \mathbb{Z}^+$. It must be a multiple of b, as each column is made out of blocks.

Each column of our matrix is split into d_{sketch}/b blocks of size b. Each block contains three sub-blocks: a random string, the matrix signature, and the column signature. To specify the column signature, we define an encoding procedure Enc which maps two bits b_m, b_s and the column index into a $(b/3)$-length sequence. $\text{Enc} : \{\pm 1\}^2 \times [d_{\text{sketch}}] \to \{\pm \frac{1}{\sqrt{d_{\text{sketch}} q}}\}^{b/3}$, which is presented as Algorithm 1. These two bits encode the parity of the other two sub-blocks, which will aid our dictionary learning procedure in recovery of the correct signs. The sampling algorithm which describes our distribution is presented as Algorithm 2.

$\diamondsuit$ blocks, each with b entries

$$\left\{\begin{bmatrix}
f_{s,1,1}\cdot\sigma_{s,1} & f_{s,1,2}\cdot\sigma_{s,2} & f_{s,1,3}\cdot\sigma_{s,3} & \vec{0} & \cdots & f_{s,1,d_{\text{sketch}}}\cdot\sigma_{s,d_{\text{sketch}}} \\
f_{c,1,1}\cdot\sigma_{c,1,1} & f_{c,1,2}\cdot\sigma_{c,1,2} & f_{c,1,3}\cdot\sigma_{c,1,3} & \vec{0} & \cdots & f_{c,1,d_{\text{sketch}}}\cdot\sigma_{c,1,d_{\text{sketch}}} \\
f_{m,1,1}\cdot\sigma_m & f_{m,1,2}\cdot\sigma_m & f_{m,1,3}\cdot\sigma_m & \vec{0} & \cdots & f_{m,1,d_{\text{sketch}}}\cdot\sigma_m \\
\hline
\vec{0} & f_{s,2,2}\cdot\sigma_{s,2} & \vec{0} & \vec{0} & \cdots & \vec{0} \\
\vec{0} & f_{c,2,2}\cdot\sigma_{c,2,2} & \vec{0} & \vec{0} & \cdots & \vec{0} \\
\vec{0} & f_{m,2,2}\cdot\sigma_m & \vec{0} & \vec{0} & \cdots & \vec{0} \\
\hline
f_{s,3,1}\cdot\sigma_{s,1} & f_{s,3,2}\cdot\sigma_{s,2} & f_{s,3,3}\cdot\sigma_{s,3} & f_{s,3,4}\cdot\sigma_{s,4} & \cdots & \vec{0} \\
f_{c,3,1}\cdot\sigma_{c,3,1} & f_{c,3,2}\cdot\sigma_{c,3,2} & f_{c,3,3}\cdot\sigma_{c,3,3} & f_{c,3,4}\cdot\sigma_{c,3,4} & \cdots & \vec{0} \\
f_{m,3,1}\cdot\sigma_m & f_{m,3,2}\cdot\sigma_m & f_{m,3,3}\cdot\sigma_m & f_{m,3,4}\cdot\sigma_m & \cdots & \vec{0} \\
\hline
f_{s,4,1}\cdot\sigma_{s,1} & f_{s,4,2}\cdot\sigma_{s,2} & f_{s,4,3}\cdot\sigma_{s,3} & \vec{0} & \cdots & f_{s,4,d_{\text{sketch}}}\cdot\sigma_{s,d_{\text{sketch}}} \\
f_{c,4,1}\cdot\sigma_{c,4,1} & f_{c,4,2}\cdot\sigma_{c,4,2} & f_{c,4,3}\cdot\sigma_{c,4,3} & \vec{0} & \cdots & f_{c,4,d_{\text{sketch}}}\cdot\sigma_{c,4,d_{\text{sketch}}} \\
f_{m,4,1}\cdot\sigma_m & f_{m,4,2}\cdot\sigma_m & f_{m,4,3}\cdot\sigma_m & \vec{0} & \cdots & f_{m,4,d_{\text{sketch}}}\cdot\sigma_m \\
\hline
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\hline
\vec{0} & \vec{0} & f_{s,\diamondsuit,3}\cdot\sigma_{s,3} & f_{s,\diamondsuit,4}\cdot\sigma_{s,4} & \cdots & \vec{0} \\
\vec{0} & \vec{0} & f_{c,\diamondsuit,3}\cdot\sigma_{c,\diamondsuit,3} & f_{c,\diamondsuit,4}\cdot\sigma_{c,\diamondsuit,4} & \cdots & \vec{0} \\
\vec{0} & \vec{0} & f_{m,\diamondsuit,3}\cdot\sigma_m & f_{m,\diamondsuit,4}\cdot\sigma_m & \cdots & \vec{0}
\end{bmatrix}\right.$$

Figure 4. Example block-random matrix from distribution $\mathcal{D}(b, q, d_{\text{sketch}})$. $\diamondsuit \coloneqq d_{\text{sketch}}/b$ denotes the number of blocks.

Algorithm 1 $\text{Enc}(j, b_m, b_s)$

Input: Column index $j \in [d_{\text{sketch}}]$, two bits $b_m, b_s \in \{\pm 1\}$, .
Output: A vector $\sigma_c \in \{\pm\frac{1}{\sqrt{d_{\text{sketch}}q}}\}^{b/3}$.

Set σ_c to be the (zero-one) binary representation of $j-1$ (a $\lceil \log_2 d_{\text{sketch}} \rceil$-dimensional vector).
Replace each zero in σ_c with -1 and each one in σ with $+1$.
Prepend σ_c with a $+1$, making it a $(\lceil \log_2 d_{\text{sketch}} \rceil + 1)$-dimensional vector.
Append σ_c with (b_m, b_s), making it a $(\lceil \log_2 d_{\text{sketch}} \rceil + 3)$-dimensional vector.
Append σ_c with enough $+1$ entries to make it a $(b/3)$-dimensional vector. Note that $b \geq 3\,(\lceil \log_2 d_{\text{sketch}} \rceil + 3)$.
Divide each entry of σ_c by $\sqrt{d_{\text{sketch}}q}$.
return σ_c.

Algorithm 2 $\mathcal{D}(b, q, d_{\text{sketch}})$

Input: Block size $b \in \mathbb{Z}^+$, block sampling probability $q \in [0, 1]$, number of rows/columns d_{sketch}.
Output: A matrix $R \in \mathbb{R}^{d_{\text{sketch}} \times d_{\text{sketch}}}$.

Initialize R to be the all-zeros $\mathbb{R}^{d_{\text{sketch}} \times d_{\text{sketch}}}$ matrix.
Sample a "matrix signature" vector σ_m from the uniform distribution over $\{\pm\frac{1}{\sqrt{d_{\text{sketch}}q}}\}^{b/3}$.
for column $j \in [d_{\text{sketch}}]$ **do**
  Sample a "random string" vector $\sigma_{s,j}$ from the uniform distribution over $\{\pm\frac{1}{\sqrt{d_{\text{sketch}}q}}\}^{b/3}$.
  for block $i \in [d_{\text{sketch}}/b]$ **do**
    Sample three coin flips $f_{m,i,j}, f_{s,i,j}, f_{c,i,j}$ as independent Rademacher random variables (i.e. uniform over $\{\pm 1\}$).
    Compute two bits $f'_{m,i,j} \coloneqq f_{m,i,j} \cdot \text{sign}(\sigma_m[1])$ and $f'_{s,i,j} \coloneqq f_{s,i,j}s \cdot \text{sign}(\sigma_{s,j}[1])$.
    Compute a "column signature" vector $\sigma_{c,i,j} \coloneqq \text{Enc}(j, f'_{m,i,j}, f'_{s,i,j})$.
    Sample $\eta_{i,j}$ to be a (zero-one) Bernoulli random variable which is one with probability q.
    if $\eta_{i,j} = 1$ **then**
      Set $R[b(i-1)+1 : bi+1, j]$ to be $f_{s,i,j} \cdot \sigma_{s,j}$ concatenated with $f_{c,i,j} \cdot \sigma_{c,i,j}$ concatenated with $f_{m,i,j} \cdot \sigma_m$.
    end if
  end for
end for
return R.

6. Conclusion

In this paper, we presented a sketching mechanism which captures several natural properties. Two potential areas for future work are our proposed use cases for this mechanism: adding a sketch repository to an existing modular network and using it

to find new concepts, and using our sketches to learn a teacher network.

References

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

Arora, S., Bhaskara, A., Ge, R., and Ma, T. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pp. 584–592, 2014a.

Arora, S., Bhaskara, A., Ge, R., and Ma, T. Provable bounds for learning some deep representations. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 584–592, Bejing, China, 22–24 Jun 2014b. PMLR. URL http://proceedings.mlr.press/v32/arora14.html.

Arora, S., Ge, R., and Moitra, A. New algorithms for learning incoherent and overcomplete dictionaries. In *Conference on Learning Theory*, pp. 779–806, 2014c.

Arora, S., Ge, R., Ma, T., and Moitra, A. Simple, efficient, and neural algorithms for sparse coding. In *Proceedings of The 28th Conference on Learning Theory*, pp. 113–149, 2015.

Azam, F. *Biologically inspired modular neural networks*. PhD thesis, Virginia Tech, 2000.

Brannon, E. M. and Roitman, J. D. Nonverbal representations of time and number in animals and human infants. 2003.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541. ACM, 2006.

Clune, J., Mouret, J.-B., and Lipson, H. The evolutionary origins of modularity. *Proc. R. Soc. B*, 280(1755):20122863, 2013.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., and Graves, A. Associative long short-term memory. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pp. 1986–1994. JMLR.org, 2016.

Du, S. S., Lee, J. D., and Tian, Y. When is a convolutional filter easy to learn? *arXiv preprint arXiv:1709.06129*, 2017a.

Du, S. S., Lee, J. D., Tian, Y., Poczos, B., and Singh, A. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779*, 2017b.

Ellis, K., Ritchie, D., Solar-Lezama, A., and Tenenbaum, J. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems*, pp. 6059–6068, 2018.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Gayler, R. W. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. *arXiv preprint cs/0412059*, 2004.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. Learning to transduce with unbounded memory. In *Advances in neural information processing systems*, pp. 1828–1836, 2015.

Hanson, D. L. and Wright, F. T. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42(3):1079–1083, 1971.

Hawkins, J. and Blakeslee, S. *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan, 2007.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hinton, G. E., Ghahramani, Z., and Teh, Y. W. Learning to parse images. In *Advances in neural information processing systems*, pp. 463–469, 2000.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *J. American Statist. Assoc.*, 58:13 – 30, 03 1963. doi: 10.2307/2282952.

Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 804–813, 2017.

Joulin, A. and Mikolov, T. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems*, pp. 190–198, 2015.

Kane, D. M. and Nelson, J. Sparser johnson-lindenstrauss transforms. *J. ACM*, 61(1):4:1–4:23, January 2014. ISSN 0004-5411. doi: 10.1145/2559902. URL http://doi.acm.org/10.1145/2559902.

Levy, S. D. and Gayler, R. Vector symbolic architectures: A new building material for artificial general intelligence. In *Proceedings of the 2008 Conference on Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, pp. 414–418. IOS Press, 2008.

Li, Y. and Yuan, Y. Convergence analysis of two-layer neural networks with ReLU activation. *arXiv preprint arXiv:1705.09886*, 2017.

Oh, J., Singh, S., Lee, H., and Kohli, P. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2661–2670. JMLR. org, 2017.

Papyan, V., Romano, Y., Sulam, J., and Elad, M. Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks. *IEEE Signal Processing Magazine*, 35(4): 72–89, 2018.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017.

Smolensky, P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

Spielman, D. A., Wang, H., and Wright, J. Exact recovery of sparsely-used dictionaries. In *Conference on Learning Theory*, pp. 37–1, 2012.

Sukhbaatar, S., Weston, J., Fergus, R., et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Wu, J., Tenenbaum, J. B., and Kohli, P. Neural scene de-rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., and Tenenbaum, J. B. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

Zaremba, W. and Sutskever, I. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521*, 2015.

Zhang, Q., Panigrahy, R., Sachdeva, S., and Rahimi, A. Electron-proton dynamics in deep learning. *arXiv preprint arXiv:1702.00458*, 2017.

Zhong, K., Song, Z., and Dhillon, I. S. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017a.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. *arXiv preprint arXiv:1706.03175*, 2017b.

A. Technical Machinery for Random Matrices

In this appendix, we present several technical lemmas which will be useful for analyzing the random matrices that we use in our sketches.

A.1. Notation

Definition 1. *Suppose we have two fixed vectors $x, y \in \mathbb{R}^{d_{sketch}}$ and random matrices $R_1, \ldots, R_k$ drawn independently from distributions $\mathcal{D}_1, ..., \mathcal{D}_k$. We have a random variable Z which is a function of x, y, and $R_1, \ldots, R_k$. Z is a δ-nice noise variable with respect to D if the following two properties hold:*

1. *(Centered Property) $\mathbb{E}_{R_1 \sim \mathcal{D}_1, \ldots R_k \sim \mathcal{D}_k}[Z] = 0$.*
2. *(δ-bounded Property) With high probability (over $R_1 \sim \mathcal{D}_1, \ldots R_k \sim \mathcal{D}_k$), $|Z| \leq \delta \cdot ||x||_2 \cdot ||y||_2$.*

Definition 2. *We say that a distribution $\mathcal{D}$ has the α_D-Leaky δ_D-Desynchronization Property if the following is true. If $R \sim \mathcal{D}$, then the random variable $Z = \langle Rx, y\rangle - \alpha_D \langle x, y\rangle$ is a δ_D-nice noise variable. If $\alpha_D = 0$ then we simply refer to this as the δ_D-Desynchronization Property.*

Definition 3. *We say that a distribution $\mathcal{D}$ has the α_I-scaled δ_I-Isometry Property if the following is true. If $R \sim \mathcal{D}$ and $x = y$, then the random variable $Z = \langle Rx, Rx\rangle - \alpha_I \langle x, x\rangle$ is a δ_I-nice noise variable. If $\alpha_I = 1$ then we simply refer to this as the δ_I-Isometry Property.*

Since our error bounds scale with the ℓ_2 norm of vectors, it will be useful to use isometry to bound the latter using the following technical lemma.

Lemma 1. *If $\mathcal{D}$ satisfies the α_I-scaled δ_I-Isometry Property, then with high probability:*

$$||Rx||_2^2 \leq (\alpha_I + \delta_I) ||x||_2^2$$

Proof. Consider the random variable:

$$Z = \langle Rx, Rx\rangle - \alpha_I \langle x, x\rangle$$

Since $\mathcal{D}$ satisfies the α_I-scaled δ_I-Isometry Property, Z is a δ_I-nice random variable. Hence with high probability:

$$\langle Rx, Rx\rangle - \alpha_I \langle x, x\rangle \leq \delta_I ||x||_2^2$$
$$\langle Rx, Rx\rangle \leq (\alpha_I + \delta_I) ||x||_2^2$$

This completes the proof. □

While isometry compares $\langle Rx, Rx\rangle$ and $\langle x, x\rangle$, sometimes we are actually interested the more general comparison between $\langle Rx, Ry\rangle$ and $\langle x, y\rangle$. The following technical lemma lets get to the general case at the cost of a constant blowup in the noise.

Lemma 2. *If $\mathcal{D}$ satisfies the α_I-scaled δ_I-Isometry Property, then the random variable $Z = \langle Rx, Ry\rangle - \alpha_I \langle x, y\rangle$ is a $(3\delta_I)$-nice noise variable.*

Proof. This proof revolves around the scalar fact that $(a + b)^2 = a^2 + 2ab + b^2$, so from squaring we can get to products.

This claim is trivially true if x or y is the zero vector. Hence without loss of generality, we can scale x and y to unit vectors, since being a nice noise variable is scale invariant.

We invoke the α_I-scaled δ_I-bounded Isometry Property three times, on $x_1 = x + y$, $x_2 = x$, and $x_3 = y$. This implies that the following three noise variables are δ_I-nice:

$$Z_1 = \langle R(x + y), R(x + y)\rangle - \alpha_I \langle x + y, x + y\rangle$$
$$Z_2 = \langle Rx, Rx\rangle - \alpha_I \langle x, x\rangle$$
$$Z_3 = \langle Ry, Ry\rangle - \alpha_I \langle y, y\rangle$$

We now relate our four noise variables of interest.

$$\begin{aligned} Z_1 &= [\langle Rx, Rx\rangle + 2\langle Rx, Ry\rangle + \langle Ry, Ry\rangle] \\ &\quad - \alpha_I [\langle x, x\rangle + 2\langle x, y\rangle + \langle y, y\rangle] \\ &= Z_2 + 2Z + Z_3 \\ Z &= \frac{1}{2}Z_1 - \frac{1}{2}Z_2 - \frac{1}{2}Z_3 \end{aligned}$$

Hence, Z is centered. We can bound it by analyzing the ℓ_2 lengths of the vectors that we applied isometry to. With high probability:

$$\begin{aligned} |Z_1| &\leq \delta_I ||x+y||_2^2 \\ &\leq \delta_I \left(||x||_2 + ||y||_2\right)^2 \\ &\leq 4\delta_I \\ |Z_2| &\leq \delta_I ||x||_2^2 \\ &\leq \delta_I \\ |Z_3| &\leq \delta_I ||y||_2^2 \\ &\leq \delta_I \end{aligned}$$

Hence Z is also $(3\delta_I)$-bounded, making it a $(3\delta_I)$-noise variable, completing the proof. □

A.2. Transparent Random Matrices

For our sketch, we sometimes want to both rotate a vector and pass it through as-is. For these cases, the distribution TRANSPARENT $(\mathcal{D})$ is useful and defined as follows. We draw R from D and return $\bar{R} = \left(\frac{I+R}{2}\right)$. This subsection explains how the desynchronization and isometry of $\mathcal{D}$ carry over to TRANSPARENT $(\mathcal{D})$. This technical lemma shows how to get desynchronization.

Lemma 3. *If $\mathcal{D}$ satisfies the δ_D-Desynchronization Property, then* TRANSPARENT $(\mathcal{D})$ *satisfies the $(1/2)$-leaky $(\delta_D/2)$-Desynchronization Property.*

Proof. We want to show that if $\bar{R}$ is drawn from TRANSPARENT $(\mathcal{D})$, then the following random variable Z is a $(\delta_D/2)$-nice noise variable.

$$\begin{aligned} Z &= \left\langle \bar{R}x, y\right\rangle - \frac{1}{2}\langle x, y\rangle \\ &= x^T\left(\frac{I+R^T}{2}\right)y - \frac{1}{2}\langle x, y\rangle \\ &= \frac{1}{2}x^Ty + \frac{1}{2}x^TR^Ty - \frac{1}{2}\langle x, y\rangle \\ &= \frac{1}{2}\underbrace{[x^TR^Ty]}_{\text{Let this be } Z'.} \end{aligned}$$

Since $\mathcal{D}$ satisfies the δ_D-Desynchronization Property, we know Z' is a δ_D-nice noise variable. Our variable Z is just a scaled version of Z', making it a $(\delta_D/2)$-nice noise variable. This completes the proof. □

This next technical lemma shows how to get isometry.

Lemma 4. *If $\mathcal{D}$ satisfies the δ_D-Desynchronization Property and the δ_I-Isometry Property, then* TRANSPARENT $(\mathcal{D})$ *satisfies the $\frac{1}{2}$-scaled $(\frac{1}{2}\delta_D + \frac{1}{4}\delta_I)$-Isometry Property.*

Proof. We want to show that if $\bar{R}$ is drawn from TRANSPARENT $(\mathcal{D})$, and $x = y$, then the following random variable Z is a $\left(\delta_D + \frac{1}{2}\delta_I\right)$-nice noise variable:

$$\begin{aligned} Z &= \left\langle \bar{R}x, \bar{R}x \right\rangle - \frac{1}{2}\left\langle x, x \right\rangle \\ &= x^T \left(\frac{I + R^T}{2}\right)\left(\frac{I + R}{2}\right) x - \frac{1}{2}x^T x \\ &= \frac{1}{4}\left(x^T x + x^T R x + x^T R^T x + x^T R^T R x\right) - \frac{1}{2}x^T x \\ &= \frac{1}{2}\underbrace{\left\langle Rx, x \right\rangle}_{\text{Let this be } Z_1.} + \frac{1}{4}\underbrace{\left(\left\langle Rx, Rx \right\rangle - \left\langle x, x \right\rangle\right)}_{\text{Let this be } Z_2.} \end{aligned}$$

Since $\mathcal{D}$ satisfies the δ_D-Desynchronization Property, we know Z_1 is a δ_D-nice noise variable. Since $\mathcal{D}$ satisfies the δ_I-Isometry Property, we know Z_2 is a δ_I-nice noise variable.

This makes our variable Z a $(\frac{1}{2}\delta_D + \frac{1}{4}\delta_I)$-nice noise variable, as desired. □

A.3. Products of Random Matrices

For our sketch, we wind up multiplying matrices. In this subsection, we try to understand the properties of the resulting product. The distribution PRODUCT $(\mathcal{D}_1, \mathcal{D}_2)$ is as follows. We independently draw $R_1 \sim \mathcal{D}_1, R_2 \sim \mathcal{D}_2$ and then return:

$$\tilde{R} = R_2 R_1$$

. This technical lemma says that the resulting product is desynchronizing.

Lemma 5. *If $\mathcal{D}_1$ satisfies the $\delta_{D,1}$-Desynchronization Property, $\mathcal{D}_1$ satisfies the $\delta_{I,1}$-Isometry Property, and $\mathcal{D}_2$ satisfies the $\delta_{D,2}$-leaky $\alpha_{D,2}$-Desynchronization Property, then* PRODUCT $(\mathcal{D}_1, \mathcal{D}_2)$ *satisfies the $\left[\delta_{D,1} + \delta_{D,2}\sqrt{1 + \delta_{I,1}}\right]$-Desynchronization Property.*

Proof. We want to show that if $\tilde{R}$ is drawn from PRODUCT $(\mathcal{D}_1, \mathcal{D}_2)$, then the following random variable Z is a $\left[\delta_{D,1} + \delta_{D,2}\sqrt{1 + \delta_{I,1}}\right]$-nice noise variable.

$$\begin{aligned} Z &= \left\langle \tilde{R}x, y \right\rangle - \left\langle x, y \right\rangle \\ &= \left\langle R_2 R_1 x, y \right\rangle - \left\langle x, y \right\rangle \end{aligned}$$

We now define some useful (noise) variables.

$$\begin{aligned} Z_1 &\coloneqq \left\langle R_1 x, y \right\rangle - \left\langle x, y \right\rangle \\ Z_2 &\coloneqq \left\langle R_2 R_1 x, y \right\rangle - \left\langle R_1 x, y \right\rangle \\ Z &= Z_1 + Z_2 \end{aligned}$$

From our assumptions about $\mathcal{D}_1$ and $\mathcal{D}_2$, we know that Z_1 is a $\delta_{D,1}$-nice noise variable and Z_2 is a $\delta_{D,2}$-nice noise variable. By Lemma 1 we know that with high probability, $||R_1 x||_2^2 \leq 1 + \delta_{I,1}$. Hence, with high probability:

$$\begin{aligned} |Z_1| &\leq \delta_{D,1} \left|\left|x\right|\right|_2 \left|\left|y\right|\right|_2 \\ |Z_2| &\leq \delta_{D,2}\sqrt{1 + \delta I, 1} \left|\left|x\right|\right|_2 \left|\left|y\right|\right|_2 \\ |Z| &\leq |Z_1| + |Z_2| \\ &\leq \left[\delta_{D,1} + \delta_{D,2}\sqrt{1 + \delta_{I,1}}\right] \left|\left|x\right|\right|_2 \left|\left|y\right|\right|_2 \end{aligned}$$

Hence Z is a $\left[\delta_{D,1} + \delta_{D,2}\sqrt{1 + \delta_{I,1}}\right]$-nice noise variable, completing the proof. □

This technical lemma says that the resulting product is isometric.

Lemma 6. *If $\mathcal{D}_1$ satisfies the $\alpha_{I,1}$-scaled $\delta_{I,1}$-Isometry Property and $\mathcal{D}_2$ satisfies the $\alpha_{I,2}$-scaled $\delta_{I,2}$-Isometry Property, then* PRODUCT $(\mathcal{D}_1, \mathcal{D}_2)$ *satisfies the $(\alpha_{I,1}\alpha_{I,2})$-scaled $[\delta_{I,2}(1+\delta_{I,1}) + \alpha_{I,2}\delta_{I,1}]$-bounded Isometry Property.*

Proof. We want to show that if $\tilde{R}$ is drawn from PRODUCT $(\mathcal{D}_1, \mathcal{D}_2)$, and $x = y$, then the following random variable Z is a $[\delta_{I,2}(1+\delta_{I,1}) + \alpha_{I,2}\delta_{I,1}]$-nice noise variable.

$$\begin{aligned} Z &\coloneqq \left\langle \tilde{R}x, \tilde{R}x \right\rangle - \alpha_{I,1}\alpha_{I,2} \langle x, x \rangle \\ &= x^T \tilde{R}^T \tilde{R} x - \alpha_{I,1}\alpha_{I,2} x^T x \\ &= x^T R_1^T R_2^T R_2 R_1 x - \alpha_{I,1}\alpha_{I,2} x^T x \end{aligned}$$

We now define some useful (noise) variables.

$$\begin{aligned} x_0 &\coloneqq x \\ x_1 &\coloneqq R_1 x_0 \\ x_2 &\coloneqq R_2 x_1 \\ Z_1 &\coloneqq \langle R_1 x_0, R_1 x_0 \rangle - \alpha_{I,1} \langle x_0, x_0 \rangle \\ &= ||x_1||_2^2 - \alpha_{I,1} ||x_0||_2^2 \\ Z_2 &\coloneqq \langle R_2 x_1, R_2 X_1 \rangle - \alpha_{I,2} \langle x_1, x_1 \rangle \\ &= ||x_2||_2^2 - \alpha_{I,2} ||x_1||_2^2 \end{aligned}$$

Z is just a weighted sum of our random variables:

$$\begin{aligned} Z_2 + \alpha_{I,2} Z_1 &= ||x_2||_2^2 - \alpha_{I,2} ||x_1||_2^2 \\ &\quad + \alpha_{I,2} ||x_1||_2^2 - \alpha_{I,1}\alpha_{I,2} ||x_0||_2^2 \\ &= ||x_2||_2^2 - \alpha_{I,1}\alpha_{I,2} ||x_0||_2^2 \\ &= Z \end{aligned}$$

Since $\mathcal{D}_1$ satisfies the $\alpha_{I,1}$-scaled $\delta_{I,1}$-Isometry Property we know that Z_1 is a $\delta_{I,1}$-nice noise variable. Similarly, we know Z_2 is a $\delta_{I,2}$-nice noise variable. As a result, Z is centered; it remains to bound it. By Lemma 1, we know that with high probability $||x_1||_2^2 \leq 1 + \delta_{I,1}$. Hence with high probability:

$$\begin{aligned} |Z_1| &\leq \delta_{I,1} ||x_0||_2^2 \\ |Z_2| &\leq \delta_{I,2}(1+\delta_{I,1}) ||x_0||_2^2 \\ |Z| &\leq |Z_2| + \alpha_{I,2} |Z_1| \\ &\leq \delta_{I,2}(1+\delta_{I,1}) + \alpha_{I,2}\delta_{I,1} \end{aligned}$$

Hence Z is a $[\delta_{I,2}(1+\delta_{I,1}) + \alpha_{I,2}\delta_{I,1}]$-nice noise variable, completing the proof. □

A.4. Applying Different Matrices

For the analysis of our sketch, we sometimes compare objects that have been sketched using different matrices. The following technical lemma is a two matrix variant of desynchronization for this situation.

Lemma 7. *Suppose that we have a distribution $\mathcal{D}$ which satisfies the α_I-scaled δ_I-Isometry Property and the α_D-leaky δ_D-Desynchronization Property. Suppose we independently draw $R_1 \sim \mathcal{D}$ and $R_2 \sim \mathcal{D}$. Then the random variable*

$$Z = \langle R_1 x, R_2 y \rangle - \alpha_D^2 \langle x, y \rangle$$

is a $(\delta_D \sqrt{\alpha_I + \delta_I} + \alpha_D \delta_D)$-nice noise variable.

Proof. We define useful (noise) variables Z_1 and Z_2.

$$\begin{aligned} x' &\coloneqq R_1 x \\ Z_1 &\coloneqq \langle x', R_2 y \rangle - \alpha_D \langle x', y \rangle \\ Z_2 &\coloneqq \langle R_1 x, y \rangle - \alpha_D \langle x, y \rangle \\ Z &= Z_1 + \alpha_D Z_2 \end{aligned}$$

Since $\mathcal{D}$ satisfies the α_D-leaky δ_D-Desynchronization Property, we know that Z_1 and Z_2 are δ_D-nice noise variables. Since D satisfies the α_I-scaled δ_I-Isometry Property, we can apply Lemma 1 to show that with high probability, $||R_1 x||_2^2 \leq (\alpha_I + \delta_I)$. Hence with high probability:

$$\begin{aligned} |Z_1| &\leq \delta_D \sqrt{\alpha_I + \delta_I} \\ |Z_2| &\leq \delta_D \\ |Z| &\leq |Z_1| + \alpha_D |Z_2| \\ &\leq \delta_D \sqrt{\alpha_I + \delta_I} + \alpha_D \delta_D \end{aligned}$$

Hence Z is a $(\delta_D \sqrt{\alpha_I + \delta_I} + \alpha_D \delta_D)$-nice noise variable, as desired. This completes the proof. □

B. Properties of the Block-Random Distribution

In this section, we prove that our block-random distribution over random matrices satisfies the isometry and desynchronization properties required by our other results. In particular, our goal is to prove the following two theorems:

Theorem 6. *Suppose that q is $\Omega\left(\frac{\sqrt{b \log N}}{\sqrt{d_{sketch}}}\right)$ and d_{sketch} is poly(N). Then there exists a $\delta = O\left(\frac{\sqrt{b \log N}}{\sqrt{d_{sketch}}}\right)$ such that the block-random distribution $\mathcal{D}(b, q, d_{sketch})$ satisfies the δ-Isometry Property.*

Theorem 7. *Suppose that q is $\Omega\left(\frac{\sqrt{b \log N}}{\sqrt{d_{sketch}}}\right)$. Then there exists a $\delta = O\left(\frac{\sqrt{b \log N}}{\sqrt{d_{sketch}}}\right)$ such that the block-random distribution $\mathcal{D}(b, q, d_{sketch})$ satisfies the $O(\frac{\log N}{\sqrt{d_{sketch}}})$-Desynchronization Property.*

Note that instantiating the theorems with the minimum necessary b gives us good error bounds.

Corollary 1. *Suppose that $b = 3 \max(\lceil \log_2 N \rceil, \lceil \log_2 d_{sketch} \rceil + 3)$, $q = \left\lceil \frac{\sqrt{(\log N + \log d_{sketch}) \log N}}{\sqrt{d_{sketch}}} \right\rceil$, and d_{sketch} is poly(N). Then the block-random distribution $\mathcal{D}(b, q, d_{sketch})$ satisfies the $\tilde{O}\left(\frac{1}{\sqrt{d_{sketch}}}\right)$-Isometry Property and the $\tilde{O}\left(\frac{1}{\sqrt{d_{sketch}}}\right)$-Desynchronization Property.*

B.1. Isometry

In this subsection, we will prove Theorem 6. Our proof essentially follows the first analysis presented by Kane and Nelson for their sparse Johnson-Lindenstrauss Transform (Kane & Nelson, 2014). Our desired result differs from theirs in several ways, which complicates our version of the proof:

- Their random matrices do not have block-level patterns like ours do. We handle this by using the small size of our blocks.
- Their random matrices have a fixed number of non-zeros per column, but the non-zeros within one of our columns are determined by independent Bernoulli random variables: $\eta_{i,j}$. We handle this by applying a standard concentration bound to the number of non-zeros.
- Some of our sub-block patterns depend on the ± 1 coin flips for other sub-blocks, ruining independence. We handle this by breaking our matrix into three sub-matrices to be analyzed separately; see Figure 5.

For the sake of completeness, we reproduce the entire proof here. The high-level proof plan is to use Hanson-Wright inequality (Hanson & Wright, 1971) to bound the ℓ^{th} moment of the noise, and then use a Markov's inequality to convert the result into a high probability bound on the absolute value of the noise.

$f_{s,1,\ }\cdot\sigma_{s,1}$	$f_{s,1,\ }\cdot\sigma_{s,2}$	$f_{s,1,\ }\cdot\sigma_{s,3}$	$\vec{0}$	$\cdots$	$f_{s,1,d_{\text{sketch}}}\cdot\sigma_{s,d_{\text{sketch}}}$
$f_{c,1,1}\cdot\sigma_{c,1,1}$	$f_{c,1,2}\cdot\sigma_{c,1,2}$	$f_{c,1,3}\cdot\sigma_{c,1,3}$	$\vec{0}$	$\cdots$	$f_{c,1,d_{\text{sketch}}}\cdot\sigma_{c,1,d_{\text{sketch}}}$
$f_{m,1,1}\cdot\sigma_m$	$f_{m,1,2}\cdot\sigma_m$	$f_{m,1,3}\cdot\sigma_m$	$\vec{0}$	$\cdots$	$f_{m,1,d_{\text{sketch}}}\cdot\sigma_m$
$\vec{0}$	$f_{s,2,\ }\cdot\sigma_{s,2}$	$\vec{0}$	$\vec{0}$	$\cdots$	$\vec{0}$
$\vec{0}$	$f_{c,2,2}\cdot\sigma_{c,2,2}$	$\vec{0}$	$\vec{0}$	$\cdots$	$\vec{0}$
$\vec{0}$	$f_{m,2,2}\cdot\sigma_m$	$\vec{0}$	$\vec{0}$	$\cdots$	$\vec{0}$
$f_{s,3,\ }\cdot\sigma_{s,1}$	$f_{s,3,\ }\cdot\sigma_{s,2}$	$f_{s,3,\ }\cdot\sigma_{s,3}$	$f_{s,3,\ }\cdot\sigma_{s,4}$	$\cdots$	$\vec{0}$
$f_{c,3,1}\cdot\sigma_{c,3,1}$	$f_{c,3,2}\cdot\sigma_{c,3,2}$	$f_{c,3,3}\cdot\sigma_{c,3,3}$	$f_{c,3,4}\cdot\sigma_{c,3,4}$	$\cdots$	$\vec{0}$
$f_{m,3,1}\cdot\sigma_m$	$f_{m,3,2}\cdot\sigma_m$	$f_{m,3,3}\cdot\sigma_m$	$f_{m,3,4}\cdot\sigma_m$	$\cdots$	$\vec{0}$
$f_{s,4,\ }\cdot\sigma_{s,1}$	$f_{s,4,\ }\cdot\sigma_{s,2}$	$f_{s,4,\ }\cdot\sigma_{s,3}$	$\vec{0}$	$\cdots$	$f_{s,4,d_{\text{sketch}}}\cdot\sigma_{s,d_{\text{sketch}}}$
$f_{c,4,1}\cdot\sigma_{c,4,1}$	$f_{c,4,2}\cdot\sigma_{c,4,2}$	$f_{c,4,3}\cdot\sigma_{c,4,3}$	$\vec{0}$	$\cdots$	$f_{c,4,d_{\text{sketch}}}\cdot\sigma_{c,4,d_{\text{sketch}}}$
$f_{m,4,1}\cdot\sigma_m$	$f_{m,4,2}\cdot\sigma_m$	$f_{m,4,3}\cdot\sigma_m$	$\vec{0}$	$\cdots$	$f_{m,4,d_{\text{sketch}}}\cdot\sigma_m$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\vec{0}$	$\vec{0}$	$f_{s,\diamondsuit,\ }\cdot\sigma_{s,3}$	$f_{s,\diamondsuit,\ }\cdot\sigma_{s,4}$	$\cdots$	$\vec{0}$
$\vec{0}$	$\vec{0}$	$f_{c,\diamondsuit,3}\cdot\sigma_{c,\diamondsuit,3}$	$f_{c,\diamondsuit,4}\cdot\sigma_{c,\diamondsuit,4}$	$\cdots$	$\vec{0}$
$\vec{0}$	$\vec{0}$	$f_{m,\diamondsuit,3}\cdot\sigma_m$	$f_{m,\diamondsuit,4}\cdot\sigma_m$	$\cdots$	$\vec{0}$

Figure 5. The proof of Theorem 6 involves three sub-matrices of a random block-random matrix. The first of these sub-matrices, R_s, is highlighted here in green. It consists of all the random string sub-blocks. Just as in Figure 4, $\diamondsuit := d_{\text{sketch}}/b$ is shorthand for the number of blocks.

Suppose we draw a random matrix $R \in \mathbb{R}^{d_{\text{sketch}} \times d_{\text{sketch}}}$ from our distribution $\mathcal{D}(b, q, d_{\text{sketch}})$. We split R into three sub-matrices: $R_s \in \mathbb{R}^{d_{\text{sketch}}/3 \times d_{\text{sketch}}}$ containing the random string sub-blocks, $R_m \in \mathbb{R}^{d_{\text{sketch}}/3 \times d_{\text{sketch}}}$ containing the matrix signature sub-blocks, and $R_c \in \mathbb{R}^{d_{\text{sketch}}/3 \times d_{\text{sketch}}}$ containing the column signature sub-blocks.

We fix $\delta = O\left(\frac{\log N}{\sqrt{d_{\text{sketch}}}}\right)$. Consider any one of these sub-matrices, $R_\star$. We wish to show that for any unit vector $x \in \mathbb{R}^{d_{\text{sketch}}}$, with high probability over $R \sim \mathcal{D}(b, q, d_{\text{sketch}})$:

$$||R_\star x||_2^2 \in \left[\frac{1}{3}(1-\delta), \frac{1}{3}(1+\delta)\right]$$

Since this would imply that with high probability (due to union-bounding over the three sub-matrices):

$$\begin{aligned} ||Rx||_2^2 &= \sum_{\star \in \{s,c,m\}} ||R_\star x||_2^2 \\ &\in [1-\delta, 1+\delta] \end{aligned}$$

We will use the variant of the Hanson-Wright inequality presented by Kane and Nelson (Kane & Nelson, 2014).

Theorem 8 (Hanson-Wright Inequality (Hanson & Wright, 1971))**.** *Let $z = (z_1, ..., z_n)$ be a vector of i.i.d. Rademacher ± 1 random variables. For any symmetric $B \in \mathbb{R}^{n \times n}$ and $\ell \geq 2$,*

$$\mathbb{E}\left|z^T B z - \text{trace}(B)\right|^\ell \leq C^\ell \cdot \max\left\{\sqrt{\ell} \cdot ||B||_F, \ell \cdot ||B||_2\right\}^\ell$$

for some universal constant $C > 0$ independent of B, n, ℓ.

Note that in the above theorem statement, $||B||_F$ is the Frobenius norm of matrix B (equal to the ℓ_2 norm of the flattened matrix) while $||B||_2$ is the operator norm of matrix B (equal to the longest length of A applied to any unit vector).

Our immediate goal is now to somehow massage $||R_\star x||_2^2$ into the quadratic form $z^T B z$ that appears in the inequality. We begin by expanding out the former. Note that for simplicity of notation, we'll introduce some extra indices: we use $\sigma_{s,i,j}$ to denote $\sigma_{s,j}$ and $\sigma_{m,i,j}$ to denote σ_m so that all three types of sub-block patterns can be indexed the same way. Also, we'll

use $\Diamond := d_{\text{sketch}}/b$ to denote the number of blocks.

$$
\begin{aligned}
R_\star x &= \sum_{j=1}^{d_{\text{sketch}}} \begin{bmatrix} \eta_{1,j} f_{\star,1,j} \sigma_{\star,1,j} \\ \eta_{2,j} f_{\star,2,j} \sigma_{\star,2,j} \\ \vdots \\ \eta_{\Diamond,j} f_{\star,\Diamond,j} \sigma_{\star,\Diamond,j} \end{bmatrix} x_j \\
||R_\star x||_2^2 &= x^T R^T R x \\
&= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} \left(\eta_{i,j} f_{\star,i,j} \sigma_{\star,i,j} x_j\right)^T \left(\eta_{i,j'} f_{\star,i,j'} \sigma_{\star,i,j'} x_{j'}\right) \\
&= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} \eta_{i,j} \eta_{i,j'} f_{\star,i,j} f_{\star,i,j'} x_j x_{j'} \left\langle \sigma_{\star,i,j}, \sigma_{\star,i,j'} \right\rangle
\end{aligned}
$$

As before, this can also be expressed as a quadratic form in the f variables. In particular, let the $\Diamond d_{\text{sketch}} \times \Diamond d_{\text{sketch}}$ block-diagonal matrix $B_\star$ is as follows. There are $\Diamond$ blocks $B_{\star,i}$, each $d_{\text{sketch}} \times d_{\text{sketch}}$ in shape. The $(j,j')^{th}$ entry of the i^{th} block is $\eta_{i,j}\eta_{i,j'} x_j x_{j'} \left\langle \sigma_{\star,i,j}, \sigma_{\star,i,j'} \right\rangle$. As a result, $||R_\star x||_2^2 = f_\star^T B_\star f_\star$, where $f_\star$ is the $\Diamond d_{\text{sketch}}$-dimensional vector $(f_{\star,1,1}, f_{\star,1,2}, ...)$.

It is important to note for the sake of the Hanson-Wright inequality that the matrix $B_\star$ is random, but it *does not depend* on the f variables. Of course, B_c depends on the f variables involved in B_s and B_m by construction, but not its own (and this is why we split our analysis over the three submatrices).

All that remains before we can effectively apply the Hanson-Wright inequality is to bound the trace, Frobenius norm, and operator norm of this matrix $B_\star$. We begin with the trace, which is a random variable depending on the zero-one Bernoulli random variables η.

$$
\begin{aligned}
\text{trace}(B_\star) &= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \eta_{i,j}^2 x_j^2 \, ||\sigma_{\star,i,j}||_2^2 \\
&= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \eta_{i,j} x_j^2 (b/3) (\frac{1}{d_{\text{sketch}} q}) \\
&= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \eta_{i,j} x_j^2 \frac{1}{3\Diamond q}
\end{aligned}
$$

Let the random variable $X_{i,j} := \eta_{i,j} x_j^2 \frac{1}{3\Diamond q}$. Since x was a unit vector, we know that the expected value of $\sum_i \sum_j X_{i,j} = \frac{1}{3}$. Additionally, note that:

$$
\sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \left(x_j^2 \frac{1}{3\Diamond q}\right)^2 = \frac{1}{9\Diamond q^2} \sum_{j=1}^{d_{\text{sketch}}} x_j^4 \leq \frac{1}{9\Diamond}
$$

By Hoeffding's Inequality (Hoeffding, 1963) we the probability that this sum (i.e. the trace) differs too much from its expectation:

$$
\begin{aligned}
\Pr\left[\left|\text{trace}(B_\star) - \frac{1}{3}\right| \geq \frac{\delta}{6}\right] &\leq \exp\left(\frac{-2(\delta/6)^2}{\sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \left(x_j^2 \frac{1}{3\Diamond q}\right)^2}\right) \\
&\leq \exp\left(\frac{-3\Diamond(\delta)^2}{2}\right)
\end{aligned}
$$

Hence there exists a sufficiently large $\delta = O\left(\frac{\sqrt{\log N}}{\sqrt{\Diamond}}\right)$ so that we will have with high probability that the trace is within $\frac{\delta}{6}$ of $\frac{1}{3}$.

Next, we want a high probability bound on the Frobenius norm of $B_\star$.

$$\begin{aligned}
||B_\star||_F^2 &= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} \left(\eta_{i,j}\eta_{i,j'}x_j x_{j'} \left\langle \sigma_{\star,i,j}, \sigma_{\star,i,j'} \right\rangle\right)^2 \\
&= \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} \eta_{i,j}\eta_{i,j'}x_j^2 x_{j'}^2 \left\langle \sigma_{\star,i,j}, \sigma_{\star,i,j'} \right\rangle^2 \\
&\leq \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} \eta_{i,j}\eta_{i,j'}x_j^2 x_{j'}^2 \left(\frac{1}{d_{\text{sketch}}q} b/3\right)^2 \\
&= \frac{1}{9\Diamond^2 q^2} \sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} x_j^2 x_{j'}^2 \sum_{i=1}^{\Diamond} \eta_{i,j}\eta_{i,j'}
\end{aligned}$$

We want to show that for any two columns j, j' there aren't too many matching non-zero blocks. The number of matching blocks is the sum of $\Diamond$ i.i.d. Bernoulli random variables with probability q^2. By a Hoeffding Bound (Hoeffding, 1963), we know that for all $t \geq 0$:

$$\Pr\left[\sum_{i=1}^{\Diamond} \eta_{i,j}\eta_{i,j'} \geq \Diamond q^2 + t\right] \leq \exp\left(-2t^2/\Diamond\right)$$

So the desired claim is true with high probability (i.e. polynomially small in both N and d_{sketch}) as long as we can choose t to be on the order of $\frac{\sqrt{\log(N d_{\text{sketch}})}}{\sqrt{\Diamond}}$. However, we want this to be $O(\Diamond q^2)$. Since by assumption q is $\Omega\left(\frac{\sqrt{\log N}}{\sqrt{\Diamond}}\right)$, we actually just want it to be $O(\log N)$. This is true since we assumed d_{sketch} was poly(N) and $\Diamond \geq 1$.

We can now safely union-bound over all pairs of columns and finish bounding the Frobenius norm (with high probability):

$$\begin{aligned}
||B_\star||_F^2 &\leq \frac{1}{9\Diamond^2 q^2} \underbrace{\sum_{j=1}^{d_{\text{sketch}}} \sum_{j'=1}^{d_{\text{sketch}}} x_j^2 x_{j'}^2}_{||x||_2^4 = 1} O(\Diamond q^2) \\
&\leq O\left(\frac{1}{\Diamond}\right)
\end{aligned}$$

Finally, we want a high probability bound on the operator norm of $B_\star$. The operator norm is just its maximum eigenvalue, and its eigenvalues are the eigenvalues of its blocks. We can write each block as a rank-$(b/3)$ decomposition $\sum_{k=1}^{b/3} u_{i,k} u_{i,k}^T$ by breaking up the inner products by coordinate (each yielding a vector $u_{i,k}$). Specifically, the vector $u_{i,k} \in \mathbb{R}^{d_{\text{sketch}}}$ is

defined by $(u_{i,k})_j = \eta_{i,j} x_j \sigma_{\star,i,j}[k]$.

$$\begin{aligned}
||B_\star||_2 &= \max_i ||B_{\star,i}||_2 \\
&= \max_i \left|\left| \sum_{k=1}^{b/3} u_{i,k} u_{i,k}^T \right|\right|_2 \\
&\leq \max_i \sum_{k=1}^{b/3} \left|\left| u_{i,k} u_{i,k}^T \right|\right|_2 \\
&= \max_i \sum_{k=1}^{b/3} ||u_{i,k}||_2^2 \\
&\leq \max_i \sum_{k=1}^{b/3} \frac{1}{d_{\text{sketch}} q} ||x||_2^2 \\
&= \frac{b}{3} \frac{1}{d_{\text{sketch}} q} \\
&= \frac{1}{3 \lozenge q}
\end{aligned}$$

Applying the Hanson-Wright inequality, Theorem 8, we know that with high probability:

$$\mathbb{E} \left| z^T B z - \frac{1}{3} \pm \frac{\delta}{2} \right|^\ell \leq C^\ell \cdot \max \left\{ \sqrt{\ell} \cdot O(1/\lozenge), \ell \cdot O(1/(\lozenge q)) \right\}^\ell$$

Applying Markov's inequality for ℓ an even integer:

$$\Pr \left[\left| z^T B z - \frac{1}{3} \pm \frac{\delta}{2} \right| > \frac{\delta}{6} \right] < \left(\frac{6}{\delta} \right)^\ell C^\ell \cdot \max \left\{ \sqrt{\ell} \cdot O(1/\sqrt{\lozenge}), \ell \cdot O(1/(\lozenge q)) \right\}^\ell$$

$$\Pr \left[\left| z^T B z - \frac{1}{3} \right| > \frac{\delta}{3} \right] < \left(\frac{6C}{\delta} \max \left\{ \sqrt{\ell} \cdot O(1/\sqrt{\lozenge}), \ell \cdot O(1/(\lozenge q)) \right\} \right)^\ell$$

We'll need to choose a sufficiently large ℓ on the order of $\log N$:

$$\begin{aligned}
\Pr \left[\left| z^T B z - \frac{1}{3} \right| > \frac{\delta}{3} \right] &< \left(\frac{6C}{\delta} \max \left\{ O\left(\frac{\sqrt{\log N}}{\sqrt{\lozenge}} \right), O\left(\frac{\log N}{\lozenge q} \right) \right\} \right)^{\Theta(\log N)} \\
&< \left(\frac{6C}{\delta} O\left(\frac{\sqrt{\log N}}{\sqrt{\lozenge}} \right) \right)^{\Theta(\log N)}
\end{aligned}$$

We again used that q is $\Omega\left(\frac{\sqrt{\log N}}{\sqrt{\lozenge}}\right)$. We conclude by choosing $\delta = O\left(\frac{\log N}{\lozenge}\right)$, to make the base of the left-hand side a constant and achieve our polynomially small failure probability. This completes the proof of Theorem 6.

B.2. Desynchronization

In this subsection, we will prove Theorem 7. Suppose we draw a random matrix $R \in \mathbb{R}^{d_{\text{sketch}} \times d_{\text{sketch}}}$ from our distribution $\mathcal{D}(b, q, d_{\text{sketch}})$, and we have two unit vectors $x, y \in \mathbb{R}^{d_{\text{sketch}}}$. For convenience, we will index x using $[d_{\text{sketch}}]$ and y using $\{s, c, m\} \times [\lozenge] \times [b/3]$.

As in the previous section, we split R into R_s, R_c, R_m (see Figure 5) so that our random variables will be independent of the sub-block patterns. For one sub-matrix:

$$y^T R_\star x = \sum_{i=1}^{\lozenge} \sum_{j=1}^{d_{\text{sketch}}} \eta_{i,j} f_{\star,i,j} \left\langle \sigma_{\star,i,j}, y_{\star,i} \right\rangle x_j$$

We view this as a sum of $\Diamond d_{\text{sketch}}$ random variables, which we will apply Bernstein's Inequality to bound. Note that:

$$\begin{aligned}
|\eta_{i,j} f_{\star,i,j} \langle \sigma_{\star,i,j}, y_{\star,i} \rangle x_j| &\leq \frac{b}{3} \frac{1}{d_{\text{sketch}} q} \\
&= \frac{1}{3 \Diamond q} \\
\sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \mathbb{E}\left[\eta_{i,j}^2 f_{\star,i,j}^2 \langle \sigma_{\star,i,j}, y_{\star,i} \rangle^2 x_j^2\right] &= q \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \langle \sigma_{\star,i,j}, y_{\star,i} \rangle^2 x_j^2 \\
&\leq q \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} ||\sigma_{\star,i,j}||_2^2 ||y_{\star,i}||_2^2 x_j^2 \\
&= q \sum_{i=1}^{\Diamond} \sum_{j=1}^{d_{\text{sketch}}} \frac{b}{3} \frac{1}{d_{\text{sketch}} q} ||y_{\star,i}||_2^2 x_j^2 \\
&= \frac{1}{3 \Diamond}
\end{aligned}$$

Combining these with Bernstein's Inequality yields:

$$\Pr\left[y^T R_\star x > \frac{\delta}{3}\right] \leq \exp\left(-\frac{\delta^2/18}{\frac{1}{3\Diamond} + \frac{1}{9 \Diamond q} \frac{\delta}{3}}\right)$$

We note that by assumption $q = \Omega\left(\frac{\sqrt{\log N}}{\sqrt{\Diamond}}\right)$ and hence we can also choose a sufficiently large $\delta = O\left(\frac{\sqrt{\log N}}{\sqrt{\Diamond}}\right)$ to make this probability polynomially small in N. This completes the proof.

C. Proof Sketches for Sketch Prototypes

In this appendix, we present proof sketches for our sketch prototypes.

C.1. Proof Sketches for Prototype A

Recall Claim 1, which we will now sketch a proof of.

Claim 1. *Prototype A has the **Attribute Recovery** property: for any object $\theta^\star$, with high probability (over the random matrices), its attribute vector $x_{\theta^\star}$ can be recovered up to at most $O(\log N/(w_{\theta^\star}\sqrt{d_{sketch}}))$ ℓ_∞-error.*

Proof Sketch. We multiply our overall sketch with $R_{M(\theta^\star)}^T = R_{M(\theta^\star)}^{-1}$. This product is $\sum_\theta w_\theta R_{M(\theta^\star)}^{-1} R_{M(\theta)} x_\theta$. Our signal term occurs when $\theta = \theta^\star$ and is $w_{\theta^\star} x_{\theta^\star}$. We think of the other terms as noise; we know from our orthonormal matrix properties that with high probability they each perturb any specific coordinate by at most $O(w_\theta \sqrt{\log N}/\sqrt{d_{\text{sketch}}})$. Taken together, no coordinate is perturbed by more than $O(\sqrt{\log N}/\sqrt{d_{\text{sketch}}})$. Dividing both our signal term and noise by $w_{\theta^\star}$, we get the desired claim. □

Recall Claim 2, which we will now sketch a proof of.

Claim 2. *Prototype A has the **Sketch-to-Sketch Similarity** property: (i) if two sketches share no modules, then with high probability they have at most $O(\sqrt{\log N}/\sqrt{d_{sketch}})$ dot-product, (ii) if two sketches share an object $\theta^\star$, which has $w_{\theta^\star}$ in the one sketch and $\bar{w}_{\theta^\star}$ in the other, then with high probability they have at least $w_{\theta^\star}\bar{w}_{\theta^\star} - O(\sqrt{\log N}/\sqrt{d_{sketch}})$ dot-product, (iii) if two sketches have objects produced by identical modules and each pair of objects produced by the same module differ by at most ϵ in ℓ_2 distance and are given equal weights, then they are guaranteed to be at most ϵ from each other in ℓ_2 distance.*

Proof Sketch. Consider two overall sketches $s = \sum_\theta w_\theta R_{M(\theta)} x_\theta$ and $\bar{s} = \sum_{\bar{\theta}} \bar{w}_{\bar{\theta}} R_{M(\bar{\theta})} x_{\bar{\theta}}$. Their dot-product can be written as $\sum_\theta \sum_{\bar{\theta}} w_\theta \bar{w}_{\bar{\theta}} x_\theta^T R_{M(\theta)}^T R_{M(\bar{\theta})} x_{\bar{\theta}}$.

First, let us suppose we are in case (i) and they share no modules. Since orthonormal matrices are desynchronizing, each term has magnitude at most $O(w_\theta \bar{w}_{\bar{\theta}} \sqrt{\log N}/\sqrt{d_{\text{sketch}}})$. Since the sum of each set of weights is one, the total magnitude is at most $O(\sqrt{\log N}/\sqrt{d_{\text{sketch}}})$. This completes the proof of case (i).

Next, let us suppose we are in case (ii) and they share an object $\theta^\star$. Since orthonormal matrices are isometric, the term $\theta = \bar{\theta} = \theta^\star$ has a value of $w_{\theta^\star}\bar{w}_{\theta^\star}$. We can analyze the remaining terms as in case (i); they have total magnitude at most $O(\sqrt{\log N}/\sqrt{d_{\text{sketch}}})$. The worst-case is that the noise is negative, which yields the claim for case (ii).

We need to analyze case (iii) a little differently. Since our objects are paired up between the sketches, we refer to the objects as $\theta_1, \ldots, \theta_k$ (in the first sketch) and $\bar{\theta}_1, \ldots, \bar{\theta}_k$ (in the second sketch. Because they are close in ℓ_2 distance, we can write $x_{\bar{\theta}_i} = x_{\theta_i} + \epsilon_i$, where each ϵ_i is a noise vector with ℓ_2 length at most ϵ.

We can write our second sketch as $\bar{s} = \sum_i w_{\theta_i} R_{M(\theta_i)} (x_{\theta_i} + \epsilon_i)$ (by assumption, the paired objects had the same weight and were produced by the same module). But then $\bar{s} = s + \sum_i w_{\theta_i} R_{M(\theta_i)} \epsilon_i$! Since orthonormal matrices are isometric and the weights sum to one, $\bar{s}$ is at most ϵ away from s in ℓ_2 norm, as desired. □

C.2. Proof Sketches for Prototype B

Recall Claim 3, which we will now sketch a proof of.

Claim 3. *Prototype B has the **Attribute Recovery** property, which is as follows. Consider an object $\theta^\star$ with weight $w_{\theta^\star}$. Then (i) if no other objects are produced by the same module in the sketch, with high probability we can recover w_θ up to $O(\frac{\sqrt{\log N}}{w_{\theta^\star}\sqrt{d_{sketch}}})$ ℓ_∞-error and (ii) even if other objects are output by the same module, if we know $\theta^\star$'s index in the overall sketch, with high probability we can recover w_θ up to $O(\frac{\sqrt{\log N}}{w_{\theta^\star}\sqrt{d_{sketch}}})$ ℓ_∞-error.*

Proof Sketch. We begin with the easier case (i). For case (i), we attempt to recover $x_{\theta^\star}$ via multiplying the sketch by $R^T_{M(\theta^\star),1}$. The signal component of our sketch has the form $\left(\frac{I+R_i}{2}\right)\frac{1}{2}R^T_{M(\theta^\star),1}x_{\theta^\star}$. The $R_i/2$ part can be considered noise, and the $I/2$ part gives us a quarter of the signal we had when analyzing sketch A. The noise calculation is the same as before because $R_{M(\theta^\star)}$ does not occur anywhere else, so we get the same additive error (up to a constant hidden in the big-oh notation).

For the harder case (ii), we multiply the sketch by $R^T_{M(\theta^\star)}R^T_i$ instead. This recovers the $R_i/2$ part instead of the $I/2$ part, but everything else is the same as the previous case. □

Recall Claim 4, which we will now sketch a proof of.

Claim 4. *Prototype B has the **Sketch-to-Sketch Similarity** property, just as in Claim 2, except in case (ii) the dot-product is at least $\frac{1}{4}w_{\theta^\star}\bar{w}_{\theta^\star} - O(\sqrt{\log N}/\sqrt{d_{sketch}})$, in case (iii) the sketches are at most $\epsilon/2$ from each other in ℓ_2 distance, and we have an additional case (iv) if one sketch has an object $\theta^\star$ of weight $w_{\theta^\star}$ and the other sketch has an object $\bar{\theta}^\star$ of weight $\bar{w}_{\bar{\theta}^\star}$ and both objects are produced by the same module, then with high probability they have at least $\frac{1}{16}w_{\theta^\star}\bar{w}_{\theta^\star} - O(\sqrt{\log N}/\sqrt{d_{sketch}})$.*

Proof Sketch. Case (i) is the same as before; the desynchronizing property of orthonormal matrices means that taking the dot-product of module sketches, even when transformed by $I/2$ or $R_i/2$, results in at most $O(\sqrt{\log N}/\sqrt{d_{\text{sketch}}})$ noise.

Regarding case (ii), our new signal term is weaker by a factor of four. Notice that for both sketches, the object sketch $s_{\text{object}}(\theta^\star)$ is still identical. We lose our factor of four when considering that a $\left(\frac{I+R_i}{2}\right)$ has been added into the tuple sketch; the $I/2$ lets through the information we care about, and compounded over two sketches results in a factor four loss. If the object were located at the same index in both sketches, we would only lose a factor of two.

Regarding case (iii), the new module sketch is only half-based on attributes; the other half agrees if the objects have the same module. As a result, changing the attributes has half the effect as before.

Case (iv) is similar to case (i), except only the e_1 component of the module sketch is similar, resulting in an additional factor two loss from each sketch, for a final factor of sixteen. □

Recall Claim 5, which we will now sketch a proof of.

Claim 5. *Prototype B has the **Summary Statistics** property, which is as follows. Consider a module $M^\star$, and suppose all objects produced by $M^\star$ have the same weight in the overall sketch. Then (i) we can estimate the number of objects produced by $M^\star$ up to an additive $O(\frac{\sqrt{\log N}}{w^\star\sqrt{d_{sketch}}})$ and (ii) we can estimate the summed attribute vector of objects produced by $M^\star$ up to $O(\frac{\sqrt{\log N}}{w^\star\sqrt{d_{sketch}}})$ ℓ_∞-error.*

Proof Sketch. It is more intuitive to begin with case (ii). Consider what would happen if we performed case (i) of **Attribute Recovery** as in Claim 3. That claim assumed that there were no other objects produced by the same module, but if there were (and they had the same weight $w^\star$), then their attribute vectors would be added together. This exactly yields the current case (ii).

Next, for case (i), we simply observe that the object sketch treats x_θ the same way as it treats e_1, so we could use the same idea to recover the sum, over all objects with the same module, of e_1. That is exactly the number of such modules times e_1, so looking at the first coordinate gives us the desired estimate for the current case (i). □

D. Proofs for Final Sketch

In this appendix, we present proofs for the properties of our final sketch. Recall that our final sketch is defined as follows.

$$\begin{aligned}
s_{\text{tuple}}(s_1, ..., s_k) &\coloneqq \sum_{i=1}^{k} w_i \left(\frac{I + R_i}{2}\right) s_i \\
s_{\text{object}}(\theta) &\coloneqq \left(\tfrac{I+R_{M(\theta),0}}{2}\right) s_{\text{tuple}}\left(s_{\text{attr}}(\theta), s_{\text{input}}(\theta), \tfrac{1}{2}, \tfrac{1}{2}\right) \\
s_{\text{attr}}(\theta) &\coloneqq \tfrac{1}{2}R_{M(\theta),1}x_\theta + \tfrac{1}{2}R_{M(\theta),2}e_1 \\
s_{\text{input}}(\theta) &\coloneqq s_{\text{tuple}}\left(s_{\text{object}}(\theta_1), ..., s_{\text{object}}(\theta_k), w_1, ..., w_k\right) \\
s_{\text{overall}} &\coloneqq s_{\text{input}}(\text{output pseudo-object})
\end{aligned}$$

If we unroll these definitions, then the overall sketch has the form:

$$s_{\text{overall}} = \sum_{\theta} w_\theta \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

where $h(\theta)$ is the depth of object θ and $\left(\frac{I+R_{\theta,j}}{2}\right)$ is the j^{th} matrix when walking down the sketch tree from the overall sketch to the θ object sketch (i.e. alternating between input index, module matrix, and a one or two index into module versus inputs).

Technical Assumptions. For this section, we need the following technical modification to the sketch. The R_i matrices for the tuple operators are *redrawn at every depth level* of the sketch, where the depth starts at one at the overall sketch and increases every time we take a tuple sketch. Additionally, we assume that we can guarantee $\delta \leq \frac{1}{4H}$ by appropriately increasing the sketch dimension (in the typical case where H is constant, this just increases the sketch dimension by a constant); this section simply accepts a δ guarantee derived in Appendix **??** and does not tinker with sketch parameters.

D.1. Proofs for Attribute Recovery

Theorem 9. *Our final sketch has the **Attribute Recovery** property, which is as follows. Consider an object $\theta^\star$ which lies at depth $h(\theta^\star)$ in the overall sketch and with effective weight $w_{\theta^\star}$.*

(i) Suppose no other objects in the overall sketch are also produced by $M(\theta^\star)$. We can produce a vector estimate of the attribute vector $x_{\theta^\star}$, which with high probability has an additive error of at most $O(2^{3h(\theta^\star)}h(\theta^\star)\delta/w_{\theta^\star})$ in each coordinate.

(ii) Suppose we know the sequence of input indices to get to $\theta^\star$ in the sketch. Then even if other objects in the overall sketch are produced by $M(\theta^\star)$, we can still produce a vector estimate of the attribute vector $x_{\theta^\star}$, which with high probability has an additive error of at most $O(2^{3h(\theta^\star)}h(\theta^\star)\delta/w_{\theta^\star})$ in each coordinate.

Proof. We begin by proving the easier case (i). Suppose that our overall sketch is s. Let $\beta = 2^{3h(\theta^\star)+1}/w_{\theta^\star}$, and imagine that we attempt to recover the j^{th} coordinate of $x_{\theta^\star}$ by computing $\beta e_j^T R_{M(\theta^\star),1}^T s$, which has the form:

$$\sum_\theta \beta w_\theta e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

When $\theta = \theta^\star$, we get the term:

$$2^{3h(\theta^\star)+1} e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta^\star)}} \left[\left(\frac{I + R_{\theta^\star,j}}{2}\right)\right] s_{\text{object}}(\theta^\star)$$

If we expand out each transparent matrix, we would get $2^{3h(\theta^\star)}$ sub-terms. Let's focus on the sub-term where we choose $I/2$ every time:

$$2e_j^T R_{M(\theta^\star),1}^T \left[\frac{1}{2} R_{M(\theta^\star),1} x_{\theta^\star} + \frac{1}{2} R_{M(\theta^\star),2} e_1\right]$$

By Lemma 2, we know that with high probability (note e_j and $x_{\theta^\star}$ are both unit vectors):

$$\left|2e_j^T R_{M(\theta^\star),1}^T \times \frac{1}{2} R_{M(\theta^\star),1} x_{\theta^\star} - e_j^T x_{\theta^\star}\right| \leq 3\delta = O(\delta)$$

This half of the sub-term is (approximately) the j^{th} coordinate of $x_{\theta^\star}$, up to an additive error of $O(\delta)$. We control the other half using Lemma 7:

$$\left|2e_j^T R_{M(\theta^\star),1}^T \times \frac{1}{2} R_{M(\theta^\star),2} e_1\right| \leq \delta\sqrt{1+\delta} = O(\delta)$$

It remains to bound the additive error from the other sub-terms and the other terms.

The other sub-terms have the form:

$$2e_j^T R_{M(\theta^\star),1}^T \tilde{R} s_{\text{object}}(\theta^\star)$$

where $\tilde{R}$ is a product of one to $3h(\theta^\star)$ R matrices. By Lemma 1, we know that with high probability $s_{\text{object}}(\theta^\star)$ and $R_{M(\theta^\star),1} e_j$ have squared ℓ_2 norm at most $(1+\delta)$.

We analyze $\tilde{R}$ by repeatedly applying Lemma 5: we know that the distribution of one matrix satisfies δ-Desynchronization and δ-Isometry. Let $\gamma = \sqrt{1+\delta}$. Then the product distribution of two matrices satisfies $[(1+\gamma)\delta]$-Desynchronization, the product distribution of three matrices satisfies $\left[(1+\gamma+\gamma^2)\delta\right]$-Desynchronization, and the product distribution of $3h(\theta^\star)$ matrices satisfies $\left[\frac{\gamma^{3h(\theta^\star)}-1}{\gamma-1}\delta\right]$-Desynchronization. The last desynchronization is the largest, therefore no matter how many matrices are being multiplied, with high probability the absolute value of any other sub-term is bounded by:

$$\begin{aligned}
&2\frac{\gamma^{3h(\theta^\star)}-1}{\gamma-1}\delta(1+\delta) \\
\leq\ &2\frac{\gamma^{6h(\theta^\star)}-1}{\gamma^2-1}\delta(1+\delta) \\
\leq\ &2\frac{(1+\delta)^{3h(\theta^\star)}-1}{\delta}\delta(1+\delta) \\
\leq\ &2(1+O(h(\theta^\star)\delta)-1)(1+\delta) \\
\leq\ &O(h(\theta^\star)\delta)
\end{aligned}$$

Note that we used the fact that $(1+x)^n \leq 1 + O(xn)$ if $xn \leq \frac{1}{2}$.

Hence, the absolute value of the sum of the other sub-terms is bounded by $O(2^{3h(\theta^\star)}h(\theta^\star)\delta)$.

We now bound the additive error from the other terms, which have the form:

$$\beta w_\theta e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

The plan is to use the desynchronization of $R_{M(\theta^\star),1}$. To do so, we will need to bound the ℓ_2 length of $s_{\text{object}}(\theta)$ as it gets multiplied by this sequence of matrices. By Lemma 4 that the distribution of these matrices is $\frac{1}{2}$-scaled $(\frac{3}{4}\delta)$-Isometric. By repeatedly applying Lemma 1 we know that the squared ℓ_2 length is upper bounded by $(\frac{1}{2} + \frac{3}{4}\delta)^{3h(\theta)}(1+\delta)$. By our assumptions, this is upper-bounded by a constant:

$$\begin{aligned}
&(\frac{1}{2} + \frac{3}{4}\delta)^{3h(\theta)}(1+\delta) \\
&\leq (1+\delta)^{3h(\theta)+1} \\
&\leq (1 + 1/(2H))^{2H} \\
&\leq e
\end{aligned}$$

Now, by the desynchronization of $R_{M(\theta^\star),1}$, we know that with high probability the absolute value of the term indexed by θ is at most $O(\beta w_\theta \delta)$. Hence with high probability the absolute value of all other terms is at most $O(\beta\delta)$. Plugging in our choice of β, it is at most $O(2^{3h(\theta^\star)+1}\delta/w_{\theta^\star})$.

Hence we have shown that with high probability, we can recover the j^{th} coordinate of $x_{\theta^\star}$ with the desired additive error. This completes the proof of case (i).

We now prove the harder case (ii). Again, we let $\beta = 2^{3h(\theta^\star)+1}/w_{\theta^\star}$, and imagine that we attempt to recover the j^{th} coordinate of $x_{\theta^\star}$ by computing $\beta e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta^\star)}} \left[R_{\theta^\star,j}^T\right] s$, which has the form:

$$\sum_\theta \beta w_\theta e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}} \left[R_{\theta^\star,j}^T\right] \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

It is worth noting that we *did not compute* the alternative product $\beta e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta^\star)}} \left[\left(\frac{I+R_{\theta^\star,j}^T}{2}\right)\right] s$ because the transparent matrices in the alternative would allow through information about other objects produced by the same module.

When $\theta = \theta^\star$, we get the term:

$$2^{3h(\theta^\star)+1} e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}} \left[R_{\theta^\star,j}^T\right] \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta^\star)}} \left[\left(\frac{I + R_{\theta^\star,j}}{2}\right)\right] s_{\text{object}}(\theta^\star)$$

If we expand out each transparent matrix, we would get $2^{3h(\theta^\star)}$ sub-terms. Let's focus on the sub-term where we choose $R/2$ every time:

$$2 e_j^T R_{M(\theta^\star),1}^T \tilde{R}^T \tilde{R} \left[\frac{1}{2} R_{M(\theta^\star),1} x_{\theta^\star} + \frac{1}{2} R_{M(\theta^\star),2} e_1\right]$$

where $\tilde{R} = \overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}}[R_{\theta^\star,j}]$.

As before, we expect to get some signal from the first half $\frac{1}{2}R_{M(\theta^\star),1}x_{\theta^\star}$ and some noise from the second half $\frac{1}{2}\mathbb{R}_{M(\theta^\star),2}e_1$. Let us begin with the first half. The matrices involved match exactly, so it is a matter of repeatedly applying Lemma 6. Note that all matrices involved are δ-Isometric. Since at each step we multiply the isometry parameter by $(1+\delta)$ and then add δ, the isometry parameter after combining $3h(\theta^\star)$ times is:

$$\begin{aligned}\sum_{i=0}^{3h(\theta^\star)} \delta(1+\delta)^i &= \delta\frac{(1+\delta)^{3h(\theta^\star)+1}-1}{1+\delta-1}\\ &= (1+\delta)^{3h(\theta^\star)+1}-1\\ &\leq 1+O(h(\theta^\star)\delta)-1\\ &\leq O(h(\theta^\star)\delta)\end{aligned}$$

Note that we used the fact that $(1+x)^n \leq 1+O(xn)$ if $xn \leq \frac{1}{2}$ along with our assumption that $\delta < 1/(4H)$.

Hence the product of all $(3h(\theta^\star)+1)$ matrices is $O(h(\theta^\star)\delta)$-Isometric. By Lemma 2 this implies that with high probability:

$$\left|\left\langle \tilde{R}R_{M(\theta^\star),1}e_j, \tilde{R}R_{M(\theta^\star),1}x_{\theta^\star}\right\rangle - \langle e_j, x_{\theta^\star}\rangle\right| \leq O(h(\theta^\star)\delta)$$

We now control the noise half. By Lemma 7, with high probability:

$$\left|\left\langle R_{M(\theta^\star),1}e_j, R_{M(\theta^\star),2}e_1\right\rangle\right| \leq \delta\sqrt{1+\delta} = O(\delta)$$

We have already computed that the remaining $3h(\theta^\star)$ matrices is $O(H\delta)$-Isometric and hence approximately preserves this. Lemma 2 says that with high probability:

$$\begin{aligned}\Bigg|&\left\langle \tilde{R}R_{M(\theta^\star),1}e_j, \tilde{R}R_{M(\theta^\star),2}e_1\right\rangle\\ &- \left\langle R_{M(\theta^\star),1}e_j, R_{M(\theta^\star),2}e_1\right\rangle\Bigg| \leq O(H\delta)\end{aligned}$$

By the triangle inequality, we now know that with high probability, the sub-term where we choose $R/2$ every time is at most $O(H\delta)$ additive error away from the j^{th} coordinate of $x_{\theta^\star}$.

We now consider the other sub-terms. These other sub-terms are of the form:

$$2e_j^T R_{M(\theta^\star),1}^T \tilde{R}^T \tilde{R}' \left[\frac{1}{2}R_{M(\theta^\star),1}x_{\theta^\star} + \frac{1}{2}R_{M(\theta^\star),2}e_1\right]$$

where $\tilde{R}$ is still $\overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}}[R_{\theta^\star,j}]$ but $\tilde{R}'$ is now the product of a subsequence of those terms; we write it as $\overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}}[S_{\theta^\star,j}]$ where $S_{\theta^\star,j}$ is either $R_{\theta^\star,j}$ or the identity matrix.

Next, we define $x_J \coloneqq \overset{\curvearrowleft}{\prod_{j=J}^{3h(\theta^\star)}}[R_{\theta^\star,j}]R_{M(\theta^\star),1}e_j$ and $x'_J \coloneqq \overset{\curvearrowleft}{\prod_{j=J}^{3h(\theta^\star)}}[S_{\theta^\star,j}]s_{\text{object}}(\theta)$. Additionally, when $J = 3h(\theta^\star)+1$, $x_J = R_{M(\theta^\star),1}e_j$ and $x'_J = s_{\text{object}}(\theta)$. Our plan is to induct backwards from $J = 3h(\theta^\star)+1$, controlling the dot product of the two.

By Lemma 1, we know that all x_J and x'_J have squared ℓ_2 length at most $(1+\delta)^{3h(\theta^\star)+1}$. Under our assumptions, this is less than e and is hence a constant.

Lemma 7 and Lemma 2 already imply that for our base case of $J = 3h(\theta^\star)+1$, the dot product of the two is at most $O(\delta)$ away from the j^{th} coordinate of $x_{\theta^\star}$.

How does the dot product change from J to $J-1$? We either multiply x_J by a random matrix, or both x_J and x'_J by the same random matrix. In the first case, the δ-Desynchronization of our matrices tells us that the new dot-product will be zero times the old dot-product plus $O(\delta)$ error. In the second case, the δ-Isometry of our matrices tells us that the new

dot-product will be the old dot-product plus $O(\delta)$ error. Since the first case must occur *at least once*, we know that the final dot product when $J = 1$ consists of no signal and $O(h(\theta^\star)\delta)$ noise.

Hence, summing up our $2^{3h(\theta^\star)} - 1$ other sub-terms, we arrive at $O(2^{3h(\theta^\star)}h(\theta^\star)\delta)$ noise. It remains to consider the other terms.

Unfortunately, we will need to analyze our other terms by breaking them into sub-terms as well, since otherwise we would need to consider the situation where R is applied to one vector and $\left(\frac{I+R}{2}\right)$ to the other. The other terms have the form:

$$\beta w_\theta e_j^T R_{M(\theta^\star),1}^T \overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}} \left[R_{\theta^\star,j}^T\right] \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

We break our terms into sub-terms, but *only for the first* $3h(\theta^\star)$ *choices*. Our analysis focuses on handling the case where θ is at a greater depth than $\theta^\star$; we handle the other case by inserting identity matrices and splitting them into $I/2 + I/2$. We wind up with $2^{3h(\theta^\star)}$ sub-terms. Such a sub-term has the form:

$$(2w_\theta / w_{\theta^\star}) e_j^T R_{M(\theta^\star),1}^T \tilde{R}^T \tilde{R}' \tilde{R}'' s_{\text{object}}(\theta)$$

where

$$\tilde{R} \coloneqq \overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta^\star)}} [R_{\theta^\star,j}]$$

$$\tilde{R}' \coloneqq \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta^\star)}} [S_{\theta,j}]$$

$$\tilde{R}'' \coloneqq \overset{\curvearrowright}{\prod_{j=3h(\theta^\star)+1}^{3h(\theta)}} \left[\frac{I + R_{\theta,j}}{2}\right]$$

and $S_{\theta,j}$ is either $R_{\theta,j}$ or the identity matrix.

We define:

$$x_J \coloneqq \overset{\curvearrowleft}{\prod_{j=J}^{3h(\theta^\star)}} [R_{\theta^\star,j}] R_{M(\theta^\star),1} e_j$$

$$x'_J \coloneqq \overset{\curvearrowright}{\prod_{j=J}^{3h(\theta^\star)}} [S_{\theta,j}] \tilde{R}'' s_{\text{object}}(\theta)$$

When $J = 3h(\theta^\star) + 1$, we define $x_J \coloneqq R_{M(\theta^\star),1} e_j$ and $x'_J = \tilde{R}'' s_{\text{object}}(\theta)$.

As usual, Lemma 1 guarantees that all x_J and x'_J have constant ℓ_2 length. For our base induction case $J = [\max(3h(\theta^\star), 3h(\theta)) + 1]$, we would like to know the dot product of $R_{M(\theta^\star),1} e_j$ with $\tilde{R}'' s_{\text{object}}(\theta)$.

We first analyze the dot product of $R_{M(\theta^\star),1}$ with $s_{\text{object}}(\theta)$. Keep in mind that it may be the case that $M(\theta) = M(\theta^\star)$. In the same module case, we know by Lemma 2 and Lemma 7 that we get a signal of the j^{th} component of x_θ along with $O(\delta)$ noise. In the different module case, Lemma 7 is enough to tell us that there is no signal and $O(\delta)$ noise. As a reminder, the goal is to show that this signal *does not get through*, since we only want the j^{th} component of $x_{\theta^\star}$, not any other x_θ.

We know by Lemma 3 that our transparent matrices are $\frac{1}{2}$-leaky $O(\delta)$-Desynchronizing. Hence after applying all the transparent matrices, we know that at most we have our original signal (the j^{th} component of x_θ or nothing) and $O(\delta)$ noise (the noise telescopes due to the $\frac{1}{2}$ leaking/scaling). This is the base case of our backwards induction, $J = 3h(\theta^\star) + 1$.

We now consider what happens to the dot product of x_J and x'_J as we move from J to $J-1$. We may either (a) multiply both by different random matrices, (b) multiply both by the same random matrix, or (c) multiply only one by a random matrix. Furthermore, we know that the sequences $R_{\theta^\star,\cdot}$ and $R_{\theta,\cdot}$ differ at some point in the first $3h(\theta^\star)$ matrices, so we must end up in case (a) or case (c) at least once.

For case (a), we know by Lemma 7 that the current dot-product will be multiplied by zero and $O(\delta)$ noise will be added. For case (b), we know by Lemma 2 that the current dot-product will be multiplied by one and $O(\delta)$ noise will be added. For case (c), we know by d-Desynchronization that the current dot-product will be multiplied by zero and $O(\delta)$ noise will be added.

Since case (a) or (c) must occur at least once, we know that we get no signal and at most $O(h(\theta^\star)\delta)$ noise. Since each of our $2^{3h(\theta^\star)}$ sub-terms has a multiplier of $(2w_\theta/w_{\theta^\star})$ on top of the dot product of x_1 and x'_1, each term has noise bounded in magnitude (still with high probability) by $O(w_\theta 2^{3h(\theta^\star)} h(\theta^\star)\delta/w_{\theta^\star})$. Summing over all objects θ, we get our final desired claim: with high probability, the total additive noise is bounded by $O(2^{3h(\theta^\star)} h(\theta^\star)\delta/w_{\theta^\star})$. This completes the proof. □

Proof Notes. Although our proofs were written as if we extracted one attribute value at a time, we can extract them simultaneously by computing the vector $\beta R^T_{M(\theta^\star),1} s$, which is exactly a stacked version of the terms we were considering. Our technique is extendable to cases between (i) and (ii). For example, suppose we know that only one object in the overall sketch was both produced by the eye module and then subsequently used by the face module. We could recover its attributes by computing the vector $\beta R^T_{\text{eye},1} R^T_{\text{face},0} s$. Finally, the way we extract the attributes from the overall sketch is the same way we could extract attributes from a module sketch, meaning we do not need to know the precise sketch we are working with when attempting to extract information (although it will of course affect the error).

D.2. Proofs for Sketch-to-Sketch Similarity

Theorem 10. *Our final sketch has the **Sketch-to-Sketch Similarity** property, which is as follows. Suppose we have two overall sketches s and $\bar{s}$.*

(i) If the two sketches share no modules, then with high probability they have at most $O(\delta)$ dot-product.

(ii) If s has an object $\theta^\star$ of weight $w_{\theta^\star}$ and $\bar{s}$ has an object $\bar{\theta}^\star$ of weight $\bar{w}_{\theta^\star}$ and both objects are produced by the same module, then with high probability they have at least $\Omega(2^{-6h(\theta^\star)} w_{\theta^\star} \bar{w}_{\bar{\theta}^\star}) - O(\delta)$ dot-product.

(iii) If the two sketches are identical except that the attributes of any object θ differ by at most ϵ in ℓ_2 distance, then with probability one the two sketches are at most $\epsilon/2$ from each other in ℓ_2 distance.

Proof. We prove cases (i) and (ii) together. Suppose that our overall sketches are $s, \bar{s}$. Then the dot-product of our sketches has the form:

$$\sum_\theta \sum_{\bar{\theta}} w_\theta \bar{w}_{\bar{\theta}} \; s_{\text{object}}(\theta)^T \overset{\curvearrowleft}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R^T_{\theta,j}}{2}\right)\right] \overset{\curvearrowright}{\prod_{j=1}^{3h(\bar{\theta})}} \left[\left(\frac{I + R_{\bar{\theta},j}}{2}\right)\right] s_{\text{object}}(\bar{\theta})$$

Our proof plan is to analyze a single term of this double-summation at a time. Fix θ and $\bar{\theta}$. We analyze this term with our toolkit of technical lemmas. The plan is to first understand $\langle s_{\text{object}}(\theta), s_{\text{object}}(\bar{\theta})\rangle$. There are two possibilities for this dot-product, depending on our two objects θ and $\bar{\theta}$. If these objects come from different modules, then we will want a small dot-product. If these objects come from the same module, then we will want a significant dot-product.

Dot-Product Under Different Modules. Since the matrices are different, we know by Lemma 7 that with high probability, all of the following are true:

$$\left|\left\langle R_{M(\theta),1}x_\theta, R_{M(\bar{\theta}),1}x_{\bar{\theta}}\right\rangle\right| \leq \delta\sqrt{1+\delta} \leq O(\delta)$$
$$\left|\left\langle R_{M(\theta),1}x_\theta, R_{M(\bar{\theta}),2}e_1\right\rangle\right| \leq \delta\sqrt{1+\delta} \leq O(\delta)$$
$$\left|\left\langle R_{M(\theta),2}e_1, R_{M(\bar{\theta}),1}x_{\bar{\theta}}\right\rangle\right| \leq \delta\sqrt{1+\delta} \leq O(\delta)$$
$$\left|\left\langle R_{M(\theta),2}e_1, R_{M(\bar{\theta}),2}e_1\right\rangle\right| \leq \delta\sqrt{1+\delta} \leq O(\delta)$$

Combining these statements and using the triangle inequality, we know that with high probability,

$$\left|\left\langle s_{\text{object}}(\theta), s_{\text{object}}(\bar{\theta})\right\rangle\right| \leq O(\delta)$$

Dot-Product Under Same Module. Since the matrices are the same, we know by Lemma 2 that with high probability, both of the following are true:

$$\left|\left\langle R_{M(\theta),1}x_\theta, R_{M(\bar{\theta}),1}x_{\bar{\theta}}\right\rangle - \langle x_\theta, x_{\bar{\theta}}\rangle\right| \leq 3\delta \leq O(\delta)$$
$$\left|\left\langle R_{M(\theta),2}e_1, R_{M(\bar{\theta}),2}e_1\right\rangle - \langle e_1, e_1\rangle\right| \leq 3\delta \leq O(\delta)$$

Also, we still know by Lemma 7 that with high probability, both of the following are true:

$$\left|\left\langle R_{M(\theta),1}x_\theta, R_{M(\bar{\theta}),2}e_1\right\rangle\right| \leq \delta\sqrt{1+\delta} \leq O(\delta)$$
$$\left|\left\langle R_{M(\theta),2}e_1, R_{M(\bar{\theta}),1}x_{\bar{\theta}}\right\rangle\right| \leq \delta\sqrt{1+\delta} \leq O(\delta)$$

Combining these statements with the triangle inequality, we know that with high probability,

$$\left|\left\langle s_{\text{object}}(\theta), s_{\text{object}}(\bar{\theta})\right\rangle - \frac{1}{4}\langle x_\theta, x_{\bar{\theta}}\rangle - \frac{1}{4}\langle e_1, e_1\rangle\right| \leq O(\delta)$$

Since attribute vectors are nonnegative, we know that:

$$\left\langle s_{\text{object}}(\theta), s_{\text{object}}(\bar{\theta})\right\rangle \geq \frac{1}{4} - O(\delta)$$

We have successfully established our desired facts about the base dot-product and are now ready to consider what happens when we begin applying matrices of the form $\left(\frac{I+R}{2}\right)$ to both vectors in this dot product. We begin by controlling the ℓ_2 norm of the various vectors produced by gradually applying these matrices. By Lemma 1, we know that every vector of the form

$$x_J = \overset{\curvearrowleft}{\prod_{j=J}^{3h(\theta)}}\left[\left(\frac{I+R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

where $J \in \{1, 2, \ldots, 3h(\theta)\}$ or of the form

$$\bar{x}_J = \overset{\curvearrowleft}{\prod_{j=J}^{3h(\bar{\theta})}}\left[\left(\frac{I+R_{\bar{\theta},j}}{2}\right)\right] s_{\text{object}}(\bar{\theta})$$

where $J \in \{1, 2, \ldots, 3h(\bar{\theta})\}$ has squared ℓ_2 norm at most $(\frac{1}{2}+\frac{3}{4}\delta)^{\max(3h(\theta),3h(\bar{\theta}))}(1+\delta)$. By our assumptions, this is upper-bounded by a constant:

$$\begin{aligned}
&(\frac{1}{2}+\frac{3}{4}\delta)^{\max(3h(\theta),3h(\bar{\theta}))}(1+\delta)\\
&\leq (1+\delta)^{\max(3h(\theta),3h(\bar{\theta}))+1}\\
&\leq (1+1/(2H))^{2H}\\
&\leq e
\end{aligned}$$

We have shown that our unit vectors maintain constant ℓ_2 norm throughout this process. This allows us to safely apply our desynchronization and isometry properties to any x_J or $\bar{x}_J$.

From here, we plan to induct in order of decreasing depth. For homogeneity of notation, we will define x_J to be $s_{\text{object}}(\theta)$ when $J > 3h(\theta)$ and $\bar{x}_J$ to be $s_{\text{object}}(\bar{\theta})$ when $J > 3h(\bar{\theta})$. We induct backwards, beginning with $J = \max(3h(\theta), 3h(\bar{\theta}))+1$ and ending with $J = 1$.

In our base case, we know that these are two possibilities for $\langle x_J, \bar{x}_J \rangle$. If we are in the different module case, then the magnitude of this quantity is $O(\delta)$. If we are in the same module case, then this quantity is at least $\frac{1}{4} - O(\delta)$.

To get from J to $J-1$, there are three possibilities. We either (a) multiply both x_J and $\bar{x}_J$ by the same matrix, (b) multiply x_J and $\bar{x}_J$ by different matrices, or (c) multiply only one of them by a matrix. Note that case (c) happens when one object is at a different depth than the other object, so our analysis first peels off the additional matrices to make them the same depth, then proceeds. This is an important point, because tuple matrices are only shared across the same depth.

In any one of these lettered cases, we know from Lemma 3 and Lemma 4 that the distribution of the matrix(ces) is $\frac{1}{2}$-leaky $O(\delta)$-Desynchronizing and $\frac{1}{2}$-scaled $O(\delta)$-Isometric.

In case (a), we apply Lemma 2 to show that with high probability, the new dot product is $\frac{1}{2}$ times the old dot product plus $O(\delta)$ additive noise. In case (b), we apply Lemma 7 to show that with high probability, the new dot product is $\frac{1}{4}$ times the old dot product plus $O(\delta)$ additive noise. In case (c), we use the fact that our distribution is $\frac{1}{2}$-leaky $O(\delta)$-Desynchronizing to show that the new dot product is $\frac{1}{2}$ times the old dot product, plus $O(\delta)$ additive noise.

Hence, if the objects were produced by different modules, then we begin with $O(\delta)$ noise and at every step we reduce by at least a factor of $\frac{1}{2}$ before adding $O(\delta)$ noise. The total noise telescopes into $O(\delta)$ total. If the objects were produced by the same module, then we begin with at least $\frac{1}{4}$ signal and $O(\delta)$ noise. The noise sums as before into $O(\delta)$ total, and the signal falls by at most $\frac{1}{4}$ per step, resulting in a final signal of $\Omega(2^{-6h(\theta)})$.

However, our original term under consideration was actually a scaled version of what we just analyzed: $w_\theta \bar{w}_{\bar{\theta}} x_1^T \bar{x}_1$. Hence in the different module case, there is $O(w_\theta \bar{w}_{\bar{\theta}} \delta)$ noise, and in the same module case there is $\Omega(2^{-6h(\theta)} w_\theta \bar{w}_{\bar{\theta}})$ signal and $O(w_\theta \bar{w}_{\bar{\theta}} \delta)$ noise.

Hence for case (i) which we were originally trying to prove, since the effective weights w_θ sum to one and the effective weights $\bar{w}_{\bar{\theta}}$ sum to one, we wind up, with high probability, at most $O(\delta)$ total noise, as desired. For case (ii) which we were originally trying to prove, we get at least $\Omega(2^{-6h(\theta^\star)} w_{\theta^\star} \bar{w}_{\bar{\theta^\star}}) - O(\delta)$ dot product.

We conclude by proving case (iii). Suppose that in sketch $\bar{s}$, the attributes of θ are $x_\theta + \epsilon_\theta$, where $||\epsilon_\theta||_2 \leq \epsilon$ for all objects θ. Let's write the difference between the two sketches.

$$\bar{s} - s = \sum_\theta w_\theta \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2} \right) \right] \frac{1}{2} R_{M(\theta),1} \epsilon_\theta$$

Applying Lemma 4, we know that $\left(\frac{I+R}{2}\right)$ is $\frac{1}{2}$-scaled $\frac{3}{4}\delta$-Isometric. Repeatedly applying Lemma 2, we can conclude that with high probability:

$$\begin{aligned}
&\left|\left| \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2} \right) \right] R_{M(\theta),1} \frac{1}{4} \epsilon_\theta \right|\right|_2^2 \\
&\leq (\frac{1}{2} + \frac{3}{4}\delta)^{3h(\theta)} (1+\delta) \frac{1}{4} \epsilon^2 \\
&\leq (\frac{1}{2} + \frac{3}{8})^{3h(\theta)+1} \frac{1}{4} \epsilon^2 \\
&\leq \frac{1}{4} \epsilon^2
\end{aligned}$$

Taking the square root of the above and applying the triangle inequality yields the following.

$$\begin{aligned} ||\bar{s} - s||_2 &\leq \sum_\theta w_\theta \frac{1}{2}\epsilon \\ &\leq \frac{1}{2}\epsilon \end{aligned}$$

This completes the proof of case (iii). □

D.3. Proofs for Summary Statistics

Theorem 11. *Our final sketch has the **Summary Statistics** property, which is as follows. Consider a module $M^\star$ which lies at depth $h(M^\star)$, and suppose all the objects produced by $M^\star$ has the same effective weight $w^\star$.*

- *(i) **Frequency Recovery**. We can produce an estimate of the number of objects produced by $M^\star$. With high probability, our estimate has an additive error of at most $O(2^{3h(M^\star)}\delta/w^\star)$.*
- *(ii) **Summed Attribute Recovery**. We can produce a vector estimate of the summed attribute vectors of the objects produced by $M^\star$. With high probability, our estimate has additive error of at most $O(2^{3h(M^\star)}\delta/w^\star)$ in each coordinate.*

Proof. Let the overall sketch be s.

We begin by proving case (i). As we did in case (ii) of **Attribute Recovery**, let $\beta = 2^{3h(M^\star)+1}/w^\star$. We compute $\beta e_1^T R_{M^\star,1}^T s$, which has the form:

$$\sum_\theta \beta w_\theta e_1^T R_{M^\star,2}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I + R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

There are two types of terms in this sum: those objects produced by $M^\star$, and those that are not. Consider some object $\theta^\star$ produced by $M^\star$. When $\theta = \theta^\star$, we get the term:

$$2^{3h(\theta^\star)+1} e_1^T R_{M^\star,2}^T \overset{\curvearrowright}{\prod_{j=1}^{3h(M^\star)}} \left[\left(\frac{I + R_{\theta^\star,j}}{2}\right)\right] s_{\text{object}}(\theta^\star)$$

If we expand out each transparent matrix, we would get $2^{3h(M^\star)}$ sub-terms. Again, focus on the sub-term where we choose $I/2$ every time:

$$2e_1^T R_{M^\star,2}^T \left[\frac{1}{2} R_{M^\star,1} x_{\theta^\star} + \frac{1}{2} R_{M^\star,2} e_1\right]$$

By the δ-Isometry property of our distribution, we know that with high probability (note e_1 is a unit vector):

$$\left|2e_1^T R_{M^\star,2}^T \times \frac{1}{2} R_{M^\star,2} e_1 - e_1^T e_1\right| \leq \delta$$

This half of the sub-term (up to additive error $O(\delta)$) contributes one to our estimate, representing one copy of an object produced by $M^\star$. We control the other half of the sub-term using Lemma 7:

$$\left|2e_1^T R_{M^\star,2}^T \times \frac{1}{2} R_{M^\star,1} x_{\theta^\star}\right| \leq \delta\sqrt{1+\delta} = O(\delta)$$

Next, we bound the additive error from the other sub-terms of such a term $\theta^\star$. These other sub-terms have the form:

$$2e_1^T R_{M^\star,2}^T \tilde{R} s_{\text{object}}(\theta^\star)$$

where $\tilde{R}$ is a product of one to $3h(M^\star)$ R matrices. As before, we apply Lemma 1 so that with high probability $s_{\text{object}}(\theta^\star)$ and $R_{M^\star,2}e_1$ have squared ℓ_2 norm at most $(1+\delta)$, and then conclude using Lemma 5 that the distribution of $\tilde{R}$ satisfies $O(\delta)$-Desynchronization (this was the worst case). Hence with high probability the absolute value of any other sub-term was bounded by $O(\delta)$, and the absolute value of the sum of the other sub-terms for $\theta^\star$ is bounded by $O(2^{3h(M^\star)}\delta)$.

There are at most $1/w^\star$ such objects, so the total error from these terms is $O(2^{3h(M^\star)}\delta/w^\star)$, as desired. It remains to show that the total error from the other terms is also $O(2^{3h(M^\star)}\delta/w^\star)$. These other terms have the form:

$$\beta w_\theta e_1^T R_{M^\star,2}^T \overset{\frown}{\prod_{j=1}^{3h(\theta)}} \left[\left(\frac{I+R_{\theta,j}}{2}\right)\right] s_{\text{object}}(\theta)$$

We prove this using the δ-Desynchronization of $R_{M^\star,2}$. To do so, we will need to bound the ℓ_2 length of $s_{\text{object}}(\theta)$ as it gets multiplied by this sequence of matrices. As before, we use Lemma 4 to know that the distribution of these matrices is $\frac{1}{2}$-scaled $(\frac{3}{4}\delta)$-Isometric, and then repeatedly apply Lemma 6 to conclude that the squared ℓ_2 length is upper bounded by a constant. Hence the δ-Desynchronization of $R_{M^\star,2}$ implies that with high probability the absolute value of the term indexed by θ is at most $O(\beta w_\theta \delta)$. Hence the absolute value of all other terms is at most $O(\beta\delta)$. Plugging in our choice of β, it is at most $O(2^{3h(M^\star)}\delta/w^\star)$, as desired.

We have finished proving case (i); with high probability, we can recover the number of objects produced by module $M^\star$, with the desired additive error.

We now prove case (ii). We do exactly what we did in case (i), except that instead of computing $\beta e_1^T R_{M^\star,2}^T s$ we compute $\beta e_j^T R_{M^\star,1}^T$. Instead of operating on the first coordinate of the e_1 in our object sketch, we now operate on the j^{th} coordinate of the x_θ in our object sketch, and hence we can recover the j^{th} coordinate of the sum of the attribute vectors of the appropriate objects to the same error as in case (i). □

Proof Notes. As when we were reasoning about the **Attribute Recovery** property, we can simultaneously extract the entire summed attribute vector at the same time by computing the vector $\beta R_{M^\star,1}^T s$. Additionally, we focus on more precise groups of objects than all those output by a particular module. For example, if we cared about objects produced by the eye module and then subsequently used by the face module, we could recover their approximate frequency and summed attribute vector by computing $\beta e_1^T R_{\text{eye},2}^T R_{\text{face},0}^T s$ and $\beta R_{\text{eye},1}^T R_{\text{face},0}^T s$.

D.4. Generalizing for Object Signature Recovery

In this subsection, we explain how to also satisfy the **Object Signature Recovery** property through a small modification to our sketch. The key idea is that each object is given a "magic number" based on its attributes: m_θ is a random $(\log N)$-sparse unit vector whose entries are in $\{0, \frac{1}{\sqrt{\log N}}\}$. We revise the sketch of object θ to be $\frac{1}{3}R_{M(\theta),1}x_\theta + \frac{1}{3}R_{M(\theta),2}e_1 + \frac{1}{3}R_{M(\theta),3}m_\theta$. The process of recovering an object signature is the same as recovering its attribute vector, except that the signature is quantized and therefore as long as the additive error is less than $\frac{1}{2\sqrt{\log N}}$, we can recover it exactly.

D.5. Generalizing for Graceful Erasure

In this subsection, we explain how to generalize our previous proofs for the **Graceful Erasure** property. We will need to better understand the behavior of our matrices when only a prefix of the result is considered. We can extend our definitions as follows:

Definition 4. *We say that a distribution D over random $d_{sketch} \times d_{sketch}$ matrices satisfies the d_{DE}-prefix α_{DE}-leaky δ_{DE}-Desynchronization Property if the following is true. If $R \sim D$, then the random variable $Z = \sum_{i=1}^{d_{DE}} (Rx)_i(y)_i - \alpha_{DE}\langle x, y\rangle$ is a δ_{DE}-nice noise variable. If $\alpha_{DE} = 0$ then we simply refer to this as the d_{DE}-prefix δ_{DE}-Desynchronization Property.*

Definition 5. *We say that a distribution D over random $d_{sketch} \times d_{sketch}$ matrices satisfies the d_{IE}-prefix α_{IE}-leaky δ_{IE}-Isometry Property if the following is true. If $R \sim D$ and $x = y$, then the random variable $Z = \sum_{i=1}^{d_{IE}} (Rx)_i^2 - \alpha_{IE}\langle x, x\rangle$ is a δ_{IE}-nice noise variable. If $\alpha_{IE} = 1$ then we simply refer to this as the d_{IE}-prefix δ_{IE}-Isometry Property.*

The next step is to extend our technical lemmas to reason about these prefixes. For example, we will again want to instead compare $\sum_{i=1}^{d_{IE}} (Rx)_i(Ry)_i$ and $\langle x, y\rangle$ instead, so Lemma 2 turns into the following:

Lemma 8. *If D satisfies the d_{IE}-prefix α_{IE}-leaky δ_{IE}-Isometry Property, then the random variable $Z = \sum_{i=1}^{d_{IE}} (Rx)_i (Ry)_i - \alpha_{IE} \langle x, y \rangle$ is a $(3\delta_{IE})$-nice noise variable.*

Proof. The proof is the same idea as before. The claim is trivially true if x or y is zero, and then without loss of generality we scale them to unit vectors.

We invoke the d_{IE}-prefix α_{IE}-leaky δ_{IE}-Isometry Property three times, on $x_1 = x + y$, $x_2 = x$, and $x_3 = y$. This implies that the following three noise variables are d_{IE}-nice:

$$Z_1 = \sum_{i=1}^{d_{IE}} (R(x+y))_i^2 - \alpha_{IE} \langle x+y, x+y \rangle$$
$$Z_2 = \sum_{i=1}^{d_{IE}} (R(x))_i^2 - \alpha_I \langle x, x \rangle$$
$$Z_3 = \sum_{i=1}^{d_{IE}} (R(y))_i^2 - \alpha_I \langle y, y \rangle$$

By construction, $Z = \frac{1}{2} Z_1 - \frac{1}{2} Z_2 - \frac{1}{2} Z_3$, so Z is centered. We bound it by noting that $||x+y||_2^2 \leq 4$. We know that with high probability, $|Z_1| \leq 4\delta_{IE}$, $|Z_2| \leq \delta_{IE}$, and $|Z_3| \leq \delta_{IE}$. By the triangle inequality, Z is $(3\delta_{IE})$-bounded, making it a $(3\delta_{IE})$-noise variable and completing the proof. □

Using extended technical lemmas, it is possible to generalize our various properties to the case where we only have a prefix of the sketch. For example, the following claim generalizes case (i) of the **Attribute Recovery** property.

Claim 6. *Suppose we have an overall sketch s and we keep only a $d'_{sketch} \leq d_{sketch}$ prefix of it, s'. Consider an object $\theta^\star$ which lay at depth $h(\theta^\star)$ and had effective weight $w_{\theta^\star}$ in the original sketch s. Suppose no other objects in s are also produced by $M(\theta^\star)$. We can produce a vector estimate of the attribute vector $x_{\theta^\star}$ from s', which with high probability has an additive error of at most $O(2^{3h(\theta^\star)} \delta_E / w_{\theta^\star})$ in each coordinate.*

Proof Sketch. We have an overall sketch s and a version s' where the last $(d_{\text{sketch}} - d'_{\text{sketch}})$ coordinates have been replaced with zeros. We compute $\beta e_j^T R_{M(\theta^\star),1}^T s'$. From our proof of case (i) of Theorem 9, there are three things to analyze: the primary signal sub-term, the other sub-terms, and the other terms.

The primary signal sub-term is now the following.

$$\sum_{i=1}^{d'_{\text{sketch}}} \left[\left(\frac{1}{2} R_{M(\theta^\star),1} x_{\theta^\star} + \frac{1}{2} R_{M(\theta^\star),2} e_1 \right)_i \left(2 R_{M(\theta^\star),1} e_j \right)_i \right]$$

Using d'_{sketch}-prefix isometry, we can show that the first half is a (scaled-down) of $e_j^T x_{\theta^\star}$ plus (scaled-down) small noise. Using a generalization of Lemma 7, we can show that the second half is (scaled-down) small noise.

When analyzing the other sub-terms, we use the d'_{sketch}-prefix desynchronization of the final matrix in $\tilde{R}$ to show the the sub-term is just (scaled-down) small noise. Finally, when analyzing the other terms, we use the d'_{sketch}-prefix desynchronization of $R_{M(\theta^\star),1}$ to show that the term is just (scaled-down) small noise. □

E. Proofs for Dictionary Learning

E.1. Proof of Network Learnability

In this subsection, we prove Theorem 5. Recall Theorem 5:

Theorem 5. *If the teacher deep modular network has constant depth, then any module $M^\star$ which satisfies the following two properties:*

- *(Robust Module Learnability) The module is learnable from $(\alpha = poly(N))$ input/output training pairs which have been corrupted by ℓ_∞ error at most a constant $\epsilon > 0$.*

- *(Sufficient Weight) In a* $\left(\beta = \frac{1}{poly(N)}\right)$*-fraction of the inputs, the module produces an object and all of the input objects to that object have effective weight at least* w*.*

can, with high probability, be learned from poly(N) *overall sketches of dimension poly*$(1/w, 1/\epsilon) \log^2 N$*.*

Suppose we additionally know that the communication graph is a fixed tree. We can identify the sub-graph of objects which each have effective weight w *in a* $\left(\beta = \frac{1}{poly(N)}\right)$*-fraction of the inputs.*

Proof. Our plan is to use Theorem 4 to unwind our sketching. Precisely, we invoke it with N as thrice the maximum between the number of modules and the number of objects in any communication graph, $\tilde{S}$ as $(\alpha/\beta) \log N = \text{poly}(N)$, H as three times the depth of our modular network, and ϵ_H as $\min(\epsilon/4, 1/4)2^{-H}w$. This yields a sketch dimension $d_{\text{sketch}} \leq \text{poly}(1/\epsilon_H) \log N \log \text{poly}(N)$ which is $\text{poly}(1/w, 1/\epsilon) \log^2 N$ as claimed. Additionally, it yields a base number of samples $S = \text{poly}(N)$ and a sequence of ℓ_∞ errors $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_{H-1} \leq \epsilon_H$. These are guarantees on an algorithm A which accepts an $h \in [3 \cdot \text{ network depth}]$ as well as $S_h = \tilde{S}N^{h-1}$ samples with up to ϵ_h ℓ_∞ error and $O(\sqrt{d_{\text{sketch}}})$ ℓ_1 error.

This base number of samples is why we will require $\text{poly}(N)$ overall sketches. By construction, overall sketches are just input subsketches of the output psuedo-object. We know that no communication graph contains more than $N/3$ objects by the definition of N, so we can write all the overall sketches in the form:

$$s_{\text{overall}} = \sum_{i=1}^{N} x_i + R_i x_i$$

where x_i is $(w_i/2)s_{\text{object}}(\theta_i)$ if there actually was an i^{th} input and the zero vector otherwise. We want to treat $\sum_i x_i$ as ℓ_1 noise, but how much do they contribute? We can only tolerate them contributing at most $O(\sqrt{d_{\text{sketch}}})$ ℓ_1 norm. In fact, we claim that any sketch vector has at most $O(1)$ ℓ_2 norm, which would imply the desired statement by Cauchy-Schwartz and noticing that we are taking a convex combination of sketch vectors. We prove an ℓ_2 bound on sketch vectors as in Appendix D, but roughly the idea is that our error parameter δ, which we know to be $O(\frac{\log N}{\sqrt{d_{\text{sketch}}}})$, needs to be less than the constant $\frac{1}{8H}$ (recall that our network depth is constant). Since our sketch dimension already scaled with $\log^2 N$, this is true up to making our sketch dimension a constant factor larger. We actually have no additional ℓ_∞ noise, which is under the ϵ_1 requirement. Hence, we can safely apply A and retrieve $(w_i/2)$ times our object sketches (and some zero vectors), up to ϵ_2 ℓ_∞ noise. Note that we now have a total of $\tilde{S}N$ vectors, since the procedure turns one vector into N vectors.

If any of our retrieved vectors have less than ϵ_2 ℓ_∞ norm, we declare them garbage and any further vectors that dictionary learning recovers from them as garbage. When learning modules, we will not use any garbage vectors. Now, we have several approximate, scaled object sketches and several garbage vectors. We note that the scaled object sketches, by definition, are of the form:

$$w \cdot s_{\text{object}}(\theta) = w \left(\frac{I + R_{M(\theta),0}}{2}\right) x \qquad \left(+ \sum_{M \neq M(\theta)} R_{M,0}\vec{0}\right)$$

where w is some weight from the previous step and x is w times a tuple sketch. Undoing the first matrix off an object sketch is exactly like the previous step; we handle the transparent matrix by noting that all sketches have constant ℓ_2 norm with high probability and hence it's a matter of ℓ_1 noise for our algorithm. Our ϵ_2 ℓ_∞ noise turns into ϵ_3 ℓ_∞ noise. The garbage vectors can be thought of as ℓ_∞ noise plus the zero vector, which also decomposes nicely:

$$\vec{0} = \sum_M R_{M,0}\vec{0}$$

so the dictionary learning procedure can safely handle them. There are at most $N/3$ matrices involved (we can imagine the problem as padded with zero matrices and additional zero vectors). We recover $\tilde{S}N^2$ vectors, which are attribute subsketches, input subsketches, and garbage vectors.

The process of undoing attribute subsketches and input subsketches is why we need N to be thrice the maximum between the number of modules and the number of objects in any communication graph. For attribute subsketches, there are up to two matrices for every module at this depth. For input subsketches, there is possibly a tuple matrix for every object at this

depth. Still, there are at most N matrices involved, so we can recover our SN^3 vectors, which are object attribute vectors, e_1's, object sketches, and garbage vectors.

Unlike previous steps, where we had one type of vector (and garbage vectors), we now have *three types of vectors* that we want to distinguish; we can only recurse on object sketches because we cannot guarantee that object attribute vectors or e_1 can be written as the correct matrix-vector product form. We can get some helpful information by looking at which vectors produced which vectors in our dictionary learning procedure. By construction, we know that one object sketch gets turned into an attribute subsketch and an input subsketch. The former gets turned into an attribute vector and e_1 while the latter gets turned into a series of object sketches. For example, if the input subsketch produces at least three non-garbage vectors, then we know which subsketch was which (the attribute subsketch only produces up to two non-garbage vectors with high probability). Of course, this idea won't help us distinguish in general; an object might only have two input objects! The better test is to see which subsketch produces a vector that looks more like e_1. To begin with, it's worth noting that the two subsketches have the same scaling: the effective weight of object θ, w_θ, times $2^{-2h(\theta)+2}$.

Now, let's consider how small the first coordinate of the (scaled) e_1 vector can be. It picks up an additional $1/2$ scaling factor, so the vector we recover is $w_\theta 2^{-2h(\theta)+1} e_1$ up to $\epsilon_{3h(\theta)-2}$ additional ℓ_∞ error. We know that $\epsilon_{3h(\theta)-2} < \epsilon_H < w2^{-H}$, so the first coordinate is at least:

$$w_\theta 2^{-2h(\theta)+1} - w2^{-H}$$

Now, let's consider how large the first (or any) coordinate of any object sketch could be. Due to Theorem 7, we know that any (at most unit in length) vector has a first coordinate of at most $O(\frac{\log N}{\sqrt{d_{\text{sketch}}}})$ after one of our block random matrices is applied to it. Furthermore, applying the various transparent matrices $\left(\frac{I+R}{2}\right)$ throughout our sketch increases this by at most a constant factor. The object sketch is just a convex combination of such vectors, so it also has at most $O(\frac{\log N}{\sqrt{d_{\text{sketch}}}})$ in its first coordinate. Since there is up to $w2^{-H}$ additional ℓ_∞ error in the recovery process and this vector is scaled too, we know that the recovered object sketch vector has first coordinate at most:

$$w_\theta 2^{-2h(\theta)+2} O(\frac{\log N}{\sqrt{d_{\text{sketch}}}}) + w2^{-H}$$

We require that $O(\frac{\log N}{\sqrt{d_{\text{sketch}}}}) < \frac{1}{8}$. As already discussed, since our sketch dimension already scaled with $\log^2 N$, this amounts to making the sketch dimension at most a constant factor larger. Now, our recovered object sketch vector has first coordinate at most:

$$w_\theta 2^{-2h(\theta)+1}/4 + w2^{-H}$$

Whether our two cases are distinguishable still depends on how θ's effective weight w_θ relates to the goal weight w. We could clearly distinguish the two if $w_\theta 2^{-2h(\theta)+1}$ was at least $4w2^{-H}$, since then the first case would yield a value of at least $3w2^{-H}$ and the second case a value of at most $2w2^{-H}$. Hence our test is the following: if some vector produced by the object sketch has a first coordinate of at least $3w2^{-H}$, then take the one with the largest first coordinate and decide that it was produced by the object attribute subsketch. If no vector produced by the object sketch has a first coordinate of at least $3w2^{-H}$, then swap them all for zero vectors and declare them garbage.

Notice that if a recovered object sketch vector manages to have a first coordinate value of at least $3w2^{-H}$, then we know that $w_\theta 2^{-2h(\theta)+1}$ is at least $8w2^{-H}$ and hence the scaled e_1 still beats all the recovered object sketch vectors. On the other hand, all the vectors are declared garbage, then we know that $w_\theta 2^{-2h(\theta)+1} < 4w2^{-H}$ and hence $w_\theta < w$ (i.e. we don't need to recover this object or any of its children). Hence our procedure works and we can continue to safely recurse.

The result of this procedure is that for each module (we can tell which vectors belong to which modules due to shared final matrices) we recover unordered pairs of scaled (attribute vector, e_1), at least every time the module produces an object with at least w weight (possibly more). We'll consider the one with larger first coordinate to be e_1, and divide the other vector by the first coordinate of e_1 to unscale it.

Note that both vectors have been scaled by $w_\theta 2^{-2h(\theta)+1}$ and then face at most ϵ_H additional ℓ_∞ noise. note that we chose a $\epsilon_H < (\epsilon/4)2^{-H}w$ and no vector could have passed our test unless $w_\theta 2^{-2h(\theta)+1} \geq 2w2^{-H}$, i.e. $w/w_\theta \leq 2^{H-2h(\theta)}$.

Combining these facts, we know that a single instance of the ℓ_∞ noise messes up any (unscaled) coordinate by at most:

$$\begin{aligned} 2^{2h(\theta)-1}\epsilon_H/w_\theta &\leq 2^{2h(\theta)-1}(\epsilon/4)2^{-H}w/w_\theta \\ &\leq 2^{2h(\theta)-1}(\epsilon/4)2^{-H}2^{H-2h(\theta)} \\ &= (\epsilon/8) \end{aligned}$$

There are two cases: (i) we consider the correct vector to be e_1 and (ii) we consider the object attribute vector to be e_1. In case (i), our object attribute vector is scaled by a number which is within a multiplicative $(1 \pm \epsilon/8)$ of correct. Additionally, its noise makes it off by an additive $\epsilon/8$ in ℓ_∞ distance. Since we capped ϵ to be at most one, the multiplicative difference is at most $(9/8)/(1-\epsilon/8) - 9/8 \leq \frac{9}{56}\epsilon$, for a total difference of $\frac{2}{7}\epsilon$ ℓ_∞ distance.

In case (ii), the original attribute vector had to have first coordinate at least $1-(\epsilon/4)$. Since it has an ℓ_1 norm of at most one, we know that it is at most $\epsilon/4$ in ℓ_∞ distance from e_1. We imagine that the noised version of e_1 is it; by triangle inequality this choice is at most $\frac{3}{8}\epsilon$ from it. We again pick up at most a $\frac{9}{56}\epsilon$ error from incorrect scaling, for a total of at most $\frac{15}{28}\epsilon$ ℓ_∞ error.

We conclude the proof of the first half of our theorem by noting that since we started with $(\alpha/\beta)\log N$ samples, with high probability we will have at least α samples where we recover an input/output pair for $M^\star$ within the robust module learnability conditions.

Regarding fixed communication graphs, we can use this same algorithm to detect the entire path to an object with high probability (i.e. what modules produce each node along the path) if the object has effective weight w along this path. This means we can identify the entire sub-graph of objects which have effective weight w "often enough". □

E.2. Proof of Recursable Dictionary Learning

In this subsection, we prove Theorem 4, which we restate next.

Theorem 4. *[Recursable Dictionary Learning] There exists a family of distributions $\{\mathcal{D}(b, q, d_{sketch})\}$ which produce $d_{sketch} \times d_{sketch}$ matrices satisfying the following. For any positive integers $N, \underset{\sim}{S}$, positive constant H, positive real ϵ_H, block size $b \geq poly(\log N, \log d_{sketch}, 1/\epsilon_H)$, nonzero block probability $q \geq poly(\log N, \log d_{sketch}, 1/\epsilon_H)/\sqrt{d_{sketch}}$, and dimension $d_{sketch} \geq poly(1/\epsilon_H, \log N, \log \underset{\sim}{S})$, there exists:*

- *a base number of samples S where $\underset{\sim}{S} \leq S \leq \underset{\sim}{S} \cdot poly(N)$,*
- *and a sequence of ℓ_∞ errors $(0 <)\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_{H-1}(\leq \epsilon_H)$ with $\epsilon_1 \geq poly(\epsilon_H)$,*

such that the following is true. For any $h \in [H-1]$, let $S_h = SN^{h-1}$. For any unknown vectors $x^{(1)}, ..., x^{(S_h)} \in \mathbb{R}^{d_{sketch} \times N}$ with ℓ_2 at most $O(1)$, if we draw $R_1, ..., R_N \sim \mathcal{D}(d_{sketch})$ and receive S_h noisy samples $y^{(k)} := [R_1 R_2 \cdots R_N]x^{(k)} + z_1^{(k)} + z_\infty^{(k)}$ where each $z_1^{(k)} \in \mathbb{R}^{d_{sketch}}$ is noise with $\left|\left|z_1^{(k)}\right|\right|_1 \leq O(\sqrt{d_{sketch}})$ (independent of our random matrices) and each $z_\infty^{(k)} \in \mathbb{R}^{d_{sketch}}$ is noise with $\left|\left|z_\infty^{(k)}\right|\right|_\infty \leq \epsilon_h$ (also independent of our random matrices), then there is an algorithm which takes as input h, $y^{(1)}$, ..., $y^{(S_h)}$, runs in time $poly(S_h, d_{sketch})$, and with high probability outputs $\hat{R}_1, ..., \hat{R}_N, \hat{x}^{(1)}, ..., \hat{x}^{(S_h)}$ satisfying the following for some permutation π of $[N]$:

- *for every $i \in [N]$ and $j \in [d_{sketch}]$, if there exists $k^\star \in [S_h]$ such that $\left|x^{(k^\star)}_{(i-1)d_{sketch}+j}\right| \geq \epsilon_{h+1}$ then the j^{th} column of $\hat{R}_{\pi(i)}$ is $0.2d_{sketch}$ in Hamming distance from the j^{th} column of R_i.*
- *for every $k \in [S_h], i \in [N], j \in [d_{sketch}]$, $\left|x^{(k)}_{(\pi(i)-1)d_{sketch}+j} - x^{(k)}_{(i-1)d_{sketch}+j}\right| \leq \epsilon_{h+1}$.*

We point out that the ℓ_∞ bound on the recovery error in Theorem 4 implies that the x vectors can be recovered exactly if they are quantized and the recovery error ϵ_{h+1} is small enough.

Notation. We next provide some notation that will be used in the proof of Theorem 4. Let H be a given positive constant. Fix $h \in [H-1]$ and let $S_h = SN^{h-1}$ where S will be specified later.

- For each sample index $i \in [S_h]$, we divide the d_{sketch}-dimensional real vector $y^{(i)}$ into d_{sketch}/b contiguous blocks each of length b, with the ℓth block being $y^{(i)}[(\ell-1)b+1 :\ \ell b+1]$ for each $\ell \in [d_{\text{sketch}}/b]$.

- For any two points $x, y \in \mathbb{R}^n$, recall that their ℓ_∞-distance is defined as

$$d_\infty(x, y) := \|y - x\|_\infty = \max_{i \in [n]} |y_i - x_i|.$$

  Moreover, we define the symmetric ℓ_∞-distance between x and y as $\min(d_\infty(x, y), d_\infty(x, -y))$.

- For any vector $y \in \mathbb{R}^n$ and any finite subset L of $\mathbb{R}^n$, we denote the ℓ_∞-distance of y from L by

$$d_\infty(y, L) := \min_{x \in L} \|y - x\|_\infty.$$

  If this distance is at least τ, then we say that y is τ-far in ℓ_∞-distance from the set L.

- For any $y, y' \in \mathbb{R}^n$, we define their Hamming distance as $\Delta(y, y') := |\{j \in [n] : y_j \neq y'_j\}|$ and the symmetric Hamming distance as $\overline{\Delta}(y, y') = \min(\Delta(y, y'), \Delta(y, -y'))$.

Algorithm.

Algorithm 3 $\mathrm{Dec}(\overline{z}_c)$

Input: A vector $\overline{z}_c$ on the hypercube $\{\pm \frac{1}{\sqrt{q d_{\text{sketch}}}}\}^{b/3}$.
Output: A pair (j, f) where j is an index in $[d_{\text{sketch}}]$ and f is a sign.

if the first coordinate of $\overline{z}_c$ is negative **then**
    Negate each coordinate of $\overline{z}_c$.
end if
Set t to $\lceil \log_2 d_{\text{sketch}} \rceil$.
Let σ_c be the binary vector obtained from $\overline{z}_c[2 : t + 2]$ by replacing $-\frac{1}{\sqrt{q d_{\text{sketch}}}}$ by 0 and $+\frac{1}{\sqrt{q d_{\text{sketch}}}}$ by 1.
Let j be the non-negative integer whose binary representation is σ_c.
Increment j by 1.
Set f to the sign of $\overline{z}_c[t + 2]$.
return (j, f).

Algorithm 4 Recursable Dictionary Learning

Input: Positive integer h, observations $y^{(1)}, \ldots, y^{(S_h)}$ and thresholds τ_1 and τ_2.
Output: Matrices $\hat{R}_1, \ldots, \hat{R}_N \in \mathbb{R}^{d_{\text{sketch}} \times d_{\text{sketch}}}$ and vectors $\hat{x}^{(1)}, \ldots, \hat{x}^{(S_h)} \in \mathbb{R}^{d_{\text{sketch}} \cdot N}$.

for each sample index $k \in [S_h]$ **do**
  for each block index $\ell \in [d_{\text{sketch}}/b]$ **do**
    Let $w_{k,\ell}$ be the product of $\sqrt{q/b}$ and the ℓ_1-norm of the ℓth block of $y^{(k)}$.
    if $w_{k,\ell}$ is not zero **then**
      Let $z_{k,\ell}$ be the coordinate-wise normalization of the ℓth block of $y^{(k)}$ by $w_{k,\ell}$.
      Let $z_{k,\ell,s}$, $z_{k,\ell,c}$ and $z_{k,\ell,m}$ be the first, middle and last $b/3$ coordinates of $z_{k,\ell}$ respectively.
    end if
  end for
end for
Initialize n to zero, $\hat{R}_1, \ldots, \hat{R}_N$ to the all-zeros matrices and $\hat{x}^{(1)}, \ldots, \hat{x}^{(S_h)}$ to the all-zeros vectors.
for each $i \in [N]$ **do**
  Let $\hat{\sigma}_{i,m}$ be a vector in $\{\pm \frac{1}{\sqrt{q d_{\text{sketch}}}}\}^{b/3}$ that is initialized arbitrarily.
end for
for each sample index $k \in [S_h]$ **do**
  for each block index $\ell \in [d_{\text{sketch}}/b]$ **do**
    if $w_{k,\ell}$ is zero **then**
      continue
    end if
    if the ℓth block of $y^{(k)}$ has a coordinate whose absolute value is smaller than τ_1 or larger than $\frac{2}{\sqrt{q d_{\text{sketch}}}}$ **then**
      continue
    end if
    if $z_{k,\ell,s}$ is τ_2-far in ℓ_∞-distance from the hypercube $\{\pm \frac{1}{\sqrt{q d_{\text{sketch}}}}\}^{b/3}$ **then**
      continue
    end if
    Let $S_{k,\ell}$ be the set of all block indices $\ell' \in [d_{\text{sketch}}/b]$ for which $w_{k,\ell'}$ is non-zero, all the coordinates of the ℓ'th block of $y^{(k)}$ are between τ_1 and $\frac{2}{\sqrt{q d_{\text{sketch}}}}$ and $z_{k,\ell',s}$ is $2\tau_2$-close in symmetric ℓ_∞-distance to $z_{k,\ell,s}$.
    if the cardinality of $S_{k,\ell}$ is less than $(0.9)^3 q d_{\text{sketch}}$ **then**
      continue
    end if
    Let $\overline{z}_{k,\ell}$ be the coordinate-wise rounding of $z_{k,\ell}$ to the hypercube $\{\pm \frac{1}{\sqrt{q d_{\text{sketch}}}}\}^b$.
    Let $\overline{z}_{k,\ell,s}$, $\overline{z}_{k,\ell,c}$, $\overline{z}_{k,\ell,m}$ be the most common first, middle and last $\frac{b}{3}$ coordinates of $\{\overline{z}_{k,\ell'} : l' \in S_{k,\ell}\}$ respectively.
    if there is $i \in [n]$ such that $\hat{\sigma}_{i,m}$ is $0.01b$-close in symmetric Hamming distance to $\overline{z}_{k,\ell,m}$ **then**
      Set i^* to i.
    else
      Increment n by 1 and set i^* to the new value of n.
      Set $\hat{\sigma}_{i^*,m}$ to the matrix signature $\overline{z}_{k,\ell,m}$.
    end if
    Let $(j, f) = \text{Dec}(\overline{z}_{k,\ell,c})$ be the decoded column index and sign respectively.
    if the sign of the first coordinate of $\overline{z}_{k,\ell,m}$ is the opposite of f **then**
      Negate $w_{k,\ell}$ and each coordinate of $\overline{z}_{k,\ell}$.
    end if
    if $\hat{R}_{i^*}[:, j]$ is the all-zeros vector **then**
      for $\ell' \in S_{k,\ell}$ **do**
        Set $\hat{R}_{i^*}[(\ell'-1)b+1 : \ell' b+1, j]$ to $\overline{z}_{k,\ell}$.
      end for
    end if
    Set $x^{(k)}_{(i^*-1)d_{\text{sketch}}+j}$ to $w_{k,\ell}$.
  end for
end for
return $\hat{R}_1, \ldots, \hat{R}_N$ and $\hat{x}^{(1)}, \ldots, \hat{x}^{(S_h)}$.

Analysis. Theorem 4 follows from Theorem 12 below which proves the guarantees on the operation of Algorithm 4.

Theorem 12 (Guarantees of Algorithm 4)**.** *Let b, q and d_{sketch} satisfy $b \geq poly(\log N, \log d_{sketch}, 1/\epsilon_H)$, $q \geq poly(\log N, \log d_{sketch}, 1/\epsilon_H)/\sqrt{d_{sketch}}$, and $d_{sketch} \geq poly(1/\epsilon_H, \log N, \log S)$. For any unknown vectors $x^{(1)}, ..., x^{(S_h)} \in \mathbb{R}^{d_{sketch} \cdot N}$ with ℓ_2-norm at most $O(1)$, if we draw $R_1, ..., R_N \sim \mathcal{D}(b, q, d_{sketch})$ and receive S_h noisy samples $y^{(k)} := [R_1 R_2 \cdots R_N] x^{(k)} + z_1^{(k)} + z_\infty^{(k)}$ where each $z_1^{(k)} \in \mathbb{R}^{d_{sketch}}$ is noise with $\left|\left|z_1^{(k)}\right|\right|_1 \leq O(\sqrt{d_{sketch}})$ (independent of our random matrices) and each $z_\infty^{(k)} \in \mathbb{R}^{d_{sketch}}$ is noise with $\left|\left|z_\infty^{(k)}\right|\right|_\infty \leq \epsilon_h$ (also independent of our random matrices), then Algorithm 4, taking as inputs h, $y^{(1)}$, ..., $y^{(S_h)}$, and thresholds $\tau_1 = \Theta\left(\frac{\epsilon_{h+1}}{\sqrt{q \cdot d_{sketch}}}\right)$ and $\tau_2 = \Theta(\epsilon_{h+1}^2)$, runs in time $poly(S_h, d_{sketch})$ and outputs matrices $\hat{R}_1, \ldots, \hat{R}_N \in \mathbb{R}^{d_{sketch} \times d_{sketch}}$ and vectors $\hat{x}^{(1)}, \ldots, \hat{x}^{(S_h)} \in \mathbb{R}^{d_{sketch} \cdot N}$ that, with high probability, satisfy the following for some permutation π of $[N]$:*

- **Matrix Recovery:** *For every column $(i-1)d_{sketch}+j$ of the matrix $[R_1 R_2 \cdots R_N]$ (where $i \in [N]$ and $j \in [d_{sketch}]$) for which there exists a sample index $k^\star \in [S_h]$ such that $\left|x^{(k^\star)}_{(i-1)d_{sketch}+j}\right| \geq \epsilon_{h+1}$, the jth column of $\hat{R}_{\pi(i)}$ is $0.2d_{sketch}$-close in Hamming distance to the jth column of R_i. Moreover, all the other columns of $\hat{R}_{\pi(i)}$ are zero.*
- **Input Recovery:** *For each $i \in [N]$ and $j \in [d_{sketch}]$, it is guaranteed that $|\hat{x}^{(k)}_{(\pi(i)-1)d_{sketch}+j} - x^{(k)}_{(i-1)d_{sketch}+j}| \leq \epsilon_{h+1}$.*

Theorem 12 directly follows from Lemmas 9 and 10 below, applied in sequence. We point out that for simplicity of exposure, we chose large constants (which we did not attempt to optimize) in the statements of Lemmas 9 and 10.

Lemma 9. *For every absolute positive constant t and every given positive real number ϵ_{h+1}, for any $b \geq \Theta\left(\frac{\log N + \log d_{sketch} + \log S_h}{\epsilon_{h+1}^2}\right)$, for $\epsilon_h = \Theta(\epsilon_{h+1}^4)$, with probability at least $1 - N^{-t}$, there exists a permutation Π of $[N]$ such that the following holds during the operation of Algorithm 4:*

- *Whenever i^* is set in lines 33-38, then $\hat{\sigma}_{i^*,m}$ is $0.001b$-close in Hamming distance to the matrix signature $\sigma_{\pi(i^*),m}$.*
- *If moreover j is set in line 39, then the jth column of $\hat{R}_{i^*}$ is $\frac{\epsilon_{h+1} d_{sketch}}{4}$-close in Hamming distance to the jth column of $R_{\pi(i^*)}$.*
- *If furthermore line 48 is executed, then*

$$|x^{(k)}_{(\pi(i^*)-1)d_{sketch}+j} - x^{(k)}_{(i^*-1)d_{sketch}+j}| \leq \epsilon_{h+1}.$$

Lemma 10. *For every absolute positive constant t and every given positive real number ϵ_{h+1}, for any q satisfying $q \geq \Theta\left(\frac{\log N + \log(d_{sketch}) + \log S_h}{\epsilon_{h+1}^2 \cdot \sqrt{d_{sketch}}}\right)$ and for $\epsilon_h = \Theta(\epsilon_{h+1}^4)$, the following holds. For the permutation π of $[N]$ satisfying the properties in Lemma 9, with probability at least $1 - N^{-t}$, for all $k \in [S_h]$, $i \in [N]$ and $j \in [d_{sketch}]$:*

- *If $|x^{(k)}_{(i-1)d_{sketch}+j}| \geq \kappa \cdot \epsilon_{h+1}$, the jth column of $\hat{R}_{\pi(i)}$ is $0.2d_{sketch}$-close in Hamming distance to the jth column of R_i.*
- *It is the case that $|\hat{x}^{(k)}_{(\pi(i)-1)d_{sketch}+j} - x^{(k)}_{(i-1)d_{sketch}+j}| \leq \epsilon_{h+1}$.*

Proof Preliminaries. To prove Lemma 9, we need the next two known theorems and the following two lemmas. Theorem 13 below provides asymptotically tight bounds on the expected moments of a linear combination of independent uniform Bernouilli random variables.

Theorem 13 (Khintchine Inequality)**.** *Let $\{\epsilon_n : n \in [N]\}$ be i.i.d. random variables with $\Pr[\epsilon_n = +1] = \Pr[\epsilon_n = -1] = 1/2$ for each $n \in [N]$, i.e., a sequence with Rademacher distribution. Let $0 < p < \infty$ and let $x_1, x_2, \ldots, x_N \in \mathbb{R}$. Then, there exist positive absolute constants A_p and B_p (depending only on p) for which*

$$A_p \cdot \left(\sum_{n=1}^N x_n^2\right)^{1/2} \leq \left(\mathbb{E}|\sum_{n=1}^N \epsilon_n x_n|^p\right)^{1/p} \leq B_p \cdot \left(\sum_{n=1}^N x_n^2\right)^{1/2}.$$

The well-known Markov's inequality allows us to lower-bound the probability that a non-negative random variable is not much larger than its mean. The next theorem allows us to lower-bound the probability that a non-negative random variable is not much *smaller* than its mean, provided that the random variable has finite variance.

Theorem 14 (Paley-Zygmund Inequality)**.** *If $Z \geq 0$ is a random variable with finite variance and if $0 \leq \theta \leq 1$, then*

$$\Pr[Z > \theta \cdot \mathbb{E}[Z]] \geq (1-\theta)^2 \cdot \frac{\mathbb{E}[Z]^2}{\mathbb{E}[Z^2]}.$$

We will also need the following two lemmas.

Lemma 11 (Separation Lemma)**.** *Let v be a positive real number. Let X and Y be two random variables that are symmetric around 0 and such that the absolute value of each of them is larger than v with probability at least α. Then, there exist real numbers $a_1 < a_2 < a_3 < a_4$ satisfying $\min(a_2 - a_1, a_3 - a_2, a_4 - a_3) \geq v$ and such that the sum $X + Y$ lies in each of the intervals $(-\infty, a_1]$, $[a_2, a_3]$ and $[a_4, +\infty)$ with probability at least $\alpha/4$.*

We will use the following multiplicative version of the Chernoff bound.

Theorem 15 (Chernoff Bound – Multiplicative Version)**.** *Suppose $X_1, \ldots, X_n$ are independent random variables taking values in $\{0, 1\}$. Let $X := \sum_{i \in [n]} X_i$ and let $\mu = \mathbb{E}[X]$. Then,*

- *For all $0 \leq \delta \leq 1$, $\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2 \mu/2}$.*
- *For all $\delta \geq 0$, $\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2 \mu/(2+\delta)}$.*

Proof of Lemma 11. Define $m_{X,-}$ and $m_{X,+}$ to be the medians of the negative and positive parts (respectively) of the random variable X conditioned on $|X| > v$ (note that $m_{X,-} = -m_{X,+}$ although we will not be using this fact in the proof). Similarly, define $m_{Y,-}$ and $m_{Y,+}$ to be the negative and positive parts (respectively) of the random variable Y conditioned on $|Y| \geq v$. Setting

$$a_1 = m_{X,-} + m_{Y,-},$$
$$a_2 = m_{X,-} + v,$$
$$a_3 = m_{Y,+} - v,$$

and

$$a_4 = m_{X,+} + m_{Y,+},$$

the statement of Lemma 11 now follows. □

We are now ready to prove the correctness of Algorithm 4.

Proof of Lemma 10. Fix $k \in [S_h]$, $i \in [N]$ and $j \in [d_{\text{sketch}}]$, and assume that $|x^{(k)}_{(i-1)d_{\text{sketch}}+j}| \geq \kappa \cdot \epsilon_{h+1}$. For each $\ell \in [d_{\text{sketch}}/b]$, we consider the random vector

$$\text{err}_{k,i,\ell,j} := \sum_{\substack{i' \in [N], j' \in [d_{\text{sketch}}]: \\ i' \neq i \text{ or } j' \neq j}} x^{(k)}_{(i'-1)d_{\text{sketch}}+j'} R_{i'}[(\ell-1)b+1 : \ell b+1,\ j']$$
$$+ z^{(k)}_{\infty}[(\ell-1)b+1 : \ell b+1] + z^{(k)}_{1}[(\ell-1)b+1 : \ell b+1].$$

We think of the b-dimensional real vector $\text{err}_{k,i,\ell,j}$ as containing the coordinate-wise errors incurred by estimating

$$x^{(k)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+1 : \ell b+1,\ j]$$

from the ℓth block of $y^{(k)}$. Let $T_{k,i,j}$ be the set of all block indices $\ell \in [d_{\text{sketch}}/b]$ for which $\eta_{i,\ell,j} = 1$ and

$$\|z^{(k)}_1[(\ell-1)b+1 : \ell b+1]\|_\infty \leq O(\frac{b}{\sqrt{d_{\text{sketch}}}}).$$

By an averaging argument, the Chernoff Bound (Theorem 15) and the union bound, we have that for every fixed $k \in [S_h]$, $i \in [N]$ and $j \in [d_{\text{sketch}}]$, with probability at least $1 - e^{-0.005 \cdot \frac{q \cdot d_{\text{sketch}}}{b}}$, it is the case that $|T_{k,i,j}| > 0.9 \cdot \frac{q \cdot d_{\text{sketch}}}{b}$.

Henceforth, we condition on a setting of $\{\eta_{i,\ell,j} : \ell \in [d_{\text{sketch}}/b]\}$ for which $|T_{k,i,j}| > 0.9 \cdot \frac{q \cdot d_{\text{sketch}}}{b}$. We will upper-bound the probability, over the randomness of $\{\eta_{i',\ell,j'} : i' \in [N], j' \in [d_{\text{sketch}}],$ such that $i' \neq i$ or $j' \neq j\}$, that the ℓ_∞-norm of the random vector $\text{err}_{k,i,\ell,j}$ is large, and then we will show, by again applying the Chernoff Bound (Theorem 15), that for at least a 0.9-fraction of the ℓ's in $T_{k,i,j}$, the ℓ_∞-norm is small. We have that

$$\begin{aligned}\|\text{err}_{k,i,\ell,j}\|_\infty &\leq \sum_{\substack{i' \in [N], j' \in [d_{\text{sketch}}]: \\ i' \neq i \text{ or } j' \neq j}} |x^{(k)}_{(i'-1)d_{\text{sketch}}+j'}| \cdot R_{i'}[(\ell-1)b+1 : \ell b+1,\ j'] + \epsilon_h + O(\frac{b}{\sqrt{d_{\text{sketch}}}}) \\ &:= \epsilon_{k,i,\ell,j},\end{aligned}$$

where the first inequality above uses the assumption that $\|z^{(k)}_\infty\|_\infty \leq \epsilon_h$. By the desynchronization property (Section B.2), the fact that the ℓ_2-norm of $x^{(k)}$ is guaranteed to be at most $O(1)$ for all $k \in [S_h]$, and the assumptions that $q \gg \frac{\sqrt{b} \cdot \log N}{\sqrt{d_{\text{sketch}}}}$ and $b \gg \log N$, the expected value of $\epsilon_{k,i,\ell,j}$ satisfies

$$\mathbb{E}[\epsilon_{k,i,\ell,j}] \leq \epsilon_h + O(\frac{b}{\sqrt{d_{\text{sketch}}}}),$$

where the last inequality follows from the fact that for all $k \in [S_h]$, it is the case that $\|x^{(k)}\|_1 \leq \gamma$. By Markov's inequality, we get that with probability at least 0.9, it is the case that

$$\epsilon_{k,i,\ell,j} \leq 10 \cdot \epsilon_1 + O(\frac{b}{\sqrt{d_{\text{sketch}}}}). \tag{1}$$

Since the random variables in the set $\{\epsilon_{k,i,\ell,j} : \ell \in T_{k,i,j}\}$ are independent, the Chernoff Bound (Theorem 15) implies that with probability at least $1 - e^{-0.1^2 \cdot 0.9 \cdot |T_{k,i,j}|/2} \geq 1 - e^{-0.004 \cdot \frac{q d_{\text{sketch}}}{b}}$, for at least a $(0.9)^2$-fraction of the ℓ's in $T_{k,i,j}$, it is the case that $\epsilon_{k,i,\ell,j}$ satisfies Equation (1) and hence $\|\text{err}_{k,i,\ell,j}\|_\infty \leq 10 \cdot \epsilon_h + O(\frac{b}{\sqrt{d_{\text{sketch}}}})$. For each of these values of ℓ, the block (k, ℓ) will fail the tests in lines 18 and 21 of Algorithm 4 as long as

$$\frac{\kappa \cdot \epsilon_{h+1}}{\sqrt{q \cdot d_{\text{sketch}}}} - 10 \cdot \epsilon_h - O(\frac{b}{\sqrt{d_{\text{sketch}}}}) > \tau_1. \tag{2}$$

Moreover for each of these values of ℓ, the block (k, ℓ) will fail the test in line 24 of Algorithm 4 as long as

$$\frac{10 \cdot \epsilon_h + O(\frac{b}{\sqrt{d_{\text{sketch}}}})}{\frac{\kappa \cdot \epsilon_{h+1}}{\sqrt{q \cdot d_{\text{sketch}}}} - 10 \cdot \epsilon_h - O(\frac{b}{\sqrt{d_{\text{sketch}}}})} < \tau_2. \tag{3}$$

Furthermore for each of these values of ℓ, we will also have that $|S_{k,\ell}| > (0.9)^3 \cdot \frac{q \cdot d_{\text{sketch}}}{b}$ (as long as Equation (2) above holds) and hence the block (k, ℓ) will fail the test in line 28 of Algorithm 4.

Thus, the block (k, ℓ) fails the tests in lines 18, 21, 24 and 28 of Algorithm 4 with probability at least $1 - 2 \cdot e^{-0.004 \cdot \frac{q \cdot d_{\text{sketch}}}{b}}$ as long as Equations (2) and (3) above hold. In this case, a vector that is at a (symmetric relative) Hamming distance of at most 0.1 from $r_{m,j}$ will be added to the set of atoms (if no such vector was already present in the set). Moreover, the value of $x^{(k)}_{(i-1)d_{\text{sketch}}+j}$ will be recovered up to an absolute error of ϵ_{h+1} as long as

$$\frac{10 \cdot \epsilon_h + O(\frac{b}{\sqrt{d_{\text{sketch}}}})}{\frac{\kappa \cdot \epsilon_{h+1}}{\sqrt{q \cdot d_{\text{sketch}}}} - 10 \cdot \epsilon_h - O(\frac{b}{\sqrt{d_{\text{sketch}}}})} < \epsilon_{h+1}. \tag{4}$$

By a union bound, we get that with probability at least $1 - 2 \cdot S_h \cdot d_{\text{sketch}} \cdot N \cdot e^{-0.004 \cdot \frac{q d_{\text{sketch}}}{b}}$, the above recovery guarantees simultaneously hold for all $k \in [S_h]$ and all $i \in [N]$ and $j \in [d_{\text{sketch}}]$ for which $|x^{(k)}_{(i-1)d_{\text{sketch}}+j}| \geq \kappa \cdot \epsilon_{h+1}$. This probability is at least the desired $1 - N^{-t}$ in the statement of Lemma 10 as long as

$$2 \cdot S_h \cdot d_{\text{sketch}} \cdot N \cdot e^{-0.004 \cdot \frac{q \cdot d_{\text{sketch}}}{b}} < N^{-t}. \tag{5}$$

Finally, we note that Equations (2), (3), (4) and (5) above are all simultaneously satisfied (assuming that t is a given positive absolute constant) if we set

$$\epsilon_h = \Theta(\epsilon_{h+1}^4),$$
$$\tau_1 = \Theta\left(\frac{\epsilon_{h+1}}{\sqrt{q \cdot d_{\text{sketch}}}}\right),$$
$$\tau_2 = \Theta(\epsilon_{h+1}^2),$$

and

$$q \geq \Theta\left(\frac{\log N + \log(d_{\text{sketch}}) + \log S_h}{\epsilon_{h+1}^2 \cdot \sqrt{d_{\text{sketch}}}}\right).$$

Proof of Lemma 9. We start by considering the special case where the "ℓ_1 error" $z_1^{(k)}$ is zero (for all $k \in [S_h]$). We will later generalize the proof to the case where we only assume that $\left|\left|z_1^{(k)}\right|\right|_1 \leq O(\sqrt{d_{\text{sketch}}})$.

For each $k \in [S_h]$ and $\ell \in [d_{\text{sketch}}/b]$, we define $x^{(k,\ell)} \in \mathbb{R}^{N \cdot d_{\text{sketch}}}$ by setting $x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} = x^{(k)}_{(i-1)d_{\text{sketch}}+j} \cdot \mathbb{1}[\eta_{i,\ell,j} = 1]$ for each $i \in [N]$ and $j \in [d_{\text{sketch}}]$. For each coordinate $u \in [b]$, we consider the scalar random variable

$$|y^{(k)}[(\ell-1)b+u]| = |\sum_{i \in [N], j \in [d_{\text{sketch}}]} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u, j] + z_\infty^{(k)}[(\ell-1)b+u]|.$$

Let $\lambda \geq 1$ be a real parameter to be specified later. If $\|x^{(k,\ell)}\|_2 \leq \epsilon_{h+1}/\lambda$, then for each $u \in [b/3]$, Khintchine's Inequality (Theorem 13) with $p = 1$, along with the fact that $\|z_\infty^{(k)}\|_\infty \leq \epsilon_h$, imply that

$$\begin{aligned}\mathbb{E}[|y^{(k)}[(\ell-1)b+u]|] &\leq \frac{B_2}{\sqrt{q \cdot d_{\text{sketch}}}} \cdot (\sum_{i \in [N], j \in [d_{\text{sketch}}]} (x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2)^{1/2} + \epsilon_h \\ &\leq \frac{B_2 \cdot \epsilon_{h+1}}{\lambda \cdot \sqrt{q \cdot d_{\text{sketch}}}} + \epsilon_h,\end{aligned}$$

where B_2 is a positive absolute constant. By Markov's inequality, we get that

$$\Pr[|y^{(k)}[(\ell-1)b+u]| > \frac{10 \cdot B_2 \cdot \epsilon_{h+1}}{\lambda \cdot \sqrt{q \cdot d_{\text{sketch}}}} + 10 \cdot \epsilon_h] \leq 0.1.$$

Since the coordinates $u \in [b/3]$ are independent, the probability that all of them satisfy $|y^{(k)}[(\ell-1)b+u]| > 10 \cdot B_2 \cdot \epsilon_{h+1}/(\lambda\sqrt{qd_{\text{sketch}}}) + 10 \cdot \epsilon_h$ is at most $(0.1)^{b/3}$. Thus, the probability that block (k,ℓ) will pass the test in line 21 of Algorithm 4 is at least $1 - (0.1)^{b/3}$ as long as

$$\frac{10 \cdot B_2 \cdot \epsilon_{h+1}}{\lambda \cdot \sqrt{q \cdot d_{\text{sketch}}}} + 10 \cdot \epsilon_h < \tau_1. \tag{6}$$

By a union bound, we get that with probability at least

$$1 - S_h \cdot \frac{d_{\text{sketch}}}{b} \cdot (0.1)^{b/3}, \tag{7}$$

for all $k \in [S_h]$ and all $\ell \in [d_{\text{sketch}}/b]$ for which $\|x^{(k,\ell)}\|_2 \leq \epsilon_{h+1}/\lambda$, block (k,ℓ) will pass the test in line 21 of Algorithm 4.

Henceforth, we assume that $\|x^{(k,\ell)}\|_2 > \epsilon_{h+1}/\lambda$. Assume moreover that for all $i \in [N]$ and $j \in [d_{\text{sketch}}]$ for which $\eta_{i,\ell,j} = 1$, it is the case that

$$(x^{(k)}_{(i-1)d_{\text{sketch}}+j})^2 \leq g \cdot \sum_{\substack{i' \in [N], j' \in [d_{\text{sketch}}]: \\ i' \neq i \text{ or } j' \neq j}} (x^{(k)}_{(i'-1)d_{\text{sketch}}+j'})^2 \cdot \mathbb{1}[\eta_{i',\ell,j'} = 1], \tag{8}$$

where $g > 1$ is an absolute constant to be specified later. Equation (8) is equivalent to saying that for all $i \in [N]$ and $j \in [d_{\text{sketch}}]$,

$$(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2 \leq g \cdot \sum_{\substack{i' \in [N], j' \in [d_{\text{sketch}}]: \\ i' \neq i \text{ or } j' \neq j}} (x^{(k,\ell)}_{(i'-1)d_{\text{sketch}}+j'})^2. \tag{9}$$

For each coordinate $u \in [b/3]$, we can then break the sum into two parts

$$\sum_{\substack{i\in[N],\\ j\in[d_{\text{sketch}}]}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j] = \sum_{(i,j)\in\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j] + \sum_{(i,j)\in[N]\times[d_{\text{sketch}}]\setminus\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j], \tag{10}$$

where $\mathcal{W} \subseteq [N]\times[d_{\text{sketch}}]$ and such that each of these two parts has a weight vector with ℓ_2-norm at least $\epsilon_{h+1}/(\lambda\cdot\sqrt{2\cdot(g+1)})$, i.e.,

$$\min\left(\sqrt{\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2}, \sqrt{\sum_{(i,j)\in[N]\times[d_{\text{sketch}}]\setminus\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2}\right) \geq \frac{\epsilon_{h+1}}{\lambda\cdot\sqrt{2\cdot(g+1)}}.$$

Such a partition can be obtained by sorting the coefficients $\{(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2 : i\in[N], j\in[d_{\text{sketch}}]\}$ in non-increasing order and then including every other coefficient in the set $\mathcal{W}$ (which will thus contain the largest, 3rd largest, 5th largest etc. coefficients). Applying Khintchine's Inequality (Theorem 13) with $p=1$ and $p=2$, we get that

$$\frac{A_1}{\sqrt{q\cdot d_{\text{sketch}}}}\cdot\sqrt{\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2} \leq \mathbb{E}[|\sum_{(i,j)\in\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j]|] \leq \frac{B_1}{\sqrt{q\cdot d_{\text{sketch}}}}\cdot\sqrt{\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2} \tag{11}$$

and

$$\frac{A_2}{\sqrt{q\cdot d_{\text{sketch}}}}\cdot\sqrt{\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2} \leq \sqrt{\mathbb{E}[|\sum_{(i,j)\in\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j]|^2]} \leq \frac{B_2}{\sqrt{q\cdot d_{\text{sketch}}}}\cdot\sqrt{\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2}, \tag{12}$$

where A_1, B_1, A_2 and B_2 are some positive absolute constants. Applying the Paley–Zygmund Inequality (Theorem 14) to the random variable $Z = |\sum_{(i,j)\in\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j]|$ with $\theta = 0.5$, we get that $Z > 0.5\cdot\mathbb{E}[Z]$ with probability at least $0.25\cdot\frac{\mathbb{E}[Z]^2}{\mathbb{E}[Z^2]}$. Note that for our setting of Z, Equation (11) implies that

$$\frac{\epsilon_{h+1}\cdot A_1}{\lambda\cdot\sqrt{2\cdot(g+1)\cdot q\cdot d_{\text{sketch}}}} \leq \frac{A_1}{\sqrt{q\cdot d_{\text{sketch}}}}\cdot(\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2)^{1/2} \leq \mathbb{E}[Z].$$

Moreover, Equation (12) implies that

$$\mathbb{E}[Z^2] \leq \frac{B_2^2}{q\cdot d_{\text{sketch}}}\cdot\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2.$$

Putting these inequalities together, we get that

$$|\sum_{(i,j)\in\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u,j]| > \frac{\epsilon_{h+1}\cdot A_1}{2\cdot\lambda\cdot\sqrt{2\cdot(g+1)\cdot q\cdot d_{\text{sketch}}}}$$

with probability at least

$$0.25\cdot\frac{A_1^2\cdot\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2}{B_2^2\cdot\sum_{(i,j)\in\mathcal{W}}(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2} = 0.25\cdot\frac{A_1^2}{B_2^2}.$$

Thus, $v_{\mathcal{W}} := \sum_{(i,j)\in\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u, j]$ is a random variable that is symmetric around 0 and whose absolute value is larger than $v := \epsilon_{h+1} \cdot A_1/(2 \cdot \sqrt{2 \cdot (g+1) \cdot q \cdot d_{\text{sketch}}} \cdot \lambda)$ with probability at least $\gamma := 0.25 \cdot A_1^2/B_2^2$. Applying the same reasoning to the second summand in Equation (10), we deduce that the random variable $v_{[N]\times[d_{\text{sketch}}]\setminus\mathcal{W}} := \sum_{(i,j)\in[N]\times[d_{\text{sketch}}]\setminus\mathcal{W}} x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j} R_i[(\ell-1)b+u, j]$ is also a random variable that is symmetric around 0 and whose absolute value is larger than v with probability at least γ. Since the random variables $v_{\mathcal{W}}$ and $v_{[N]\times[d_{\text{sketch}}]\setminus\mathcal{W}}$ are independent, Lemma 11 then implies that their sum belongs to each of the 3 intervals $(-\infty, a_1]$, $[a_2, a_3]$ and $[a_4, +\infty)$ with probability at least $\gamma/4$ where $\min(a_2 - a_1, a_3 - a_2, a_4 - a_3) \geq v$. Thus, any normalization of this random variable by a positive scalar that has value smaller than $2\sqrt{b/q}$ (in particular, normalizing by $w_{k,\ell}$ which has value smaller than $2\sqrt{b/q}$ if block (k,ℓ) fails the test in line 21 of Algorithm 4) results in a value which is at an absolute distance of more than

$$\frac{v}{2} \cdot \frac{\sqrt{q}}{2\sqrt{b}} = \frac{\epsilon_{h+1} \cdot A_1}{8\sqrt{2 \cdot (g+1) \cdot b \cdot d_{\text{sketch}}}}$$

from the set $\{\pm\frac{1}{\sqrt{q d_{\text{sketch}}}}\}$ with probability at least $\gamma/4 = A_1^2/(16 \cdot B_2^2)$. Since the $b/3$ coordinates are independent, we get that block (k,ℓ) fails the test in line 24 of Algorithm 4 with probability at most $(A_1^2/(16 \cdot B_2^2))^{b/3}$ as long as

$$\frac{\epsilon_{h+1} \cdot A_1}{8\sqrt{2 \cdot (g+1) \cdot b \cdot d_{\text{sketch}}}} - \epsilon_h > \tau_2. \tag{13}$$

By a union bound, we get that with probability at least

$$1 - N \cdot \frac{d_{\text{sketch}}}{b} \cdot (A_1^2/(16 \cdot B_2^2))^{b/3}, \tag{14}$$

for all $k \in [S_h]$ and all $\ell \in [d_{\text{sketch}}/b]$ for which $\|x^{(k,\ell)}\|_2 > \epsilon_2/\lambda$ and for which Equation (9) is satisfied (for all $i \in [N]$ and all $j \in [d_{\text{sketch}}]$), block (k,ℓ) will pass the test in line 24 of Algorithm 4.

Next, we still assume that $\|x^{(k,\ell)}\|_2 > \epsilon_{h+1}/\lambda$ but we assume that Equation (9) is violated for some $i \in [N]$ and $j \in [d_{\text{sketch}}]$, i.e.,

$$(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2 > g \cdot \sum_{\substack{i'\in[N], j'\in[d_{\text{sketch}}]: \\ i'\neq i \text{ or } j'\neq j}} (x^{(k,\ell)}_{(i'-1)d_{\text{sketch}}+j'})^2. \tag{15}$$

This implies that

$$(x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j})^2 > \frac{g}{g+1} \cdot \|x^{(k,\ell)}\|_2^2 \tag{16}$$

and

$$\sum_{\substack{i'\in[N], j'\in[d_{\text{sketch}}]: \\ i'\neq i \text{ or } j'\neq j}} (x^{(k,\ell)}_{(i'-1)d_{\text{sketch}}+j'})^2 < \frac{1}{g+1} \cdot \|x^{(k,\ell)}\|_2^2. \tag{17}$$

We now consider the rounding $\overline{y}^{(k)}$ of $y^{(k)}$ to the hypercube $\{\pm\frac{1}{\sqrt{q d_{\text{sketch}}}}\}^{d_{\text{sketch}}}$ obtained by setting each coordinate of $y^{(k)}$ to $+\frac{1}{\sqrt{q d_{\text{sketch}}}}$ if it is non-negative and to $-\frac{1}{\sqrt{q d_{\text{sketch}}}}$ otherwise. We argue that $\overline{y}^{(k)}[(\ell-1)b+1 : \ell b+1]$ is at a Hamming distance of at most $0.05b$ from the vector $R_i[(\ell-1)b+1 : \ell b+1, j]$ (note that Equation (15) and the assumption that $\|x^{(k,\ell)}\|_2 > \alpha/\lambda$ imply that $\eta_{i,\ell,j} = 1$). To prove this, we consider, for every coordinate $u \in [b]$, the random variable

$$\Delta_{k,i,j,\ell,u} = |\sum_{\substack{i'\in[N], j'\in[d_{\text{sketch}}]: \\ i'\neq i \text{ or } j'\neq j}} x^{(k,\ell)}_{(i'-1)d_{\text{sketch}}+j'} R_i[(\ell-1)b+u, j] + z^{(k)}_\infty[(\ell-1)b+u]|.$$

Note that the u-th coordinate $\overline{y}^{(k)}[(\ell-1)b+u]$ of $\overline{y}^{(k)}$ is equal to $R_i[(\ell-1)b+u, j]$ if

$$\frac{|x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j}|}{\sqrt{q \cdot d_{\text{sketch}}}} > \Delta_{k,i,j,\ell,u}. \tag{18}$$

Thus, to show that the Hamming distance between $\overline{y}^{(k)}[(\ell-1)b+1 : \ell b+1]$ and $R_i[(\ell-1)b+1 : \ell b+1, j]|]$ is at most $\epsilon_{h+1}b/4$, it suffices to show that for at least a $(1-\epsilon_2/4)$-fraction of the coordinates $u \in [b]$ it is the case that

$|x^{(k,\ell)}_{(i-1)d_{\text{sketch}}+j}| > \Delta_{k,i,j,\ell,u} \cdot \sqrt{q \cdot d_{\text{sketch}}}$. Since $\|z^{(k)}_{\infty}\|_{\infty} < \epsilon_h$, we have that

$$\begin{aligned}
\mathbb{E}[\Delta_{k,i,j,\ell,u}] &\leq |\sum_{\substack{i' \in [N], j' \in [d_{\text{sketch}}]: \\ i' \neq i \text{ or } j' \neq j}} x^{(k,\ell)}_{(i'-1)d_{\text{sketch}}+j'} R_i[(\ell-1)b+u, j]| + \epsilon_h \\
&\leq \frac{B_1}{\sqrt{q \cdot d_{\text{sketch}}}} \cdot \sqrt{\sum_{\substack{i' \in [N], j' \in [d_{\text{sketch}}]: \\ i' \neq i \text{ or } j' \neq j}} (x^{(k,\ell)}_{(i'-1)d_{\text{sketch}}+j'})^2} + \epsilon_h \\
&\leq \frac{B_1}{\sqrt{q \cdot d_{\text{sketch}}}} \cdot \frac{\|x^{(k,\ell)}\|_2}{\sqrt{g+1}} + \epsilon_h, \qquad (19)
\end{aligned}$$

where the penultimate inequality follows from Khintchine's Inequality (Theorem 13) with $p = 1$ and the last inequality follows from Equation (17). By Markov's inequality, we get that for each fixed $u \in [b]$, with probability at least $1 - \epsilon_{h+1}/8$,

$$\Delta_{k,i,j,\ell,u} \leq \frac{80 \cdot B_1 \cdot \|x^{(k,\ell)}\|_2}{\sqrt{q \cdot d_{\text{sketch}}} \cdot \epsilon_{h+1} \cdot \sqrt{g+1}} + \frac{80 \cdot \epsilon_h}{\epsilon_{h+1}},$$

and Equation (16) then implies that in this case Equation (18) is satisfied as long as

$$\sqrt{\frac{g}{g+1}} \cdot \|x^{(k,\ell)}\|_2 > \frac{80 \cdot B_1 \cdot \|x^{(k,\ell)}\|_2}{\sqrt{q \cdot d_{\text{sketch}}} \cdot \epsilon_{h+1} \cdot \sqrt{g+1}} + \frac{80 \cdot \epsilon_h}{\epsilon_{h+1}}. \qquad (20)$$

Since the first $b/3$ coordinates are independent, the Chernoff Bound (Theorem 15) hence implies that the Hamming distance between the first $b/3$ coordinates of $\overline{y}^{(k)}[(\ell-1)b+1 : \ell b+1]$ and the first $b/3$ coordinates of $R_i[(\ell-1)b+1 : \ell b+1, j]|]$ is at most $\epsilon_{h+1}b/4$ with probability at least

$$1 - e^{-\epsilon^2_{h+1} \cdot (1 - \frac{\epsilon_{h+1}}{8}) \cdot \frac{b}{128}}. \qquad (21)$$

We now briefly explain why in this case, $\overline{z}_{k,\ell,c}$ and $\overline{z}_{k,\ell,m}$ (which are set in line 32 of Algorithm 4) will be exactly equal to the middle and last $b/3$ coordinates of $R_i[(\ell-1)b+1 : \ell b+1, j]|]$ respectively. We describe the argument for the middle coordinates (the same holds for the last coordinates). By Equation (19), we have that the expected sum of the $\Delta_{k,i,j,\ell,u}$ values for all middle coordinates u is at most

$$\frac{b}{3} \cdot (\frac{B_1}{\sqrt{q \cdot d_{\text{sketch}}}} \cdot \frac{\|x^{(k,\ell)}\|_2}{\sqrt{g+1}} + \epsilon_h).$$

By Markov's inequality and the Chernoff bound, we thus get that with probability at least $1 - 1/\text{poly}(N)$, it is the case that $\overline{z}_{k,\ell,c}$ is exactly equal to the middle $b/3$ coordinates of $R_i[(\ell-1)b+1 : \ell b+1, j]|]$, as long as:

$$100 \cdot \frac{b}{3} \cdot (\frac{B_1}{\sqrt{q \cdot d_{\text{sketch}}}} \cdot \frac{\|x^{(k,\ell)}\|_2}{\sqrt{g+1}} + \epsilon_h) < \sqrt{\frac{g}{g+1}} \cdot \|x^{(k,\ell)}\|_2, \qquad (22)$$

We also note that as long as

$$(1 - \frac{\epsilon_{h+1}}{4}) \cdot (80 \cdot \frac{B_1}{\sqrt{q \cdot d_{\text{sketch}}}} \cdot \frac{\|x^{(k,\ell)}\|_2}{\sqrt{g+1}} + 80 \cdot \epsilon_h) + \frac{\epsilon_{h+1}}{4} \cdot \frac{2}{\sqrt{q \cdot d_{\text{sketch}}}} < \epsilon_{h+1}, \qquad (23)$$

an identical application of the Chernoff bound implies that with probability at least the quantity in Equation (21), if line 48 is executed when Algorithm 4 examines block (k, ℓ), then $|x^{(k)}_{(i^*-1)d_{\text{sketch}}+j} - x^{(k)}_{(i-1)d_{\text{sketch}}+j}| \leq \epsilon_{h+1}$ (we note that we are here using the fact that the block (k, ℓ) has failed the test in line 21 of Algorithm 4, hence the multiplicative factor of $2/\sqrt{q \cdot d_{\text{sketch}}}$ on the left-hand side of Equation (23)). By a union bound, we get that these properties simultaneously hold for all blocks $(k, \ell) \in [S_h] \times [d_{\text{sketch}}/b]$ for which $\|x^{(k,\ell)}\|_2 > \epsilon_{h+1}/\lambda$ and Equation (15) holds (for some $i \in [N]$ and $j \in [d_{\text{sketch}}]$), with probability at least

$$1 - 2 \cdot S_h \cdot \frac{d_{\text{sketch}}}{b} \cdot e^{-\epsilon^2_{h+1} \cdot (1 - \frac{\epsilon_{h+1}}{8}) \cdot \frac{b}{128}}. \qquad (24)$$

By another union bound and using Equations (7), (14) and (24), we get with probability at least

$$1 - \frac{S_h \cdot d_{\text{sketch}}}{b/} \cdot (0.1)^{b/3} - \frac{N \cdot d_{\text{sketch}}}{b} \cdot (\frac{A_1^2}{16 \cdot B_2^2})^{b/3} - \frac{2 \cdot S_h \cdot d_{\text{sketch}}}{b} \cdot e^{-\epsilon_{h+1}^2 \cdot (1 - \frac{\epsilon_{h+1}}{8}) \cdot \frac{b}{128}}, \tag{25}$$

that for all $k \in [S_h]$ and $\ell \times [d_{\text{sketch}}/b]$, if, when examining block (k, ℓ), the tests in lines 18, 21 and 24 of Algorithm 4 all fail, then we are sure that $\overline{z}_{k,\ell}$ is ϵ_{h+1}-close in Hamming distance to $R_i[(\ell-1)b+1 : \ell b+1, j]$. Moreover, we have that if i^* is set in lines 33-38, then $\hat{\sigma}_{i^*,m}$ is identical to to the matrix signature $\sigma_{i,m}$. Similarly, we can recover the correct column index $j \in [d_{\text{sketch}}]$. We are now ready to define our permutation π of $[N]$. For each $i \in [N]$, we let $\pi(i)$ be the first index i^* set in line 36 of Algorithm 4 for which $\hat{\sigma}_{i^*,m}$ is $10\epsilon_{h+1}$-close in Hamming distance to $\hat{\sigma}_{i,m}$. If no such i^* exists, we define $\pi(i)$ in an arbitrary way that doesn't result in a collision.

We now describe how to handle the general case where $z_1^{(k)}$ is not necessarily zero but where we are guaranteed that $\left|\left|z_1^{(k)}\right|\right|_1 \leq O(\sqrt{d_{\text{sketch}}})$. This can be done by observing that the number of blocks $\ell \in [d_{\text{sketch}}/b]$ for which

$$\|z_1^{(k)}[(\ell-1)b+1 : \ell b+1]\|_\infty \leq O(\frac{b}{\sqrt{d_{\text{sketch}}}})$$

is at most $0.001 \cdot q d_{\text{sketch}}/b$. Thus, the same reasoning above combined with the test in line 28 of Algorithm 4 implies that even in the presence of ℓ_1 noise, the probability bound derived above on spurious blocks (k, ℓ) failing all tests still holds.

To sum up, the probability in Equation (25) is at least the desired $1 - N^{-t}$ probability lower bound in the statement of Lemma 9 as long as

$$\frac{S_h \cdot d_{\text{sketch}}}{b} \cdot (0.1)^b - \frac{N \cdot d_{\text{sketch}}}{b} \cdot (\frac{A_1^2}{16 \cdot B_2^2})^{b/3} - \frac{2 \cdot S_h \cdot d_{\text{sketch}}}{b} \cdot e^{-\epsilon_{h+1}^2 \cdot (1 - \frac{\epsilon_{h+1}}{8}) \cdot \frac{b}{128}} < N^{-t}. \tag{26}$$

Lemma 9 now follows by noting that Equations (6), (13), (20), (22), (23) and (26) can be simultaneously satisfied (assuming that $S_h \leq \text{poly}(N)$ and t is a given positive absolute constant) if we set

$$g = \Theta\left(\frac{1}{\epsilon_{h+1}^2 \cdot \sqrt{q \cdot d_{\text{sketch}}}}\right),$$

$$\lambda = \Theta(1),$$

$$\epsilon_h = \Theta(\epsilon_{h+1}^4),$$

$$\kappa = \Theta(1),$$

and

$$b \geq \Theta\left(\frac{\log N + \log d_{\text{sketch}} + \log S_h}{\epsilon_{h+1}^2}\right).$$