A SHORT SURVEY ON ALMOST ORTHOGONAL VECTORS IN A FEW SPECIFIC LARGE DIMENSIONS

ℝAMI LUISTO

Abstract. The concept of *almost orthogonal vectors*, i.e. vectors whose cosine similarities is close to 0, relates to topics both in pure mathematics and in coding theory under the guises of spherical packing and spherical codes. In recent years the rise of advanced language models in AI has created new interest in this concept as the models seem to store certain concepts as almost orthogonal directions in high-dimensional spaces. In this survey we represent some ideas regarding almost orthogonal vectors through three approaches: (1) the mathematical theory of almost orthogonality, (2) some observations from the embedding spaces of language models, and (3) generation of large sets of almost orthogonal vectors by simulations.

Contents

Date: September 24, 2025.

1. Introduction

In a given Euclidean space $\mathbb{R}^n$ there can exist at most n directions that are pairwise orthogonal [Axl15]. If we relax the condition of orthogonality by requiring the inner product of unit vectors to be merely *close* to 0 the situation changes drastically, especially in higher dimensions. This phase transition in high and low dimensions relates to a concept sometimes known as *the curse (or blessing) of high dimensionality*. We'll discuss this further in the coming sections, see especially Section 3.3

We say that two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ are *ε-almost orthogonal* for $\varepsilon > 0$ if their cosine similarity lies between $-\varepsilon$ and ε. The main motivating question for this paper is as follows:

> How many pairwise ε-almost orthogonal vectors can fit in $\mathbb{R}^n$ for various values of ε and n?

As we'll discuss in Section 4.3 this question is closely tied to various open problems related to both *spherical codes* and *sphere packing*. In particular, exact solutions to the best configurations of vectors are beyond the scope of this survey, and instead we will focus on various estimates.

Since e.g. spherical codes have many practical applications, there already exists literature on approximate solutions. However, these do tend to focus more on low-dimensional examples – see e.g. [Coh23] where the examples only go up to dimension 32. This is natural if you aim to use this with e.g. binary blocks that tend to have word sizes of 8, 16, or 32 bits. By contrast, the existing results for almost orthogonal vectors in higher dimensions tend to focus on the *asymptotic* behaviour of the results, i.e. what happens when the ambient dimension grows very large. In particular, the typical results like the Johnson-Lindenstrauss Lemma tend not to yield good estimates in dimensions below several thousands – see Section 4.2.

The contextual motivation for us, on the other hand, arises from the so called *embedding vectors* of various AI-models. For a contemporary example, various language models based on the transformer architecture (including GPT, LLaMa, Gemini and Mistral) encode the meaning of text through embedding vectors that live in high-dimensional Euclidean spaces like $\mathbb{R}^{768}$ or $\mathbb{R}^{1024}$ (see e.g.[Eth19, Ope23, RKR21]). These dimensions lie between the small dimensions of coding theory and high dimensions of asymptotical results. This gap is of primary interest in this survey.

Based on current understanding, see e.g. [EHO$^+$22, HCH$^+$23, BTB$^+$23, TCM$^+$24], modern language models models seem to store different semantic concepts as different directions in these embedding spaces. Crucially for us, these different concepts seem to be encoded as *almost* orthogonal vectors, and it turns out that the geometry of e.g. $\mathbb{R}^{768}$ can encode much more such directions than just 768, though lossily. A driving motivation in this survey is to better understand how many almost orthogonal vectors can be fitted in various high-dimensional Euclidean spaces where the particular dimensions we study are the embedding space dimensions of particular contemporary language models – for the sake of exposition we will often focus on dimensions 512, 768, 1024 and 2048 when concrete examples are needed.

As per the parameter ε in ε-almost orthogonality we will in most examples focus on either $\varepsilon = 0.1$ or $\varepsilon = 0.01$ when we need to fix the parameter ourselves. This is again purely for the sake of exposition, as the values themselves do not carry any special meaning. Note that $\varepsilon = 0.1$ corresponds to angles between 84 and 96 degrees, while $\varepsilon = 0.01$ corresponds to angles of roughly 90 ± 0.5 degrees. Also note that if unit vectors $\mathbf{v}$ and $\mathbf{w}$ have an inner product of 0.1, then $\mathbf{v} = 0.1\mathbf{w} + 0.9\mathbf{u}$ where $\mathbf{u}$ is orthogonal to $\mathbf{w}$, meaning that "10% of $\mathbf{v}$ is explained by $\mathbf{w}$".

1.1. **Structure of this survey.** We'll start by "setting the scene" and describe in a bit more detail on why we are interested in almost orthogonality in high-dimensional spaces when studying modern (Large) Language models. We will then go through some mathematical preliminaries, including some more heuristical comments on how the geometry of low and high dimensions differs. We'll then move on to studying some (trivial) mathematical bounds we can get to the amount of almost orthogonal vectors in Euclidean spaces. We then look at some other classical results, in particular the Johnson-Lindenstrauss Lemma. The bounds we see here turn out to be somewhat meagre compared to practical situations. Thus we turn into simulations and compare various methods of generating almost orthogonal vectors. In the final section we'll draw conclusions on all of this.

We wish to emphasize that this paper is structured as a *survey*; it claims no new mathematical theorems, rather it tries to provide a convenient summary of useful known information. On the other hand it does not claim to be a *comprehensive* survey as many areas are only briefly mentioned, and we lack the expertise to comment authoratively to many topics discussed here.

2. Context – Latent spaces of (Large) Language Models

We will not repeat here a detailed introduction to the architecture of transformer models, rather we given a quick overview of the terminology. For a proper introduction, see e.g. [TVWW22].

Most classical transformer-based language models break incoming text into *tokens*. These tokens tend to be common words or word fragments. The collection of tokens a model has is called the *vocabulary*. The vocabulary size of e.g. the different BERT[1] variants are in the ballpark of 30–50 thousand tokens. For a more thorough introduction to tokens and their usage we refer to [DCLT18] and [Mik13].

For each token in the vocabulary the model has a learned embedding vector of a particular dimension d_{model}. These vectors are called the *input embeddings* of the model. Thus a given input sequence of natural text is broken into a sequence of tokens, and each of these tokens is then converted into the corresponding embedding vector. This sequence of embedding vectors is then passed through several *transformer blocks* by the model. These tranformer blocks update each of the embedding vectors of the input sequence based on other vectors in the sequence. This evolution of the sequence of

[1]BERT is what I would call the archetypical small language model, see e.g. [RKR21] for a comprehensive text on the topic.

embedding vectors is sometimes called *the residual stream*. For the standard transformer architecture, the d_{model} stays the same for the embeddings as they progress through the transformer blocks.[2] We've listed some of the values[3] of d_{model} in Tables 1, 2 and 3 for a few common SLM and LLM architectures.

Model	d_{model}
Encoder-only (BERT-style)	
BERT (base) [DCLT18]	768
BERT (large) [DCLT18]	1024
RoBERTa (base) [LOG+19]	768
RoBERTa (large) [LOG+19]	1024
DistilBERT [SDCW19]	768
ALBERT (base) [LCG+19]	128
XLM-RoBERTa [CKG+19]	768
XLNet (base) [YDY+19]	768
XLNet (large) [YDY+19]	1024
ERNIE [SWL+19]	768
ModernBERT (base) [WCC+24]	768
ModernBERT (large) [WCC+24]	1024

TABLE 1. Token-embedding (input) vector sizes for a wide range of language model families of the *encoder* architecture.

For us the input embeddings are of particular interest because they are context free – the embeddings do not depend on any way on the other tokens in the sequence nor on the relative or absolute position of a particular token. Thus we can study these vectors as "independent embedding vectors". For latter parts in the residual stream any embedding vector is affected by the other tokens and their positions in the input – i.e. the context. Thus the study of the embedding vectors and their geometry in the latter layers of transformers usually requires sampling of text examples and their careful analysis. The selection of texts to use naturally has a considerable effect, so for the examples in this section we simply restrict ourselves to the input embeddings of various models.

To get an idea on what kind of cosine similarities we see in the real world, we've taken a few common transformer-based models, extracted their input embeddings and calculated the cosine similarities between each embedding vector pair (excluding the trivial cosine similarities of a vector with itself). In Figure 1 we plot the distribution of these cosine similarities for each model.

A noteworthy property noticed already in [MBV17] and [Eth19] is that the cosine similarities are not at all equally distributed around the origin. Instead for almost all models the main mass of cosine similarities is strictly

[2]Though e.g. the ALBERT model uses factorised embedding; token embeddings are 128-dimensional but are expanded internally to a 768-dimensional vector at various points in the processing.

[3]We list the exact dimensions when known, but for some closed source models list the best known guesses based on public information available.

Model	d_{model}
Decoder-only (GPT, LLaMA, etc.)	
GPT-1 [RNS+18]	768
GPT-2 (small) [RWC+19]	768
GPT-2 (medium) [RWC+19]	1024
GPT-2 (large) [RWC+19]	1280
GPT-2 (XL) [RWC+19]	1600
GPT-3 (175B) [BMR+20]	12288
GPT-3.5	$\sim$12288†
GPT-4	Not disclosed
Codex (12B)	$\sim$5120†
LLaMA-1 (7B) [TLI+23]	4096
LLaMA-1 (13B) [TLI+23]	5120
LLaMA-1 (33B) [TLI+23]	6656
LLaMA-1 (65B) [TLI+23]	8192
LLaMA-2 (7B) [TMS+23]	4096
LLaMA-2 (13B) [TMS+23]	5120
LLaMA-2 (70B) [TMS+23]	8192
Mistral (7B) [Ope23]	4096
Mixtral (8$\times$7B) [JSR+24]	4096†
Bloom (176B) [LSFA+23]	14336
Falcon (40B) [AAA+23]	$\sim$8192†
Grok-1 [xT24]	Not disclosed

TABLE 2. Token-embedding (input) vector sizes for a wide range of language-model families of the *decoder* architecture. †Estimated values; "Not disclosed" indicates the vendor has not released architecture details.

Model	d_{model}
Encoder–decoder (T5, PaLM, ...)	
T5 (small) [RSR+19]	512
T5 (base) [RSR+19]	768
T5 (large) [RSR+19]	1024
T5 (3B) [RSR+19]	1024
T5 (11B) [RSR+19]	1024
PaLM (540B) [CND+23]	18432
Gopher (280B) [RBC+21]	16384
Chinchilla (70B) [HBM+22]	8192
AlphaCode (41B) [LCC+22]	6144
Claude 1/2 [Ant23]	Not disclosed
Gemini Ultra [PH23]	Not disclosed

TABLE 3. Token-embedding (input) vector sizes for a wide range of language-model families for the *encoder-decoder* architecture. "Not disclosed" indicates the vendor has not released architecture details.

positive, sometimes by a wide margin. With the interpretation that different

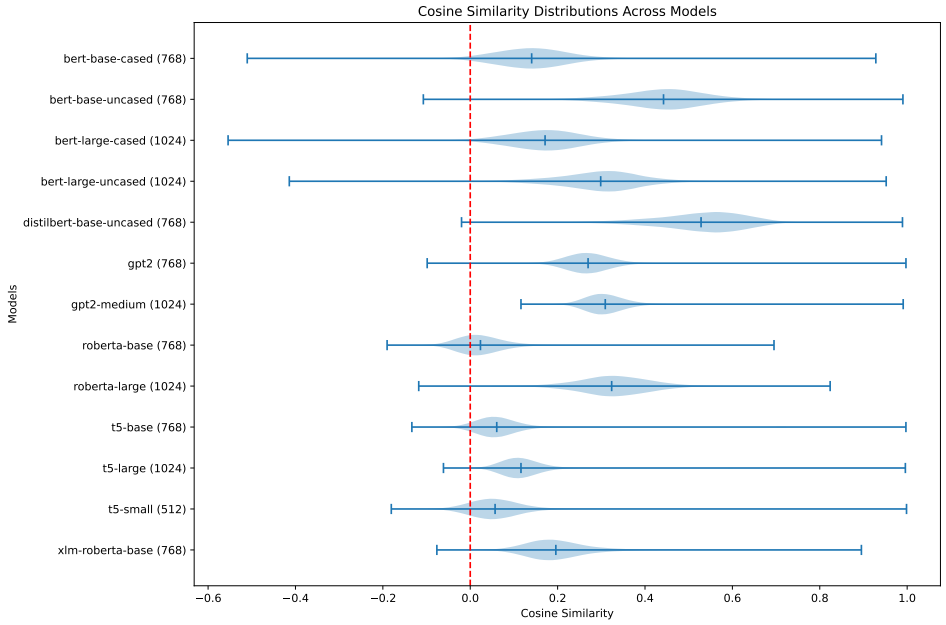

FIGURE 1. The distribution of pairwise cosine similarities of input embeddings for various models. The dimension of the embedding space is in parenthesis after the model name.

directions encode various semantic meanings in any transformer model, the conclusion here would then be that almost all transformer models tend to see quite a lot of similarities between any two words or tokens – at least at the input embedding phase of things. Even if there are some meanings that are somewhat orthogonal, strong negative correlation in the form of a cosine similarity near -1 seems to be unheard of. We note that it is not the "purpose" of the language models or their input embedding vectors to minimize pairwise absolute cosine similarities. The models do want to understand differences between tokens, but they also really "want" to find similarities. Furthermore even if the model uses some "basis" collection of almost orthogonal directions to encode various meanings that correlate more or less, most of the embedding vectors are then in a sense mapped to be linear combinations of these major dimensions and not dominant directions themselves.

We've listed some more detailed statistical numbers in Tables 4 and 5.

We further note that if we rescale the distributions as probability distributions and then normalize them by translating by the mean and scaling by the standard deviation, the distributions are very alike to each other and the normal distribution. See Figure 2. We are not quite sure if this is expected or surprising.

3. MATHEMATICAL PRELIMINARIES

We'll start off with some mathematical preliminaries. We go through basic notations and then define some special functions in order to approximate the volumes and ares of high-dimensional balls, spheres and their subsets. We

Model	Emb. Dim	Mean CosSim	Std CosSim	CosSim 25%	CosSim 50%	CosSim 75%
xlm-roberta-base	768	0.1959	0.0695	0.1487	0.1896	0.2358
t5-small	512	0.0569	0.0592	0.0170	0.0537	0.0925
t5-large	1024	0.1161	0.0472	0.0849	0.1126	0.1425
t5-base	768	0.0607	0.0491	0.0281	0.0576	0.0889
roberta-large	1024	0.3237	0.0912	0.2727	0.3273	0.3823
roberta-base	768	0.0234	0.0543	-0.0136	0.0189	0.0550
gpt2-medium	1024	0.3091	0.0471	0.2775	0.3058	0.3365
gpt2	768	0.2697	0.0537	0.2349	0.2679	0.3022
distilbert-base-uncased	768	0.5283	0.1023	0.4668	0.5398	0.5998
bert-large-uncased	1024	0.2986	0.0999	0.2397	0.3036	0.3598
bert-large-cased	1024	0.1713	0.0859	0.1155	0.1716	0.2261
bert-base-uncased	768	0.4424	0.0983	0.3811	0.4453	0.5052
bert-base-cased	768	0.1406	0.0848	0.0845	0.1397	0.1944

TABLE 4. Cosine similarity statistics: Embedding dimension, mean/std and quartiles of the cosine similarity.

Model	Emb. Dim	Mean Norm	Std Norm	Norm 25%	Norm 50%	Norm 75%
xlm-roberta-base	768	5.8701	0.4551	5.6919	5.9038	6.1314
t5-small	512	522.4708	65.9444	483.9047	520.5771	556.7801
t5-large	1024	457.1157	63.6800	416.2583	449.0090	488.0606
t5-base	768	517.7155	72.0042	473.5927	510.2301	555.2615
roberta-large	1024	4.3211	0.5238	4.0697	4.4603	4.6806
roberta-base	768	3.6286	0.3494	3.4261	3.6920	3.8746
gpt2-medium	1024	3.6835	0.4126	3.4084	3.6646	3.9646
gpt2	768	3.9556	0.4379	3.6681	3.9405	4.2474
distilbert-base-uncased	768	1.6640	0.2713	1.4537	1.6626	1.8302
bert-large-uncased	1024	1.4575	0.2002	1.3131	1.4350	1.5864
bert-large-cased	1024	1.5238	0.1908	1.3917	1.5058	1.6478
bert-base-uncased	768	1.4036	0.1989	1.2556	1.4006	1.5312
bert-base-cased	768	1.2871	0.1511	1.1886	1.2687	1.3790

TABLE 5. Embedding norm statistics: Embedding dimension, mean/std and quartiles of the norm.

then present some high-level observations on the geometrical differences of high and low dimensions.

3.1. **Notation.** We denote by $\mathbb{N} = \{0, 1, 2, \ldots\}$ the set of natural numbers and use the shorthand of $\mathbb{N}_+$ for the strictly positive natural numbers $\{1, 2, 3, \ldots\}$. By $\mathbb{R}$ we denote the set of real numbers, while $\mathbb{R}_+$ stands for the non-negative real numbers $[0, \infty)$.

By $\mathbb{R}^n$ we denote the n-dimensional Euclidean space for $n \geq 1$, i.e. the set

$$\{(x_1, x_2, \ldots, x_n) \mid x_i \in \mathbb{R}\}.$$

We often denote the vectors in $\mathbb{R}^n$ by $\mathbf{v}$, $\mathbf{w}$, $\mathbf{u}$ or some other bold font letter. In these cases we implicitly assume that the coordinates of the vector are denoted by subscripts of an unboldened letter, e.g. the components of the vector $\mathbf{w}$ are $(w_1, w_2, \ldots, w_n)$, and we tacitly assume that the dimension n is clear from the context. The *origin* is the point with all zero coordinates and we denote it as $\mathbf{0} := (0, \ldots, 0)$.

We equip $\mathbb{R}^n$ with the inner product

$$\langle \cdot, \cdot \rangle \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad \langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{n} v_i w_i$$

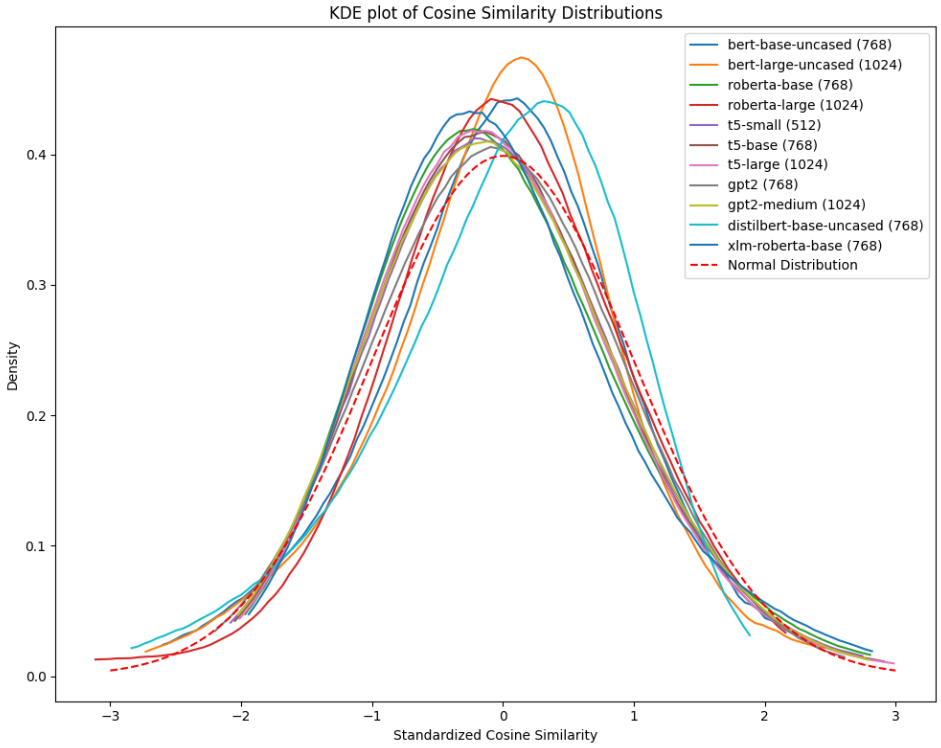

FIGURE 2. The distribution of pairwise cosine similarities of input embeddings for various models after being scaled as a probability distribution and normalized. Normal distribution is shown for comparison.

and the Euclidean norm

$$\|\cdot\|\colon \mathbb{R}^n \to \mathbb{R}_+, \quad \|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v}\rangle} = \sqrt{\sum_{i=1}^{n} v_i^2} \quad \text{for } \mathbf{v} \in \mathbb{R}^n.$$

A vector with length 1 is called a *unit vector*. We call two non-zero vectors $\mathbf{v}$ and $\mathbf{w}$ *orthogonal* if $\langle \mathbf{v}, \mathbf{w}\rangle = 0$, and *orthonormal* if they are orthogonal unit vectors.

With the inner product we can also define a crucial concept in this survey - *the cosine similarity*.

Definition 3.1. Let $\mathbf{v}, \mathbf{w}$ be two vectors in $\mathbb{R}^n \setminus \{\mathbf{0}\}$. We set their *cosine similarity* to be the number

$$\operatorname{cs}(\mathbf{v}, \mathbf{w}) = \frac{\langle \mathbf{v}, \mathbf{w}\rangle}{\|\mathbf{v}\| \cdot \|\mathbf{w}\|}.$$

Note that orthogonal vectors have cosine similarity of 0, and the cosine similarity of a vector with itself is always 1. Furthermore, scaling of either vector by a non-zero scalar has no effect on the vectors' cosine similarity.

For any two non-zero vectors $\mathbf{v}, \mathbf{w}$,

$$\cos(\theta) = \frac{\langle \mathbf{v}, \mathbf{w} \rangle}{\|\mathbf{v}\| \cdot \|\mathbf{w}\|} = \mathrm{cs}(\mathbf{v}, \mathbf{w}),$$

where θ is the angle between the vectors.

3.1.1. *A few special functions.* When studying the volumes and areas of high-dimensional balls and spheres the formulas tend to involve a few *special functions*. These are akin to the so called *special constants* like π or e, except that the Gamma and Beta functions are *functions* and not constants. They are natural objects that pop up[4] in many different settings, just like π. Regardless, they are needed for many volume and area estimates of high-dimensional balls and spheres; for further details and proofs we refer the reader to [AS48].

The Gamma function can be defined in a subdomain of the complex plane, but for us it is sufficient to study it for positive real numbers.

Definition 3.2. The *Gamma function* Γ is defined as

$$\Gamma\colon (0,\infty) \to \mathbb{R}, \quad \Gamma(x) = \int_0^\infty t^{x-1} e^{-x} \, \mathrm{d}t.$$

For our purposes, the following properties of the Gamma function will be sufficient, see again [AS48] for the proofs.

(Γ1) The Gamma function generalizes the factorial, i.e. for all $n \in \mathbb{N}_+$

$$\Gamma(n) = (n-1)! = (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1.$$

(Γ2) More generally, we have the following recursive relation for all $x \in (0,\infty)$

$$\Gamma(x+1) = x \cdot \Gamma(x).$$

(Γ3) We have an explicit value for $\frac{1}{2}$, namely $\Gamma(\frac{1}{2}) = \sqrt{\pi}$.

(Γ4) Stirling's formula approximation[5] applies for the Gamma function, i.e.

$$\Gamma(x+1) \approx \sqrt{2\pi x} \left(\frac{x}{e}\right)^x.$$

Definition 3.3. We define the *incomplete Beta function* $B\colon [0,1]\times(0,\infty)^2 \to \mathbb{R}$ by

$$B(x;\, a, b) = \int_0^x t^{a-1} \, (1-t)^{b-1} \, dt.$$

And the *Beta function* is defined simply by setting the parameter $x = 1$ and denoted with a slight abuse of notation as $B(a, b)$.

[4]One could take the Platonic view that these are actually existing objects that have some special property, or a more Kolmogorov-complexity style of approach that these hapepn to be concepts that we come across so frequently that we've given them special short names.

[5]We'll ignore here the exact accuracy of the approximation here. In the range of values we study it is less than a percent and, as we shall see, our estimations are not dependent on so fine values.

Definition 3.4. The *regularized incomplete Beta function* $I\colon [0,1]\times(0,\infty)^2 \to \mathbb{R}$ is defined as

$$I_x(a,b) = \frac{B(x;\, a,b)}{B(a,b)}.$$

3.2. **Volumes and areas of spheres and balls.** With the Gamma function and the various Beta function variants, we can now move on to studying the volumes and areas for spherical objects in high dimensions.

By *the unit cube* in $\mathbb{R}^n$ we mean the set $[0,1]^n$, and by *the unit cube centered at the origin* the set $\left[-\frac{1}{2},\frac{1}{2}\right]^n$. When we talk of a *cube of side length* a we mean a unit cube that has been scaled by the scalar a in all coordinates.

In $\mathbb{R}^n$, the volume of a cube of side length a is simply a^n, and the diameter of such a cube is $a\sqrt{n}$. In particular the unit cube has volume 1 and diameter $\sqrt{n}$.

We denote by $\overline{B}_n(r)$ the *closed ball of radius* r *in* $\mathbb{R}^n$, i.e. the set

$$\overline{B}_n(r) = \{\mathbf{v} \in \mathbb{R}^n \mid \|\mathbf{v}\| \leq r\}.$$

Furthermore we denote its volume by $V_n(r)$.

Similarly the $n-1$ dimensional *sphere of radius* r is the set

$$\mathbb{S} = \{\mathbf{v} \in \mathbb{R}^n \mid \|\mathbf{v}\| = r\}.$$

The volume of a ball of radius r in $\mathbb{R}^n$ is given by

$$V_n(r) := \frac{\pi^{n/2}}{\Gamma(n/2+1)} f^n. \tag{3.1}$$

(See again [AS48] for details.)

With the Stirling formula we can get an approximation for the volume as

$$\begin{aligned} V_n(r) = \frac{\pi^{n/2}}{\Gamma(n/2+1)} r^n &\approx \frac{\pi^{n/2}}{\sqrt{2\pi(n/2+1)}\cdot\left(\frac{n/2+1}{e}\right)^{n/2+1}} r^n \\ &= \frac{1}{\sqrt{n\pi}}\left(\frac{2\pi e}{n}\right)^{n/2} r^n. \end{aligned}$$

From this we can then calculate an approximate value for the function $R_n\colon \mathbb{R}_+ \to \mathbb{R}_+$ defined by setting $R_n(V)$ to be the radius r_V for which $V_n(r_V) = V$:

$$R_n(V) \approx (\pi n)^{1/(2n)} \sqrt{\frac{n}{2\pi e}} V^{1/n}.$$

Note also that for the unit ball we have

$$V_n(1) = \frac{1}{\sqrt{n\pi}}\left(\frac{2\pi e}{n}\right)^{n/2}.$$

From these formulae the crucial thing to note is that in $V_n(r)$ for large enough n the value is completely dominated by the term $n^{-n/2}$, which goes to zero faster than any polynomial or exponential function. To demonstrate this, in Figure 3 we plot some volumes of balls in various dimensions. The key takeaway here is that in higher dimensions the volume of the unit ball starts to converge to zero as $\mathcal{O}(n^{-n/2})$, i.e. exceedingly fast.

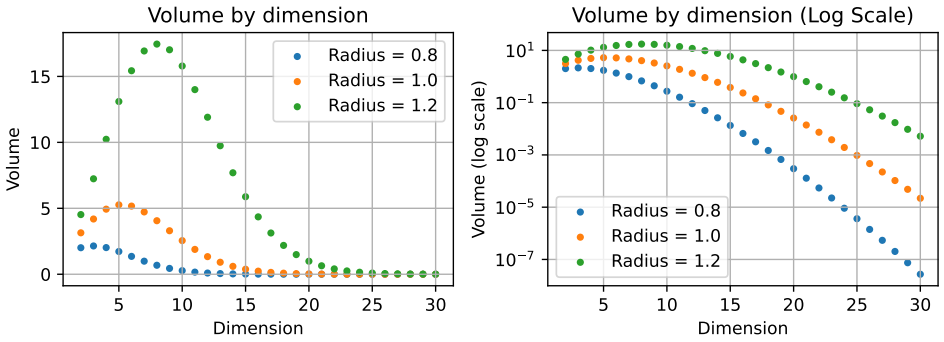

FIGURE 3. Volume comparisons of balls in different dimensions. The maximum value for the unit ball occurs at $n = 5$.

In dimension 768, which is important for us, trying to directly calculate the volume with the Gamma function creates an integer overflow in Python3, and probably also in most other non-symbolic math libraries. With the Stirling formula we can approximate that the unit ball in dimension 768 has volume below 10^{-300}. Conversely, to get a ball with volume 1 we can calculate that we would need a radius of around $R_{768}(1) \approx 6.7$. This is our first important geometric observation concerning high-dimensional geometry – *the volumes of n-dimensional unit balls are very very small.*

On the other hand, as we mentioned earlier, the volume of a unit cube in $\mathbb{R}^n$ is 1, and the diameter is $\sqrt{n}$. So for example in dimension 768 we get the diameter of the unit cube to be about 27.7. This property is a first hint at a phenomenon that is sometimes described as "pointyness of high-dimensional cubes". The idea being that even though in all dimensions some parts of a unit cube centered at the origin are inside the circle (e.g. any point in the cube of the form $(0, \ldots, 0, 1/2, 0, \ldots, 0)$ is on the boundary of the cube and within a unit ball), the corners of the cube start to point out of the ball more and more. Also note that in 1D, looking at the edge vertex of a cube we see that half of the ambient space is in the cube. In 2D, only a fourth and in 3D only an eighth – see Figure 4. This theme continues in higher dimensions, with the edge vertex of a cube in dimension 768 having less than one part in 10^{-200} of the ambient space being part of the cube. This can be thought as another measure of the pointyness. In dimension 768, considering that the unit cube has a diameter of around 27, we note that the corners also point out of the ball quite a lot. Furthermore there are 2^{768} of them, and the distance between two neighbouring ones is only 1, so the cube does seem to have a more "hedgehog-style" quality to it.

For our geometrical considerations, the concept of *spherical caps* is crucial. To construct a spherical cap on a sphere, we simply fix a point on the sphere take the intersection of the sphere with a ball centered on that point. There are various ways to parametrize the spherical cap, we refer to Figure 5 for the notation. In studying almost orthonormal vectors, we note that two unit vectors $\mathbf{v}$ and $\mathbf{w}$ are ε-almost orthonormal exactly when on the unit sphere the spherical caps with $\vartheta = \varepsilon/2$ around them are disjoint.

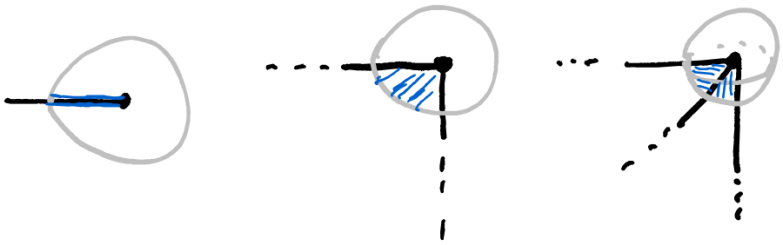

FIGURE 4. An illustration on how much of the ambient space of a cube edge is part of the cube.

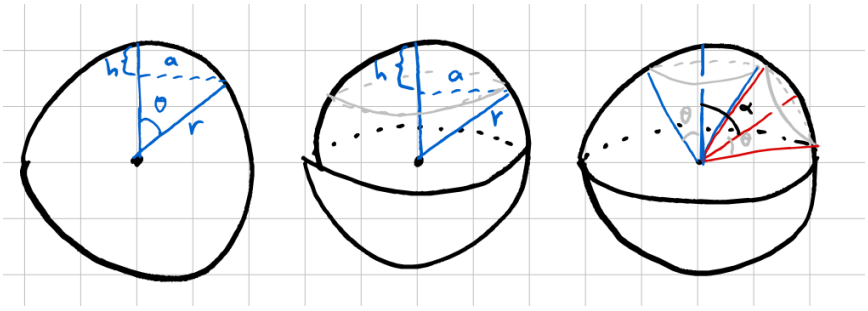

FIGURE 5. Example image of spherical caps.

We are interested in calculating the area[6] of such spherical caps on unit spheres, which we denote by SC_ϑ, in order to calculate an area-based bound on the amount of disjoint spherical caps on a unit sphere.

In dimension n we get the following formula for the area of a spherical cap, see [Led01] for details:

$$A_h = \frac{1}{2}\frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} r^{n-1} I_{(2rh-h^2)/r^2}\left(\frac{n-1}{2}, \frac{1}{2}\right), \quad 0 \leq h \leq r, \tag{3.2}$$

where I_x is the regularized incomplete Beta function previously defined. For a unit cube this simplifies to

$$A_h = \frac{1}{2}\frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} I_{(2h-h^2)}\left(\frac{n-1}{2}, \frac{1}{2}\right), \quad 0 \leq h \leq 1. \tag{3.3}$$

We note here that if we plot A_h for $h \in [0,1]$ in $\mathbb{R}^{256}$, see Figure 6, we see that the supermajority of the area of the sphere is concentrated on the equator.

3.3. **High and low dimensions in general.** Low dimensions have several "exotic" properties, both geometrical and topological. For example:

[6]For simplicity, we'll use the term "area" even though we are not working with 2-dimensional objects. Here "$n-1$ volume" or "$n-1$ dimensional Hausdorff measure" would be more appropriate, but a mouthful.

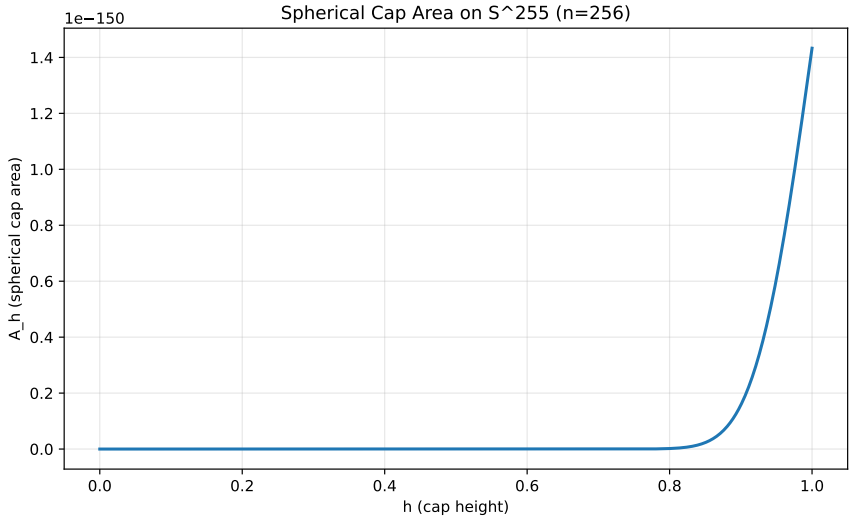

FIGURE 6. A plot of the area of a spherical cap of a unit sphere in $\mathbb{R}^{256}$.

(1) We've already noted above some aspects of high-dimensional geometry that are different from the more "usual" low-dimensional one. Another point of view is that it is the low dimensions that are the strange ones. In high dimensions cubes are pointy, unit balls have low volumes and even what they have is concentrated near the boundary – indeed with the volume of a ball of radius r behaving as $\mathcal{O}(r^n)$, it is easy to see that for larger n the volume of $B(\mathbf{0}, 0.99)$ is considerably smaller than the volume of $B(\mathbf{0}, 1)$, meaning that the volume of the ball is concentrated near the boundary. Furthermore as we observed in Figure 6, for a sphere in high dimensions the supermajority of the volume of the sphere is concentrated near the equator.[7]
(2) The only way to equip a Euclidean space with a product that produces a division algebra is restricted to dimensions 1, 2, 4 and 8 [Ada60]. As a more simple example, the only way to equip a Euclidean space with a product that produces an algebraic field structure is to take $\mathbb{R}$ with the standard product or $\mathbb{R}^2$ with the product of complex numbers, see [Hat05].
(3) Knot theory works only in dimension 3; in lower dimensions a string cannot be knotted and in higher dimensions any knot can be trivially untied [Rol03].
(4) Stable orbits of planets require 3+1 dimensions, i.e. three spatial dimensions and one temporal dimension, see e.g. [Ehr17, Teg97].
(5) Exotic differentiable structures[8] occur only in $\mathbb{R}^4$ [Fre82, Don83].

[7]This effect is also known as "the concentration of measure", see [Led01, Ver18].

[8]We won't go to the very technical details here, but we can equip an Euclidean space with a so called differentiable structure that specifies how calculus works on that space. There is only one choice in all the Euclidean spaces except for $\mathbb{R}^4$, which has more.

4. Mathematical bounds to almost orthogonality

Our main question in this section is to find out how many almost orthogonal vectors can we fit in a given vector space. After the previous section on how different the behaviour of low and high dimensional spaces can be, we should expect the dimension to play a large role. Indeed, let's begin by doing a quick simulation. In the Figure 7 we have sampled 1000 random vectors in a few different dimensions and plotted the distribution of their cosine similarities - note that we have included for any pair of vectors $\mathbf{v}, \mathbf{w}$ both the cosine similarity $\mathrm{cs}(\mathbf{v}, \mathbf{w})$ and $\mathrm{cs}(\mathbf{w}, \mathbf{v})$, which does not effect the shape here as $\mathrm{cs}(\mathbf{v}, \mathbf{w}) = \mathrm{cs}(\mathbf{w}, \mathbf{v})$.

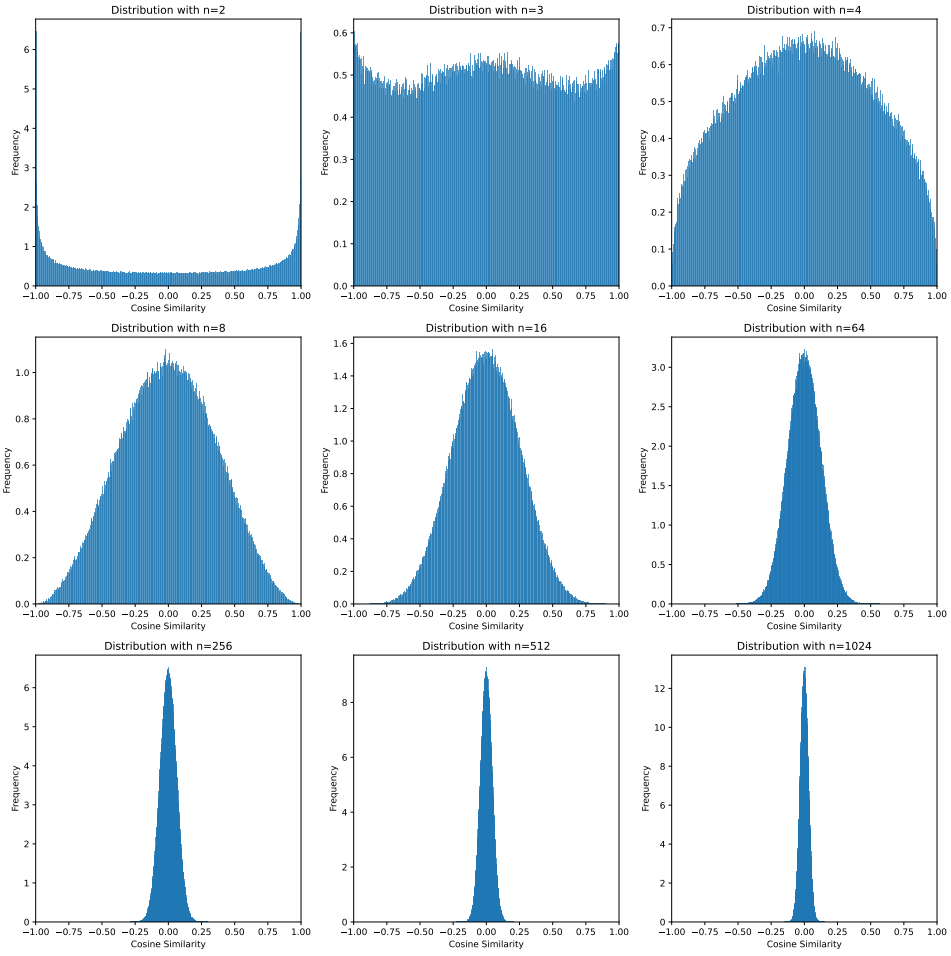

Figure 7. Cosine similarities of randomly sampled vectors in various dimensions.

Indeed, Vershynin [Ver18][9] shows that the excpected value of the cosine similarity of two random n-dimensional unit vectors is $1/\sqrt{n}$. This is in some way expected; see again Figure 6 describing how the mass of a sphere

[9]See also https://math.stackexchange.com/questions/4555166/the-probability-for-inner-product-for-two-unit-sphere-uniformly-distributed-rand.

is concentrated on the equator. From this point of view if you choose one vector as your basis, then any random vector is most likely chosen from near the equator relative to the first vector, meaning that they are nearly orthogonal.

4.1. **Volume-based limits.** What's the optimal amount then? How many ε-almost orthogonal vectors can we have in $\mathbb{R}^n$ for a given ε and n? Here it turns out that an exact answer is really hard to figure out. The problem is related to the problem of *sphere packings* in high dimensions. Indeed, suppose we have a set of vectors in $\mathbb{R}^n$ that have some upper bound on their pairwise cosine similarities. Since the cosine similarity is not altered by scalar scaling, we can assume these are all unit vectors. The upper bound on the cosine similarities implies that these vectors, when thought of as points in the $n-1$-dimensional unit sphere $\mathbb{S}^{n-1}$, have a lower bound $b = b(\varepsilon)$ on their pairwise distances in the arc-distance of $\mathbb{S}^{n-1}$. This correspondence between the angle bound and arc-distance is one-to-one, meaning that looking for an optimal ε-almost orthogonal set of vectors in $\mathbb{R}^n$ is quantitatively *equivalent* to finding a set of points $\mathbb{S}^{n-1}$ with a uniform lower bound b on their pairwise distances. This in turn translates to trying to find out how many disjoint balls of radius b can we have in the space $\mathbb{S}^{n-1}$. This problem is a variation of the so called *sphere packing problem*, and we do not know how to solve it even in most Euclidean spaces. (The solution is currently known in dimensions 3, 8 and 24: [Hal05, Via17, CKM+17].) So suffice it to say that we can't prove in this paper what is the maximum amount of t-almost orthogonal vectors in dimension 768.

So if an optimal bound is impossible, can we at least estimate an upper bound of these sphere packings? Let's try an approach based on volume estimates; two distinct unit vectors give rise to disjoint spherical caps, and we can calculate how many disjoint spherical caps can at most fit on a sphere based on volume. First recall Figure 5 on the terminology of spherical caps and Section 3.2 for the area of such caps. We want to calculate the For ε-almost orthogonal vectors, the angle between them is $\alpha = \cos^{-1}(\varepsilon)$, and the disjoint spherical caps have angle $\alpha/2$.

$$h = 1 - \cos(\alpha/2) = 1 - \cos\left(\frac{1}{2}\cos^{-1}(\varepsilon)\right).$$

In particular, for us this boils down to

$$A_h = \frac{1}{2}\frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} I_{2h-h^2}\left(\frac{n-1}{2}, \frac{1}{2}\right)$$

and with $h = 1 - \cos(\alpha/2)$ we get, by momentarily denoting $\cos(\alpha/2) =: t$, that

$$\begin{aligned} 2h - h^2 &= 2 - 2\cos(\alpha/2) - (1 - \cos(\alpha/2))^2 \\ &= 2 - 2t - (1-t)^2 \\ &= 2 - 2t - 1 + 2t - t^2 \\ &= 1 - t^2 = 1 - \cos(\alpha/2)^2. \end{aligned}$$

So in particular we get that

$$\mathrm{vol}(SC_{\alpha/2}) = \frac{1}{2}\frac{2\pi^{n/2}}{\Gamma\left(\frac{n}{2}\right)} I_{1-\cos(\alpha/2)^2}\left(\frac{n-1}{2}, \frac{1}{2}\right).$$

On the other hand for the area of the whole n-dimensional unit sphere, $\mathbb{S}^n$, we get

$$A = \frac{2\pi^{n/2}}{\Gamma(n/2)}.$$

In particular, a single spherical cap takes up a fraction of

$$\frac{SC_{\alpha/2}}{A} = \frac{1}{2} I_{1-\cos(\alpha/2)^2}\left(\frac{n-1}{2}, \frac{1}{2}\right)$$

of the area of the total unit n-sphere. Note that each direction produces two such spherical caps at the antipodal points. Thus the inverse of twice this fraction gives an absolute area-based upper bound for the amount of ε-almost orthogonal vectors in $\mathbb{R}^n$.

Let's get estimates for values of ε being 0, 0.1 and 0.01 in the dimensions of $n = 2$, $n = 3$, $n = 32$, $n = 768$ and $n = 4096$. For each ε we have $\alpha = \cos^{-1}(\varepsilon)$, and for these values we wish to estimate

$$I_{1-\cos(\alpha/2)^2}\left(\frac{n-1}{2}, \frac{1}{2}\right)^{-1},$$

where $I_x(a, b)$ is the incomplete Beta function. The results are listed in table

$n\backslash\varepsilon$	0.1	0.01	0
2	2.136	2.013	2.0
3	3.87	3.456	3.414
4	6.605	5.602	5.504
8	45.08	31.36	30.17
16	1526	720.9	665.9
32	1.267×10^6	2.784×10^5	2.372×10^5
768	2.537×10^{134}	3.249×10^{118}	6.849×10^{116}
4096	6.635×10^{711}	1.127×10^{627}	1.296×10^{618}

TABLE 6. Some area-based bounds for the amount of ε-almost orthogonal vectors in $\mathbb{R}^n$.

So what we note here is that even though these area estimates do give an upper bound to the almost orthogonal vectors, the estimates are quite useless. Indeed, when applied to the cosine similarity of 0 which corresponds to actually orthogonal vectors, we get an estimate much larger than the dimension which we know to be the real bound in dimensions above three. So the *geometry* of the situation is really crucial here; the dominant limit to the amount of spherical caps is not lack of volume. Indeed, as we'll discuss in Section 4.3, the so called *packing density* goes down at an exponential speed for high dimensional sphere packings.

So this purely area-based bound does tell us that we can't pack more than about 10^{100} of 0.1-almost orthogonal vectors in $\mathbb{R}^{768}$. Later on we'll see that

the actually achievable amounts seem to be in the realm of $10^3 - -10^5$, so this bound is not very useful in practice.

4.2. **The Johnson-Lindenstrauss Lemma.** We next study some existing strong results on how point clouds can be embedded into a lower ambient dimension while limiting the amount of distortion. First we study a classical result of Johnson-Lindenstrauss that gives excellent asymptotic bounds on a slightly more complex question.

One approach to get almost orthogonal vectors in $\mathbb{R}^n$ is to try to take a set of orthogonal vectors in a higher dimensional space, say $\mathbb{R}^N$, and then see if we can have a mapping $f\colon \mathbb{R}^N \to \mathbb{R}^n$ that doesn't distort the cosine similarities "too much". This type of dimensionality reduction is a very important tool in e.g. data-analysis, compressed sensing, and other fields where we get algorithmic complexity penalties from high dimensions. A classical result to aid in this is the Johson-Lindenstrauss lemma, see [JL$^+$84]

Lemma 4.1 (Johnson-Lindenstrauss Lemma)**.** *Let $0 < \varepsilon < 1$ and let $S := \{x_1, \dots, x_k\} \subset \mathbb{R}^N$. Then for any $n > \frac{8\log(k)}{\varepsilon^2}$ there exists a linear map $L\colon \mathbb{R}^N \to \mathbb{R}^n$ such that*

$$(1-\varepsilon)\|x_i - x_j\|^2 \leq \|Lx_i - Lx_j\|^2 \leq (1+\varepsilon)\|x_i - x_j\|^2 \tag{4.1}$$

for all $x_i, x_j \in S$.

Remark 4.2. We have used in the statement of the theorem the constant 8 in the bound $n > \frac{8\log(k)}{\varepsilon^2}$, as this seems to be the common one listed e.g. in Wikipedia. However, the source for this constant are the lecture notes [FG16, Lemma 2.6] which are not peer-reviewed, though they do contain a proof. The discussion on the Wikipedia articles talk page[10] lists various variants on this bound, which we summarize here:

- For $m > 4$ we get $n > 20(\ln m)/\varepsilon^2$ in [MRT18, Lemma 15.4].
- In [Mat13, p. 300] they have $n > 200(\ln m)/\varepsilon^2$.
- [DG03] lists a different "style" of bound: $n > 4\left(\varepsilon^2/2 - \varepsilon^3/3\right)^{-1}\log(k)$.
- The lecture notes [Duc24, Section 3.1.3] prove $n > 16(\ln m)/\varepsilon^2$.

For our purposes it turns out that the exact choice here does not matter, as we shall soon see. The issue is that these results are by nature asymptotic and do not give strong estimates in the small-ish dimensions like 768.

Note that the Johson-Lindenstrauss Lemma can be widely useful in dimension reduction. Suppose we are studying a number of grayscale images that have been taken with a megapixel resolution, i.e. we have a set of vectors in $\mathbb{R}^{1000000}$. Many algorithms will balk at such a high dimension, e.g. if their complexity is polynomial w.r.t. the dimension. But if we are willing to tolerate some error, then Johnson-Lindenstrauss can help us by a lot. Indeed, imagine we have 10k images we want to study and we are okay with an error rate of $\varepsilon = 0.1$. Now the Johnson-Lindenstrauss lemma gives us a mapping into an Euclidean space of dimension around 7000. Now suppose we have 100k images, 1M or 10M images. What we see now is that we only need to map our data to dimension of about 9k, 11k or 13k, respectively.

[10] `https://en.wikipedia.org/wiki/Talk:Johnson\%E2\%80\%93Lindenstrauss_lemma`,viewed 24.09.2025.

This reduction in dimension from a million to around ten thousand can have a drastic effect in practice. Especially since the amount of images is only increasing the end dimension in a logarithmic way, as we see in these examples.

Now let's turn to see how we might use Johson-Lindenstrauss with our problem. The natural idea would be, of course, to take a large-dimensional $\mathbb{R}^N$, and choose its basis plus the origin as our set of vectors. Now the first question is then about how the cosine similarity will react to the Johnson-Lindenstrauss map. First we see that

$$\begin{aligned}\frac{\langle L\mathbf{v}, L\mathbf{w}\rangle}{\|L\mathbf{v}\|\|L\mathbf{w}\|} &= \frac{1}{2}\frac{\|L\mathbf{v}\|^2 + \|L\mathbf{w}\|^2 - \|L\mathbf{v} - L\mathbf{w}\|^2}{\|L\mathbf{v}\|\|L\mathbf{w}\|} \\ &\leq \frac{1}{2}\frac{(1+\varepsilon)\|\mathbf{v}\|^2 + (1+\varepsilon)\|\mathbf{w}\|^2 - (1-\varepsilon)\|\mathbf{v} - \mathbf{w}\|}{\sqrt{1-\varepsilon}\|\mathbf{v}\|\sqrt{1-\varepsilon}\|\mathbf{w}\|}.\end{aligned}$$

Now if $\mathbf{v}$ and $\mathbf{w}$ are orthonormal, then $\|\mathbf{v}\| = \|\mathbf{w}\| = 1$, and $\|\mathbf{v} - \mathbf{w}\|^2 = 2$, so we get that this equals to

$$\begin{aligned}&\frac{1}{2}\frac{(1+\varepsilon)\|\mathbf{v}\|^2 + (1+\varepsilon)\|\mathbf{w}\|^2 - (1-\varepsilon)\|\mathbf{v} - \mathbf{w}\|}{\sqrt{1-\varepsilon}\|\mathbf{v}\|\sqrt{1-\varepsilon}\|\mathbf{w}\|} \\ &\quad= \frac{1}{2}\frac{(1+\varepsilon) + (1+\varepsilon) - 2(1-\varepsilon)}{(1-\varepsilon)} \\ &\quad= \frac{1}{2}\frac{4\varepsilon}{(1-\varepsilon)} \\ &\quad= \frac{2\varepsilon}{(1-\varepsilon)}.\end{aligned}$$

Similarly we can derive a lower bound, and so we have

$$-\frac{2\varepsilon}{(1+\varepsilon)} \leq \frac{\langle L\mathbf{v}, L\mathbf{w}\rangle}{\|L\mathbf{v}\|\|L\mathbf{w}\|} \leq \frac{2\varepsilon}{(1-\varepsilon)} \tag{4.2}$$

Thus if we apply the Johnson-Lindenstrauss Lemma to the set of unit vectors in $\mathbb{R}^N$ appended with the origin with our aim to generate almost orthogonal vectors in $\mathbb{R}^{768}$, we see that we have to choose ε to be roughly half that of the desired threshold. For an example, if we want the threshold to be around $\frac{1}{3}$, then we have to choose $\varepsilon = \frac{1}{6}$. With this choice the Johnson-Lindenstrauss lemma then gives us a bound from

$$\begin{aligned}768 > 8\frac{\ln(N+1)}{\varepsilon^2} &\Leftrightarrow 96 \cdot \varepsilon^2 > ln(N+1) \\ &\Leftrightarrow \frac{96}{36} > \ln(N+1) \\ &\Leftrightarrow \frac{8}{3} > \ln(N+1) \\ &\Leftrightarrow N < e^{8/3} - 1 \approx 13.4.\end{aligned}$$

So for the threshold of $\frac{1}{3}$, a direct application of the Johnson-Lindenstrauss technique guarantees at least 13 almost orthogonal vectors in $\mathbb{R}^{768}$. This is a far cry from the 768 orthogonal base vectors we know that exist in $\mathbb{R}^{768}$.

More generally Johnson-Lindenstrauss gives the bound

$$N < \exp\left(\frac{n}{8}\left(\frac{2t}{1-t}\right)^2\right) - 1. \tag{4.3}$$

And so for the threshold of 0.1-almost orthogonal vectors, to get more than the basis-guaranteed $N \geq n$ almost orthogonal vectors, then we have to have n at least 29748. Though after that bound the exponential behaviour takes over, and e.g. for $n = 40000$ and $\varepsilon = 0.1$ we already get $N > 10^6$.

So turns out that for our "not quite so large" dimension of 768, the Johnson-Lindenstrauss result does not yield results applicable for our almost orthogonal vectors. The issue here stems largely from the fact that the Johnson-Lindenstrauss result is an asymptotic result and gives a much stronger output than just almost orthogonal vectors - it roughly preserves distances of an *arbitrary* set of vectors.

We know, however, that more almost orthogonal vectors can be fitted into $\mathbb{R}^{768}$. For example, the vectors $(1, 1, 1, \ldots)/\sqrt{768}$ and $(1, -1, 1, -1, \ldots)/\sqrt{768}$ have a very small cosine similarity (below 0.04) both with all the standard basis vectors and each other. With a bit of clever combinatorics it's not hard to get many more such examples. Since these theoretical results don't seem to provide impressive results in this realm of dimensions, let us turn to less theoretical approaches. Though as we will later see the underlying idea behind the proof of the Johnson-Lindenstrauss Lemma is still very applicable for our purposes.

4.3. **Packing density, spherical codes and the Kabatianskii-Levenshtein bound.** In this section we'll briefly mention a few related topics. We won't dive deeper to these ideas, but note them for context.

As we noted in Section 4.1, the pure volume-based bounds on almost orthogonality were completely useless in high dimensions. The issue here is related to the so called *packing density* of spherical packings. We refer to [Coh16] for an overview of the topic of sphere packing and packing density, but the main point for us is that for high dimensions sphere packings are not dense; they cover an exponentially diminishing fraction of the ambient space as the dimension increases. We saw a part of this effect on the rightmost column of Table 6 – there the true maximum amount of disjoint spherical caps with antipodal pairs is dictated by linear algebra, but the bounds provided by essentially area density grow in some superexponential fashion.

So almost orthogonal vectors correspond to pairwise disjoint spherical caps. Another way of phrasing this is that they correspond to a set of points on a sphere with an upper bound to their pairwise inner products. Such point sets on a sphere are known as *spherical codes* and have direct applications in coding theory. We refer the reader to [CZ14] for details, but note that the study of spherical codes is also an unsolved field of study.

A related problem is known as *the Tammes problem* (see e.g. [Mus18]) where we ask to minimize the pairwise distances of a set of points on the sphere. Anecdotally we've seen the Tammes problem used as a term in the realm of "small balls and distances", whereas the "almost orthogonal" approach is more in the realm of large distances as the points induced on the sphere aim to be almost a quarter apart.

Note again that both the Tammes problem and spherical codes usually focuse on *points on a sphere* while we work on *directions in space*. Any direction naturally induces two antipodal points on a sphere, but this extra assumption of antipodal inclusion is not present on the more general questions of spherical codes or the Tammes problem.

5. Simulation approaches

Our aim is to next turn into generating collections of almost orthogonal vectors in various Euclidean spaces. As we've learned so far, there is very little hope of proving anything about the optimality of any of these collections. But if different methods seem to yield similar results, we could consider that this as weak evidence that an actual mathematical limit might be near.

5.1. **Methods.** Our three main generation approaches are:

(1) Sampling random unit vectors.
(2) Random projections of higher-dimensional bases in the spirit of the proof idea behind the Johnson-Lindenstrauss Lemma.
(3) Energy minimization through ML methods.

Each of these have their own subsection below, explaining the idea of the approach. Besides these generative techniques, we also test *pruning* methods, i.e. generating extra vectors and then dropping a subset with an aim of reducing the maximum absolute cosine similarity.

5.1.1. *Randomly sampled unit vectors.* For us this is the most straightforward method. If we sample unit vectors randomly, then they should not have massive pairwise cosine similarities. Indeed, the expected value of the cosine similarity between two randomly sampled unit vectors in $\mathbb{R}^n$ should be around $\frac{1}{\sqrt{n}}$. (See again the discussion in Section 4.)

It is not trivial[11] to derive what the expected maximum absolute cosine similarity then would be for N randomly generated unit vectors, which is why we turned to simulations. Though see again Figure 7 where we show a descriptive example of the distributions of pairwise cosine similarities of random unit vectors in various dimensions.

After some heuristical tests, we settled to generating twice the amount of vectors here and then pruning half of them.

5.1.2. *Random projections of higher-dimensional bases.* Instead of raw sampling, we can apply the procedure behind the proof of the Johnson-Lindenstrauss Lemma discussed earlier (Section 4.2). Here the basic idea is that if we take a collection of vectors in a high dimension, then its projection to a random subspace has a non-zero probability to weakly preserve its various (geo)metric properties. So for our purposes of trying to find almost orthogonal collections of vectors, a natural approach here would be to take a random orthonormal basis in a high-dimensional space $\mathbb{R}^N$ and project it to a lower n-dimensional space subspace $V \subset \mathbb{R}^N$.

Due to rotational symmetry, it suffices to either take the standard basis of $\mathbb{R}^N$ and project it to a random subspace or to take a random orthonormal basis and project it to the subspace spanned by the first n coordinates i.e.

[11] Or, it was not trivial for us.

take the first n coordinates of the vectors. We'll opt for the latter here, though the former could probably be optimized to run a bit faster and with less memory.

Also in this generation approach we opted to generating twice the amount of vectors and pruning half.

5.1.3. *Minimizing the absolute value of cosine similarity.* Partially motivated by a video of the math youtuber 3Blue1Brown[12] we wanted to try out a more machine-learning based method. Here we create a nice collection of vectors by minimizing a type of energy function on the pairwise distances of vectors. To this end we take a sequence of unit vectors representing directions $[v_1, \ldots, v_n]$, append them with their antipodal vectors $[-v_1, \ldots, -v_n]$, and then look at the energy

$$\sum_{i<j} \frac{1}{(d(w_i, w_j))^p},$$

where the $(w_j)_{j=1}^{2n}$ is the set of vectors and their antipodal vectors. We start with a random sampling of vectors, and then "nudge" each of the vectors v_i in direction that most reduces this energy.

In this approach the pruning methods proved, unsurprisingly, mostly detrimental.

5.2. **Combinations and hyperparameters.** We tested several variations and combinations of these generation and pruning techniques. For the energy minimization there was little effect on the results based on whether we started with a random collection of vectors or e.g. something generated with the random projections method. We also tested some variations on how we sample random unit vectors, but the results were indistinguishable.

For the energy minmization there were various hyperparameters, but as our aim here is to show more qualitative than quantitative results we won't go to details on our choices. The source will be made available. We will, however, mention that the choice for th exponent in our energy functional had a big effect on how fast the process converged. Figures 8 and 9 below show how the maximum absolute cosine similarity evolves with various exponent choices. We see that the evolution converges much faster when the exponent is increased until around exponent 16 after which results start to worsen. This is most likely due to the fact that the higher exponent produces stronger gradients, until the floating point numbers start to round off to zero. There is probably some clever way of optimizing this procedure, but we leave it out of the scope of this article.

We also tested a few variants of the energy minimization approach where we used other loss functions that targeted the max or top N absolute cosine similarities directly. They did not prove to be very effective in practice.[13]

In the pruning methods we tried a few variations of the amount of oversampling and then pruning. The factor of 2.0 was chosen quite heuristically

[12] `https://www.youtube.com/watch?v=9-Jl0dxWQs8`

[13] We also note that the 3Blue1Brown youtube video mentioned earlier also used a gradient flow method but aimed to minimize the *average* cosine similarity which produced good average results but strong outliers.

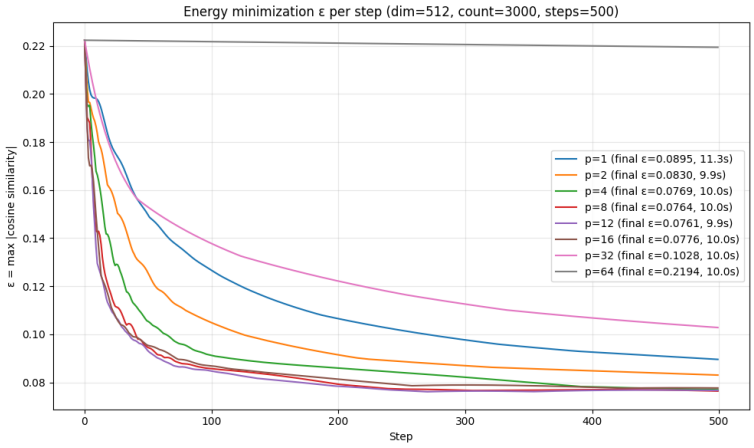

FIGURE 8. The effect of the energy exponent and step count in generating 3000 unit vectors in $\mathbb{R}^{512}$ minimizing their energy.

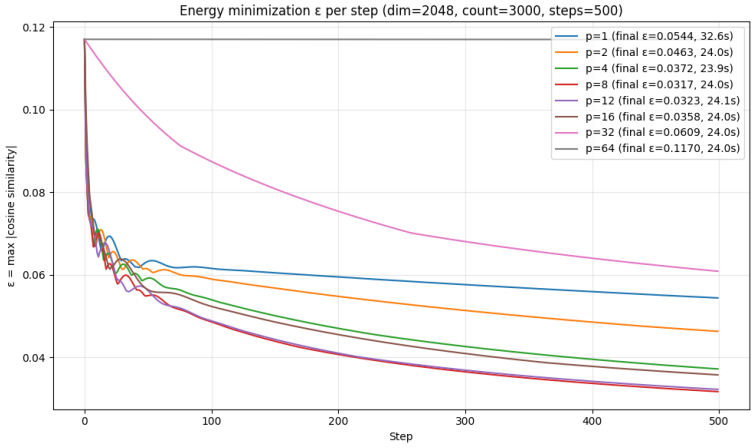

FIGURE 9. The effect of the energy exponent and step count in generating 3000 unit vectors in $\mathbb{R}^{2048}$ and minimizing their energy.

as a balance between compute time and results. We note that besides just dropping the "worst offenders" from the generated set one by one, a more subtle approach could be to try and detect what is in some sense the best subset of vectors from a given generated set. If we phrase this as a graph problem where each vector is a node and two nodes are connected when the corresponding two vectors have their absolute cosine similarity below some threshold, then this turns into the NP-complete *Clique problem*. This formulation does naturally discard a lot of geometric information, and finding a best (or good) almost orthogonal subset might be computationally much easier in our case. But with the problem adjacent to an NP-complete problem we would be wary.

5.3. **Generation results.** First of all, in Figures 10, 11, and 12 we show what kind of distribution of cosine similarities our generators produce in dimensions 32, 768 and 2048 for specific sizes of vector collections. What we note is that, especially in the lower dimensions, the energy minimizers create a bimodal distributions. The others seem more normal distribution-y.

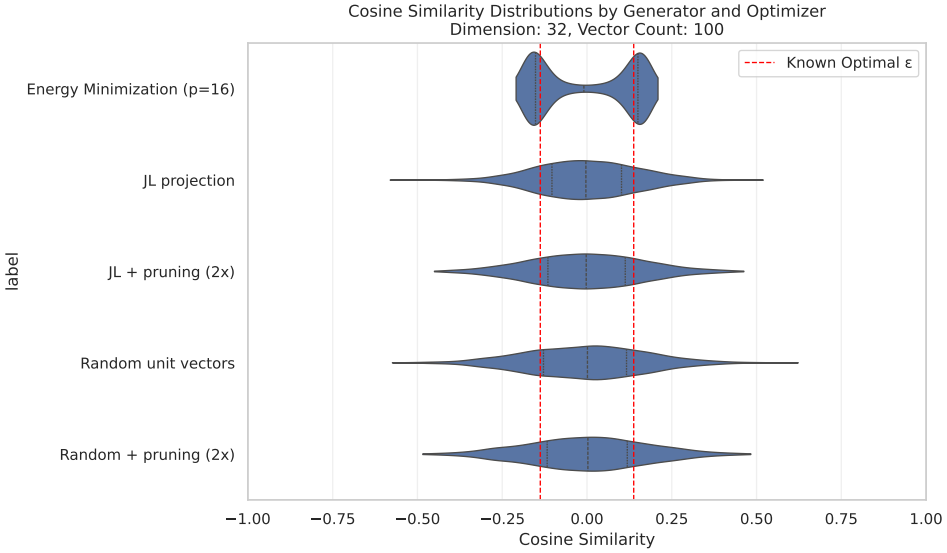

FIGURE 10. Violin plot of the pairwise cosine similarity distributions generated by various methods in dimension 32 and 100 vectors. The "known optimal" is from spherical codes and most likely unattainable with directions.

Then in Figures 13, 14, 15, 16, and 17 we show how the various techniques compare with a varying amount of vectors in dimensions 32, 128, 512, 768 and 1024. In dimension 32 we also have plotted some data from [Coh23] that shows the max cosine similarity of the best known corresponding sphere packing.[14]

5.4. **Simulation conclusions.** In the following Table 7 we've collected our best results and the method we used to acquire it. As thoroughly discussed above, these should not be considered to exact or even near exact but at least they should be indicative on what simple approaches can produce.

6. AFTERWORD

So after all of our analysis we can say that there is clearly an interesting phenomenon present in finding sets of almost orthogonal vectors in high dimensions. The existing theoretical bounds are still quite far from practical results.

[14]Here we again emphasize that our collections of almost orthogonal vectors correspond to collections of pairs of antipodal points on the sphere. The spherical packing minimizes have no such limitation of needing to have the antipodal point also present, and thus can naturally yield better results.

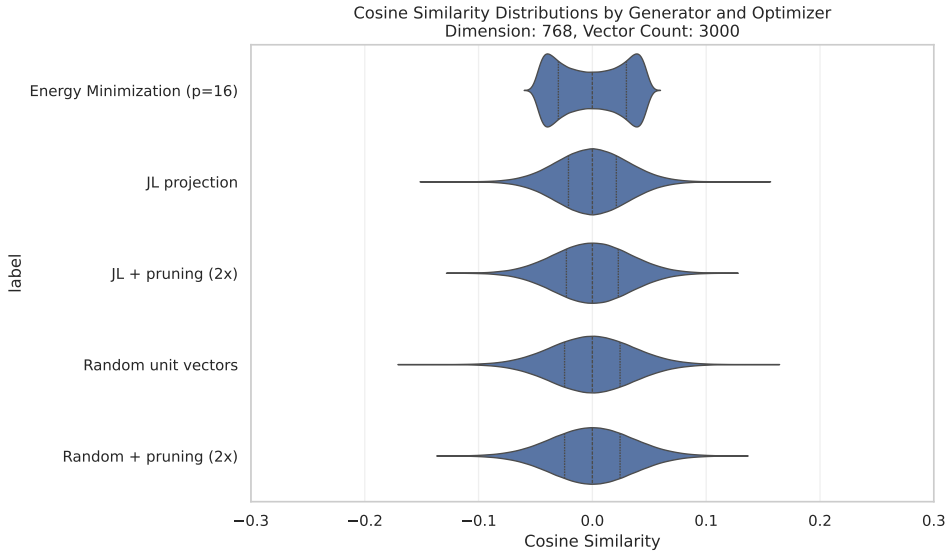

FIGURE 11. Violin plot of the pairwise cosine similarity distributions generated by various methods in dimension 768 and $3,000$ vectors.

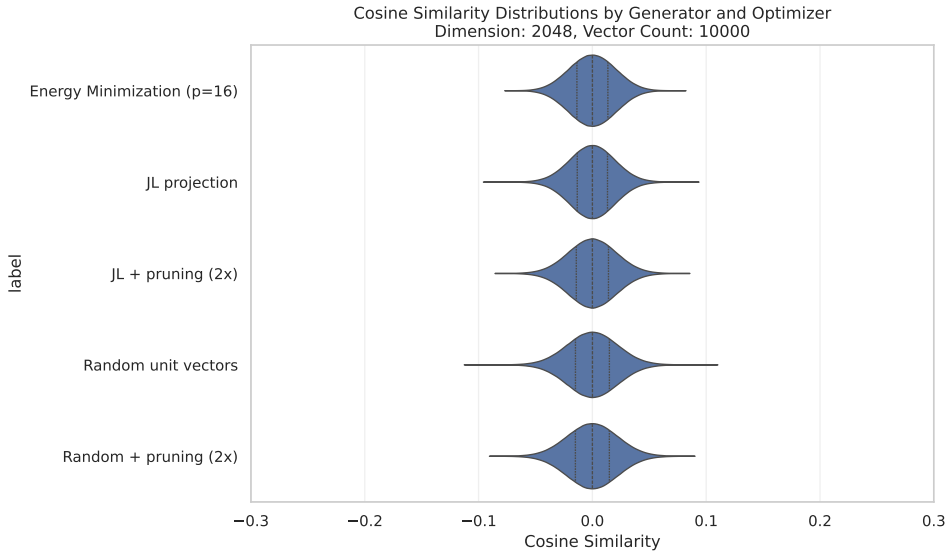

FIGURE 12. Violin plot of the pairwise cosine similarity distributions generated by various methods in dimension 2048 and $10,000$ vectors.

From the point of view of e.g. the BERT language model, what we've noted is that with thresholds around 0.1 we can fit an order of magnitude more almost orthogonal vectors in $\mathbb{R}^{768}$ than what we get from the basis vectors, which goes to show tha there is "a lot of space in the geometry" of $\mathbb{R}^{768}$ for lossily storing information as directions.

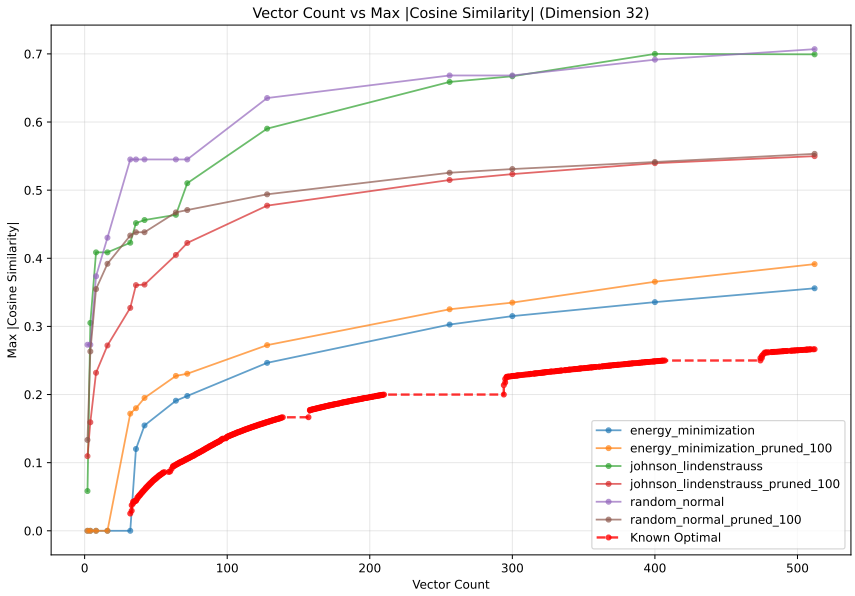

FIGURE 13. All vector counts in dimension 32.

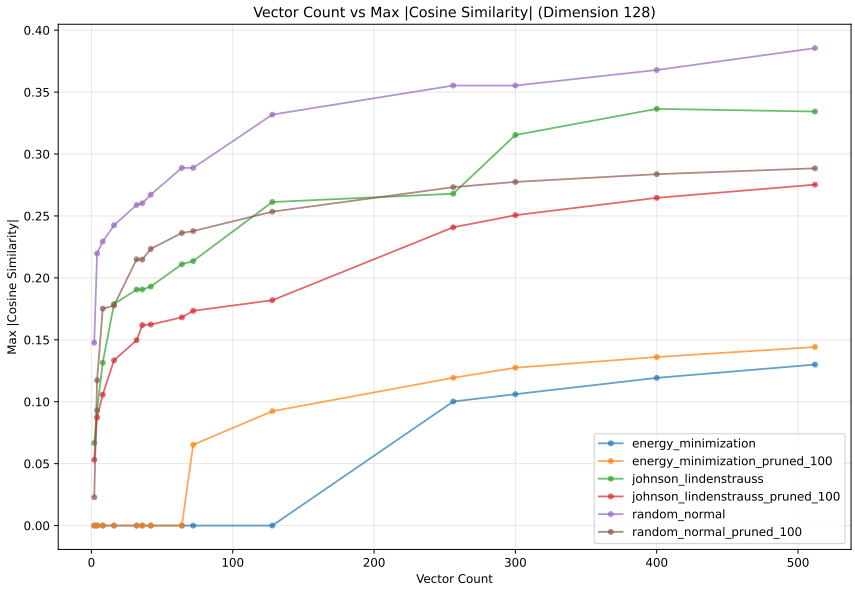

FIGURE 14. All vector counts in dimension 128.

Many of the approaches mentioned here would benefit from a deep dive analysis, and we list some of them in the next subsection.

6.1. **Possible future avenues of study.** We list here some research questions that we found interesting, but did not have time to go more into and are not planning to work on in the near future.

(1) Let's assume that the internal model of an language model has fixed K content directions in $\mathbb{R}^{768}$. Then look at various shapes in $\mathbb{R}^K$,

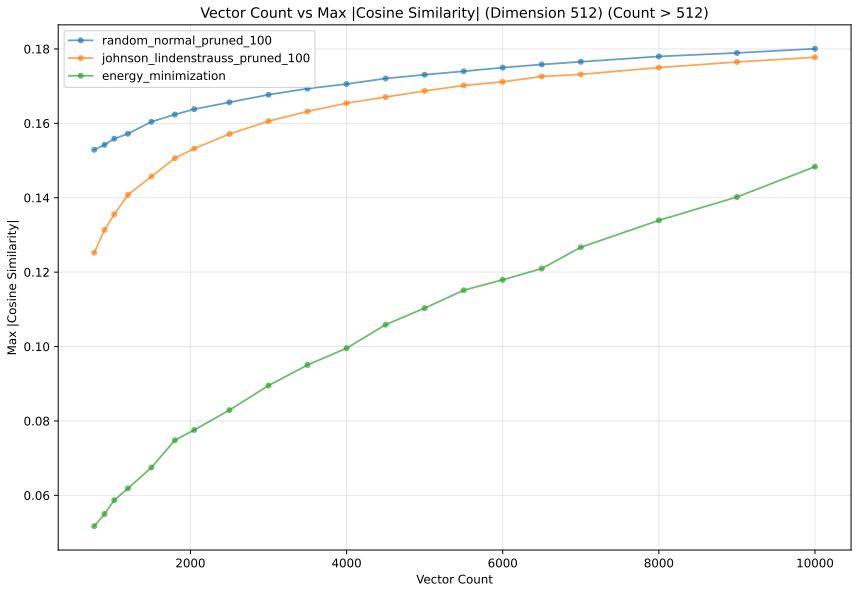

FIGURE 15. Generating more vectors than the dimension in dimension 512.

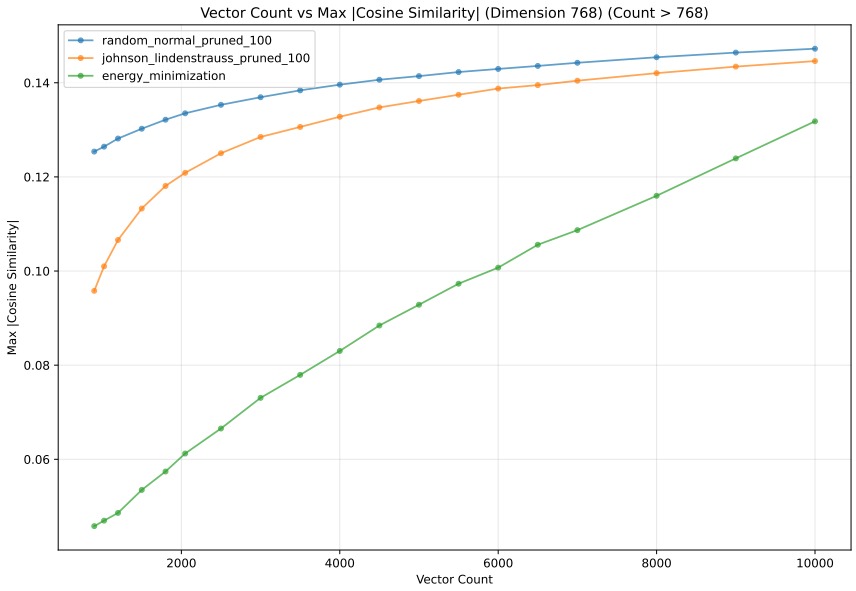

FIGURE 16. Generating more vectors than the dimension in dimension 768.

and map these to $\mathbb{R}^{768}$ in some way that maps the K basiss vectors to K-almost orthogonal vectors. (E.g. Johnson-Lindenstrauss style random projection techniques). Look at the distributions of the cosine similarities, is it common for the distributions to resemble those in Figure 1?

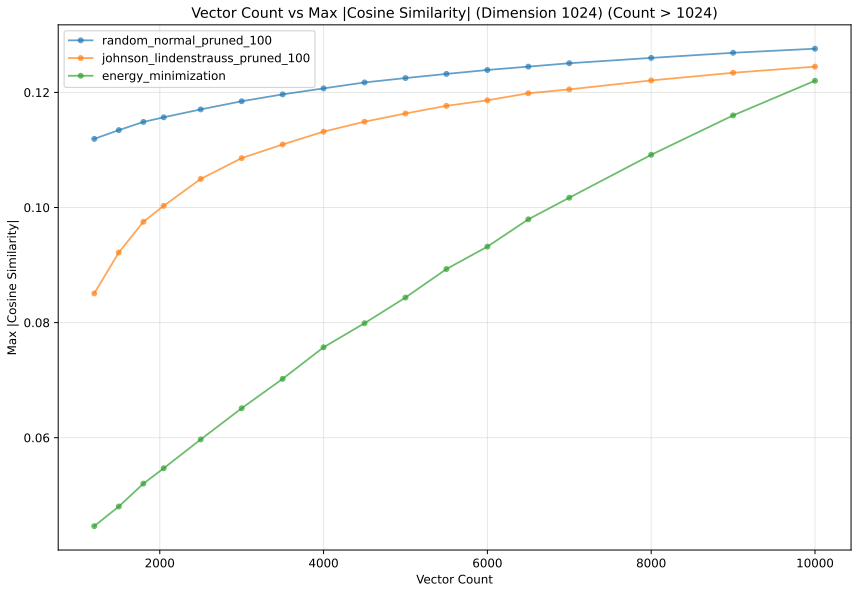

FIGURE 17. Generating more vectors than the dimension in dimension 1024.

$k\backslash d$	**32**	**64**	**128**	**256**	**512**	**768**	**1024**	**2048**
40	0.1184	0	0	0	0	0	0	0
60	0.1637	0	0	0	0	0	0	0
100	0.2092	0.1099	0	0	0	0	0	0
200	0.2577	0.1503	0.0788	0	0	0	0	0
400	0.3110	0.1903	0.1102	0.0564	0	0	0	0
600	0.3416	0.2129	0.1258	0.0706	0.0330	0	0	0
800	0.3713	0.2286	0.1388	0.0786	0.0422	0.0252	0	0
1000	0.3887	0.2407	0.1482	0.0865	0.0472	0.0320	0	0
1500	0.4109	0.2654	0.1654	0.1008	0.0573	0.0410	0.0323	0
2000	0.4241	0.2838	0.1764	0.1075	0.0644	0.0467	0.0376	0
3000	0.4439	0.3116	0.1968	0.1218	0.0749	0.0567	0.0458	0.0316
5000	0.4752	0.3432	0.2219	0.1406	0.0900	0.0673	0.0567	0.0406
8000	0.4976	0.3658	0.2455	0.1604	0.1034	0.0823	0.0707	0.0525

TABLE 7. Best achieved $\max|\cos|$ by (k, d).

(2) Compressed sensing (see e.g. [Mac09] or [Don06]) feels to have some similar ideas of doing high-dimensional analysis in smaller dimensions. In a few brief attempst we didn't find a good connection yet, theoretically or numerically, but we feel there might be something useful in here.
(3) A deeper dive to the ideas behind the Kabatianskii-Levenshtein bound would be interesting. We doubt that the bounds are closer to optimal than our numerical examples, but it would be valuable to know.

References

[AAA+23] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models, 2023. *arXiv preprint arXiv:2311.16867*, 2023.

[Ada60] John Frank Adams. On the non-existence of elements of hopf invariant one. *Annals of Mathematics*, 72(1):20–104, 1960.

[Ant23] Anthropic. Claude 2. `https://www.anthropic.com/news/claude-2`, July 2023. Blog post, Jul. 11, 2023.

[AS48] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1948.

[Axl15] Sheldon Axler. *Linear algebra done right*. Springer, 2015.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[BTB+23] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

[CKG+19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[CKM+17] Henry Cohn, Abhinav Kumar, Stephen Miller, Danylo Radchenko, and Maryna Viazovska. The sphere packing problem in dimension 24. *Annals of mathematics*, 185(3):1017–1033, 2017.

[CND+23] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[Coh16] Henry Cohn. A conceptual breakthrough in sphere packing. *arXiv preprint arXiv:1611.01685*, 2016.

[Coh23] Henry Cohn. Table of spherical codes. `https://spherical-codes.org/`, 2023. Archived at `https://hdl.handle.net/1721.1/153543`.

[CZ14] Henry Cohn and Yufei Zhao. Sphere packing bounds via spherical codes. 2014.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[Don83] Simon K Donaldson. An application of gauge theory to four-dimensional topology. *Journal of Differential Geometry*, 18(2):279–315, 1983.

[Don06] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[Duc24] John C. Duchi. Statistics and information theory (lecture notes). `https://web.stanford.edu/class/stats311/lecture-notes.pdf`, 2024.

[EHO+22] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[Ehr17] Paul Ehrenfest. In what way does it become manifest in the fundamental laws of physics that space has three dimensions. In *Proc. Amsterdam Acad*, volume 20, page 200, 1917.

[Eth19] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[FG16] Carlos Fernandez-Granda. Lecture notes 5: Random projections. `https://cims.nyu.edu/~cfgranda/pages/OBDA_spring16/material/random_projections.pdf`, 2016. Accessed: 2025-08-20.

[Fre82] Michael Hartley Freedman. The topology of four-dimensional manifolds. *Journal of Differential Geometry*, 17(3):357–453, 1982.

[Hal05] Thomas C Hales. A proof of the kepler conjecture. *Annals of mathematics*, pages 1065–1185, 2005.

[Hat05] Allen Hatcher. *Algebraic topology*. 2005.

[HBM+22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[HCH+23] Tom Henighan, Shan Carter, Tristan Hume, Nelson Elhage, Robert Lasenby, Stanislav Fort, Nicholas Schiefer, and Christopher Olah. Superposition, memorization, and double descent. *Transformer Circuits Thread*, 6:24, 2023.

[JL+84] William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[JSR+24] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[LCC+22] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[LCG+19] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[Led01] Michel Ledoux. *The concentration of measure phenomenon*. Number 89. American Mathematical Soc., 2001.

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[LSFA+23] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

[Mac09] Dana Mackenzie. Compressed sensing makes every pixel count. *What's Happening in the Mathematical Sciences*, 7:114–127, 2009.

[Mat13] Jiri Matousek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

[MBV17] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*, 2017.

[Mik13] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[Mus18] Oleg R Musin. Five essays on the geometry of lászló fejes tóth. In *New Trends in Intuitive Geometry*, pages 321–333. Springer, 2018.

[Ope23] OpenAI. Mistral: Efficient super-resolution language models. *arXiv preprint arXiv:2306.09617*, 2023.

[PH23] Sundar Pichai and Demis Hassabis. Introducing gemini: our largest and most capable ai model. `https://blog.google/technology/ai/google-gemini-ai/`, December 2023. Google Keyword blog post, Dec. 6, 2023.

[RBC+21] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[RKR21] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the association for computational linguistics*, 8:842–866, 2021.

[RNS+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[Rol03] Dale Rolfsen. *Knots and links*. Number 346. American Mathematical Soc., 2003.

[RSR+19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[SDCW19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[SWL+19] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[TCM+24] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

[Teg97] Max Tegmark. On the dimensionality of spacetime. *Classical and Quantum Gravity*, 14(4):L69, 1997.

[TLI+23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[TMS+23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[TVWW22] Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

[Ver18] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[Via17] Maryna S Viazovska. The sphere packing problem in dimension 8. *Annals of mathematics*, pages 991–1015, 2017.

[WCC+24] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional

encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*, 2024.

[xT24] xAI Team. Open release of grok-1. `https://x.ai/news/grok-os`, March 2024. Blog post, March 17, 2024.

[YDY+19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Faculty of Information Technology, P.O. Box 35 (Mattilanniemi 2), FI-40014 University of Jyväskylä, Finland and Digital Workforce Services Mechelininkatu 1 a, 00180 Helsinki, Finland and Department of Mathematics and Statistics, P.O. Box 68 (Pietari Kalmin katu 5), FI-00014 University of Helsinki, Finland

Email address: `rami.luisto@gmail.com`