A Coding-Theoretic Analysis of Hyperspherical Prototypical Learning Geometry

Martin Lindström [1] **Borja Rodríguez-Gálvez** [1] **Ragnar Thobaben** [1] **Mikael Skoglund** [1]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

Hyperspherical Prototypical Learning (HPL) is a supervised approach to representation learning that designs class prototypes on the unit hypersphere. The prototypes bias the representations to class separation in a scale invariant and known geometry. Previous approaches to HPL have either of the following shortcomings: (i) they follow an unprincipled optimisation procedure; or (ii) they are theoretically sound, but are constrained to only one possible latent dimension. In this paper, we address both shortcomings. To address (i), we present a principled optimisation procedure whose solution we show is optimal. To address (ii), we construct well-separated prototypes in a wide range of dimensions using linear block codes. Additionally, we give a full characterisation of the optimal prototype placement in terms of achievable and converse bounds, showing that our proposed methods are near-optimal.

GitHub: martinlindstrom/coding_theoretic_hpl

1. Introduction

Representation learning addresses the problem of learning a mapping from a high-dimensional input space to a lower-dimensional representation space subject to suitable inductive biases. These biases are imposed on learning algorithms as additional constraints on, for example, the network architecture or the optimisation algorithm (Goyal & Bengio, 2022). Geometry-based inductive biases have long been popular in representation learning. For instance, imposing unit norm constraints on the representations has been employed in different unsupervised learning methods, either through explicit normalisation or norm-invariant loss functions in variational autoencoders (Davidson et al., 2018; Xu & Durrett, 2018), or in self-supervised learning (Wang & Isola, 2020; Wang et al., 2017; Wu et al., 2018; Chen et al., 2020; Bachman et al., 2019; Caron et al., 2020). Imposing representation separation is another common inductive bias used, for example, in contrastive learning (Wang & Isola, 2020; Rodríguez-Gálvez et al., 2023; Chen et al., 2020; Tian et al., 2020) and supervised representation learning (Khosla et al., 2020; Guerriero et al., 2018; Hasnat et al., 2017).

In the supervised learning setting, one way to impose representation separation is through *prototypical learning* (Snell et al., 2017; Jetley et al., 2015). Each class is assigned a prototype, and these are specified *a priori* to maximise their separation and are held fixed during training, where the algorithm attempts to map input samples to their class prototypes. Therefore, the representations are biased towards being separated based on their class. Recently, Hyperspherical Prototypical Learning (HPL) (Mettes et al., 2019; Kasarla et al., 2022) started imposing unit norm constraints to prototypical learning. This places the representations on the hypersphere and thus enhances representation separation bias in a scale invariant and known geometry.

To illustrate the idea behind HPL, consider the naïve approach of selecting prototypes as the familiar one-hot encoding that, for K classes, picks the canonical basis vectors $\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_K\}$ in dimension K as prototypes. As illustrated in Figure 1 (left), this results in a suboptimal class separation on the hypersphere. Instead, HPL attempts at placing K maximally separated prototypes on the n-dimensional hypersphere $\mathbb{S}^{n-1}$, hopefully with $n < K$. This combinatorial and non-convex problem is well-studied (Conway & Sloane, 1999), but even on $\mathbb{S}^2$ the problem is unsolved for general K, and optimal solutions are only known for $K = 1, \dots, 14$, and 24 (Musin & Tarasov, 2015). Despite this, approximate solutions have been proposed. Mettes et al. (2019) propose a relaxation of the problem that however only achieves suboptimal separation. Kasarla et al. (2022), on the other hand, propose a closed-form solution that we show to be optimal; however, it is only applicable in dimension $n = K - 1$.

[1]Division of Information Science and Engineering, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Martin Lindström <martin12@kth.se>, Ragnar Thobaben <ragnart@kth.se>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at the 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number 251, 2024.

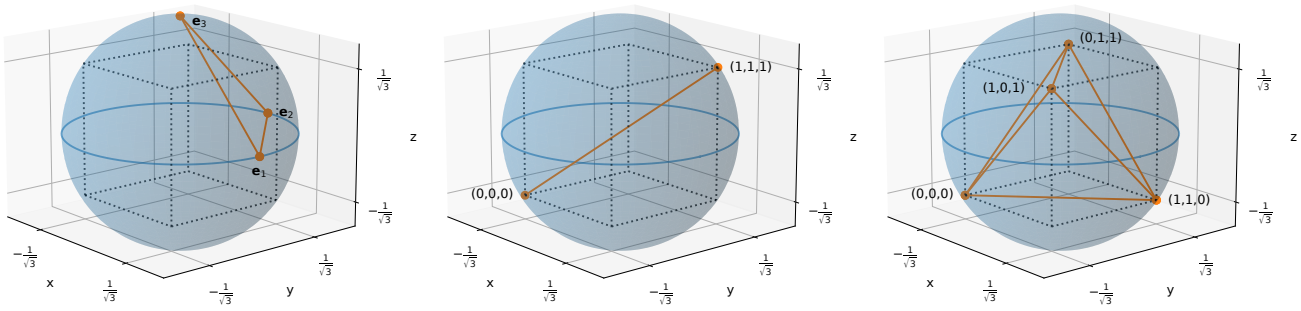

Figure 1. Prototypes on $\mathbb{S}^2$. The **left** image shows the naïve one-hot encoding approach, which has cosine similarity 0. One can do much better with binary code-based prototypes. In the **centre** image, by reducing to two prototypes, the optimal distance with cosine similarity -1 is achieved. The **right** image shows how one can fit four prototypes with a better separation, arriving at a cosine similarity of $-1/3$.

In this paper, we propose two new methods for designing hypershperical prototypes and present sharp bounds on the optimal separation that can be achieved by placing an arbitrary number of prototypes $K \leq 2^n$ on a hypershpere of dimension n. Our approach rests on theory and concepts from error correcting codes, and our contributions are threefold:

(i) We provide a new design approach for hyperspherical prototypes that maps binary linear codes defined over the n-dimensional Hamming space onto the n-dimensional hypersphere $\mathbb{S}^{n-1}$. Our approach provides guarantees on the class separation by design, at the same time that it enables a more flexible trade-off between separation and the dimension n for a given number of classes K.

(ii) We derive a *converse* bound on the guaranteed minimum prototype separation as well as an *achievable* bound that certifies that well-separated code-based prototypes exist. These bounds imply that for a large number of classes K and in high dimensions n, the worst-case cosine similarity converges to zero. The bounds also show that our code-based prototypes closely approach optimal separation for $n \approx K/2$.

(iii) Finally, we provide alternative optimization-based hyperspherical prototypes which achieve the converse bound through a convex relaxation. These improve on the prototypes obtained by Mettes et al. (2019), which do not achieve the converse bound.

The paper is organised as follows. In Section 2, we motivate the connection between binary codes and hyperspherical prototypes and give a primer on the theory of error correcting codes. In Section 3, we apply this theory in order to give both coding-theoretic prototype constructions and bounds thereon. Also in Section 3, we provide a novel optimisation-based prototype scheme. The performance of the proposed schemes is evaluated and compared to the state of the art in HPL in Section 4, and Section 5 concludes the paper.

2. Background

We start with a formal problem formulation for designing a codebook of hyperspherical prototypes in the HPL setting. Then, we connect binary error correcting codes defined in the Hamming space to hyperspherical prototypes. After that, we provide a brief overview of fundamental concepts in coding theory that are used in this paper to derive code-based hyperspherical prototypes with good separation properties. The interested reader is encouraged to consult MacWilliams & Sloane (1977) for a more detailed treatment.

2.1. Problem Formulation

We consider the HPL setting with K classes and n dimensions; that is, we are interested in placing K prototypes $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$ on the n-dimensional unit hypersphere $\mathbb{S}^{n-1}$, where the dimension n is a hyperparameter. Our objective is maxmising the Euclidean distance between every pair of prototypes $\boldsymbol{c}_i, \boldsymbol{c}_j \in \mathbb{S}^{n-1}$ $(i \neq j)$. Clearly, the Euclidean distance $d_{\mathrm{E}}(\boldsymbol{c}_i, \boldsymbol{c}_j)$ is bounded in the range $[0, 2]$ and satisfies that $\|\boldsymbol{c}_i - \boldsymbol{c}_j\|^2 = 2 - 2\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle$ for all $\boldsymbol{c}_i, \boldsymbol{c}_j \in \mathbb{S}^{n-1}$. Hence, maximising the Euclidean distance is equivalent to minimising the cosine similarity $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle$. In turn, this is equivalent to maximising the angle α between $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ since $\alpha = \arccos \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle$. Therefore, we will use the cosine similarity as a notion of separation throughout this paper. The objective in HPL is then designing a codebook $\mathcal{C} := \{\boldsymbol{c}_i \in \mathbb{S}^{n-1} : i = 1, \ldots, K\}$ of K well-separated hyperspherical prototypes, which can be summarized in the following optimisation problem:

$$\min_{\mathcal{C}} \max_{i \neq j} \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle. \tag{P}$$

Unfortunately, this problem is both non-convex due to the unit norm constraint, and combinatorial due to the search of the worst pair of prototypes $\boldsymbol{c}_i, \boldsymbol{c}_j$ $(i \neq j)$, requiring tractable relaxations that yield approximate solutions.

2.2. Connecting Codes to Protypes

The approach in this paper leverages coding theory to design n-dimensional binary vectors (that is, members of the n-dimensional Hamming space), which are mapped onto the n-dimensional hypersphere $\mathbb{S}^{n-1}$, thereby creating prototypes with good separation. To provide some intuition as to how error correcting codes relate to placing prototypes that are maximally spaced apart, consider the mapping $\pi : \{0,1\}^n \to \mathbb{S}^{n-1}$ that maps n-dimensional binary vectors $\boldsymbol{b}$ from the n-dimensional Hamming space to points $\boldsymbol{c}$ on the hypersphere. More precisely, the mapping is defined as

$$\boldsymbol{c} = \pi(\boldsymbol{b}) := \frac{2(\boldsymbol{b} - 1/2)}{\sqrt{n}} \tag{1}$$

and enforces that $\boldsymbol{c}(\ell) \in \{-1/\sqrt{n}, +1/\sqrt{n}\}$ and $\|\boldsymbol{c}\|^2 = 1$. This approach allows us to place 2^n points on the unit hypersphere $\mathbb{S}^{n-1}$ with a cosine similarity of at most $\langle \boldsymbol{c}, \boldsymbol{c}' \rangle \leq 1 - 2/n$ for every pair $\boldsymbol{c} = \pi(\boldsymbol{b})$ and $\boldsymbol{c}' = \pi(\boldsymbol{b}')$ with $\boldsymbol{b} \neq \boldsymbol{b}'$. For $K < 2^n$, we can improve the separation guarantees by carefully selecting the K binary vectors placed on the hypersphere via the mapping π. Error correcting codes provide a systematic way to achieve this, and the concept is illustrated for $n = 3$ in Figure 1.

As a baseline, consider one-hot encoding (Figure 1, left), which provides $K = 3$ orthogonal prototypes. Using carefully selected corners of the unit cube and the mapping π, we can substantially improve on one-hot encoding. Reducing to $K = 2$ prototypes (Figure 1, centre), the unit cube corners corresponding to $\boldsymbol{b}_1 = (0,0,0)$ and $\boldsymbol{b}_2 = (1,1,1)$ are diametrically opposed with a cosine similarity of -1, which is optimal. From a coding theory perspective, this corresponds to a repetition code with *Hamming distance* $d_{\mathrm{H}}(\boldsymbol{b}_1, \boldsymbol{b}_2) = 3$ between codewords; that is, the codewords differ in 3 bits. Increasing to $K = 4$ prototypes (Figure 1, right), the unit cube corners corresponding to $\boldsymbol{b}_1 = (0,0,0)$, $\boldsymbol{b}_2 = (0,1,1)$, $\boldsymbol{b}_3 = (1,0,1)$, and $\boldsymbol{b}_4 = (1,1,0)$ have a mutual cosine similarity of $-1/3$, which is an improvement over one-hot encoding (Figure 1, left) while increasing the number of classes at the same time. Again, the set $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_4\}$ constitutes a binary linear code with Hamming distance $d_{\mathrm{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j) = 2$ between its codewords.

This example demonstrates that due to the definition of the mapping function π, there exists a relation between the separation of vectors $\boldsymbol{c} = \pi(\boldsymbol{b})$ and $\boldsymbol{c}' = \pi(\boldsymbol{b}')$ on the unit hypersphere and the Hamming distance $d_{\mathrm{H}}(\boldsymbol{b}, \boldsymbol{b}')$ of the binary vectors $\boldsymbol{b}$ and $\boldsymbol{b}'$ in the sense that a large Hamming distance implies a large separation. We will make this result explicit in Section 3.1. Error correcting codes, designed to have a large Hamming distance $d_{\mathrm{H}}(\boldsymbol{b}, \boldsymbol{b}')$ between every pair of codewords $\boldsymbol{b}$ and $\boldsymbol{b}'$, are hence a well suited tool for designing hyperspherical prototypes, and coding theory provides us with useful bounds on the achievable separation.

2.3. A Primer on Coding Theory

The systematic study of error correcting codes dates back to the seminal work of Hamming (1950). By introducing redundancy in a structured way, error correcting codes allow for error detection and correction in messages and data, and are essential for guaranteeing the reliability of today's digital communication, computation, and storage systems. Linear codes defined over the Galois field $\mathrm{GF}(q)$, where $q = p^m$ and p is a prime, are of special interest as they offer a structure that can be used for efficient encoding, decoding, and analysis of the distance properties of the code.

In this paper, we mainly restrict ourselves to binary linear block codes (that is, $q = p = 2$), and only briefly discuss extensions to q-ary codes with $q > 2$. A binary block code with parameters $[n, k]$ is specified by a codebook $\mathcal{B}$ of 2^k binary codewords of length n and a bijective encoder mapping that maps the set of all length-k binary vectors into the code $\mathcal{B}$. This adds $n - k$ bits of redundancy, which we can use to detect and correct errors in the codeword. The *rate* of the code is defined as $R = k/n$, where a low rate corresponds to a high redundancy. The error detection and correction capabilities rely on the *separation* of codeword pairs $\boldsymbol{b}_i, \boldsymbol{b}_j \in \mathcal{B}$ in Hamming distance, namely

$$d_{\mathrm{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j) := \sum_{\ell=1}^{n} \mathbb{I}\{\boldsymbol{b}_i(\ell) \neq \boldsymbol{b}_j(\ell)\}.$$

A binary code with minimum Hamming distance

$$d_{\min} := \min_{\boldsymbol{b}_i, \boldsymbol{b}_j \in \mathcal{B}, i \neq j} d_{\mathrm{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j)$$

can detect $d_{\min} - 1$ errors and correct $\lfloor (d_{\min} - 1)/2 \rfloor$ errors.

In this paper, we fix $k = \lceil \log_2(K) \rceil$ given K classes, suggesting that we are in the low-rate or high-redundancy regime if n is of the order of K. Noting that low-rate codes offer large separation in terms of minimum Hamming distance, we can expect good separation by adopting a code-based approach. To this end, a fundamental result in coding theory is that *good codes exist*. This is formalised by the well-known Gilbert-Varshamov bound (MacWilliams & Sloane, 1977, Chapter 1, Theorem 12).

Lemma 2.1 (Gilbert-Varshamov Bound)**.** *There exists an* $[n, k]$ *code with minimum distance at least* $d_{\min}$*, provided that*

$$2^{n-k} > \sum_{i=0}^{d_{\min}-2} \binom{n-1}{i}. \tag{2}$$

The Gilbert-Varshamov bound for the largest $d_{\min}$ gives a lower bound on $d_{\min}$. However, the bound only guarantees that good codes *exist*, but not how to find them. Luckily, as we will show in Section 3, several linear binary codes with better minimum distance exist.

Remark 2.2. To derive some of our results, the bound in (2) needs to be evaluated carefully in order to avoid overflow problems. Notice that the bound can be rewritten as

$$\begin{aligned} 2^{-k} &> \sum_{i=0}^{d_{\min}-2} \binom{n-1}{i} \left(\frac{1}{2}\right)^{i} \left(\frac{1}{2}\right)^{n-i} \\ &= F_{\text{bin}}\left(d_{\min}-2; n-1, \tfrac{1}{2}\right), \end{aligned}$$

where $F_{\text{bin}}(\cdot\,; n-1, 1/2)$ denotes the cumulative distribution function of a binomial distribution with $n-1$ trials with success probability $1/2$. This is implemented in a numerically stable manner in many computational libraries.

3. Hyperspherical Prototype Design

In this section, we present our main contributions towards solving the optimization problem (P). As mentioned earlier, this problem is both non-convex due to the unit norm constraint, and combinatorial due to the search of the worst pair of prototypes $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ with $i \neq j$. Our contributions are focused on relaxations to the problem (P) and bounds on the optimal solution. Section 3.1 formalises the coding-theoretic approach introduced with the example in Figure 1; Section 3.2 uses coding-theoretic tools to bound the optimal solution to (P); and Section 3.4 presents a relaxation to (P) which achieves the bound on the optimal solution.

3.1. Coding-Theoretic Prototypes

We begin by formalising the intuition provided in Section 2.2, namely that a binary $[n, k]$ code with the codebook $\mathcal{B}$ and a large minimum distance $d_{\min}$ gives good prototypes. Firstly, notice that if we want K prototypes, then we need $|\mathcal{B}| = 2^k \geq K$. Additionally, recall that the mapping $\boldsymbol{c} = \pi(\boldsymbol{b})$ defined in (1) produces a unit-norm vector for any binary vector $\boldsymbol{b}$. Then, we can guarantee that binary code-based constructions guarantee good separation.

Proposition 3.1. *Assume that $\mathcal{B}$ is the codebook of a binary $[n, k]$ code with minimum distance $d_{\min}$. Then, for every pair $\boldsymbol{b}_i, \boldsymbol{b}_j \in \mathcal{B}$ with $i \neq j$, the cosine similarity between $\boldsymbol{c}_i = \pi(\boldsymbol{b}_i)$ and $\boldsymbol{c}_j = \pi(\boldsymbol{b}_j)$ is upper bounded by*

$$\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle = 1 - \frac{2d_{\text{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j)}{n} \leq 1 - \frac{2d_{\min}}{n}.$$

Proof. To show the bound on the cosine similarity, notice that two binary vectors differing in $d_{\text{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j)$ positions obey that $\sum_{\ell=1}^{n} (\boldsymbol{b}_i(\ell) - \boldsymbol{b}_j(\ell))^2 = d_{\text{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j)$. Expanding $\|\boldsymbol{c}_i - \boldsymbol{c}_j\|^2$ in two ways gives

$$\begin{aligned} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|^2 &= 2 - 2\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \\ &= \sum_{\ell=1}^{n} \left(\frac{2(\boldsymbol{b}_i(\ell) - 1/2) - 2(\boldsymbol{b}_j(\ell) - 1/2)}{\sqrt{n}} \right)^2. \end{aligned}$$

Rearranging this and recalling that $d_{\text{H}}(\boldsymbol{b}_i, \boldsymbol{b}_j) \geq d_{\min}$ yields the desired result. □

Since good codes exist (see Lemma 2.1), we provide two examples of binary codes whose minimum distance is close to $d_{\min} = n/2$ in dimensions where $n < K$. That is, there exist no worse than orthogonal prototypes with zero worst-case cosine similarity in dimensions $n \approx K/2$. Additionally, these codes are easy to implement in Python, and hence are easily integrable in modern machine learning software. The codes are the Bose–Chaudhuri–Hocquenghem (BCH) and Reed–Muller (RM) codes. Other code families like low-density parity-check (LDPC) codes and sparse graph codes are not competitive in terms of minimum distance guarantees since their minimum distance is usually lower than the one predicted by the Gilbert-Varshamov bound, see *e.g.*, Mitchell et al. (2015, Figure 9). These popular code families are hence not further considered in this paper. Polar codes, on the other hand, belong to the same code family as RM codes, see *e.g.*, Abbe et al. (2021, Section IV-D), and are hence implicitly covered.

Prototypes from BCH Codes BCH codes are known to have good minimum distance in low dimensions (MacWilliams & Sloane, 1977, pp. 258), and although the exact minimum distance is not known in general (Li, 2017), we find empirically that it approaches $d_{\min} = n/2$ in dimensions $n < K$. They are implemented in the `Galois` Python library (Hostetter, 2020, v0.3.8).

Prototypes from RM Codes The distance properties of RM codes are easier to characterise. Their construction is simple and gives straight-forward distance guarantees (Abbe et al., 2021). In fact, they provide no worse than orthogonal prototypes in dimension $n \approx K/2$.

Lemma 3.2 (Separation Guarantees for RM Codes)**.** *Let $\widetilde{K}$ be the smallest power of 2 such that $\widetilde{K} \geq K$. Then, RM codes in dimension $n \geq \widetilde{K}/2$ have minimum distance $d_{\min} = n/2$ and guarantee that $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 0$.*

Proof. By construction, RM codes are $[n, k]$ codes with $n = 2^m$, $k = \sum_{i=0}^{r} \binom{m}{i}$, and minimum distance $d_{\min} = 2^{m-r}$. Hence, we have cosine similarity $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 0$ if $r = 1$, namely if $1 + m = 1 + \log_2 n = k \geq \log_2 \widetilde{K} \geq \log_2 K$ or if $n \geq \widetilde{K}/2 \geq K/2$. □

Remark 3.3. It is important to note that the cosine similarity guarantee $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 0$ is not equivalent to pairwaise orthogonality. However, every codeword is *locally* orthogonal to all its minimum-distance neighbours and has no worse than orthogonal separation *globally*. We investigate the global cosine similarity distribution for all prototype generation schemes in Figure 3.

Realisable Dimensions with Codes As has been shown, binary codes provide a flexible way to derive prototypes. However, there is a restriction on the dimensions which are realisable: RM codes are defined for $n = 2^m$, and BCH codes are defined for $n = 2^m - 1$ for every $m \in \mathbb{N}_+$. Moreover, additional dimensions are realisable: codes can be *punctured* by removing dimensions, thereby creating a lower-dimensional code, and they can be *extended* by adding more dimensions (MacWilliams & Sloane, 1977, Chapter 1, §9). In general, puncturing a code by 1 bit will reduce its minimum distance by 1. Similarly, extending a code can (but is not guaranteed to) increase the minimum distance. Therefore, RM and BCH codes can have good distance properties around dimensions $n = 2^m$ and $n = 2^m - 1$, respectively, but not for general n. Compared to Kasarla et al. (2022), which is only valid in $n = K - 1$, coding-based prototypes hence improve flexibility by guaranteeing good separation for a larger set of admissible dimensions.

3.2. Coding-Theoretic Bounds

In this section, we provide both upper and lower bounds the worst-case cosine similarity of hyperspherical prototypes in (P). Our achievable (upper) bound is based on Lemma 2.1, which states that good binary codes exist. Our converse (lower) bound is based on results from spherical coding theory, which shows that *the minimum separation cannot be improved beyond near orthogonality*. We begin by recalling the Rankin bound from spherical coding theory (Ericson & Zinoviev, 2001, Theorem 1.4.1).

Lemma 3.4 (Rankin Bound)**.** *Any set of K hyperspherical prototypes $\mathcal{C}$ satisfies that*

$$\max_{\mathcal{C}} \min_{i \neq j} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|^2 \leq \frac{2K}{K-1}.$$

Now, we may state the achievable and converse bounds.

Theorem 3.5. *There exists a set of hyperspherical prototypes $\mathcal{C}$ with cosine similarity at most (separation at least)*

$$\max_{i \neq j} \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 1 - \frac{2d_{GV}}{n}, \tag{3}$$

where d_{GV} denotes the largest solution to the Gilbert-Varshamov bound in (2). Conversely, no set of prototypes exists with maximum cosine similarity smaller (better separation) than

$$\max_{i \neq j} \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \geq \frac{-1}{K-1}. \tag{4}$$

Proof. The achievable bound (3) follows directly from combining Proposition 3.1 and Lemma 2.1. For the converse bound, recalling that $\|\boldsymbol{c}_i - \boldsymbol{c}_j\|^2 = 2 - 2\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle$, applying Lemma 3.4, and simplifying yields (4). □

The bounds are numerically evaluated in Section 4. A number of remarks are in order. In many practical settings, the converse bound $-1/(K-1)$ is close to 0. It is therefore impossible to achieve a maximum cosine similarity that is notably better than one-hot encoding. However, as Lemma 3.2 shows, it is possible to have no worse than orthogonal prototypes in low dimension $n = \tilde{K}/2$. Moreover, as we will show with numerical examples, it is possible to have approximately orthogonal prototypes in much lower dimension than $n = K$. Finally, we note that the upper bound can be tightened for $n \geq K$ by recalling one-hot encoding. The bounds are therefore sharp, meaning that orthogonal prototypes are achievable and near-optimal for a large number of classes K. Finally, it is interesting to note that the mapping proposed by Kasarla et al. (2022) is *optimal* since it achieves the converse bound.

3.3. Beyond Binary Codes

In this section, we briefly discuss the generalisation of our results to the case of q-ary codes and motivate the choice of restricting our attention to binary codes.

Assume a construction that combines an $[n_q, k_q]$ code $\mathcal{U}$ over the Galois field GF(q), with minimum Hamming distance $d^{(\mathcal{U})}_{\mathrm{H,min}}$, with a mapping $\pi^{(l)}_q$ that maps q-ary symbols $u \in \{0, \ldots, q-1\}$ to points $\tilde{\boldsymbol{c}} = \pi^{(l)}_q(u)$ on the l-dimensional unit hypersphere $\mathbb{S}^{l-1}$, and with pairwise Euclidean distance of at least $d^{(\pi^{(l)}_q)}_{\mathrm{E,min}}$. Then, an n_q-dimensional q-ary vector $\boldsymbol{u}$ can be mapped to the unit hypersphere $\mathbb{S}^{n-1}$ in $n = n_q \cdot l$ dimensions by realising the mapping

$$\boldsymbol{c} = \pi_q(\boldsymbol{u}) \coloneqq \frac{1}{\sqrt{n_q}} \left(\pi^{(l)}_q(\boldsymbol{u}(1)), \ldots, \pi^{(l)}_q(\boldsymbol{u}(n_q)) \right). \tag{5}$$

Then, similarly to the binary case, by a proper choice of the code parameters, hyperspherical prototypes with good separation guarantees can be obtained.

Proposition 3.6. *Assume $\mathcal{U}$ is the codebook of a q-ary $[k_q, n_q]$ code with a minimum Hamming distance $d^{(\mathcal{U})}_{\mathrm{H,min}}$ that is mapped into the unit hyperspehere $\mathbb{S}^{n-1}$ in $n = n_q \cdot l$ dimensions with the mapping π_q from (5). Furthermore, let $d^{(\pi^{(l)}_q)}_{\mathrm{E,min}}$ be the minimum Euclidean distance achieved by the component mapping $\pi^{(l)}_q$. Then, for every codeword pair $\boldsymbol{u}_i, \boldsymbol{u}_j \in \mathcal{U}$ with $i \neq j$, the cosine similarity between $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ is upper bounded by*

$$\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 1 - \frac{d^{(\mathcal{U})}_{\mathrm{H,min}}}{n_q} \frac{\left[d^{(\pi^{(l)}_q)}_{\mathrm{E,min}} \right]^2}{2}.$$

Proof. The proof follows along the same lines as the proof of Proposition 3.1. □

Assume we want to minimise the upper bound on the cosine similarity. We then want to find a q-ary code with as large minimum distance as possible. The Singleton bound (MacWilliams & Sloane, 1977, Chapter 1, Theorem 11) states that every $[n_q, k_q]$ linear code has minimum distance $d_{\min} \leq n_q - k_q + 1$. Reed-Solomon (RS) codes with parameters $\log_q K \leq k_q \leq n_q \leq q$ achieve the Singleton bound with equality, and the only binary code achieving the Singleton bound is the repetition code (MacWilliams & Sloane, 1977, Chapter 11) (see Figure 1, middle and Section 2.2).

Consider now combining RS codes with the Kasarla et al. (2022) mapping, which has $\left[d_{\mathrm{E,min}}^{(\pi_q^{(l)})}\right]^2 = 2q/(q-1)$. Then, the cosine similarity for this prototype construction is guaranteed to be upper bounded by

$$\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 1 - \frac{q}{q-1} \cdot \frac{n_q - k_q + 1}{n_q}.$$

From this result, it follows that the cosine similarity of this construction becomes strictly negative if, and only if $k_q < n_q/q + 1 \leq 2$, where the second inequality comes from the requirement $n_q \leq q$ on the length of RS codes. That is, RS codes only guarantee a strictly negative cosine similarity for $k_q = 1$, given that $q \geq K$. However, in that case, the mapping by Kasarla et al. (2022) already guarantees the optimal separation, and there is no benefit by further extending the dimensions beyond $n_q = 1$ with an additional code. For $k_q = 2$, $n_q = q$, and under the condition $q^2 \geq K$, we can guarantee a cosine similarity $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 0$ for $n = (q-1) \cdot q$ dimensions. In the favourable case where $q = 2^m$ and $K = 2^{2m}$, the construction achieves $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle \leq 0$ for $n = 2^{2m} - 2^m = K - \sqrt{K}$, which for $m > 1$ is larger than $n = 2^{2m-1} = K/2$. $n = K/2$ is however obtained by the RM-code-based construction as demonstrated in Lemma 3.2. Hence, there is no benefit employing RS codes in conjunction with the mapping by Kasarla et al. (2022) compared to the RM-code-based construction.

3.4. Optimisation-Based Prototypes

In this section, we compare numerical approaches to approximately solve the non-convex and combinatorial problem (P). Throughout, we employ projected gradient descent to deal with the non-convexity introduced by the unit norm constraint, and compare different relaxations to the combinatorial part of the problem.

Minimising the Average Worst-Case Similarity Mettes et al. (2019) note that solving the combinatorial minimisation in (P) is numerically inefficient. Instead, they minimise the average maximum cosine similarity per prototype. More specifically, they define the matrix of prototypes $\boldsymbol{C} := [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K] \in \mathbb{R}^{n \times K}$ describing the codebook $\mathcal{C}$ and propose the problem

$$\begin{aligned} \min_{\boldsymbol{C}} \quad & \frac{1}{K} \sum_{i=1}^{K} \max_j \boldsymbol{M}_{i,j}, \\ \text{s.t.} \quad & \boldsymbol{M} = \boldsymbol{C}^\mathsf{T} \boldsymbol{C} - 2\boldsymbol{I}, \\ & \|\boldsymbol{c}_i\| = 1, \end{aligned} \tag{P$_{\mathrm{AVG}}$}$$

where $\boldsymbol{M}_{i,j}$ denotes the (i,j)-th element of the matrix $\boldsymbol{M}$ and $\boldsymbol{I}$ denotes the identity matrix. Since the diagonal elements of $\boldsymbol{C}^\mathsf{T}\boldsymbol{C}$ are always 1, subtracting twice the identity matrix avoids selecting these. The improvement over the original problem (P) is that multiple prototypes are updated at each gradient step, which improves the convergence speed. However, no proof of convergence or optimality is presented.

Log-Sum-Exp Relaxation We propose a convex relaxation to the combinatorial problem, which we show numerically that it closely approximates the converse bound in Theorem 3.5. Specifically, we propose to use the log-sum-exp approximation of the maximum. It is folklore knowledge that for $\boldsymbol{x} \in \mathbb{R}^n$ we have

$$\max_i \boldsymbol{x}_i \leq \frac{1}{t} \log \left(\sum_{i=1}^{n} \exp(t \boldsymbol{x}_i) \right) \leq \max_i \boldsymbol{x}_i + \frac{\log n}{t},$$

for any temperature $t > 0$. Moreover, the function is convex. For a large temperature t, this problem approaches the original problem (P). By carefully choosing a scheduler for the temperature, we are able to balance the need to update multiple prototypes, and to approximate the original problem. Hence, we propose the problem

$$\min_{\mathcal{C}} \frac{1}{t} \log \sum_{i \neq j} \exp(t \langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle), \tag{P$_{\mathrm{LSE}}$}$$

which can be rewritten as a sum over the upper (or lower) triangular part of $\exp(t\boldsymbol{C}^\mathsf{T}\boldsymbol{C})$, excluding the diagonal.

3.5. Computational Complexity

In this section, we comment on the computational complexity of the prototype generation schemes in order to give a complete characterisation of the methods. We note however that the computational complexity of generating prototypes is negligible in comparison to network training.

Optimisation-Based Prototypes The optimisation-based prototypes from (P$_{\mathrm{LSE}}$) and (P$_{\mathrm{AVG}}$) both require $\mathcal{O}(nK^2)$ operations per gradient step, since all the K^2 inner products $\langle \boldsymbol{c}_i, \boldsymbol{c}_j \rangle$ of n-dimensional vectors need to be calculated. In practice, the wall-clock time spent on these calculations is small: Even on a laptop, the computations take on the order of seconds even for $K = 1000$ and $n \approx K$.

Coding-Theoretic Prototypes Very efficient implementations of error correcting codes exist: They are implemented and run in real time on billions of light-weight wireless devices. Since the codebooks are fixed, they need only be computed once, and can later be re-used across runs. For BCH codes, tables of their so called *generator polynomials* are available, and with those, all the K codewords can be enumerated quickly with a complexity of $\mathcal{O}(nK\log(K))$ operations. As an example, the generator polynomials of BCH codes are tabulated up to $n = 2^{10} - 1 = 1023$ in Lin & Costello (2004, Appendix C), and up to $n = 2^{16} - 1 = 65\,535$ in the MATLAB function `bchgenpoly` (The MathWorks Inc., 2024). For RM codes, due to their simple structure, it is fast to generate the entire codebook, again with a complexity of $\mathcal{O}(nK\log(K))$ operations. This is done in fractions of a second even for $K = 1000$ and $n \approx K$ on a laptop.

Kasarla et al. (2022) Prototypes The prototypes from Kasarla et al. (2022) also take less than a second to generate on a laptop. However, their implementation is recursive, and therefore requires increasing the recursion limit for large K. Similar to the coding-theoretic prototypes, they can be pre-computed and re-used across runs.

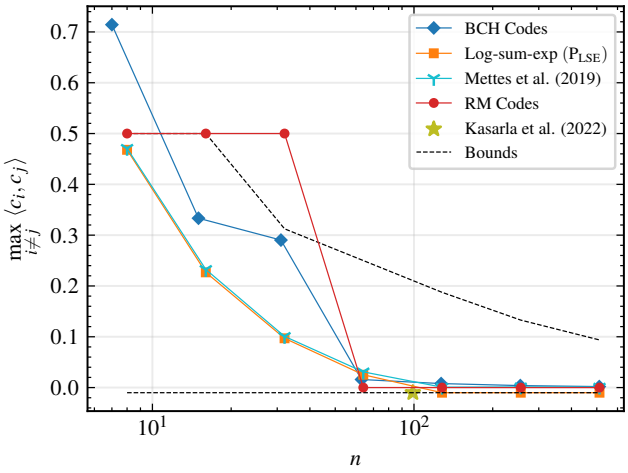

Figure 2. Maximum pairwise cosine similarity of $K = 100$ prototypes in various latent space dimensions (logarithmic scale). Coding-theoretic approaches provide additional flexibility over the Kasarla et al. (2022) mapping. The optimisation-based (P_{LSE}) prototypes achieve slightly better separation than the (P_{AVG}) scheme. All schemes (except for RM codes with $n = 32$) fall within the achievable and converse bounds from Theorem 3.5. For large n, the BCH, RM, and (P_{AVG}) prototypes yield no worse than othogonal prototypes. The (P_{LSE}) and Kasarla et al. (2022) prototypes achieve the converse bound and perform therefore slightly better.

4. Experiments

In this section, we evaluate the separation in terms of maximum cosine similarity for the considered prototype generation schemes and present numerical results on CIFAR-100 (Krizhevsky, 2009). We also present supplementary results on MNIST (LeCun et al., 1998) in Appendix C. We emphasise that our aim with these experiments is to investigate the relation between prototype separation and performance, and not necessarily to find the best performing realisations of the algorithms.

4.1. Experimental Setup

For optimisation-based prototypes, we follow Mettes et al. (2019) and use stochastic gradient descent (SGD) with learning rate 0.1 and momentum 0.9 over 1 000 epochs. For the log-sum-exp prototypes, we scale the temperature linearly with epochs from 1 to K. For CIFAR-100, we use a ResNet-34 backbone (He et al., 2016) as implemented by Kasarla et al. (2022), and we also use the same hyperparameters (SGD with a cosine annealing learning rate scheduler, learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, and batch size 512 over 200 epochs), with standard data augmentation schemes (random 32×32 crops with padding 4, random horizontal flips with probability $1/2$, and random rotations with up to 15°). We use cross-entropy loss on the cosine similarities $\boldsymbol{C}^\mathsf{T}\boldsymbol{z}$ between the prototypes $\boldsymbol{C}$ and the output $\boldsymbol{z}$ from the ResNet-34 backbone. At test time, classification is done via nearest-neighbour decoding, or equivalently, maximum cosine similarity decoding, by choosing the class of the nearest prototype. All our results are averaged over 5 runs, and we use a randomised validation set (20 % of the training set) for every run. For MNIST, we use the lightweight network proposed by Simard et al. (2003), with the same hyperparameters and data augmentations as for CIFAR-100, except that we rotate by up to 30° (instead of 15°) and do not flip horizontally.

Some additional remarks on prototype generation are in order. For BCH and RM codes, the mapping between class and prototype is fixed, while the optimisation-based mappings from (P_{LSE}) and (P_{AVG}) randomises the assignment across different runs. Therefore, for a fair comparison for BCH and RM codes, we both average over different class to prototype mappings, as well as over a fixed class to prototype mapping. Note that for both one-hot encoding and the Kasarla et al. (2022) mapping, the mutual cosine similarity is constant for all prototype pairs, and hence the class to prototype mapping does not matter.

4.2. Prototype Separation Guarantees

We evaluate the achieved prototype separation in terms of maximum cosine similarity over the dimension n for $K \in \{10, 100, 1\,000\}$ classes. The results for $K = 100$, corresponding to the models trained on CIFAR-100 in Sec-

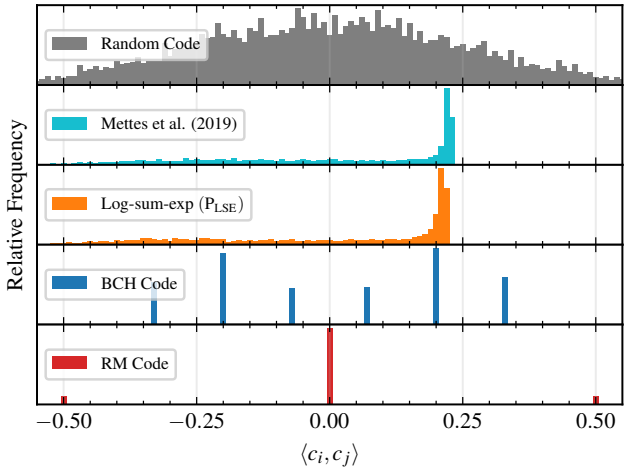

Figure 3. Cosine similarity histograms for different prototype generation schemes for $K = 100$ classes in dimension $n = 16$. All schemes have average cosine similarity close to 0. Note that there are $K/2$ RM code-based prototype pairs with cosine similarity -1 which have been omitted for clarity.

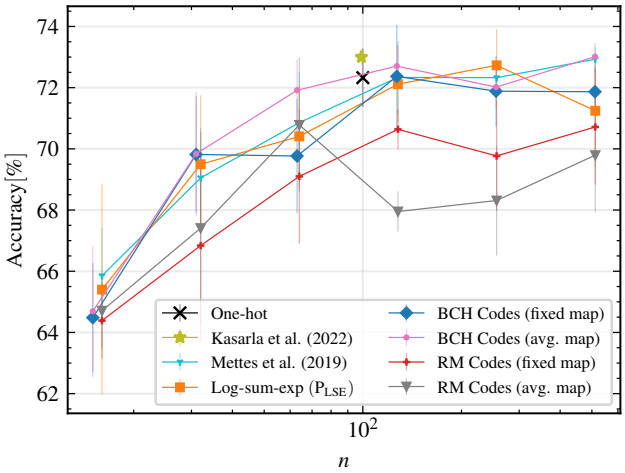

Figure 4. Top-1 accuracy results for CIFAR-100 for different prototype generation schemes, averaged over 5 runs, with errorbars corresponding to one standard deviation.

tion 4.3, are presented in Figure 2 along with the achievable and converse bounds from Theorem 3.5. Plots for $K = 10$ and $K = 1\,000$ classes are provided in Appendix A.

For $K = 100$, the results confirm our theoretical analysis in Sections 3.1 and 3.2. For $n = 99$, the mapping by Kasarla et al. (2022) achieves the lower bound as expected. RM codes achieve zero worst-case cosine similarity for $n \geq 64$, and the maximum cosine similarity achieved by BCH codes for $n \geq 63$ is slightly above zero. That is, the code-based designs give close-to-optimal separation guarantees with only approximately half the number of dimensions. Below $n = 63$, the optimisation-based schemes outperform the code-based designs, where the proposed log-sum-exp relaxation gives a slight advantage over Mettes et al. (2019). For fewer classes ($K = 10$, see Appendix A), the optimisation-based methods outperform coding-based approaches, and solving the proposed relaxation (P_{LSE}) instead of (P_{AVG}) results in a big improvement. For a large number of classes ($K = 1\,000$, see Appendix A), the optimisation-based methods perform poorly, and coding-theoretic methods guarantee better separation in lower dimensions $n < K$. We therefore conclude that code-based prototypes are beneficial if the number of classes is large, in which case the achievable dimension compression also becomes an attractive feature.

To provide further insights, we provide histograms of the cosine similarities for the different prototype schemes in Figure 3. We compare the schemes for $K = 100$ classes in dimension $n = 16$ and, for the sake of illustration, include prototypes created uniformly at random as a baseline. As the histograms show, the optimisation moves probability mass from the right tails towards lower cosine similarity, where the log-sum-exp relaxation achieves a slightly lower maximum cosine similarity. For the coding-based methods, the cosine similarities concentrate in a few different values which can be directly calculated from the weight distribution (or weight-enumerator) function of the linear codes (MacWilliams & Sloane, 1977, Chapter 2, §1).

4.3. Experiments on CIFAR-100

We now turn to results on CIFAR-100, where we compare the performance of different prototype schemes. Figure 4 shows classification accuracy on the test set for different prototype schemes in different dimensions. We notice a thresholding effect around $n = K$, indicating that little is gained by adding dimensions, which is consistent with the observed worst-case similarity in Figure 2. For $n \in \{31, 63, 127\}$, the performance of BCH-code-based prototypes averaged over the label mapping dominates the optimization-based schemes. For $n = 63$, the performance is close to the performance of one-hot encoding. The mapping by Kasarla et al. (2022) still performs best at $n = 99$, which can be expected since it guarantees a slightly lower worst-case cosine similarity.

To illustrate the separation/accuracy tradeoff for the different prototype schemes, Figure 5 plots the accuracy over the maximum cosine similarity across all our trained models. Through linear regression we find that, as expected, more dissimilar prototypes tend to yield better results. However, there is a significant variance in the accuracy within models trained with the same class of prototypes, and moreover, across different methods with the same maximum similarity. Part of the variance is explained by our use of a randomised validation set. However, as the difference in BCH code performance between the fixed mapping and the average

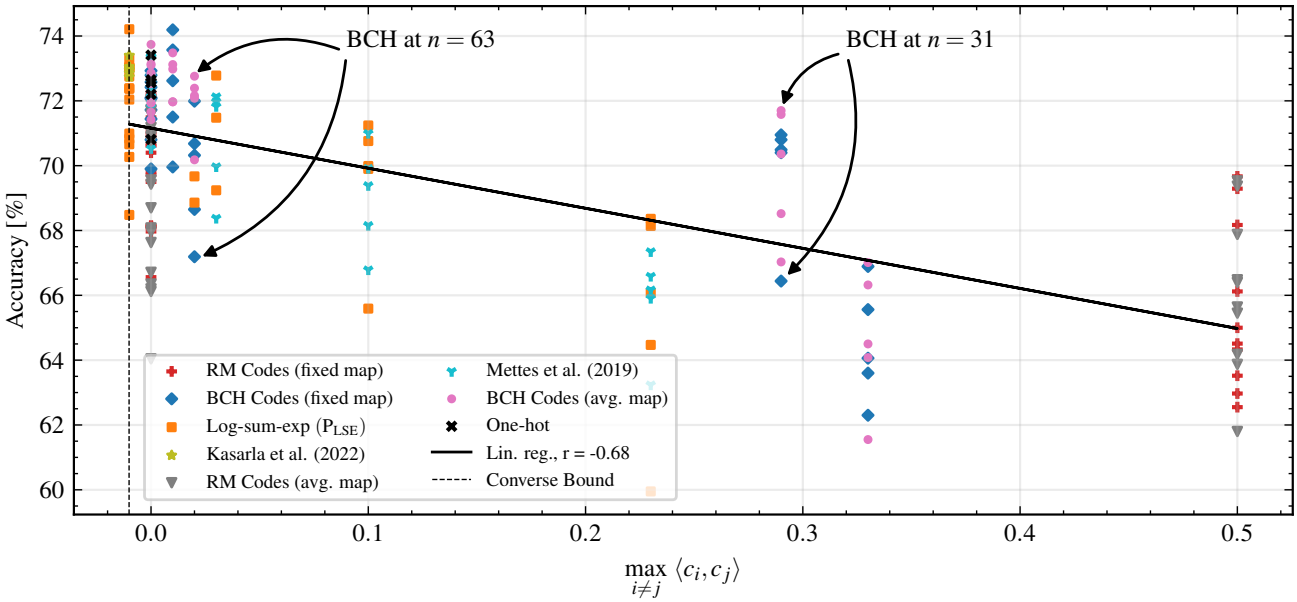

Figure 5. Comparison of accuracy on CIFAR-100 and maximum cosine similarity between the $K = 100$ prototypes. More dissimilar prototypes are correlated with higher accuracy, but there is significant variance within, and between, models corresponding to different prototype generation schemes. Note that the same maximum similarity may correspond to different n, see Figure 2. Interestingly, some BCH codes in lower dimension and with worse cosine similarity yield better performance than BCH codes in higher dimension.

mappings, and the lower performance of RM codes show (see Figure 4), the alignment of prototype similarities with *semantic* similarities of classes appears important.

In particular, the lower accuracy of RM codes in dimensions $n = 64$ and $n = 128$ is insightful. In these dimensions, they provide no worse than orthogonal prototypes (see Lemma 3.2), as well as $K/2$ prototype pairs which are diametrically opposed with cosine similarity -1. Investigating pairs of classes which were assigned diametrically opposed prototypes, we find several pairs with high semantic similarity, for example `leopard` and `lion`; `shrew` and `skunk`; and `seal` and `shark`. Compared to Mettes et al. (2019), who argued for the importance of the maximum and average similarity, our results indicate that additional properties beyond maximum and average similarity are important. Hence, we argue that further investigation on incorporating the semantics in the data in the labelling of the prototypes is needed.

5. Conclusion

In this paper, we have analysed the geometry of hyperspherical prototypical learning with tools from coding theory. Firstly, we have presented new code-based constructions to generate hyperspherical prototypes with strong minimum separation guarantees (in terms of worst-case cosine similarity). Secondly, we fully characterised the worst-case cosine similarity of these prototypes (in terms of achievable and converse bounds). Our prototypes are flexible and near-optimal in low dimension, thereby enabling a trade-off between dimension and separation for a given number of classes. Our experimental results furthermore indicate that the classification accuracy does not only depend on the worst-case separation of prototypes, but also depends on the mapping from class labels to prototypes. We thus conclude that the alignment of semantic similarity with prototype separation is an important problem for further investigation. Additionally, the impact of prototype distance on prototype-based self-supervised learning schemes is also an important future consideration.

Acknowledgements

This work was funded in part by the Swedish Research Council (VR) through grant agreements 2019-03606 and 2021-05266. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Chalmers Centre for Computational Science and Engineering (C3SE), partially funded by the Swedish Research Council through grant agreement 2022-06725.

References

Abbe, E., Shpilka, A., and Ye, M. Reed–Muller Codes: Theory and Algorithms. *IEEE Transactions on Information Theory*, 67(6):3251–3277, June 2021.

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning Representations by Maximizing Mutual Information Across Views. In *Advances in Neural Information Processing Systems*, 2019.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9912–9924, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, November 2020.

Conway, J. and Sloane, N. J. A. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, New York, 3rd edition, 1999.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical Variational Auto-Encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018.

Ericson, T. and Zinoviev, V. *Codes on Euclidean Spheres*. Elsevier Science, Amsterdam, 2001.

Goyal, A. and Bengio, Y. Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478:20210068, October 2022. doi: 10.1098/rspa.2021.0068.

Guerriero, S., Caputo, B., and Mensink, T. DeepNCM: Deep Nearest Class Mean Classifiers. *ICLR - Workshop Track*, 2018.

Hamming, R. W. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. doi: 10.1002/j.1538-7305.1950.tb00463.x.

Hasnat, M. A., Bohné, J., Milgram, J., Gentric, S., and Chen, L. von Mises-Fisher Mixture Model-based Deep learning: Application to Face Verification, December 2017. URL http://arxiv.org/abs/1706.04264.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

Hostetter, M. Galois: A performant NumPy extension for Galois fields, November 2020. URL https://github.com/mhostetter/galois.

Jetley, S., Romera-Paredes, B., Jayasumana, S., and Torr, P. Prototypical Priors: From Improving Classification to Zero-Shot Learning. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2015. doi: 10.5244/C.29.120.

Kasarla, T., Burghouts, G., van Spengler, M., van der Pol, E., Cucchiara, R., and Mettes, P. Maximum Class Separation as Inductive Bias in One Matrix. In *Advances in Neural Information Processing Systems*, 2022.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems*, 2020.

Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

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

Li, S. The Minimum Distance of Some Narrow-Sense Primitive BCH Codes. *SIAM Journal on Discrete Mathematics*, 31(4):2530–2569, January 2017.

Lin, S. and Costello, D. J. *Error Control Coding: Fundamentals and Applications*. Pearson Prentice Hall, 2 edition, 2004.

MacWilliams, F. J. and Sloane, N. J. A. *The Theory of Error-Correcting Codes*. North-Holland Publishing Company, 1977.

Mettes, P., van der Pol, E., and Snoek, C. Hyperspherical Prototype Networks. In *Advances in Neural Information Processing Systems*, 2019.

Mitchell, D. G. M., Lentmaier, M., and Costello, D. J. Spatially Coupled LDPC Codes Constructed From Protographs. *IEEE Transactions on Information Theory*, 61 (9):4866–4889, 2015. doi: 10.1109/TIT.2015.2453267.

Musin, O. R. and Tarasov, A. S. The Tammes Problem for N = 14. *Experimental Mathematics*, 24(4):460–468, October 2015.

Rodríguez-Gálvez, B., Blaas, A., Rodriguez, P., Golinski, A., Suau, X., Ramapuram, J., Busbridge, D., and Zappella, L. The Role of Entropy and Reconstruction in Multi-View Self-Supervised Learning. In *Proceedings of the 40th International Conference on Machine Learning*, July 2023.

Simard, P. Y., Steinkraus, D., and Platt, J. C. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 3. IEEE Computer Society, 2003.

Snell, J., Swersky, K., and Zemel, R. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems*, 2017.

The MathWorks Inc. bchgenpoly, 2024. URL https://www.mathworks.com/help/comm/ref/bchgenpoly.html. Accessed 4 July, 2024.

Tian, Y., Krishnan, D., and Isola, P. Contrastive Multiview Coding. In *Computer Vision – ECCV 2020*. Springer International Publishing, 2020.

Wang, F., Xiang, X., Cheng, J., and Yuille, A. L. NormFace: L2 Hypersphere Embedding for Face Verification. In *Proceedings of the 25th ACM International Conference on Multimedia*, 2017.

Wang, T. and Isola, P. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, November 2020.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Xu, J. and Durrett, G. Spherical Latent Spaces for Stable Variational Autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

A. Separation Guarantees for $K = 10$ and $K = 1\,000$ Prototypes

In this section, we provide additional results for prototype generation schemes for $K = 10$ and $K1\,000$ prototypes, see Figures 6 and 7. For $K = 10$ prototypes, the optimisation-based approaches work well: in particular, solving ($\mathrm{P}_{\mathrm{LSE}}$) provides no worse than orthogonal prototypes in dimension $n = 8$, and optimally separated prototypes in $n = 16$. However, the improvement over one-hot encoding and the Kasarla et al. (2022) mapping is small. On the other hand, for $K = 1\,000$ prototypes, the coding-theoretic approaches can guarantee no worse than orthogonal (and therefore near-optimal) separation in $n \approx {}^{K}/_{2}$, while optimisation-based prototypes require high dimension to give approximately orthogonal prototypes.

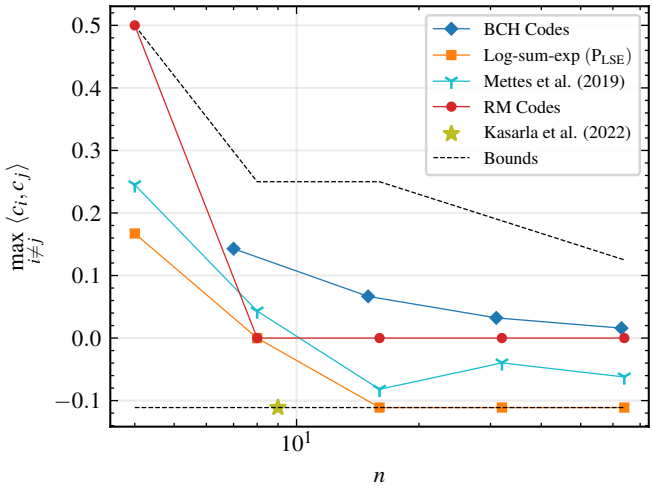

Figure 6. Maximum pairwise cosine similarity of $K = 10$ prototypes in various latent space dimensions (logarithmic scale). The optimisation-based ($\mathrm{P}_{\mathrm{LSE}}$) prototypes achieve better separation than the ($\mathrm{P}_{\mathrm{AVG}}$) scheme. One-hot encoding and the Kasarla et al. (2022) are competitive in this regime. All schemes fall within the achievable and converse bounds from Theorem 3.5. For large n, the the BCH and RM prototypes yield no worse than othogonal prototypes. The ($\mathrm{P}_{\mathrm{LSE}}$) and Kasarla et al. (2022) prototypes achieve the converse bound, and are therefore better. The ($\mathrm{P}_{\mathrm{AVG}}$) prototypes are more than orthogonal, but do not achieve the converse bound.

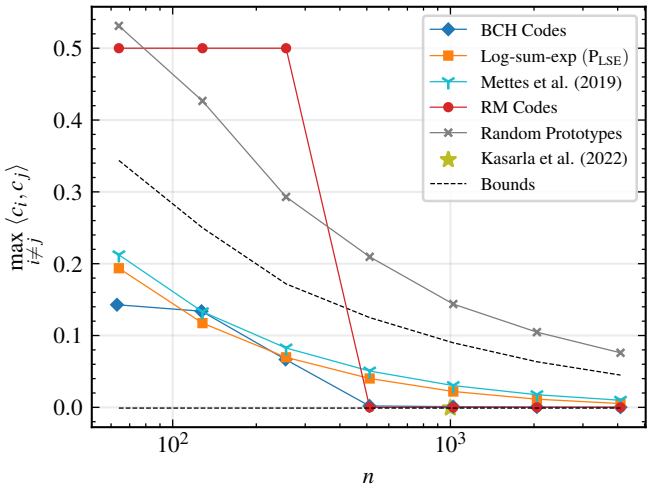

Figure 7. Maximum pairwise cosine similarity of $K = 1\,000$ prototypes in various latent space dimensions (logarithmic scale). Coding-theoretic approaches give near-optimal separation in low dimension $n \geq 511$, and offer better flexibility than the Kasarla et al. (2022) mapping. The optimisation schemes require high dimension to be competitive, while the coding-theoretic prototypes are no worse than orthogonal in low dimension. In very high dimensions, even uniformly random prototypes are near-orthogonal.

B. Cosine Similarity Histograms for $K = 10$ and $K = 1\,000$ prototypes

In this section, we show cosine similarity histograms for $K = 10$ and $K = 1\,000$ prototypes, see Figures 8 and 9. The figures illustrate that optimisation-based methods are effective for a small number of prototypes, where the density can be moved around effectively to create good maximum cosine similarity properties. The coding-theoretic approaches concentrate the density in a few cosine similarities, which is beneficial for a large number of prototypes K, where they outperform the optimisation-based approaches in lower dimensions n.

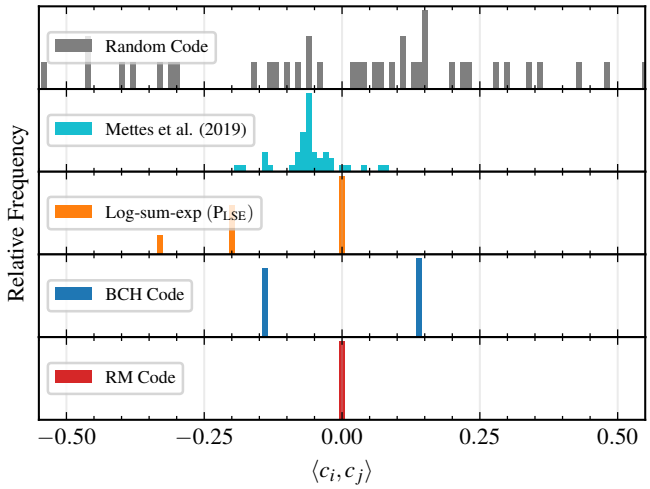

Figure 8. Cosine similarity histogram for different prototype generation schemes for $K = 10$ classes and dimension $n = 8$. For a small number of classes, the optimisation-based approaches are more efficient in moving probability mass to lower similarities. Note that there is a low but non-zero density for the RM code at similarity -1, which has been omitted for clarity.

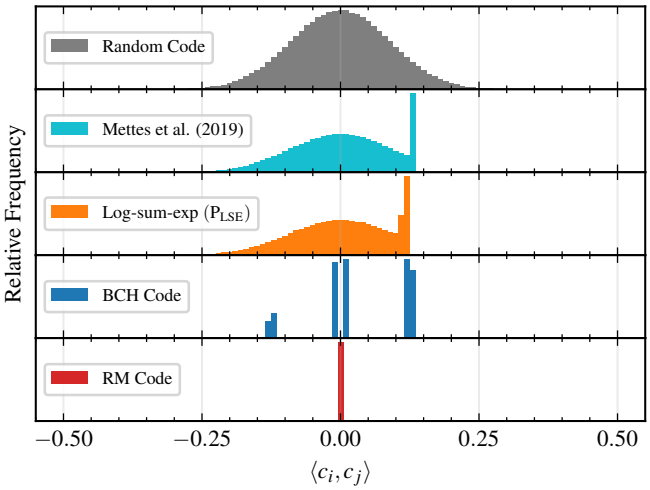

Figure 9. Cosine similarity histogram for different prototype generation schemes for $K = 1\,000$ classes and dimension $n = 128$. The coding-based approaches concentrate density at a few similarities, which outperforms optimisation-based methods for a large number of classes. Note that for the RM code there is non-zero density at similarity -0.5 and 0.5 which are too small to be visible, and that there is density at similarity -1 which has been omitted for clarity.

C. Results on MNIST

In this section, we provide results on MNIST, similar to the ones on CIFAR-100, in Figures 10 and 11. We notice that while all the schemes perform well on MNIST, we are able to beat one-hot encoding and the Kasarla et al. (2022) mapping using prototypes obtained from solving ($\mathrm{P}_{\mathrm{LSE}}$) in lower dimension. We also find that a smaller maximum cosine similarity is correlated with better performance, although the correlation is weaker here than for CIFAR-100.

Similar to CIFAR-100, we notice a high variance. In particular, notice the averaged mappings outperforms the fixed mappings for both BCH and RM codes. This again indicates that the semantic class to prototype mapping is important. Again, investigating mappings for RM codes for $n = 8$, where the prototypes are no worse than orthogonal, we find the following diametrically opposed pairs in the first trained model. For the fixed mapping, the pairs `4` and `5`; and `8` and `9` were diametrically opposed. These pairs are semantically similar, especially when hand-written. A randomised class to prototype assignment has a chance of avoiding these semantically similar pairings.

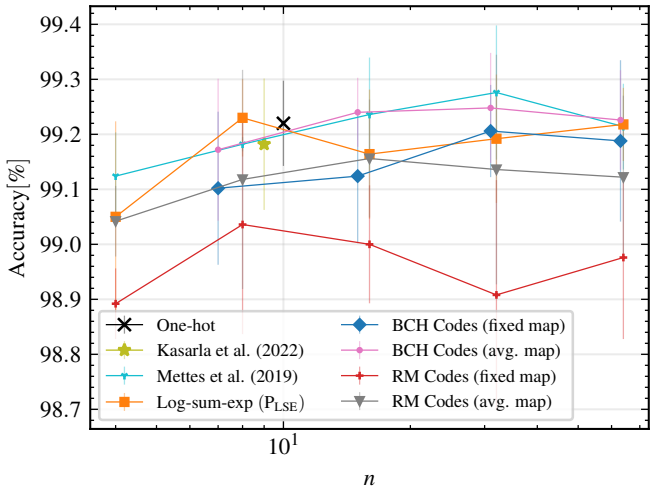

Figure 10. Top-1 accuracy results for MNIST over different prototype generation schemes, averaged over 5 runs, with error bars corresponding to one standard deviation.

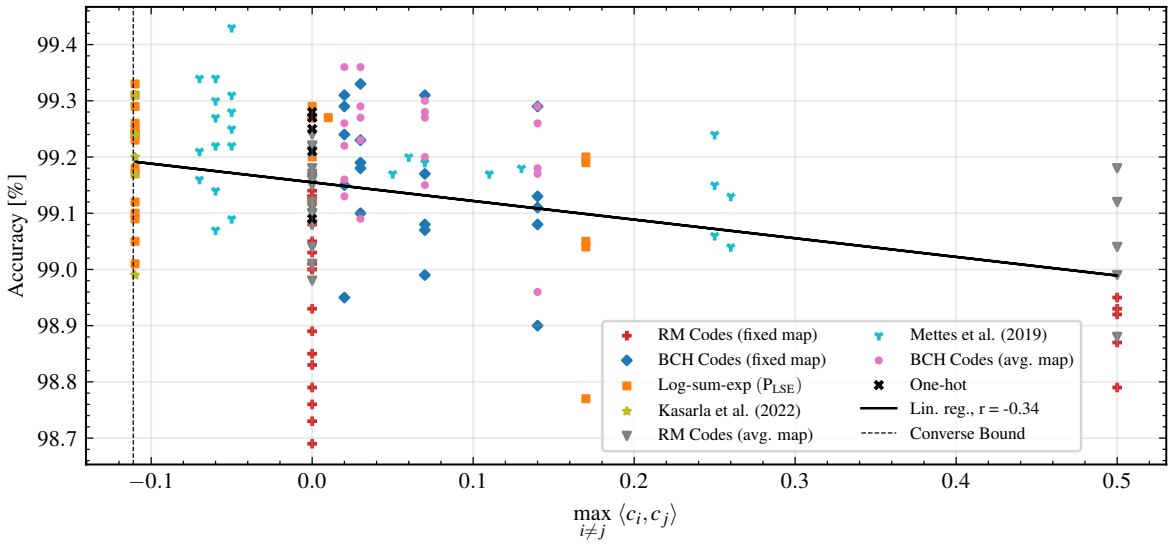

Figure 11. Comparison of accuracy on MNIST and maximum cosine similarity between the $K = 10$ prototypes. Higher accuracy is correlated with smaller maximum cosine similarity. However, there is a large variance within, and between, different prototype generation scheme. Note that the same maximum similarity may correspond to different n, see Figure 6.