MRL-AdANNS: Matryoshka Representation Learning for Web-Scale Adaptive Semantic Search

Aniket Rege

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

Master of Science

University of Washington

2023

Committee:

Ali Farhadi

Linda Shapiro

Jenq-Neng Hwang

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

MRL-AdANNS: Matryoshka Representation Learning for
Web-Scale Adaptive Semantic Search

Aniket Rege

Co-Chairs of the Supervisory Committee:
Ali Farhadi
Department of Computer Science & Engineering

Linda Shapiro
Department of Electrical & Computer Engineering

Learned representations are essential in modern ML systems, but often struggle to adapt to the required capacity of various downstream tasks. In this thesis, we propose 🪆 Matryoshka Representation Learning (MRL) [64] to address this challenge, which learns coarse-to-fine representations with minimal overhead to existing representation learning frameworks at no additional training or inference cost. MRL achieves accuracy and robustness comparable to low-dimensional representations, with benefits like up to $14\times$ smaller ImageNet-1K embeddings and $14\times$ speed-ups for large-scale retrieval. It extends seamlessly to web-scale datasets (ImageNet, JFT) across Vision (ResNet, ViT), Language (BERT), and V+L (ALIGN) modalities. In modern web-scale search systems, *rigid high-dimensional* representations are learned via a deep encoder and hooked into an approximate nearest neighbor search (ANNS) pipeline to retrieve similar data points. Using these rigid representations is computationally expensive and inflexible to compute-constrained environments. To overcome this, we introduce the novel AdANNS framework [92] to leverage the flexibility of Matryoshka Representations at each stage of the ANNS pipeline and provide compute-aware elastic search. We demonstrate state-of-the-art accuracy-compute trade-offs using novel AdANNS-based key ANNS building blocks like search data structures (AdANNS-IVF) [102] and quantization (AdANNS-OPQ) [29]. For example on ImageNet retrieval, AdANNS-IVF is

up to $\mathbf{1.5}\%$ more accurate than the rigid representations-based IVF [102] at the same compute budget; and matches accuracy while being up to $\mathbf{90}\times$ faster in *wall-clock time*. For Natural Questions, 32-byte AdANNS-OPQ matches the accuracy of the 64-byte OPQ baseline [29] constructed using rigid representations – *same accuracy at half the cost!* We further show that the gains from AdANNS translate to modern-day composite ANNS indices that combine search structures and quantization. Finally, we demonstrate that AdANNS can enable inference-time adaptivity for compute-aware search on ANNS indices built non-adaptively on matryoshka representations. The code is open-sourced at `https://github.com/RAIVNLab/MRL` and `https://github.com/RAIVNLab/AdANNS`.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

GLOSSARY

MRL: Matryoshka Representation Learning, a training paradigm for adaptive deployment of deep learning models. Also interchangeably used for a deep encoder trained with Matryoshka Loss.

MR: Matryoshka Representation, an embedding from a deep encoder trained with MRL with information packed in a nested logarithmic manner.

FF: Fixed Feature, an independently trained baseline encoder trained without nested Matryoshka Loss.

RR: Rigid Representation, an embedding from an independently trained Fixed Feature encoder with information diffused across it.

RETRIEVAL: the task of retrieving the nearest neighbors of a given query from an indexed database. This is done via a semantic search by embedding the database and queryset in d-dimensional latent space via an MRL or FF encoder.

ANNS: Approximate Nearest Neighbor Search, to retrieve the "approximate" nearest neighbors of a given query from a database without exhaustively searching all items.

AdANNS: Adaptive ANNS, a framework to decouple the building blocks of ANNS and provide flexible accuracy-compute tradeoff.

ACKNOWLEDGMENTS

For my joint work on MRL and AdANNS, I am grateful to my co-authors Aditya Kusupati, Gantavya Bhatt, Sharan Ranjit, Alan Fan, Matthew Wallingford, Qingqing Cao, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain and Ali Farhadi. I am also grateful to my thesis committee members Ali Farhadi, Linda Shapiro and Jenq-Neng Hwang for their helpful discussion and feedback.

Aditya Kusupati thanks Srinadh Bhojanapalli, Lovish Madaan, Raghav Somani and Ludwig Schmidt for helpful discussions and feedback, as well as Tom Duerig and Rahul Sukthankar for their support. Part of the paper's large-scale experimentation is supported through a research GCP credit award from Google Cloud and Google Research. Gantavya Bhatt is supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. Sham Kakade acknowledges funding from the NSF award CCF-1703574 and ONR N00014-22-1-2377. Ali Farhadi acknowledges funding from the NSF awards IIS 1652052, IIS 17303166, DARPA N66001-19-2-4031, DARPA W911NF-15-1-0543 and gifts from Allen Institute for Artificial Intelligence.

DEDICATION

I dedicate this dissertation work to my family, academic peers and advisors and many, many friends. A special mention to my parents Devdatta and Shilpa Rege, my sister Apoorva Rege, and my grandmother Jyoti Mahajan for their unwavering support through many difficult moments through the years.

I would also like to thank Anmol Varma, Sudha Velusamy and Shankar Venkatesan for their many years of mentorship at Samsung Research, Bangalore, and the many amazing colleagues who pushed me for grad school.

During my time at the University of Washington, I want to especially thank my mentor Aditya Kusupati for many valuable discussions throughout my degree. I also want to thank Kavya Balasubramanian for her unwavering support and encouragement.

Chapter 1

INTRODUCTION

Learned representations [70] are fundamental building blocks of real-world ML systems [83, 111]. Trained once and frozen, d-dimensional representations encode rich information and can be used to perform multiple downstream tasks [5]. The deployment of deep representations has two steps: (1) an expensive yet constant-cost forward pass to compute the representation [40] and (2) utilization of the representation for downstream applications [59, 109]. Compute costs for the latter part of the pipeline scale with the embedding dimensionality as well as the data size (N) and label space (L). At web-scale [19, 105] this utilization cost overshadows the feature computation cost. The rigidity in these representations forces the use of high-dimensional embedding vectors across multiple tasks despite the varying resource and accuracy constraints that require flexibility.

Human perception of the natural world has a naturally coarse-to-fine granularity [36, 41]. However, perhaps due to the inductive bias of gradient-based training [104], deep learning models tend to diffuse "information" across the entire representation vector. The desired elasticity is usually enabled in the existing flat and fixed representations either through training multiple low-dimensional models [40], jointly optimizing sub-networks of varying capacity [12, 122] or post-hoc compression [45, 74]. Each of these techniques struggle to meet the requirements for adaptive large-scale deployment either due to training/maintenance overhead, numerous expensive forward passes through all of the data, storage and memory cost for multiple copies of encoded data, expensive on-the-fly feature selection or a significant drop in accuracy. By encoding coarse-to-fine-grained representations, which are as accurate as the independently trained counterparts, we learn with minimal overhead a representation that can be deployed *adaptively* at no additional cost during inference.

We introduce 🪆 Matryoshka Representation Learning (MRL) to induce flexibility in the

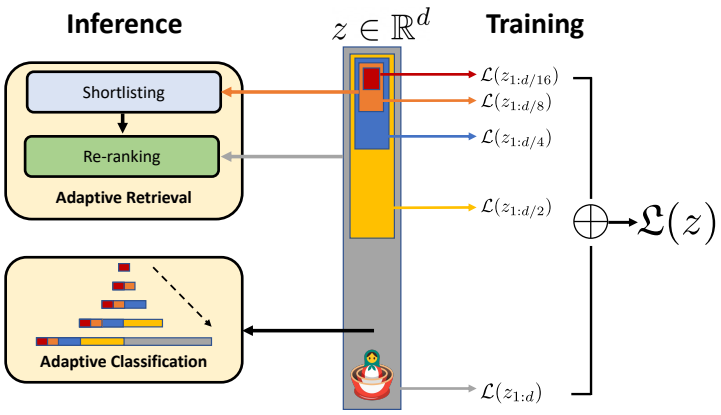


Figure 1.1: Matryoshka Representation Learning is adaptable to any representation learning setup and begets a Matryoshka Representation z by optimizing the original loss $\mathcal{L}(.)$ at $O(\log(d))$ chosen representation sizes. Matryoshka Representation can be utilized effectively for adaptive deployment across environments and downstream tasks.

learned representation. MRL learns representations of varying capacities within the same high-dimensional vector through explicit optimization of $O(\log(d))$ lower-dimensional vectors in a nested fashion, hence the name Matryoshka. MRL can be adapted to any existing representation pipeline and is easily extended to many standard tasks in computer vision and natural language processing. Figure 1.1 illustrates the core idea of Matryoshka Representation Learning (MRL) and the adaptive deployment settings of the learned Matryoshka Representations (MRs).

The first m-dimensions, $m \in [d]$, of the Matryoshka Representation is an information-rich low-dimensional vector, at no additional training cost, that is as accurate as an independently trained m-dimensional representation. The information within the Matryoshka Representation increases with the dimensionality creating a coarse-to-fine grained representation, all without significant training or additional deployment overhead. MRL equips the representation vector with the desired flexibility and multifidelity that can ensure a near-optimal accuracy-vs-compute trade-off. With these advantages, MRL enables adaptive deployment based on accuracy and compute constraints.

The MRs improve efficiency for large-scale classification and retrieval without any significant

loss of accuracy. While there are potentially several applications of coarse-to-fine MRs, in this work we focus on two key building blocks of real-world ML systems: large-scale classification and retrieval. For classification, we use adaptive cascades with the variable-size representations from a model trained with MRL, significantly reducing the average dimension of embeddings needed to achieve a particular accuracy. For example, on ImageNet-1K, MRL + adaptive classification results in up to a $14\times$ smaller representation size at the same accuracy as baselines (Section 4.3). Similarly, we use MRL in a simple adaptive retrieval system. Given a query, we shortlist retrieval candidates using the first few dimensions of the query embedding, and then successively use more dimensions to re-rank the retrieved set. A simple implementation of this approach leads to $128\times$ theoretical (in terms of FLOPS) and $14\times$ wall-clock time speedups compared to a single-shot retrieval system that uses a standard embedding vector; note that MRL's retrieval accuracy is comparable to that of single-shot retrieval (Section 4.5). Finally, as MRL explicitly learns coarse-to-fine representation vectors, it can also be used as method to analyze hardness of classification among instances and information bottlenecks.

1.1 AdANNS *Framework*

Semantic search [54] on learned representations [83, 84, 111] is a major component in retrieval pipelines [9, 19]. In its simplest form, semantic search methods learn a neural network to embed queries as well as a large number (N) of data points in a d-dimensional vector space. For a given query, the nearest (in embedding space) point is retrieved using either an exact search or using approximate nearest neighbor search (ANNS) [49] which is now indispensable for real-time large-scale retrieval.

Existing semantic search methods learn fixed or *rigid* representations (RRs) which are used as is in all the stages of ANNS. That is, while ANNS indices allow a variety of parameters for searching the design space to optimize the accuracy-compute trade-off, the provided data dimensionality is typically assumed to be an *immutable* parameter. To make it concrete, let us consider inverted file index (IVF) [102], a popular web-scale ANNS technique [33]. IVF has two stages (Figure 1.2) during inference: (a) *cluster mapping*: mapping the query to a cluster of data points [76], and

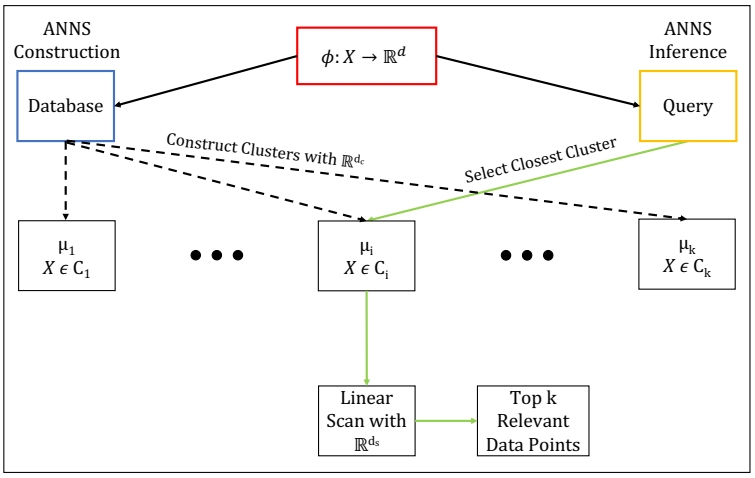


Figure 1.2: The schematic of inverted file index (IVF) outlaying the construction and inference (cluster mapping followed by linear scan) phases. Adaptive representations can be utilized effectively in the decoupled components of clustering and searching for a better accuracy-compute trade-off (AdANNS-IVF, Section 5.1).

(b) *linear scan*: distance computation w.r.t all points in the retrieved cluster to find the nearest neighbor (NN). Standard IVF utilizes the same high-dimensional RR for both phases, which can be sub-optimal.

The proposed Matryoshka Representations (MRs) satisfy the specifications for adaptive representations due to their hierarchical nested information packing. This allows us to deploy MRs in two major and novel ways as part of ANNS: (a) low-dimensional representations for accuracy-compute optimal clustering and quantization, and (b) high-dimensional representations for precise re-ranking when feasible.

To this effort, we introduce AdANNS, a novel design framework for semantic search that uses matryoshka representation-based *adaptive representations* across different stages of ANNS to ensure significantly better accuracy-compute trade-off than the state-of-the-art baselines. Typical ANNS systems have two key components: (a) search data structure to store datapoints, (b) distance computation to map a given query to points in the data structure. Through AdANNS, we address both these components and significantly improve their performance.

While MR already has multi-granular representations, careful integration with ANNS building blocks is critical to obtain a practical method and is *our main contribution*. We compare AdANNS to our simple adaptive retrieval setup (Section 4.5) that uses smaller-dimensional MR for shortlisting in retrieval followed by precise re-ranking with a higher-dimensional MR. Such techniques, unfortunately, cannot be scaled to industrial systems as they require forming a new index for every shortlisting provided by low-dimensional MR. Ensuring that the method aligns well with the modern-day ANNS pipelines is important as they already have mechanisms to handle real-world constraints like load-balancing [33] and random access from disk [51]. So, AdANNS is a step towards making the abstraction of adaptive search and retrieval feasible at the web-scale.

1.2 Summary of Research Contributions

The key contributions of this thesis are:

- We introduce Matryoshka Representation Learning (MRL) to obtain flexible representations (Matryoshka Representations) for adaptive deployment (Section 3.1).
- We also introduce AdANNS, a novel framework for semantic search that leverages Matryoshka Representations for designing ANNS systems with better accuracy-compute tradeoffs (Section 3.2).
- Up to $14\times$ faster yet accurate large-scale classification and retrieval using MRL (Section 4).
- Seamless adaptation of MRL across modalities (vision - ResNet & ViT, vision + language - ALIGN, language - BERT) and to web-scale data (ImageNet-1K/4K, JFT-300M and ALIGN data).
- AdANNS powered search data structure (AdANNS-IVF, Section 5.1) and quantization (AdANNS-OP, Section 5.2) show a significant improvement in accuracy-compute tradeoff compared to existing solutions.
- Further analysis of MRL's representations in the context of other downstream tasks (Section 6.1).
- AdANNS generalizes to modern-day composite ANNS indices and can also enable compute-aware elastic search during inference with no modifications (Section 6.2).

1.3 Thesis Outline

Chapter 2 provides an overview of the related work in representation learning, classification, retrieval and large scale approximate nearest neighbor search (ANNS).

Chapter 3 formalizes Matryoshka Representation Learning and the AdANNS framework in the context of representation learning and ANNS.

In Chapter 4 we discuss MRL as a representation learning paradigm (Section 4.1) and its application for downstream Classification (Section 4.2) and Retrieval (Section 4.4) tasks. We also discuss a natural extension of MRL to adaptive classification using cascades (Section 4.3). Finally, we discuss a precursor to the AdANNS framework via a simple shortlist+rerank adaptive retrieval setup using MRs (Section 4.5).

In Chapter 5, we propose AdANNS-IVF (Section 5.1) which tackles the first component of ANNS systems. AdANNS-IVF uses standard full-precision computations but uses adaptive representations for different IVF stages. We then propose AdANNS-OPQ (Section 5.2) which addresses the second component by using AdANNS-based quantization (OPQ [29]) – here we use exhaustive search overall points. Finally, we combine the two techniques to obtain AdANNS-IVFOPQ (Section 5.3). To demonstrate generality of our technique, we adapt AdANNS to DiskANN [51] which provides interesting accuracy-compute tradeoff; see Table 5.1. Through extensive experimentation, we also show that AdANNS generalizes across search data structures, distance approximations, modalities (text & image), and encoders (CNNs & Transformers) while still translating the theoretical gains to latency reductions in deployment. While we have mainly focused on IVF and OPQ-based ANNS in this work, AdANNS also blends well with other ANNS pipelines.

In Chapter 6, we perform a thorough analysis of MRs and their applicability (Section 6.1). We also show that AdANNS can enable compute-aware elastic search on prebuilt indices without making any modifications (Section 6.2). We provide an extensive analysis on the alignment of matryoshka representation for better semantic search (Section 6.3) and current limitations of MRL and AdANNS (Section 6.4). Lastly, Chapter 7 provides several exciting possible future research directions.

Chapter 2

RELATED WORK

2.1 Representation Learning.

Large-scale datasets like ImageNet [21, 95] and JFT [105] enabled the learning of general purpose representations for computer vision [5, 120]. These representations are typically learned through supervised and un/self-supervised learning paradigms. Supervised pretraining [40, 62, 101] casts representation learning as a multi-class/label classification problem, while un/self-supervised learning learns representation via proxy tasks like instance classification [119] and reconstruction [38, 81]. Recent advances [15, 39] in contrastive learning [35] enabled learning from web-scale data [25] that powers large-capacity cross-modal models [22, 53, 88, 123]. Similarly, natural language applications are built [47] on large language models [10] that are pretrained [86, 94] in a un/self-supervised fashion with masked language modelling [23] or autoregressive training [89].

Matryoshka Representation Learning (MRL) is complementary to all these setups and can be adapted with minimal overhead (Section 3.1). MRL equips representations with multifidelity at no additional cost which enables adaptive deployment based on the data and task (Section 4).

2.2 Efficient Classification and Retrieval.

Efficiency in classification and retrieval during inference can be studied with respect to the high yet constant deep featurization costs or the search cost which scales with the size of the label space and data. Efficient neural networks address the first issue through a variety of algorithms [30, 65] and design choices [46, 66, 107]. However, with a strong featurizer, most of the issues with scale are due to the linear dependence on number of labels (L), size of the data (N) and representation size (d), stressing RAM, disk and processor all at the same time.

The sub-linear complexity dependence on number of labels has been well studied in context of

compute [4, 50, 87] and memory [24] using Approximate Nearest Neighbor Search (ANNS) [79] or leveraging the underlying hierarchy [20, 67]. In case of the representation size, often dimensionality reduction [96, 108], hashing techniques [18, 63, 97] and feature selection [82] help in alleviating selective aspects of the $O(d)$ scaling at a cost of significant drops in accuracy.

MRL tackles the linear dependence on embedding size, d, by learning multifidelity Matryoshka Representations. Lower-dimensional MRs are as accurate as independently trained counterparts without the multiple expensive forward passes. MRs provide an *intermediate abstraction* between high-dimensional vectors and their efficient ANNS indices through the adaptive embeddings nested within the original representation vector, which power *Adaptive ANNS* and our proposed AdANNS framework (Section 5). All other aforementioned efficiency techniques are complementary and can be readily applied to the learned MRs obtained from MRL.

Several works in efficient neural network literature [12, 112, 122] aim at packing neural networks of varying capacity within the same larger network. However, the weights for each progressively smaller network can be different and often require distinct forward passes to isolate the final representations. This is detrimental for adaptive inference due to the need for re-encoding the entire retrieval database with expensive sub-net forward passes of varying capacities. Finally, ordered representations proposed by Rippel et al. [93] use nested dropout in the context of autoencoders to learn nested representations. MRL differentiates itself in formulation by optimizing only for $O(\log(d))$ nesting dimensions instead of $O(d)$. Despite this, MRL diffuses information to intermediate dimensions interpolating between the optimized Matryoshka Representation sizes accurately (Figure 4.4); making web-scale feasible.

2.3 Approximate Nearest Neighbors Search

Most real-world search systems [13, 19] are often powered by large-scale embedding based retrieval [14, 83] that scales in cost with the ever increasing web-data. While categorization [109, 121] clusters similar things together, it is imperative to be equipped with retrieval capabilities that can bring forward every instance [9]. Approximate nearest neighbour search (ANNS) is a paradigm to come as close as possible [17] to retrieving the "true" nearest neighbor (NN) without the exor-

bitant search costs associated with exhaustive search [49, 116]. ANNS performs this via with efficient indexing [18] and traversal [6, 8]. The "approximate" nature comes from data pruning as well as the cheaper distance computation that enable real-time web-scale search. In its naive form, NN-search has a complexity of $\mathcal{O}(dN)$; d is the data dimensionality used for distance computation and N is the size of the database. ANNS employs each of these approximations to reduce the linear dependence on the dimensionality (cheaper distance computation) and data points visited during search (data pruning).

Tackling the sub-optimality of Rigid ANNS. Imagine one needs to partition a dataset into k clusters for IVF and the dimensionality of the data is d – IVF uses full d representation to partition into k clusters. However, suppose we have an alternate approach that somehow projects the data in $d/2$ dimensions and learns $2k$ clusters. Note that the storage and computation to find the nearest cluster remains the same in both cases, i.e., when we have k clusters of d dimensions or $2k$ clusters of $d/2$ dimensions. $2k$ clusters can provide significantly more refined partitioning, but the distances computed between queries and clusters could be significantly more inaccurate after projection to $d/2$ dimensions.

So, if we can find a mechanism to obtain a $d/2$-dimensional representation of points that can accurately approximate the topology/distances of d-dimensional representation, then we can potentially build significantly better ANNS structure that utilizes different capacity representations for the cluster mapping and linear scan phases of IVF. But how do we find such *adaptive representations*? These desired adaptive representations should be cheap to obtain and still ensure distance preservation across dimensionality. Post-hoc dimensionality reduction techniques like SVD [31] and random projections [55] on high-dimensional RRs are potential candidates, but our experiments indicate that in practice they are highly inaccurate and do not preserve distances well enough (Figure 5.1).

Cheaper distance computation. From a bird's eye view, cheaper distance computation is always obtained through dimensionality reduction (quantization included). PCA and SVD [31, 56] can re-

duce dimensionality and preserve distances only to a limited extent without sacrificing accuracy. On the other hand, quantization-based techniques [16, 32] like (optimized) product quantization ((O)PQ) [29, 52] have proved extremely crucial for relatively accurate yet cheap distance computation and simultaneously reduce the memory overhead significantly. Another naive solution is to independently train the representation function with varying low-dimensional information bottlenecks [64] which is rarely used due to the costs of maintaining multiple models and databases.

Data pruning. Enabled by various data structures, data pruning reduces the number of data points visited as part of the search. This is often achieved through hashing [18, 97], trees [7, 28, 33, 102] and graphs [51, 78]. For example, the widely adopted HNSW [79] ($O(d\log(N))$) is as accurate as exact retrieval ($O(dN)$) at the cost of a graph-based index overhead for RAM and disk [51]. More recently there have been efforts towards end-to-end learning of the search data structures [34, 61, 67]. However, web-scale ANNS indices are often constructed on rigid d-dimensional real vectors using the aforementioned data structures that assist with the real-time search. For a more comprehensive review of ANNS structures please refer to [11, 73, 114].

Composite indices. ANNS pipelines often benefit from the complementary nature of various building blocks [54, 88]. In practice, often the data structures (coarse-quantizer) like IVF [102] and HNSW [80] are combined with cheaper distance alternatives like PQ [52] (fine-quantizer) for massive speed-ups in web-scale search. While the data structures are built on d-dimensional real vectors, past works consistently show that PQ can be safely used for distance computation during search time. As evident in modern web-scale ANNS systems like DiskANN [51], the data structures are built on d-dimensional real vectors but work with PQ vectors ($32-64$-byte) for fast distance computations.

ANNS benchmark datasets. Despite the Herculean advances in representation learning [40, 88], ANNS progress is often only benchmarked on fixed representation vectors provided for about a dozen million to billion scale datasets [2, 100] with limited access to the raw data. This resulted

in the improvement of algorithmic design for rigid representations (RRs) that are often not specifically designed for search. All the existing ANNS methods work with the assumption of using the provided d-dimensional representation which might not be Pareto-optimal for the accuracy-compute trade-off in the first place. Note that the lack of raw-image and text-based benchmarks led us to using ImageNet-1K [95] (1.3M images, 50K queries) and Natural Questions [68] (21M passages, 3.6K queries) for experimentation. While not billion-scale, the results observed on ImageNet often translate to real-world progress [60], and Natural Questions is one of the largest question answering datasets benchmarked for dense passage retrieval [58], making our results generalizable and widely applicable.

In this work, we investigate the utility of adaptive representations – embeddings of different dimensionalities having similar semantic information – in improving the design of ANNS algorithms. This helps in transitioning out of restricted construction and inference on rigid representations for ANNS. To this end, we extensively use Matryoshka Representations (MRs) which have desired adaptive properties in-built. To the best of our knowledge, this is the first work that improves accuracy-compute trade-off in ANNS by leveraging adaptive representations on different phases of construction and inference for ANNS data structures.

Chapter 3

PRELIMINARIES

This chapter describes the problem setup and summarizes the technical background required to follow this thesis. This chapter is organized as follows. Section 3.1 provides a detailed description of the problem setup for Matryoshka Representation Learning for a supervised learning task on ImageNet-1K with ResNet50, and a discussion of its adaptation to learning frameworks. Section 3.2 formalizes the AdANNS framework in the context of Approximate Nearest Neighbor Search and describes how flexible representations can be used to decouple the fundamental building blocks of ANNS to provide state-of-the-art accuracy-compute tradeoff for retrieval and search. Finally, metrics to evaluate the effectiveness of Matryoshka Representations for AdANNS are discussed in Section 3.2 and in more detail in Appendix D.

3.1 Matryoshka Representation Learning

For $d \in \mathbb{N}$, consider a set $\mathcal{M} \subset [d]$ of representation sizes. For a datapoint x in the input domain $\mathcal{X}$, our goal is to learn a d-dimensional representation vector $z \in \mathbb{R}^d$. For every $m \in \mathcal{M}$, Matryoshka Representation Learning (MRL) enables each of the first m dimensions of the embedding vector, $z_{1:m} \in \mathbb{R}^m$ to be independently capable of being a transferable and general purpose representation of the datapoint x. We obtain z using a deep neural network $F(\,\cdot\,;\theta_F)\colon \mathcal{X} \to \mathbb{R}^d$ parameterized by learnable weights θ_F, i.e., $z \coloneqq F(x;\theta_F)$. The multi-granularity is captured through the set of the chosen dimensions $\mathcal{M}$, that contains less than $\log(d)$ elements, i.e., $|\mathcal{M}| \leq \lfloor \log(d) \rfloor$. The usual set $\mathcal{M}$ consists of consistent halving until the representation size hits a low information bottleneck. We discuss the design choices in Section 4 for each of the representation learning settings.

For the ease of exposition, we present the formulation for fully supervised representation learn-

ing via multi-class classification. Matryoshka Representation Learning modifies the typical setting to become a multi-scale representation learning problem on the same task. For example, we train ResNet50 [40] on ImageNet-1K [95] which embeds a 224×224 pixel image into a $d = 2048$ representation vector and then passed through a linear classifier to make a prediction, $\hat{y}$ among the $L = 1000$ labels. For MRL, we choose $\mathcal{M} = \{8, 16, \ldots, 1024, 2048\}$ as the nesting dimensions.

Suppose we are given a labelled dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x_i \in \mathcal{X}$ is an input point and $y_i \in [L]$ is the label of x_i for all $i \in [N]$. MRL optimizes the multi-class classification loss for each of the nested dimension $m \in \mathcal{M}$ using standard empirical risk minimization using a separate linear classifier, parameterized by $\mathbf{W}^{(m)} \in \mathbb{R}^{L \times m}$. All the losses are aggregated after scaling with their relative importance $(c_m \geq 0)_{m \in \mathcal{M}}$ respectively. That is, we solve

$$\min_{\{\mathbf{W}^{(m)}\}_{m \in \mathcal{M}},\, \theta_F} \frac{1}{N} \sum_{i \in [N]} \sum_{m \in \mathcal{M}} c_m \cdot \mathcal{L}\left(\mathbf{W}^{(m)} \cdot F(x_i; \theta_F)_{1:m}\ ;\ y_i\right) , \tag{3.1}$$

where $\mathcal{L}\colon \mathbb{R}^L \times [L] \to \mathbb{R}_+$ is the multi-class softmax cross-entropy loss function. This is a standard optimization problem that can be solved using sub-gradient descent methods. We set all the importance scales, $c_m = 1$ for all $m \in \mathcal{M}$; see Appendix M.1 for ablations. Lastly, despite only optimizing for $O(\log(d))$ nested dimensions, MRL results in accurate representations, that interpolate, for dimensions that fall between the chosen granularity of the representations (Section 4.2).

We call this formulation as Matryoshka Representation Learning (MRL). A natural way to make this efficient is through weight-tying across all the linear classifiers, i.e., by defining $\mathbf{W}^{(m)} = \mathbf{W}_{1:m}$ for a set of common weights $\mathbf{W} \in \mathbb{R}^{L \times d}$. This would reduce the memory cost due to the linear classifiers by almost half, which would be crucial in cases of extremely large output spaces [109, 121]. This variant is called *Efficient* Matryoshka Representation Learning (MRL–E). Refer to Alg 1 and Alg 2 in Appendix A for the building blocks of MRL.

Adaptation to Learning Frameworks. MRL can be adapted seamlessly to most representation learning frameworks at web-scale with minimal modifications (Section 4.1). For example, MRL's adaptation to masked language modelling reduces to MRL–E due to the weight-tying between the input embedding matrix and the linear classifier. For contrastive learning, both in context of vision

& vision + language, MRL is applied to both the embeddings that are being contrasted with each other. The presence of normalization on the representation needs to be handled independently for each of the nesting dimension for best results (see Appendix C for more details).

3.2 AdANNS

The problem setup of approximate nearest neighbor search (ANNS) [49] consists of a database of N data points, $[x_1, x_2, \ldots, x_N]$, and a query, q, where the goal is to "approximately" retrieve the nearest data point to the query. Both the database and query are embedded to $\mathbb{R}^d$ using a representation function $\phi : \mathcal{X} \to \mathbb{R}^d$, often a neural network that can be learned through various representation learning paradigms [5, 40, 39, 83, 88].

Inverted File Index (IVF). IVF [102] is an ANNS data structure used in web-scale search systems [33] owing to its simplicity, minimal compute overhead, and high accuracy. IVF construction involves clustering (coarse quantization through k-means) [76] on d-dimensional representation that results in an inverted file list [117] of all the data points in each cluster. During search, d-dimensional query representation is assigned to the most relevant cluster (C_i; $i \in [k]$) by finding the closest centroid (μ_i) using an appropriate distance metric (L_2 or cosine). This is followed by an exhaustive linear search across all data points in the cluster which gives the closest NN (see Figure 1.2 in Appendix A for IVF overview). Lastly, IVF can scale to web-scale by utilizing a hierarchical IVF structure within each cluster [33]. Table H.1 in Appendix A describes the retrieval formula for multiple variants of IVF.

Optimized Product Quantization (OPQ). Product Quantization (PQ) [52] works by splitting a d-dimensional real vector into m sub-vectors and quantizing each sub-vector with an independent 2^b length codebook across the database. After PQ, each d-dimensional vector can be represented by a compact $m \times b$ bit vector; we make each vector m bytes long by fixing $b = 8$. During search time, distance computation between the query vector and PQ database is extremely efficient with only m codebook lookups. The generality of PQ encompasses scalar/vector quantization [32, 76] as special

cases. However, PQ can be further improved by rotating the d-dimensional space appropriately to maximize distance preservation after PQ. Optimized Product Quantization (OPQ) [29] achieves this by learning an orthonormal projection matrix R that rotates the d-dimensional space to be more amenable to PQ. OPQ shows consistent gains over PQ across a variety of ANNS tasks and has become the default choice in standard composite indices [51, 54].

Datasets. We evaluate the ANNS algorithms while changing the representations used for the search thus making it impossible to evaluate on the usual benchmarks [2]. Hence we experiment with two public datasets: (a) ImageNet-1K [95] dataset on the task of image retrieval – where the goal is to retrieve images from a database (1.3M image train set) belonging to the same class as the query image (50K image validation set) and (b) Natural Questions (NQ) [68] dataset on the task of question answering through dense passage retrieval – where the goal is to retrieve the relevant passage from a database (21M Wikipedia passages) for a query (3.6K questions).

Metrics Performance of ANNS is often measured using recall score [51], k-recall@N – recall of the exact NN across search complexities which denotes the recall of k "true" NN when N data points are retrieved. However, the presence of labels allows us to compute 1-NN (top-1) accuracy. Top-1 accuracy is a harder and more fine-grained metric that correlates well with typical retrieval metrics like recall and mean average precision (mAP@k). Even though we report top-1 accuracy by default during experimentation, we discuss other metrics in Appendix D. Finally, we measure the compute overhead of ANNS using MFLOPS/query and also provide wall-clock times (see Appendix C.2).

Encoders. For ImageNet, we encode both the database and query set using a ResNet50 (ϕ_I) [40] trained on ImageNet-1K. For NQ, we encode both the passages in the database and the questions in the query set using a BERT-Base (ϕ_N) [23] model fine-tuned on NQ for dense passage retrieval [58].

We use the trained RR-ResNet50 models with varying representation sizes ($d = [8, 16, \ldots, 2048]$;

default being 2048) as described in Section 3.1 alongside the MR-ResNet50 models trained with MRL for the same dimensionalities. The RR and MR models are trained to ensure the supervised one-vs-all classification accuracy across all data dimensionalities is nearly the same – 1-NN accuracy of 2048-d RR and MR models are 71.19% and 70.97% respectively on ImageNet-1K. Independently trained models, $\phi_I^{\mathrm{RR}(d)}$, output $d = [8, 16 \ldots, 2048]$ dimensional RRs while a single MRL-ResNet50 model, $\phi_I^{\mathrm{MR}(d)}$, outputs a $d = 2048$-dimensional MR that contains all the 9 granularities.

We also train BERT-Base models in a similar vein as the aforementioned ResNet50 models. The key difference is that we take a pre-trained BERT-Base model and fine-tune on NQ as suggested by Karpukhin et al. [58] with varying (5) representation sizes (bottlenecks) ($d = [48, 96, \ldots, 768]$; default being 768) to obtain $\phi_N^{\mathrm{RR}(d)}$ that creates RRs for the NQ dataset. To get the MRL-BERT-Base model, we fine-tune a pre-trained BERT-Base encoder on the NQ train dataset using the MRL objective with the same granularities as RRs to obtain $\phi_N^{\mathrm{MR}(d)}$ which contains all five granularities. Akin to ResNet50 models, the RR and MR BERT-Base models on NQ are built to have similar 1-NN accuracy for 768-d of 52.2% and 51.5% respectively. More implementation details can be found in Appendix C and additional experiment-specific information is provided at the appropriate places.

Chapter 4

MRL – MATRYOSHKA REPRESENTATION LEARNING

In this section, we discuss Matryoshka Representation Learning (MRL) for a diverse set of applications along with an extensive evaluation of the learned multifidelity representations. Further, we showcase the downstream applications of the learned Matryoshka Representations for flexible large-scale deployment through (a) Adaptive Classification (AC) and (b) Adaptive Retrieval (AR).

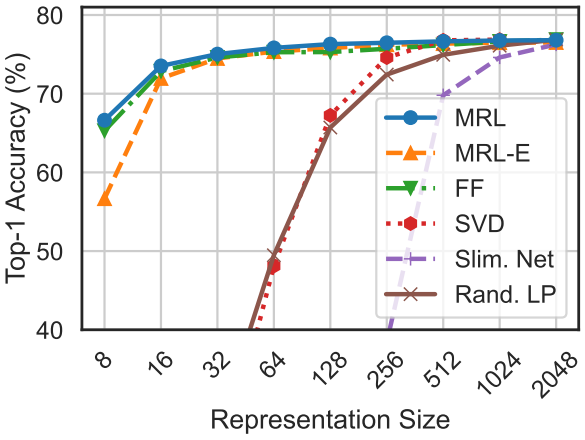


Figure 4.1: ImageNet-1K linear classification accuracy of ResNet50 models. MRL is as accurate as the independently trained FF models for every representation size.

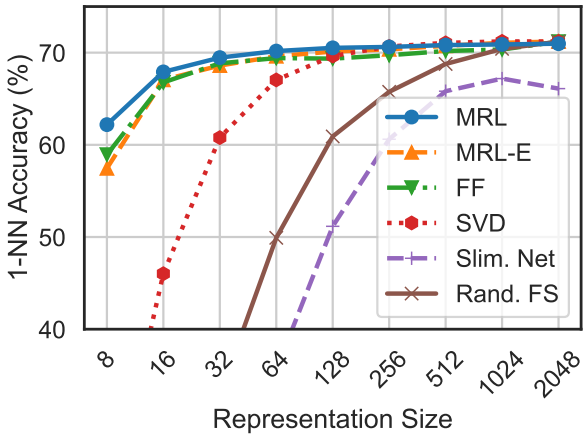


Figure 4.2: ImageNet-1K 1-NN accuracy of ResNet50 models measuring the representation quality. MRL outperforms all baselines across all representation sizes.

4.1 Representation Learning

We adapt Matryoshka Representation Learning (MRL) to various representation learning setups (a) Supervised learning for vision: ResNet50 [40] on ImageNet-1K [95] and ViT-B/16 [26] on JFT-300M [105], (b) Contrastive learning for vision + language: ALIGN model with ViT-B/16 vision encoder and BERT language encoder on ALIGN data [53] and (c) Masked language modelling: BERT [23] on English Wikipedia and BooksCorpus [124]. Please refer to Appendices B and C for details regarding the model architectures, datasets and training specifics.

We do not search for best hyper-parameters for all MRL experiments but use the same hyper-parameters as the independently trained baselines. ResNet50 outputs a 2048-dimensional representation while ViT-B/16 and BERT-Base output 768-dimensional embeddings for each data point. We use $\mathcal{M} = \{8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ and $\mathcal{M} = \{12, 24, 48, 96, 192, 384, 768\}$ as the explicitly optimized nested dimensions respectively. Lastly, we extensively compare the MRL and MRL–E models to independently trained low-dimensional (fixed feature) representations (FF), dimensionality reduction (SVD), sub-net method (slimmable networks [122]) and randomly selected features of the highest capacity FF model.

In section 4.2, we evaluate the quality and capacity of the learned representations through linear classification/probe (LP) and 1-nearest neighbour (1-NN) accuracy. Experiments show that MRL models remove the dependence on $|\mathcal{M}|$ resource-intensive independently trained models for the coarse-to-fine representations while being as accurate. Lastly, we show that despite optimizing only for $|\mathcal{M}|$ dimensions, MRL models diffuse the information, in an interpolative fashion, across all the d dimensions providing the finest granularity required for adaptive deployment.

4.2 Classification

Figure 4.1 compares the linear classification accuracy of ResNet50 models trained and evaluated on ImageNet-1K. ResNet50–MRL model is at least as accurate as each FF model at every representation size in $\mathcal{M}$ while MRL–E is within 1% starting from 16-dim. Similarly, Figure 4.2 showcases the comparison of learned representation quality through 1-NN accuracy on ImageNet-

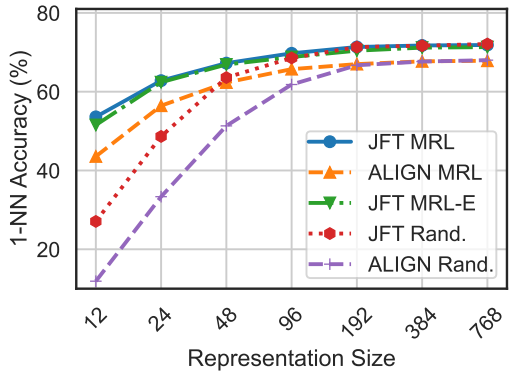


Figure 4.3: ImageNet-1K 1-NN accuracy for ViT-B/16 models trained on JFT-300M & as part of ALIGN. MRL scales seamlessly to web-scale with minimal training overhead.

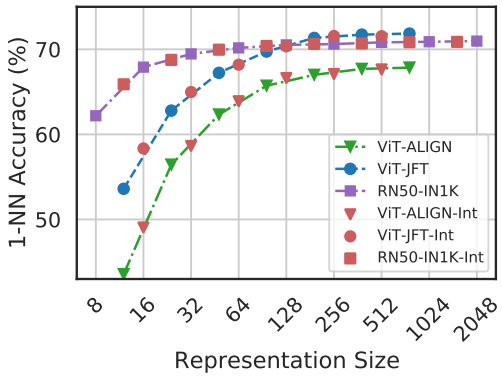


Figure 4.4: Despite optimizing MRL only for $O(\log(d))$ dimensions for ResNet50 and ViT-B/16 models; the accuracy in the intermediate dimensions shows interpolating behaviour.

1K (trainset with 1.3M samples as the database and validation set with 50K samples as the queries). Matryoshka Representations are up to 2% more accurate than their fixed-feature counterparts for the lower-dimensions while being as accurate elsewhere. 1-NN accuracy is an excellent proxy, at no additional training cost, to gauge the utility of learned representations in the downstream tasks.

We also evaluate the quality of the representations from training ViT-B/16 on JFT-300M alongside the ViT-B/16 vision encoder of the ALIGN model – two web-scale setups. Due to the expensive nature of these experiments, we only train the highest capacity fixed feature model and choose random features for evaluation in lower-dimensions. Web-scale is a compelling setting for MRL due to its relatively inexpensive training overhead while providing multifidelity representations for downstream tasks. Figure 4.3, evaluated with 1-NN on ImageNet-1K, shows that all the MRL models for JFT and ALIGN are highly accurate while providing an excellent cost-vs-accuracy trade-off at lower-dimensions. These experiments show that MRL seamlessly scales to large-scale models and web-scale datasets while providing the otherwise prohibitively expensive multi-granularity in the process. We also have similar observations when pretraining BERT; please see Appendix E.2 for more details. Our experiments also show that post-hoc compression (SVD),

linear probe on random features, and sub-net style slimmable networks drastically lose accuracy compared to MRL as the representation size decreases. Finally, Figure 4.4 shows that, while MRL explicitly optimizes $O(\log(d))$ nested representations – removing the $O(d)$ dependence [93] –, the coarse-to-fine grained information is interpolated across all d dimensions providing highest flexibility for adaptive deployment.

4.3 Adaptive Classification

The flexibility and coarse-to-fine granularity within Matryoshka Representations allows model cascades [110] for Adaptive Classification (AC) [36]. Unlike standard model cascades [115], MRL does not require multiple expensive neural network forward passes. To perform AC with an MRL trained model, we learn thresholds on the maximum softmax probability [43] for each nested classifier on a holdout validation set. We then use these thresholds to decide when to transition to the higher dimensional representation (e.g $8 \rightarrow 16 \rightarrow 32$) of the MRL model. Appendix E.1 discusses the implementation and learning of thresholds for cascades used for adaptive classification in detail.

Figure 4.5 shows the comparison between cascaded MRL representations (MRL–AC) and independently trained fixed feature (FF) models on ImageNet-1K with ResNet50. We computed the expected representation size for MRL–AC based on the final dimensionality used in the cascade. We observed that MRL–AC was as accurate, 76.30%, as a 512-dimensional FF model but required an expected dimensionality of ~ 37 while being only 0.8% lower than the 2048-dimensional FF baseline. Note that all MRL–AC models are significantly more accurate than the FF baselines at comparable representation sizes. MRL–AC uses up to $\sim 14\times$ smaller representation size for the same accuracy which affords computational efficiency as the label space grows [109]. Lastly, our results with MRL–AC indicate that instances and classes vary in difficulty which we analyze in Section 6.1 and Appendix L.

4.4 Retrieval

Nearest neighbour search with learned representations powers a plethora of retrieval and search applications [19, 111, 13, 83]. In this section, we discuss the image retrieval performance of the pretrained ResNet50 models (Section 4.1) on two large-scale datasets ImageNet-1K [95] and ImageNet-4K. ImageNet-1K has a database size of $\sim$1.3M and a query set of 50K samples uniformly spanning 1000 classes. We also introduce ImageNet-4K which has a database size of $\sim$4.2M and query set of $\sim$200K samples uniformly spanning 4202 classes (see Appendix B for details). A single forward pass on ResNet50 costs 4 GFLOPs while exact retrieval costs 2.6 GFLOPs per query for ImageNet-1K. Although retrieval overhead is 40% of the total cost, retrieval cost grows linearly with the size of the database. ImageNet-4K presents a retrieval benchmark where the exact search cost becomes the computational bottleneck (8.6 GFLOPs per query). In both these settings, the memory and disk usage are also often bottlenecked by the large databases. However, in most real-world applications exact search, $O(dN)$, is replaced with an approximate nearest neighbor search (ANNS) method like HNSW [79], $O(d\log(N))$, with minimal accuracy drop at the cost of additional memory overhead.

The goal of image retrieval is to find images that belong to the same class as the query using representations obtained from a pretrained model. In this section, we compare retrieval performance using mean Average Precision @ 10 (mAP@10) which comprehensively captures the setup of relevant image retrieval at scale. We measure the cost per query using exact search in MFLOPs. All embeddings are unit normalized and retrieved using the L2 distance metric. Lastly, we report an extensive set of metrics spanning mAP@k and P@k for $k = \{10, 25, 50, 100\}$ and real-world wall-clock times for exact search and HNSW. See Appendices F and F.1 for more details.

Figure 4.6 compares the mAP@10 performance of ResNet50 representations on ImageNet-1K across dimensionalities for MRL, $\mathrm{MRL\text{–}E}$, FF, slimmable networks along with post-hoc compression of vectors using SVD and random feature selection. $\mathrm{Matryoshka\ Representations}$ are often the most accurate while being up to 3% better than the FF baselines. Similar to classification, post-hoc compression and slimmable network baselines suffer from significant drop-off in retrieval

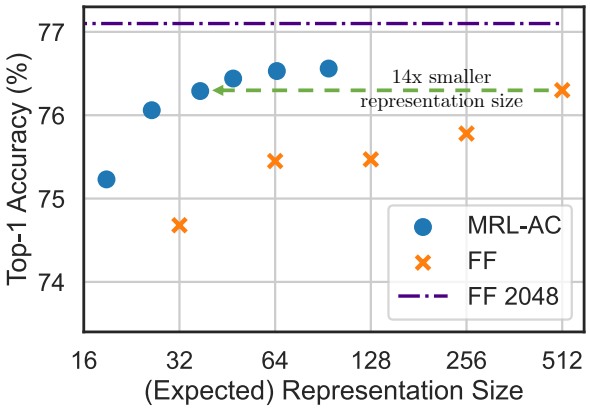


Figure 4.5: Adaptive classification on MRL ResNet50 using cascades results in $14\times$ smaller representation size for the same level of accuracy on ImageNet-1K ($\sim$ 37 vs 512 dims for 76.3%).

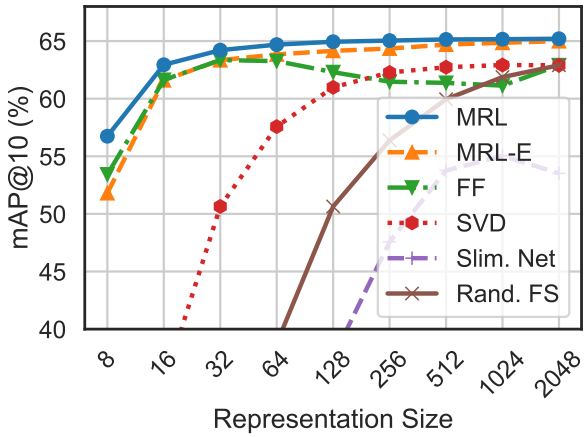


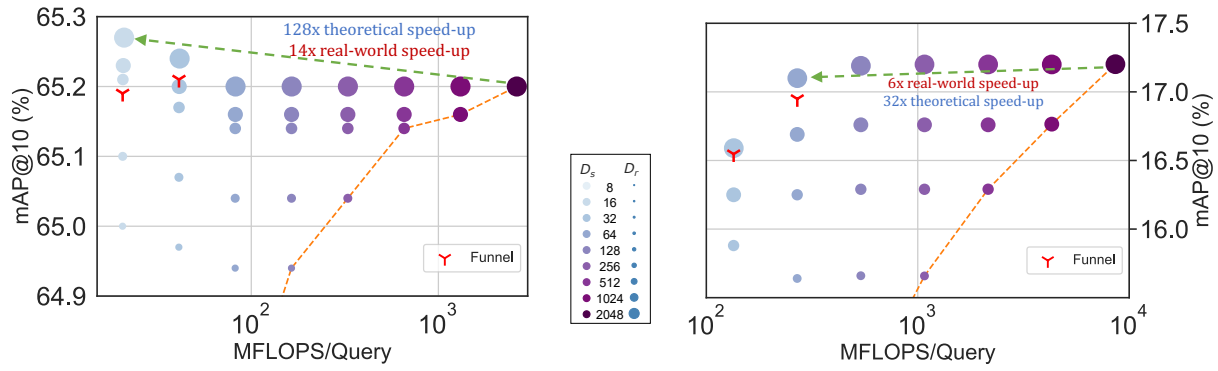
Figure 4.6: mAP@10 for Image Retrieval on ImageNet-1K with ResNet50. MRL consistently produces better retrieval performance over the baselines across all the representation sizes.

mAP@10 with $\leq$ 256 dimensions. Appendix F discusses the mAP@10 of the same models on ImageNet-4K.

MRL models are capable of performing accurate retrieval at various granularities without the additional expense of multiple model forward passes for the web-scale databases. FF models also generate independent databases which become prohibitively expense to store and switch in between. Matryoshka Representations enable adaptive retrieval (AR) which alleviates the need to use full-capacity representations, $d = 2048$, for all data and downstream tasks. Lastly, all the vector compression techniques [74, 52] used as part of the ANNS pipelines are complimentary to Matryoshka Representations and can further improve the efficiency-vs-accuracy trade-off.

4.5 Adaptive Retrieval

We benchmark MRL in the adaptive retrieval setting (AR) [59]. For a given query image, we obtained a shortlist, $K = 200$, of images from the database using a lower-dimensional representation,



(a) ImageNet-1K (b) ImageNet-4K

Figure 4.7: The trade-off between mAP@10 vs MFLOPs/Query for Adaptive Retrieval (AR) on ImageNet-1K (left) and ImageNet-4K (right). Every combination of D_s & D_r falls above the Pareto line (orange dots) of single-shot retrieval with a fixed representation size while having configurations that are as accurate while being up to $14\times$ faster in real-world deployment. Funnel retrieval is almost as accurate as the baseline while alleviating some of the parameter choices of Adaptive Retrieval.

e.g. $D_s = 16$ followed by reranking with a higher capacity representation, e.g. $D_r = 2048$. In real-world scenarios where top ranking performance is the key objective, measured with mAP@k where k covers a limited yet crucial real-estate, AR provides significant compute and memory gains over single-shot retrieval with representations of fixed dimensionality. Finally, the most expensive part of AR, as with any retrieval pipeline, is the nearest neighbour search for shortlisting. For example, even naive re-ranking of 200 images with 2048 dimensions only costs 400 KFLOPs. While we report exact search cost per query for all AR experiments, the shortlisting component of the pipeline can be sped-up using ANNS, such as IVF, HNSW and DiskANN. Appendix C.1 has a detailed discussion on compute cost for exact search, memory overhead of HNSW indices and wall-clock times for both implementations. We note that using HNSW with 32 neighbours for shortlisting does not decrease accuracy during retrieval.

Figure 4.7 showcases the compute-vs-accuracy trade-off for adaptive retrieval using MRs com-

pared to single-shot using fixed features with ResNet50 on ImageNet-1K. We observed that all AR settings lied above the Pareto frontier of single-shot retrieval with varying representation sizes. In particular for ImageNet-1K, we show that the AR model with $D_s = 16$ & $D_r = 2048$ is as accurate as single-shot retrieval with $d = 2048$ while being $\sim \mathbf{128}\times$ more efficient in theory and $\sim \mathbf{14}\times$ faster in practice (compared using HNSW on the same hardware). We show similar trends with ImageNet-4K, but note that we require $D_s = 64$ given the increased difficulty of the dataset. This results in $\sim 32\times$ and $\sim 6\times$ theoretical and in-practice speedups respectively. Lastly, while $K = 200$ works well for our adaptive retrieval experiments, we ablated over the shortlist size k in Appendix M.2 and found that the accuracy gains stopped after a point, further strengthening the use-case for $\mathrm{Matryoshka\ Representation\ Learning}$ and adaptive retrieval.

Even with adaptive retrieval, it is hard to determine the choice of D_s & D_r. In order to alleviate this issue to an extent, we propose **Funnel Retrieval**, a consistent cascade for adaptive retrieval. Funnel thins out the initial shortlist by a repeated re-ranking and shortlisting with a series of increasing capacity representations. Funnel halves the shortlist size and doubles the representation size at every step of re-ranking. For example on ImageNet-1K, a funnel with the shortlist progression of $200 \rightarrow 100 \rightarrow 50 \rightarrow 25 \rightarrow 10$ with the cascade of $16 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 2048$ representation sizes within $\mathrm{Matryoshka\ Representation}$ is as accurate as the single-shot 2048-dim retrieval while being $\sim 128\times$ more efficient theoretically (see Appendix F.1 for more results).

While the gains provided by AR and Funnel Retrieval over vanilla single-shot retrieval showcase the potential of MRL for large-scale multi-stage search systems [19], AR requires forming a new index for every shortlisting provided by low-dimensional MR. The AdANNS framework allows adaptive nearest neighbors search across compute budgets with a single index, and is thus a step towards making AR feasible at web-scale.

Chapter 5

ADANNS – ADAPTIVE ANNS

In this section, we present our proposed AdANNS framework that exploits the inherent flexibility of matryoshka representations to improve the accuracy-compute trade-off for semantic search components. Standard ANNS pipeline can be split into two key components: (a) search data structure that indexes and stores data points, (b) query-point computation method that outputs (approximate) distance between a given query and data point. For example, standard IVFOPQ [54] method uses an IVF structure to index points on full-precision vectors and then relies on OPQ for more efficient distance computation between the query and the data points during the linear scan.

Below, we show that AdANNS can be applied to both the above-mentioned ANNS components and provides significant gains on the computation-accuracy tradeoff curve. In particular, we present AdANNS-IVF which is AdANNS version of the standard IVF index structure [102], and the closely related ScaNN structure [33]. We also present AdANNS-OPQ which introduces representation adaptivity in the OPQ, an industry-default quantization. Then, in Section 5.3 we further demonstrate the combination of the two techniques to get AdANNS-IVFOPQ – an AdANNS version of IVFOPQ [54] – and AdANNS-DiskANN, a similar variant of DiskANN [51]. Overall, our experiments show that AdANNS-IVF is significantly more accuracy-compute optimal compared to the IVF indices built on RRs and AdANNS-OPQ is as accurate as the OPQ on RRs while being significantly cheaper.

5.1 AdANNS-*IVF*

Recall from Section 1 that IVF has a clustering and a linear scan phase, where both phase use same dimensional rigid representation. Now, AdANNS-IVF allows the clustering phase to use the first d_c dimensions of the given matryoshka representation (MR). Similarly, the linear scan within

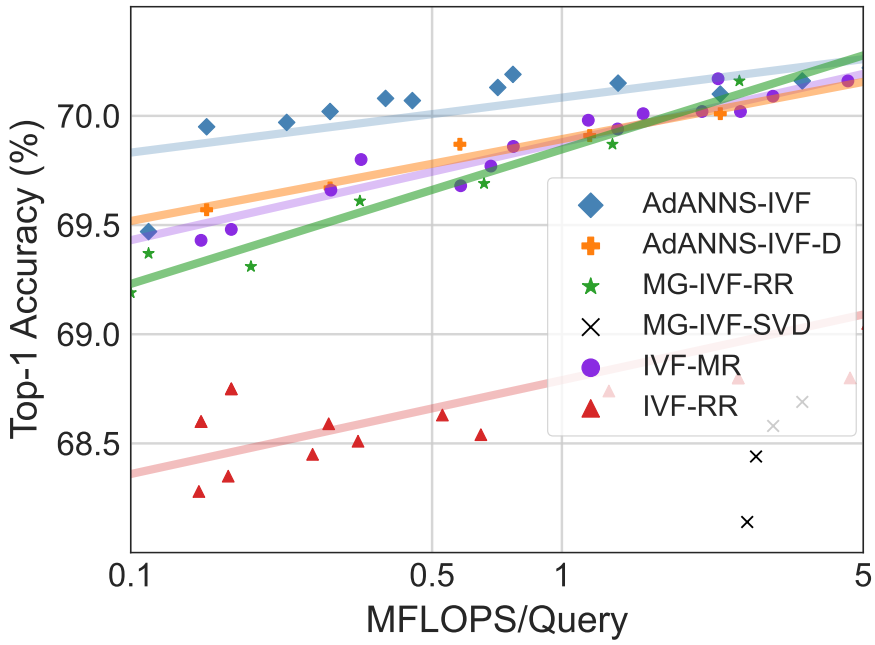


Figure 5.1: 1-NN accuracy on ImageNet retrieval shows that AdANNS-IVF achieves near-optimal accuracy-compute trade-off compared across various rigid and adaptive baselines. Both adaptive variants of MR and RR significantly outperform their rigid counterparts (IVF-XX) while post-hoc compression on RR using SVD for adaptivity falls short.

each cluster uses d_s dimensions, where again d_s represents top d_s coordinates from MR. Note that setting $d_c = d_s$ results in non-adaptive regular IVF. Intuitively, we would set $d_c \ll d_s$, so that instead of clustering with a high-dimensional representation, we can approximate it accurately with a low-dimensional embedding of size d_c followed by a linear scan with a higher d_s-dimensional representation. Intuitively, this helps in the smooth search of design space for state-of-the-art accuracy-compute trade-off. Furthermore, this can provide a precise operating point on accuracy-compute tradeoff curve which is critical in several practical settings.

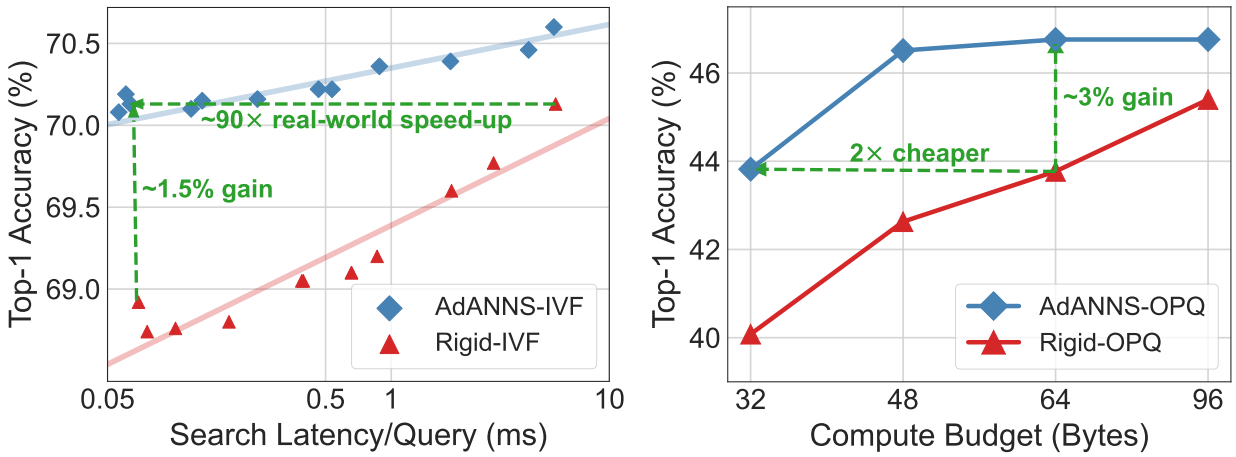
Our experiments on regular IVF with MRs and RRs (IVF-MR & IVF-RR) of varying dimensionalities and IVF configurations (# clusters, # probes) show that (Figure 5.1) matryoshka representations result in a significantly better accuracy-compute trade-off. We further studied and

found that learned lower-dimensional representations offer better accuracy-compute trade-offs for IVF than higher-dimensional embeddings (see Appendix H for more results).

AdANNS utilizes d-dimensional matryoshka representation to get accurate d_c and d_s dimensional vectors at no extra compute cost. The resulting AdANNS-IVF provides a much better accuracy-compute trade-off (Figure 5.1) on ImageNet-1K retrieval compared to IVF-MR, IVF-RR, and MG-IVF-RR – multi-granular IVF with rigid representations (akin to AdANNS without MR) – a strong baseline that uses d_c and d_s dimensional RRs. Finally, we exhaustively search the design space of IVF by varying $d_c, d_s \in [8, 16, \ldots, 2048]$ and the number of clusters $k \in [8, 16, \ldots, 2048]$. Please see Appendix H for more details. For IVF experiments on the NQ dataset, please refer to Appendix J.

Empirical results. Figure 5.1 shows that AdANNS-IVF outperforms the baselines across all accuracy-compute settings for ImageNet-1K retrieval. AdANNS-IVF results in $10\times$ lower compute for the best accuracy of the extremely expensive MG-IVF-RR and non-adaptive IVF-MR. Specifically, as shown in Figure 5.2a, AdANNS-IVF is up to 1.5% more accurate for the same compute and has up to $100\times$ lesser FLOPS/query ($90\times$ real-world speed-up!) than the status quo ANNS on rigid representations (IVF-RR). We filter out points for the sake of presentation and encourage the reader to check out Figure H.1 in Appendix H for an expansive plot of all the configurations searched.

The advantage of AdANNS for construction of search structures is evident from the improvements in IVF (AdANNS-IVF) and can be easily extended to other ANNS structures like ScaNN [33] and HNSW [78]. For example, HNSW consists of multiple layers with graphs of NSW graphs [80] of increasing complexity. AdANNS can be adopted to HNSW, where the construction of each level can be powered by appropriate dimensionalities for an optimal accuracy-compute trade-off. In general, AdANNS provides fine-grained control over compute overhead (storage, working memory, inference, and construction cost) during construction and inference while providing the best possible accuracy.



(a) Image retrieval on ImageNet-1K.

(b) Passage retrieval on Natural Questions.

Figure 5.2: AdANNS helps design search data structures and quantization methods with *better accuracy-compute trade-offs* than the existing solutions. In particular, (a) AdANNS-IVF improves on standard IVF by up to 1.5% in accuracy while being $90\times$ faster in deployment and (b) AdANNS-OPQ is as accurate as the baseline at *half the cost!* Rigid-IVF and Rigid-OPQ are standard techniques that are built on rigid representations (RRs) while AdANNS uses matryoshka representations (MRs) [64].

5.2 AdANNS-*OPQ*

Standard Product Quantization (PQ) essentially performs block-wise vector quantization via clustering. For example, suppose we need 32-byte PQ compressed vectors from the given 2048 dimensional representations. Then, we can chunk the representations in 32 equal blocks/sub-vectors of 64-d each, and each sub-vector space is clustered into $2^8 = 256$ partitions. That is, the representation of each point is essentially cluster-id for each block. Optimized PQ (OPQ) [29] further refines this idea, by first rotating the representations using a learned orthogonal matrix, and then applying PQ on top of the rotated representations. In ANNS, OPQ is used extensively to compress vectors and improves approximate distance computation primarily due to significantly lower memory

overhead than storing full-precision data points IVF.

AdANNS-OPQ utilizes MR representations to apply OPQ on lower-dimensional representations. That is, for a given quantization budget, AdANNS allows using top $d_s \ll d$ dimensions from MR and then computing clusters with d_s/B-dimensional blocks where B is the number of blocks. Depending on d_s and B, we have further flexibility of trading-off dimensionality/capacity for increasing the number of clusters to meet the given quantization budget. AdANNS-OPQ tries multiple d_s, B, and number of clusters for a fixed quantization budget to obtain the best performing configuration.

We experimented with $8 - 128$ byte OPQ budgets for both ImageNet and Natural Questions retrieval with an exhaustive search on the quantized vectors. We compare AdANNS-OPQ which uses MRs of varying granularities to the baseline OPQ built on the highest dimensional RRs. We also evaluate OPQ vectors obtained projection using SVD [31] on top of the highest-dimensional RRs.

Empirical results. Figures 5.3 and 5.2b show that AdANNS-OPQ significantly outperforms – up to 4% accuracy gain – the baselines (OPQ on RRs) across compute budgets on both ImageNet and NQ. In particular, AdANNS-OPQ tends to match the accuracy of a 64-byte (a typical choice in ANNS) OPQ baseline with only a 32-byte budget. This results in a $2\times$ reduction in both storage and compute FLOPS which translates to significant gains in real-world web-scale deployment (see Appendix G).

We only report the best AdANNS-OPQ for each budget typically obtained through a much lower-dimensional MR (128 & 192; much faster to build as well) than the highest-dimensional MR (2048 & 768) for ImageNet and NQ respectively (see Appendix J for more details). At the same time, we note that building compressed OPQ vectors on projected RRs using SVD to the smaller dimensions (or using low-dimensional RRs, see Appendix G) as the optimal AdANNS-OPQ does not help in improving the accuracy. The significant gains we observe in AdANNS-OPQ are purely due to better information packing in MRs – we hypothesize that packing the most important information in the initial coordinates results in a better PQ quantization than RRs where

the information is uniformly distributed across all the dimensions [64, 104]. See Appendix G for more details and experiments.

5.3 AdANNS *for Composite Indices*

We now extend AdANNS to composite indices [54] which put together two main ANNS building blocks – search structures and quantization – together to obtain efficient web-scale ANNS indices used in practice. A simple instantiation of a composite index would be the combination of IVF and OPQ – IVFOPQ – where the clustering in IVF happens with full-precision real vectors but the linear scan within each cluster is approximated using OPQ-compressed variants of the representation – since often the full-precision vectors of the database cannot fit in RAM. Contemporary

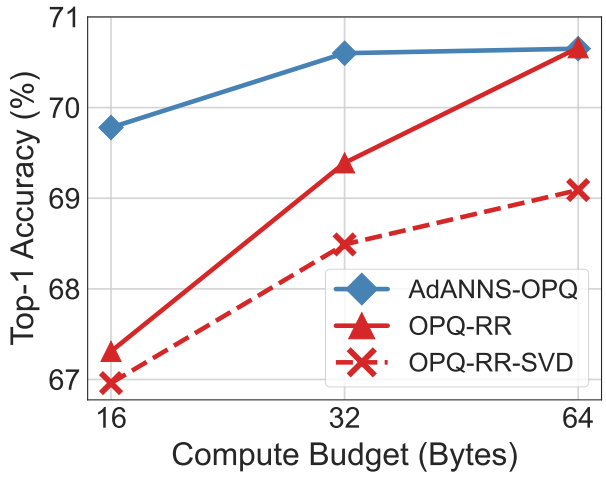


Figure 5.3: AdANNS-OPQ matches the accuracy of 64-byte OPQ on RR using only 32-bytes for ImageNet retrieval. AdANNS provides large gains at lower compute budgets and saturates to baseline performance for larger budgets.

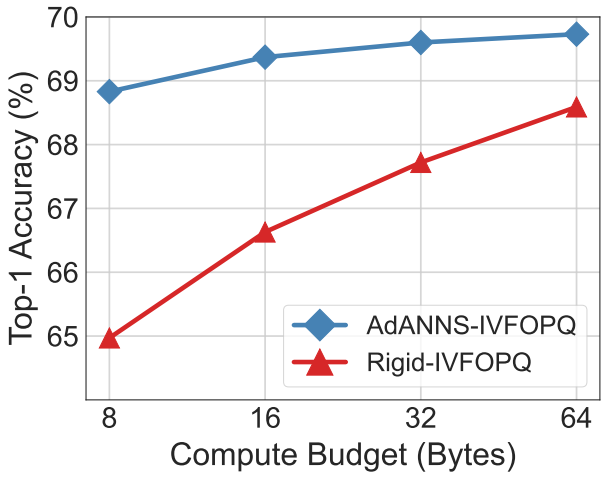


Figure 5.4: Combining the gains of AdANNS for IVF and OPQ leads to better IVFOPQ composite indices. On ImageNet retrieval, AdANNS-IVFOPQ is 8× cheaper for the same accuracy and provides 1 - 4% gains over IVFOPQ on RRs.

Table 5.1: AdANNS-DiskANN using a 16-d MR + re-ranking with the 2048-d MR outperforms DiskANN built on 2048-d RR at *half* the compute cost on ImageNet retrieval.

	RR-2048	AdANNS
PQ Budget (Bytes)	32	**16**
Top-1 Accuracy (%)	70.37	**70.56**
mAP@10 (%)	62.46	**64.70**
Precision@40 (%)	65.65	**68.25**

ANNS indices like DiskANN [51] make this a default choice where they build the search graph with a full-precision vector and approximate the distance computations during search with an OPQ-compressed vector to obtain a very small shortlist of retrieved datapoints. In DiskANN, the shortlist of data points is then re-ranked to form the final list using their full-precision vectors fetched from the disk. AdANNS is naturally suited to this shortlist-rerank framework: we use a low-d MR for forming index, where we could tune AdANNS parameters according to the accuracy-compute trade-off of the graph and OPQ vectors. We then use a high-d MR for re-ranking.

Empirical results. Figure 5.4 shows that AdANNS-IVFOPQ is $1-4\%$ better than the baseline at all the PQ compute budgets. Furthermore, AdANNS-IVFOPQ has the same accuracy as the baselines at $8\times$ lower overhead. With DiskANN, AdANNS accelerates shortlist generation by using low-dimensional representations and recoups the accuracy by re-ranking with the highest-dimensional MR at negligible cost. Table 5.1 shows that AdANNS-DiskANN is more accurate than the baseline for both 1-NN and ranking performance at only $half$ the cost. Using low-dimensional representations further speeds up inference in AdANNS-DiskANN (see Appendix I).

These results show the generality of AdANNS and its broad applicability across a variety of ANNS indices built on top of the base building blocks. Currently, AdANNS piggybacks on typical ANNS pipelines for their inherent accounting of the real-world system constraints [33, 51, 55].

However, we believe that AdANNS's flexibility and significantly better accuracy-compute trade-off can be further informed by real-world deployment constraints. We leave this high-potential line of work that requires extensive study to future research.

Chapter 6

FURTHER ANALYSIS AND DISCUSSION

6.1 Matryoshka Representation Learning

Robustness. We evaluate the robustness of the MRL models trained on ImageNet-1K on out-of-domain datasets, ImageNetV2/R/A/Sketch [91, 42, 44, 113], and compare them to the FF baselines. Table K.1 in Appendix K demonstrates that Matryoshka Representations for classification are at least as robust as the original representation while improving the performance on ImageNet-A by 0.6% – a 20% relative improvement. We also study the robustness in the context of retrieval by using ImageNetV2 as the query set for ImageNet-1K database. Table F.2 in Appendix F shows that MRL models have more robust retrieval compared to the FF baselines by having up to 3% higher mAP@10 performance. This observation also suggests the need for further investigation into robustness using nearest neighbour based classification and retrieval instead of the standard linear probing setup. We also find that the zero-shot robustness of ALIGN-MRL (Table K.2 in Appendix K) agrees with the observations made by Wortsman et al. [118]. Lastly, Table E.6 in Appendix E.2 shows that MRL also improves the cosine similarity span between positive and random image-text pairs.

Disagreement across Dimensions. The information packing in Matryoshka Representations often results in gradual increase of accuracy with increase in capacity. However, we observed that this trend was not ubiquitous and certain instances and classes were more accurate when evaluated with lower-dimensions (Figure L.1 in Appendix L). With perfect routing of instances to appropriate dimension, MRL can gain up to 4.6% classification accuracy. At the same time, the low-dimensional models are less accurate either due to confusion within the same superclass [27] of the ImageNet hierarchy or presence of multiple objects of interest. Figure 6.1 showcases 2

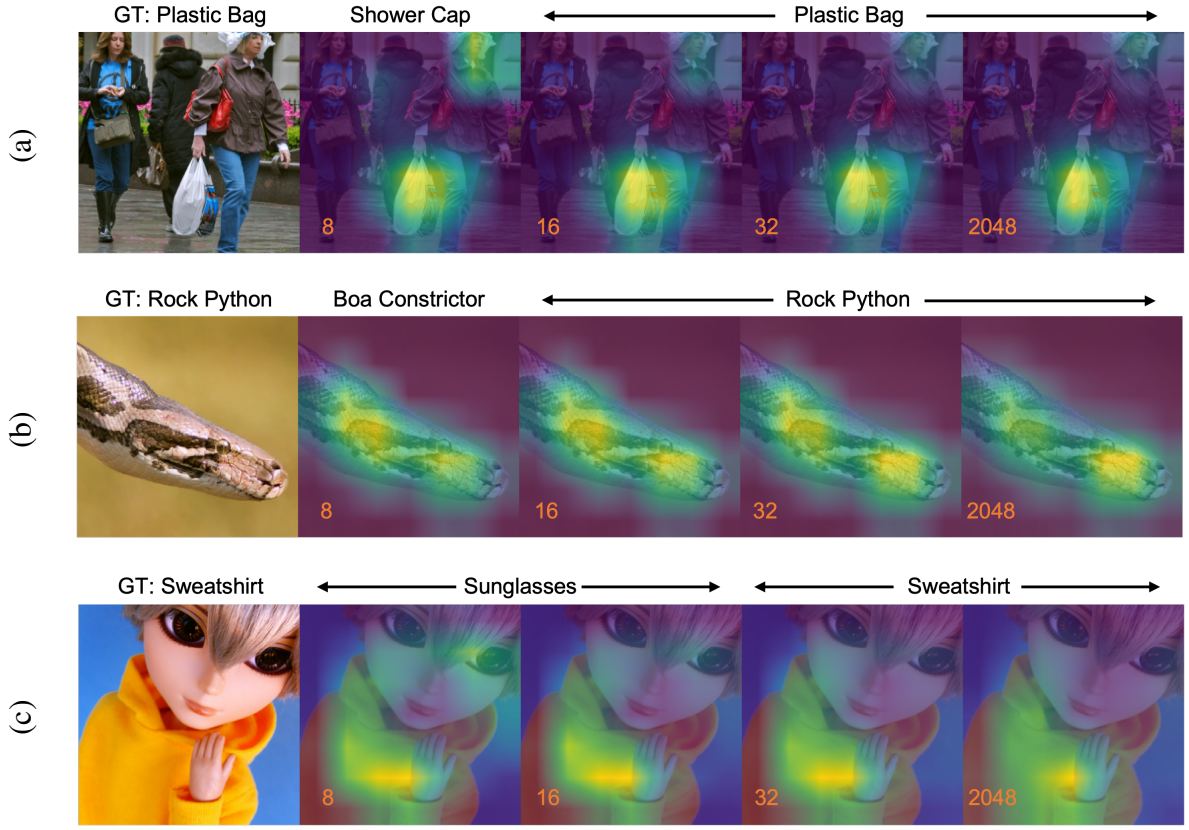


Figure 6.1: Grad-CAM [98] progression of predictions in MRL model across $8, 16, 32$ and 2048 dimensions. (a) 8-dimensional representation confuses due to presence of other relevant objects (with a larger field of view) in the scene and predicts "shower cap" ; (b) 8-dim model confuses within the same super-class of "boa" ; (c) 8 and 16-dim models incorrectly focus on the eyes of the doll ("sunglasses") and not the "sweatshirt" which is correctly in focus at higher dimensions; MRL fails gracefully in these scenarios and shows potential use cases of disagreement across dimensions.

such examples for 8-dimensional representation. These results along with Appendix L put forward the potential for MRL to be a systematic framework for analyzing the utility and efficiency of information bottlenecks.

Superclass Accuracy. As the information bottleneck becomes smaller, the overall accuracy on fine-grained classes decreases rapidly (Figure 4.2). However, the drop-off is not as significant when evaluated at a superclass level (Table L.3 in Appendix L). Figure 6.2 presents that this phenomenon occurs with both MRL and FF models; MRL is more accurate across dimensions. This shows that tight information bottlenecks while not highly accurate for fine-grained classification, do capture required semantic information for coarser classification that could be leveraged for adaptive routing for retrieval and classification. Mutifidelity of Matryoshka Representation naturally captures the underlying hierarchy of the class labels with one single model. Lastly, Figure 6.3 showcases the accuracy trends per superclass with MRL. The utility of additional dimensions in distinguishing a class from others within the same superclass is evident for "garment" which has up to 11% improvement for $8 \rightarrow 16$ dimensional representation transition. We also observed that superclasses such as "oscine (songbird)" had a clear visual distinction between the object and background and thus predictions using 8 dimensions also led to a good inter-class separability within the superclass.

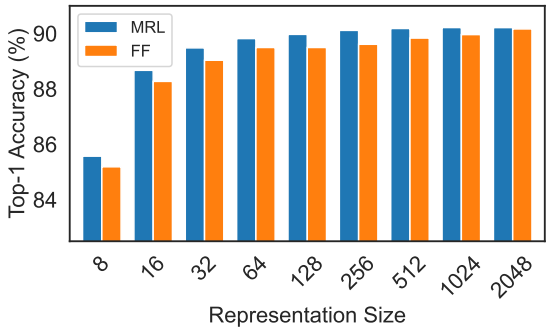


Figure 6.2: 31-way ImageNet-1K superclass classification across representation size for MRL & FF models showing the capture of underlying hierarchy through tight information bottlenecks.

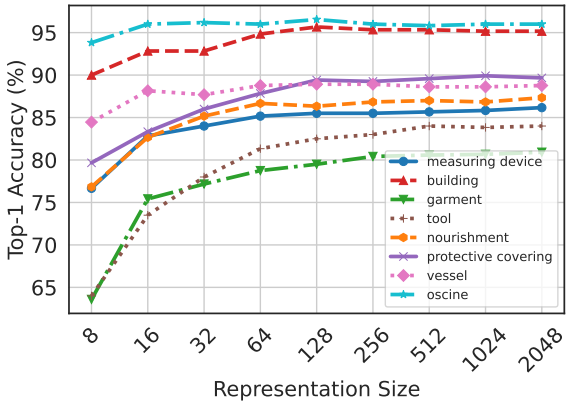


Figure 6.3: Diverse per-superclass accuracy trends across representation sizes for ResNet50-MRL on ImageNet-1K.

MRL Ablations Table M.1 in Appendix M presents that Matryoshka Representations can be enabled within off-the-shelf pretrained models with inexpensive partial finetuning thus paving a way for ubiquitous adoption of MRL. At the same time, Table M.2 in Appendix C indicates that with optimal weighting of the nested losses we could improve accuracy of lower-dimensions representations without accuracy loss. Tables M.3 and M.4 in Appendix C ablate over the choice of initial granularity and spacing of the granularites. Table M.3 reaffirms the design choice to shun extremely low dimensions that have poor classification accuracy as initial granularity for MRL while Table M.4 confirms the effectiveness of logarthmic granularity spacing inspired from the behaviour of accuracy saturation across dimensions over uniform. Lastly, Tables M.5 and M.6 in Appendix M.2 show that the retrieval performance saturates after a certain shortlist dimension and length depending on the complexity of the dataset.

6.2 Compute-aware Elastic Search During Inference

AdANNS search structures cater to many specific large-scale use scenarios that need to satisfy precise resource constraints during construction as well as inference. However, in many cases, construction and storage of the indices are not the bottlenecks or the user is unable to search the design space. In these settings, AdANNS-D enables adaptive inference through accurate yet cheaper distance computation using the low-dimensional prefix of matryoshka representation. Akin to composite indices (Section 5.3) that use PQ vectors for cheaper distance computation, we can use the low-dimensional MR for faster distance computation on ANNS structure built *non-adaptively* with a high-dimensional MR without any modifications to the existing index.

Empirical results. Figure 5.1 shows that for a given compute budget using IVF on ImageNet-1K retrieval, AdANNS-IVF is better than AdANNS-IVF-D due to the explicit control during the building of the ANNS structure which is expected. However, the interesting observation is that AdANNS-D *matches or outperforms* the IVF indices built with MRs of varying capacities for ImageNet retrieval.

However, these methods are applicable in specific scenarios of deployment. Obtaining optimal

AdANNS search structure (highly accurate) or even the best IVF-MR index relies on a relatively expensive design search but delivers indices that fit the storage, memory, compute, and accuracy constraints all at once. On the other hand AdANNS-D does not require a precisely built ANNS index but can enable compute-aware search during inference. AdANNS-D is a great choice for setups that can afford only one single database/index but need to cater to varying deployment constraints, e.g., one task requires 70% accuracy while another task has a compute budget of 1 MFLOPS/query.

6.3 Why MRs *over* RRs*?*

Quite a few of the gains from AdANNS are owing to the quality and capabilities of matryoshka representations. So, we conducted extensive analysis to understand why matryoshka representations seem to be more aligned for semantic search than the status-quo rigid representations.

Difficulty of NN search. Relative contrast (C_r) [37] is inversely proportional to the difficulty of nearest neighbor search on a given database. On ImageNet-1K, Figure M.5 shows that MRs have better C_r than RRs across dimensionalities, further supporting that matryoshka representations are more aligned (easier) for NN search than existing rigid representations for the same accuracy. More details and analysis about this experiment can be found in Appendix M.5.

Clustering distributions. We also investigate the potential deviation in clustering distributions for MRs across dimensionalities compared to RRs. Unlike the RRs where the information is uniformly diffused across dimensions [104], MRs have hierarchical information packing. Figure M.2 in Appendix M.3 shows that matryoshka representations result in clusters similar (measured by total variation distance [72]) to that of rigid representations and do not result in any unusual artifacts.

Robustness. Figure K.1 shows that MRs continue to be better than RRs even for out-of-distribution (OOD) image queries (ImageNetV2 [91]) using ANNS. It also shows that the highest data dimensionality need not always be the most robust which is further supported by the higher recall using lower dimensions. Further details about this experiment can be found in Appendix K.2.

Generality across encoders. IVF-MR consistently has higher accuracy than IVF-RR across dimensionalities despite having similar accuracies with exact NN search (for ResNet50 on Ima-

geNet and BERT-Base on NQ). We find that our observations on better alignment of MRs for NN search hold across neural network architectures, ResNet18/34/101 [40] and ConvNeXt-Tiny [75]. Appendix M.6 delves deep into the experimentation done using various neural architectures on ImageNet-1K.

Recall score analysis. Analysis of recall score (see Appendix D) in Appendix M.4 shows that for a similar top-1 accuracy, lower-dimensional representations have better 1-Recall@1 across search complexities for IVF and HNSW on ImageNet-1K. Across the board, MRs have higher recall scores and top-1 accuracy pointing to easier "searchability" and thus suitability of matryoshka representations for ANNS. Larger-scale experiments and further analysis can be found in Appendix M.

Through these analyses, we argue that matryoshka representations are better suited for semantic search than rigid representations, thus making them an ideal choice for AdANNS.

6.4 Limitations

MRL. The results in Section 6.1 reveal interesting weaknesses of MRL that would be logical directions for future work. (1) Optimizing the weightings of the nested losses to obtain a Pareto optimal accuracy-vs-efficiency trade-off – a potential solution could emerge from adaptive loss balancing aspects of anytime neural networks [48]. (2) Using different losses at various fidelities aimed at solving a specific aspect of adaptive deployment – e.g. high recall for 8-dimension and robustness for 2048-dimension. (3) Learning a search data-structure, like differentiable k-d tree, on top of Matryoshka Representation to enable dataset and representation aware retrieval. (4) Finally, the joint optimization of multi-objective MRL combined with end-to-end learnable search data-structure to have data-driven adaptive large-scale retrieval for web-scale search applications.

AdANNS. AdANNS's core focus is to improve the design of the existing ANNS pipelines. To use AdANNS on a corpus, we need to back-fill [90] the MRs of the data – a significant yet a one-time overhead. We also notice that high-dimensional MRs start to degrade in performance when optimizing also for an extremely low-dimensional granularity (e.g., < 24-d for NQ) – other-

wise is it quite easy to have comparable accuracies with both RRs and MRs. Lastly, the existing dense representations can only in theory be converted to MRs with an auto-encoder-style non-linear transformation. We believe most of these limitations form excellent future work to improve AdANNS further.

Chapter 7

CONCLUSIONS AND FUTURE DIRECTIONS

In conclusion, we presented Matryoshka Representation Learning (MRL), a flexible representation learning approach that encodes information at multiple granularities in a single embedding vector. This enables the MRL to adapt to a downstream task's statistical complexity as well as the available compute resources. We demonstrate that MRL can be used for large-scale adaptive classification as well as adaptive retrieval. On standard benchmarks, MRL matches the accuracy of the fixed-feature baseline despite using $14\times$ smaller representation size on average. Furthermore, the Matryoshka Representation based adaptive shortlisting and re-ranking system ensures comparable mAP@10 to the baseline while being $128\times$ cheaper in FLOPs and $14\times$ faster in wall-clock time. In addition, most of the efficiency techniques for model inference and vector search are complementary to MRL further assisting in deployment at the compute-extreme environments.

This adaptability of MRL to all real-world compute budgets motivated our proposed novel framework, AdANNS, that leverages adaptive representations for different phases of ANNS pipelines to improve the accuracy-compute tradeoff. AdANNS utilizes the inherent flexibility of Matryoshka Representations to design better ANNS building blocks than the standard ones which use the rigid representation in each phase. AdANNS achieves SOTA accuracy-compute trade-off for the two main ANNS building blocks: search data structures (AdANNS-IVF) and quantization (AdANNS-OPQ). Finally, the combination of AdANNS-based building blocks leads to the construction of better real-world composite ANNS indices – with as much as $8\times$ reduction in cost at the same accuracy as strong baselines – while also enabling compute-aware elastic search.

BIBLIOGRAPHY

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020.

[3] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

[4] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. *Advances in Neural Information Processing Systems*, 23, 2010.

[5] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36. JMLR Workshop and Conference Proceedings, 2012.

[6] Jon Louis Bentley. K-d trees for semidynamic point sets. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197, 1990.

[7] Erik Bernhardsson. *Annoy: Approximate Nearest Neighbors in C++/Python*, 2018. Python package version 1.13.0.

[8] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104, 2006.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Deng Cai. A revisit of hashing algorithms for approximate nearest neighbor search. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2337–2348, 2021.

[12] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[13] Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2643–2651, 2021.

[14] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932*, 2020.

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[16] Ting Chen, Lala Li, and Yizhou Sun. Differentiable product quantization for end-to-end embedding compression. In *International Conference on Machine Learning*, pages 1617–1626. PMLR, 2020.

[17] Kenneth L Clarkson. An algorithm for approximate closest-point queries. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 160–164, 1994.

[18] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[19] Jeffrey Dean. Challenges in building large-scale information retrieval systems. In *Keynote of the 2nd ACM International Conference on Web Search and Data Mining (WSDM)*, volume 10, 2009.

[20] Jia Deng, Alexander C Berg, and Li Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *CVPR 2011*, pages 785–792. IEEE, 2011.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11162–11173, 2021.

[23] 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.

[24] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

[25] Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3277, 2014.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019.

[28] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[29] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2953, 2013.

[30] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[31] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

[32] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[33] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pages 3887–3896. PMLR, 2020.

[34] Nilesh Gupta, Patrick H Chen, Hsiang-Fu Yu, Cho-Jui Hsieh, and Inderjit S Dhillon. End-to-end learning to index and search in large output spaces. *arXiv preprint arXiv:2210.08410*, 2022.

[35] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[36] Mike G Harris and Christos D Giachritsis. Coarse-grained information dominates fine-grained information in judgments of time-to-contact from retinal flow. *Vision research*, 40(6):601–611, 2000.

[37] Junfeng He, Sanjiv Kumar, and Shih-Fu Chang. On the difficulty of nearest neighbor search. In *International Conference on Machine Learning (ICML)*, 2012.

[38] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[39] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Jay Hegdé. Time course of visual perception: coarse-to-fine processing and beyond. *Progress in neurobiology*, 84(4):405–439, 2008.

[42] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.

[43] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[44] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021.

[45] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[46] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[47] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[48] Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3812–3821, 2019.

[49] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[50] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In

Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pages 528–536, 2019.

[51] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[53] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[54] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[55] William B Johnson. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[56] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[57] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[58] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[59] Tomoyuki Chikanaga Kaz Sato. Vertex ai matching engine. *Microsoft AI Blog*, 2021.

[60] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

[61] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*, pages 489–504, 2018.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[63] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2143–2157, 2009.

[64] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, December 2022.

[65] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

[66] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. *Advances in Neural Information Processing Systems*, 31, 2018.

[67] Aditya Kusupati, Matthew Wallingford, Vivek Ramanujan, Raghav Somani, Jae Sung Park, Krishna Pillutla, Prateek Jain, Sham Kakade, and Ali Farhadi. Llc: Accurate, multi-purpose learnt low-dimensional binary codes. *Advances in Neural Information Processing Systems*, 34:23900–23913, 2021.

[68] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[69] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. `https://github.com/libffcv/ffcv/`, 2022. commit 607d117.

[70] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[71] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. *Advances in Neural Information Processing Systems*, 29, 2016.

[72] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[73] W Li, Y Zhang, Y Sun, W Wang, W Zhang, and X Lin. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[74] Yoseph Linde, Andres Buzo, and Robert Gray. An algorithm for vector quantizer design. *IEEE Transactions on communications*, 28(1):84–95, 1980.

[75] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[76] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[77] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[78] Yu A Malkov and DA Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 42(04):824–836, 2020.

[79] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[80] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.

[81] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[82] Pabitra Mitra, CA Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[83] Pandu Nayak. Understanding searches better than ever before. *Google AI Blog*, 2019.

[84] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

[85] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[86] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[87] Yashoteja Prabhu, Aditya Kusupati, Nilesh Gupta, and Manik Varma. Extreme regression for dynamic search advertising. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 456–464, 2020.

[88] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[89] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

[90] Vivek Ramanujan, Pavan Kumar Anasosalu Vasu, Ali Farhadi, Oncel Tuzel, and Hadi Pouransari. Forward compatible training for representation learning. *arXiv preprint arXiv:2112.02805*, 2021.

[91] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[92] Aniket Rege, Aditya Kusupati, Alan Fan, Qingqing Cao, Sham Kakade, Prateek Jain, Ali Farhadi, et al. Adanns: A framework for adaptive semantic search. *arXiv preprint arXiv:2305.19435*, 2023.

[93] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014.

[94] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18, 2019.

[95] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[96] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419. PMLR, 2007.

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

[98] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-

based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[99] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[100] Harsha Vardhan Simhadri, George Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamy, Gopal Srinivasa, et al. Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search. *arXiv preprint arXiv:2205.03763*, 2022.

[101] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[102] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, IEEE International Conference on*, volume 3, pages 1470–1470. IEEE Computer Society, 2003.

[103] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[104] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[105] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[106] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[107] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[108] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[109] Manik Varma. Extreme classification. *Communications of the ACM*, 62(11):44–45, 2019.

[110] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[111] Charlie Waldburger. As search needs evolve, microsoft makes ai tools for better search available to researchers and developers. *Microsoft AI Blog*, 2019.

[112] Matthew Wallingford, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul Bhotika, and Stefano Soatto. Task adaptive parameter sharing for multi-task learning. *arXiv preprint arXiv:2203.16708*, 2022.

[113] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

[114] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 14(11):1964–1978, 2021.

[115] Xiaofang Wang, Dan Kondratyuk, Kris M Kitani, Yair Movshovitz-Attias, and Elad Eban. Multiple networks are more efficient than one: Fast and accurate models via ensembles and cascades. *arXiv preprint arXiv:2012.01988*, 2020.

[116] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[117] Ian H Witten, Ian H Witten, Alistair Moffat, Timothy C Bell, Timothy C Bell, Ed Fox, and Timothy C Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[118] Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*, 2021.

[119] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *arXiv preprint arXiv:1805.01978*, 2018.

[120] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[121] Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23(98):1–32, 2022.

[122] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[123] Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. Merlot reserve: Neural script knowledge through vision and language and sound. *arXiv preprint arXiv:2201.02639*, 2022.

[124] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

Appendix A

PSUEDOCODE

Algorithm 1 Pytorch code for Matryoshka Cross-Entropy Loss

```
class Matryoshka_CE_Loss(nn.Module):
   def __init__(self, relative_importance, **kwargs):
            super(Matryoshka_CE_Loss, self).__init__()
            self.criterion = nn.CrossEntropyLoss(**kwargs)
            self.relative_importance = relative_importance # usually set to
               all ones

  def forward(self, output, target):
            loss=0
            for i in range(len(output)):
              loss+= self.relative_importance[i] * self.criterion(output[i],
                 target)
            return loss
```

Algorithm 2 Pytorch code for MRL Linear Layer

```
class MRL_Linear_Layer(nn.Module):
      def __init__(self, nesting_list: List, num_classes=1000, efficient=False
          , **kwargs):
         super(MRL_Linear_Layer, self).__init__()
         self.nesting_list=nesting_list # set of m in M (Eq. 1)
         self.num_classes=num_classes
         self.is_efficient=efficient # flag for MRL-E

         if not is_efficient:
            for i, num_feat in enumerate(self.nesting_list):
               setattr(self, f"nesting_classifier_{i}", nn.Linear(num_feat,
                   self.num_classes, **kwargs))
         else:
            setattr(self, "nesting_classifier_0", nn.Linear(self.nesting_list
                [-1], self.num_classes, **kwargs)) # Instantiating one nn.
                Linear layer for MRL-E

      def forward(self, x):
            nesting_logits = ()
            for i, num_feat in enumerate(self.nesting_list):
                  if(self.is_efficient):
                        efficient_logit = torch.matmul(x[:, :num_feat], (self.
                            nesting_classifier_0.weight[:, :num_feat]).t())
                  else:
                        nesting_logits.append(getattr(self, f"
                            nesting_classifier_{i}")(x[:, :num_feat]))

            if(self.is_efficient):
                  nesting_logits.append(efficient_logit)

            return nesting_logits
```

Algorithm 3 AdANNS-IVF Psuedocode

```
# Index database to construct clusters and build inverted file system

def adannsConstruction(database, d_cluster, num_clusters):
   # Slice database with cluster construction dim (d_cluster)
   xb = database[:d_cluster]
   cluster_centroids = constructClusters(xb, num_clusters)

   return cluster_centroids


def adannsInference(queries, centroids, d_shortlist, d_search, num_probes, k):
   # Slice queries and centroids with cluster shortlist dim (d_shortlist)
   xq = queries[:d_shortlist]
   xc = centroids[:d_shortlist]

   for q in queries:
      # compute distance of query from each cluster centroid
      candidate_distances = computeDistances(q, xc)
      # sort cluster candidates by distance and choose small number to probe
      cluster_candidates = sortAscending(candidate_distances)[:num_probes]
      database_candidates = getClusterMembers(cluster_candidates)
      # Linear Scan all shortlisted clusters with search dim (d_search)
      k_nearest_neighbors[q] = linearScan(q, database_candidates, d_search, k)

   return k_nearest_neighbors
```

Appendix B

DATASETS

ImageNet-1K [95] contains 1,281,167 labeled train images, and 50,000 labelled validation images across 1,000 classes. The images were transformed with standard procedures detailed by FFCV [69].

ImageNet-4K dataset[1] was constructed by selecting 4,202 classes, non-overlapping with ImageNet-1K, from ImageNet-21K [21] with 1,050 or more examples. The train set contains 1,000 examples and the query/validation set contains 50 examples per class totalling to $\sim$4.2M and $\sim$200K respectively.

JFT-300M [105] is a large-scale multi-label dataset with 300M images labelled across 18,291 categories.

ALIGN [53] utilizes a large scale noisy image-text dataset containing 1.8B image-text pairs.

ImageNetV2 [91] is a collection of 10K images sampled a decade after the original construction of ImageNet [21]. ImageNetV2 contains 10 examples each from the 1,000 classes of ImageNet-1K.

ImageNet-A [44] contains 7.5K real-world adversarially filtered images from 200 ImageNet-1K classes.

ImageNet-R [42] contains 30K artistic image renditions for 200 of the original ImageNet-1K classes.

ImageNet-Sketch [113] contains 50K sketches, evenly distributed over all 1,000 ImageNet-1K classes.

ObjectNet [3] contains 50K images across 313 object classes, each containing $\sim$160 images each.

---

[1]More details can be found at `https://github.com/RAIVNLab/MRL/tree/main/imagenet-4k`

Appendix C

MODEL TRAINING AND COMPUTE COSTS

We trained all ResNet50–MRL models using the efficient dataloaders of FFCV [69]. We utilized the `rn50_40_epochs.yaml` configuration file of FFCV to train all MRL models defined below:

- MRL: ResNet50 model with the fc layer replaced by `MRL_Linear_Layer(efficient=False)`
- MRL–E: ResNet50 model with the fc layer replaced by `MRL_Linear_Layer(efficient=True)`
- FF–k: ResNet50 model with the fc layer replaced by `torch.nn.Linear(k, num_classes)`, where k $\in [8, 16, 32, 64, 128, 256, 512, 1024, 2048]$. We will henceforth refer to these models as simply FF, with the k value denoting representation size.

We trained all ResNet50 models with a learning rate of 0.475 with a cyclic learning rate schedule [103]. This was after appropriate scaling ($0.25\times$) of the learning rate specified in the configuration file to accommodate for 2xA100 NVIDIA GPUs available for training, compared to the 8xA100 GPUs utilized in the FFCV benchmarks. We trained with a batch size of 256 per GPU, momentum [106] of 0.9, and an SGD optimizer with a weight decay of 1e-4.

Our code (Appendix A) makes minimal modifications to the training pipeline provided by FFCV to learn Matryoshka Representations. As MRL makes minimal modifications to the ResNet50 model in the final fc layer via multiple heads for representations at various scales, it has only an 8MB storage overhead when compared to a standard ResNet50 model. MRL–E has no storage overhead as it has a shared head for logits at the final fc layer.

We trained ViT-B/16 models for JFT-300M on a 8x8 cloud TPU pod [57] using Tensorflow [1] with a batchsize of 128 and trained for 300K steps. Similarly, ALIGN models were trained using

Tensorflow on 8x8 cloud TPU pod for 1M steps with a batchsize of 64 per TPU. Both these models were trained with adafactor optimizer [99] with a linear learning rate decay starting at 1e-3.

Lastly, we trained a BERT-Base model on English Wikipedia and BookCorpus. We trained our models in Tensorflow using a 4x4 cloud TPU pod with a total batchsize of 1024. We used AdamW [77] optimizer with a linear learning rate decay starting at 1e-4 and trained for 450K steps.

In each configuration/case, if the final representation was normalized in the FF implementation, MRL models adopted the same for each nested dimension for a fair comparison. All MRL code is available at `https://github.com/RAIVNLab/MRL`.

Table C.1: Retrieval k-NN wall clock search times (s) over the entire validation (query) set of ImageNet-1K and ImageNet-4K, containing 50K and 200K samples respectively.

Rep. Size	ImageNet-1K		ImageNet-4K	
	ExactL2	HNSW32	ExactL2	HNSW32
8	0.60	0.14	35.70	1.17
16	0.57	0.18	36.16	1.65
32	0.60	0.20	36.77	1.75
64	0.66	0.24	27.88	2.21
128	0.86	0.32	30.10	4.15
256	1.29	0.46	34.97	3.39
512	2.17	0.68	46.97	4.83
1024	3.89	1.05	70.59	7.14
2048	7.31	2.05	117.78	13.43

C.1 Retrieval Experimentation and Costs

A bulk of our retrieval experimentation was written with Faiss [54], a library for efficient similarity search and clustering. More implementation details for retrieval are provided in Appendix F.

AdANNS was implemented from scratch due to difficulty in decoupling clustering and linear scan with Faiss, with code available at `https://github.com/RAIVNLab/AdANNS`. We also provide a version of AdANNS with Faiss optimizations with the restriction that $D_c \geq D_s$ as a limitation of the current implementation. All ANNS experiments (AdANNS-IVF, MG-IVF-RR, IVF-MR, IVF-RR, HNSW, HNSWOPQ, IVFOPQ) were run on an Intel Xeon 2.20GHz CPU with 12 cores. Exact Search experiments (Flat L2, PQ, OPQ) were run with CUDA 11.0 on a A100-SXM4 NVIDIA GPU with 40G RAM. The wall-clock inference times quoted in Figure 5.2a are reported on CPU with Faiss optimizations, and are averaged over three inference runs. Note that Exact search has a search time complexity of $O(dkN)$, and HNSW has a search time complexity of $O(dk\log(N))$, where N is the database size, d is the representation size, and k is the shortlist length. To examine real-world performance, we tabulated wall clock search time for every query in the ImageNet-1K and ImageNet-4K validation sets over all representation sizes d in Table C.1

Table C.2: FAISS [54] index size and build times for exact k-NN search with L2 Distance metric and approximate k-NN search with HNSW32 [79].

Rep. Size	Exact Search				HNSW32			
	ImageNet-1K		ImageNet-4K		ImageNet-1K		ImageNet-4K	
	Index Size (MB)	Index Build Time (s)	Index Size (MB)	Index Build Time (s)	Index Size (MB)	Index Build Time (s)	Index Size (MB)	Index Build Time (s)
8	40	0.04	131	0.33	381	4.87	1248	24.04
16	80	0.08	263	0.27	421	6.15	1379	33.31
32	160	0.16	525	0.52	501	6.80	1642	37.41
64	320	0.38	1051	1.05	661	8.31	2167	47.23
128	641	0.64	2101	2.10	981	11.73	3218	89.87
256	1281	1.27	4202	4.20	1622	17.70	5319	102.84
512	2562	2.52	8404	8.39	2903	27.95	9521	158.47
1024	5125	5.10	16808	17.20	5465	44.02	17925	236.30
2048	10249	10.36	33616	41.05	10590	86.15	34733	468.18

for both Exact Search and HNSW32, and ablated wall clock query time over shortlist length k on the ImageNet-1K validation set in Table C.3. The wall clock time to build the index and the index size is also shown in Table C.2.

Table C.3: Retrieval k-NN wall clock search times (s) over entire validation (query) set of ImageNet-1K over various shortlist lengths k.

Index	k = 50	k = 100	k = 200	k = 500	k = 1000	k = 2048
Exact L2	0.4406	0.4605	0.5736	0.6060	1.2781	2.7047
HNSW32	0.1193	0.1455	0.1833	0.2145	0.2333	0.2670

DPR [58] on NQ [68]. We follow the setup on the DPR repo[1]: the Wikipedia corpus has 21 million passages and Natural Questions dataset for open-domain QA settings. The training set contains 79,168 question and answer pairs, the dev set has 8,757 pairs and the test set has 3,610 pairs.

C.2 AdANNS *Inference Compute Cost*

We evaluate inference compute costs for IVF in MegaFLOPS per query (MFLOPS/query) as shown in Figures 5.1, M.1a, and H.1 as follows:

$$C = d_s k + \frac{n_p d_s N_D}{k}$$

where d_c is the **cluster** construction embedding dimensionality, d_s is the embedding dim used for linear **scan** within each **probed** cluster, which is controlled by # of search probes n_p. Finally, k is the number of clusters $|C_i|$ indexed over database of size N_D. The default setting in this work, unless otherwise stated, is $n_p = 1$, $k = 1024$, $N_D = 1281167$ (ImageNet-1K trainset). Vanilla IVF supports only $d_c = d_s$, while AdANNS-IVF-C provides flexibility via decoupling clustering

[1] https://github.com/facebookresearch/DPR

and search (Section 5). AdANNS-IVF-D is a special case of AdANNS-IVF-C with the flexibility restricted to inference, i.e., d_c is a fixed high-dimensional MR.

Appendix D

EVALUATION METRICS

In this work, we primarily use top-1 accuracy (i.e. 1-Nearest Neighbor), recall@k, corrected mean average precision (mAP@k) [67] and k-Recall@N (recall score), which are defined over all queries Q over indexed database of size N_D as:

$$\text{top-1} = \frac{\sum_Q \text{correct\_pred@1}}{|Q|}$$

$$\text{recall@k} = \frac{\sum_Q \text{correct\_pred@}k}{|Q|} * \frac{\text{num\_classes}}{|N_D|}$$

$$\text{precision@k} = \frac{\sum_Q \text{correct\_pred@}k}{|Q|\ k}$$

where correct_pred@k is the number of k-NN with correctly predicted labels for a given query. As noted in Section 3, k-Recall@N is the overlap between k exact search nearest neighbors (considered as ground truth) and the top N retrieved documents. As Faiss [54] supports a maximum of 2048-NN while searching the indexed database, we report 40-Recall@2048 in Figures M.3 and M.4. Also note that for ImageNet-1K, which constitutes a bulk of the experimentation in this work, $|Q| = 50000$, $|N_D| = 1281167$ and num_classes $= 1000$. For ImageNetv2, $|Q| = 10000$ and num_classes $= 1000$, and for ImageNet-4K, $|Q| = 210100$, $|N_D| = 4202000$ and num_classes $= 4202$.

Appendix E

MRL CLASSIFICATION

Table E.1: Top-1 classification accuracy (%) for ResNet50 MRL and baseline models on ImageNet-1K.

Rep. Size	Rand. LP	SVD	FF	Slim. Net	MRL	MRL–E
8	4.56	2.34	65.29	0.42	**66.63**	56.66
16	11.29	7.17	72.85	0.96	**73.53**	71.94
32	27.21	20.46	74.60	2.27	**75.03**	74.48
64	49.47	48.10	75.27	5.59	**75.82**	75.35
128	65.70	67.24	75.29	14.15	**76.30**	75.80
256	72.43	74.59	75.71	38.42	**76.47**	76.22
512	74.94	**76.78**	76.18	69.80	76.65	76.36
1024	76.10	**76.87**	76.63	74.61	76.76	76.48
2048	76.87	–	**76.87**	76.26	76.80	76.51

We show the top-1 classification accuracy of ResNet50–MRL models on ImageNet-1K in Table E.1 and Figure 4.1. We compare the performance of MRL models (MRL, MRL–E) to several baselines:

- FF: We utilize the FF-k models described in Appendix C for $k \in \{8, ...2048\}$.
- **SVD**: We performed a low rank approximation of the 1000-way classification layer of FF-2048, with rank = 1000.
- **Rand. LP**: We compared against a linear classifier fit on randomly selected features [39].

- **Slim. Net**: We take pretrained slimmable neural networks [122] which are trained with a flexible width backbone (25%, 50%, 75% and full width). For each representation size, we consider the first k dimensions for classification. Note that training of slimmable neural networks becomes unstable when trained below 25% width due to the hardness in optimization and low complexity of the model.

At lower dimensions ($d \leq 128$), MRL outperforms all baselines significantly, which indicates that pretrained models lack the multifidelity of Matryoshka Representations and are incapable of fitting an accurate linear classifier at low representation sizes.

Table E.2: 1-NN accuracy (%) on ImageNet-1K for various ResNet50 models.

Rep. Size	Rand. FS	SVD	JL	FF	Slimmable	MRL	MRL–E
8	2.36	19.14	0.11	58.93	1.00	62.19	57.45
16	12.06	46.02	0.09	66.77	5.12	67.91	67.05
32	32.91	60.78	0.06	68.84	16.95	69.46	68.6
64	49.91	67.04	0.05	69.41	35.60	70.17	69.61
128	60.91	69.63	0.06	69.35	51.16	70.52	70.12
256	65.75	70.67	0.04	69.72	60.61	70.62	70.36
512	68.77	71.06	0.03	70.18	65.82	70.82	70.74
1024	70.41	71.22	-	70.34	67.19	70.89	71.07
2048	71.19	71.21	-	71.19	66.10	70.97	71.21

We compared the performance of MRL models at various representation sizes via 1-nearest neighbors (1-NN) image classification accuracy on ImageNet-1K in Table E.2 and Figure 4.2. We provide detailed information regarding the k-NN search pipeline in Appendix F. We compared against a baseline of attempting to enforce nesting to a FF-2048 model by 1) Random Feature Selection (Rand. FS): considering the first m dimensions of FF-2048 for NN lookup, and 2) FF +SVD: performing SVD on the FF-2048 representations at the specified representation size, 3)

FF +JL: performing random projection according to the Johnson-Lindenstrauss lemma [55] on the FF-2048 representations at the specified representation size. We also compared against the 1-NN accuracy of slimmable neural nets [122] as an additional baseline. We observed these baseline models to perform very poorly at lower dimensions, as they were not explicitly trained to learn Matryoshka Representations.

*E.1 Adaptive Classification (*MRL*–AC)*

Table E.3: Threshold-based adaptive classification performance of ResNet50 MRL on a 40K sized held-out subset of the ImageNet-1K validation set. Results are averaged over 30 random held-out subsets.

Expected Rep. Size	Accuracy
13.43 ± 0.81	73.79 ± 0.10
18.32 ± 1.36	75.25 ± 0.11
25.87 ± 2.41	76.05 ± 0.15
36.26 ± 4.78	76.28 ± 0.16
48.00 ± 8.24	76.43 ± 0.18
64.39 ± 12.55	76.53 ± 0.19
90.22 ± 20.88	76.55 ± 0.20
118.85 ± 33.37	76.56 ± 0.20

In an attempt to use the smallest representation that works well for classification for every image in the ImageNet-1K validation set, we learned a policy to increase the representation size from m_i to m_{i+1} using a 10K sized subset of the ImageNet-1K validation set. This policy is based on whether the prediction confidence p_i using representation size m_i exceeds a learned threshold t_i^*. If $p_i \geq t_i^*$, we used predictions from representation size m_i otherwise, we increased to representation size m_{i+1}. To learn the optimal threshold t_i^*, we performed a grid search between 0 and 1 (100 samples). For each threshold t_k, we computed the classification accuracy over our 10K image sub-

set. We set t_i^* equal to the smallest threshold t_k that gave the best accuracy. We use this procedure to obtain thresholds for successive models, i.e., $\{t_j^* \mid j \in \{8, 16, 32, 64, \ldots, 2048\}\}$. To improve reliability of threshold based greedy policy, we use test time augmentation which has been used successfully in the past [101].

For inference, we used the remaining held-out 40K samples from the ImageNet-1K validation set. We began with smallest sized representation ($m = 8$) and compared the computed prediction confidence p_8 to learned optimal threshold t_8^*. If $p_8 \leq t_8^*$, then we increased $m = 16$, and repeated this procedure until $m = d = 2048$. To compute the expected dimensions, we performed early stopping at $m = \{16, 32, 64, \ldots 2048\}$ and computed the expectation using the distribution of representation sizes. As shown in Table E.3 and Figure 4.5, we observed that in expectation, we only needed a ~ 37 sized representation to achieve 76.3% classification accuracy on ImageNet-1K, which was roughly $14\times$ smaller than the FF–512 baseline. Even if we computed the expectation as a weighted average over the cumulative sum of representation sizes $\{8, 24, 56, \ldots\}$, due to the nature of multiple linear heads for MRL, we ended up with an expected size of 62 that still provided a roughly $8.2\times$ efficient representation than the FF–512 baseline. However, MRL–E alleviates this extra compute with a minimal drop in accuracy.

E.2 JFT, ALIGN and BERT

We examine the k-NN classification accuracy of learned $\mathrm{Matryoshka\ Representations}$ via ALIGN–MRL and JFT-ViT–MRL in Table E.4. For ALIGN [53], we observed that learning $\mathrm{Matryoshka\ Representations}$ via ALIGN–MRL improved classification accuracy at nearly all dimensions when compared to ALIGN. We observed a similar trend when training ViT-B/16 [26] for JFT-300M [105] classification, where learning $\mathrm{Matryoshka\ Representations}$ via MRL and MRL–E on top of JFT-ViT improved classification accuracy for nearly all dimensions, and significantly for lower ones. This demonstrates that training to learn $\mathrm{Matryoshka\ Representations}$ is feasible and extendable even for extremely large scale datasets. We also demonstrate that $\mathrm{Matryoshka\ Representations}$ are learned at interpolated dimensions for both ALIGN and JFT-ViT, as shown in Table E.5, despite not being trained explicitly at these dimensions. Lastly, Ta-

ble E.6 shows that MRL training leads to a increase in the cosine similarity span between positive and random image-text pairs.

Table E.4: ViT-B/16 and ViT-B/16-MRL top-1 and top-5 k-NN accuracy (%) for ALIGN and JFT. Top-1 entries where MRL–E and MRL outperform baselines are bolded for both ALIGN and JFT-ViT.

Rep. Size	ALIGN		ALIGN-MRL		JFT-ViT		JFT-ViT-MRL		JFT-ViT-MRL–E	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
12	11.90	28.05	**43.57**	67.36	27.07	48.57	**53.61**	75.30	**51.54**	73.94
24	33.35	55.58	**56.44**	78.19	48.64	70.20	**62.80**	81.51	**62.40**	81.36
48	51.32	73.15	**62.33**	82.30	63.58	81.80	**67.24**	84.37	**66.89**	83.80
96	61.82	81.97	**65.72**	84.61	68.56	85.13	**69.74**	85.86	**68.80**	85.13
192	66.71	85.27	**67.00**	85.36	71.32	86.21	**71.34**	86.62	**70.41**	86.01
384	67.65	85.70	**67.70**	85.73	71.67	86.98	**71.73**	87.08	71.18	86.46
768	68.00	86.10	67.85	85.85	72.10	87.20	71.85	86.92	71.31	86.62

We also evaluated the capability of Matryoshka Representations to extend to other natural language processing via masked language modeling (MLM) with BERT [23], whose results are tabulated in Table E.7. Without any hyper-parameter tuning, we observed Matryoshka Representations to be within 0.5% of FF representations for BERT MLM validation accuracy. This is a promising initial result that could help with large-scale adaptive document retrieval using BERT–MRL.

Table E.5: Examining top-1 and top-5 k-NN accuracy (%) at interpolated hidden dimensions for ALIGN and JFT. This indicates that MRL is able to scale classification accuracy as hidden dimensions increase even at dimensions that were not explicitly considered during training.

Interpolated Rep. Size	ALIGN-MRL		JFT-ViT-MRL	
	Top-1	Top-5	Top-1	Top-5
16	49.06	72.26	58.35	78.55
32	58.64	79.96	64.98	82.89
64	63.90	83.39	68.19	84.85
128	66.63	85.00	70.35	86.24
256	67.10	85.30	71.57	86.77
512	67.64	85.72	71.55	86.67

Table E.6: Cosine similarity between embeddings

Avg. Cosine Similarity	ALIGN	ALIGN-MRL
Positive Text to Image	0.27	0.49
Random Text to Image	8e-3	-4e-03
Random Image to Image	0.10	0.08
Random Text to Text	0.22	0.07

Table E.7: Masked Language Modelling (MLM) accuracy(%) of FF and MRL models on the validation set.

Rep. Size	BERT-FF	BERT-MRL
12	60.12	59.92
24	62.49	62.05
48	63.85	63.40
96	64.32	64.15
192	64.70	64.58
384	65.03	64.81
768	65.54	65.00

Appendix F

MRL IMAGE RETRIEVAL

We evaluated the strength of Matryoshka Representations via image retrieval on ImageNet-1K (the training distribution), as well as on out-of-domain datasets ImageNetV2 and ImageNet-4K for all MRL ResNet50 models. We generated the database and query sets, containing N and Q samples respectively, with a standard PyTorch [85] forward pass on each dataset. We specify the representation size at which we retrieve a shortlist of k-nearest neighbors (k-NN) by D_s. The database is a thus a [N, D_s] array, the query set is a [Q, D_s] array, and the neighbors set is a [Q, k] array. For metrics, we utilized corrected mean average precision (mAP@k) [67] and precision@k (Appendix D).

We performed retrieval with FAISS [54], a library for efficient similarity search. To obtain a shortlist of k-NN, we built an index to search the database. We performed an exhaustive NN search with the L2 distance metric with `faiss.IndexFlatL2`, as well as an approximate NN search (ANNS) via HNSW [54] with `faiss.IndexHNSWFlat`. We used HNSW with $M = 32$ unless otherwise mentioned, and henceforth referred to as HNSW32. The exact search index was moved to the GPU for fast k-NN search computation, whereas the HNSW index was kept on the CPU as it currently lacks GPU support. We show the wall clock times for building the index as well as the index size in Table C.2. We observed exact search to have a smaller index size which was faster to build when compared to HNSW, which trades off a larger index footprint for fast NN search (discussed in more detail in Appendix M). The database and query vectors are normalized with `faiss.normalize_L2` before building the index and performing search.

Retrieval performance on ImageNet-1K, *i.e.* the training distribution, is shown in Table F.1. MRL outperforms FF models for nearly all representation size for both top-1 and mAP@10, and especially at low representation size ($D_s \leq 32$). MRL–E loses out to FF significantly only at

$D_s = 8$. This indicates that training ResNet50 models via the MRL training paradigm improves retrieval at low representation size over models explicitly trained at those representation size (FF-8...2048).

We carried out all retrieval experiments at $D_s \in \{8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$, as these were the representation sizes which were a part of the `nesting_list` at which losses were added during training, as seen in Algorithm 1, Appendix A. To examine whether MRL is able to learn Matryoshka Representations at dimensions in between the representation size for which it was trained, we also tabulate the performance of MRL at interpolated $D_s \in \{12, 24, 48, 96, 192, 384, 768, 1536\}$ as MRL–Interpolated and MRL–E–Interpolated (see Table F.1). We observed that performance scaled nearly monotonically between the original representation size and the interpolated representation size as we increase D_s, which demonstrates that MRL is able to learn Matryoshka Representations at nearly all representation size $m \in [8, 2048]$ despite optimizing only for $|\mathcal{M}|$ nested representation sizes.

We examined the robustness of MRL for retrieval on out-of-domain datasets ImageNetV2 and ImageNet-4K, as shown in Table F.2 and Table F.3 respectively. On ImageNetV2, we observed that MRL outperformed FF at all D_s on top-1 Accuracy and mAP@10, and MRL–E outperformed FF at all D_s except $D_s = 8$. This demonstrates the robustness of the learned Matryoshka Representations for out-of-domain image retrieval.

F.1 Adaptive Retrieval

The time complexity of retrieving a shortlist of k-NN often scales as $O(d)$, where $d = D_s$, for a fixed k and N. We thus will have a theoretical $256\times$ higher cost for $D_s = 2048$ over $D_s = 8$. We discuss search complexity in more detail in Appendix C.1. In an attempt to replicate performance at higher D_s while using less FLOPs, we perform adaptive retrieval via retrieving a k-NN shortlist with representation size D_s, and then re-ranking the shortlist with representations of size D_r. Adaptive retrieval for a shortlist length $k = 200$ is shown in Table F.4 for ImageNet-1K, and in Table F.6 for ImageNet-4K. On ImageNet-1K, we are able to achieve comparable performance to retrieval with $D_s = 2048$ (from Table F.1) with $D_s = 16$ at $128\times$ less MFLOPs/Query (used interchangeably

with MFLOPs). Similarly, on ImageNet-4K, we are able to achieve comparable performance to retrieval with $D_s = 2048$ (from Table F.3) with $D_s = 64$ on ImageNet-1K and ImageNet-4K, at $32\times$ less MFLOPs. This demonstrates the value of intelligent routing techniques which utilize appropriately sized $\mathrm{Matryoshka\ Representations}$ for retrieval.

Funnel Retrieval. We also designed a simple cascade policy which we call funnel retrieval to successively improve and refine the k-NN shortlist at increasing D_s. This was an attempt to remove the dependence on manual choice of D_s & D_r. We retrieved a shortlist at D_s and then re-ranked the shortlist five times while simultaneously increasing D_r (rerank cascade) and decreasing the shortlist length (shortlist cascade), which resembles a funnel structure. We tabulate the performance of funnel retrieval in various configurations in Table F.5 on ImageNet-1K, and in Table F.7 on ImageNet-4K. With funnel retrieval on ImageNet-1K, we were able to achieve top-1 accuracy within 0.1% of retrieval with $D_s = 2048$ (as in Table F.1) with a funnel with $D_s = 16$, with $128\times$ less MFLOPs. Similarly, we are able to achieve equivalent top-1 accuracy within 0.15% of retrieval at $D_s = 2048$ (as in Table F.3) with funnel retrieval at $D_s = 32$ on ImageNet-4K, with $64\times$ less MFLOPs. This demonstrates that with funnel retrieval, we can emulate the performance of retrieval with $D_s = 2048$ with a fraction of the MFLOPs.

Table F.1: Retrieve a shortlist of 200-NN with D_s sized representations on ImageNet-1K via exact search with L2 distance metric. Top-1 and mAP@10 entries (%) where MRL–E and MRL outperform FF at their respective representation sizes are bolded.

Model	D_s	MFlops	Top-1	Top-5	Top-10	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
	8	10	58.93	75.76	80.25	53.42	52.29	51.84	51.57	59.32	59.28	59.25	59.21
	16	20	66.77	80.88	84.40	61.63	60.51	59.98	59.62	66.76	66.58	66.43	66.27
	32	41	68.84	82.58	86.14	63.35	62.08	61.36	60.76	68.43	68.13	67.83	67.48
	64	82	69.41	83.56	87.33	63.26	61.64	60.63	59.67	68.49	67.91	67.38	66.74
FF	128	164	69.35	84.23	88.24	62.30	60.16	58.73	57.29	67.84	66.83	65.96	64.92
	256	328	69.72	84.71	88.54	61.47	58.85	57.02	55.13	67.19	65.82	64.64	63.24
	512	656	70.18	85.04	88.91	61.37	58.41	56.26	53.98	67.12	65.49	64.07	62.35
	1024	1312	70.34	85.38	89.19	61.13	57.87	55.47	52.90	66.93	65.08	63.43	61.45
	2048	2624	71.19	85.66	89.17	62.90	60.06	57.99	55.76	68.46	66.9	65.52	63.83
	8	10	57.39	74.18	79.16	51.80	50.41	49.60	48.86	57.50	57.16	56.81	56.36
	16	20	**67.08**	81.38	85.15	61.60	60.36	59.66	59.04	66.79	66.53	66.24	65.87
	32	41	68.62	82.92	86.44	63.34	61.97	61.14	60.39	68.49	68.06	67.65	67.17
	64	82	**69.56**	83.49	86.85	**63.84**	62.33	61.43	60.57	68.93	68.4	67.96	67.38
MRL–E	128	164	**70.13**	83.63	87.07	**64.15**	62.58	61.61	60.70	69.19	68.62	68.11	67.50
	256	328	**70.39**	83.8	87.28	**64.35**	62.76	61.76	60.82	69.36	68.79	68.26	67.63
	512	656	**70.74**	83.91	87.33	**64.69**	63.05	62.06	61.14	69.63	69.00	68.50	67.88
	1024	1312	**71.05**	84.13	87.46	**64.85**	63.22	62.19	61.26	69.78	69.16	68.60	67.99
	2048	2624	71.17	84.27	87.67	**64.99**	63.33	62.29	61.33	69.90	69.24	68.68	68.05
	12	15	64.25	79.21	83.29	58.83	57.50	56.71	56.02	64.10	63.78	63.42	63.02
	24	31	68.28	82.31	85.89	62.75	61.41	60.62	59.92	67.89	67.49	67.11	66.69
	48	61	69.20	83.15	86.67	63.58	62.12	61.23	60.42	68.71	68.19	67.75	67.22
MRL–E	96	123	70.05	83.63	87.11	64.04	62.46	61.52	60.63	69.10	68.51	68.04	67.45
Interpolated	192	246	70.36	83.72	87.21	64.26	62.65	61.65	60.72	69.26	68.67	68.15	67.53
	384	492	70.54	83.88	87.28	64.55	62.94	61.93	61.01	69.51	68.92	68.40	67.78
	768	984	70.96	84.05	87.44	64.79	63.15	62.15	61.22	69.72	69.10	68.56	67.95
	1536	1968	71.19	84.17	87.57	64.94	63.29	62.26	61.32	69.85	69.21	68.66	68.04
	8	10	**62.19**	77.05	81.34	**56.74**	55.47	54.76	54.12	62.06	61.81	61.54	61.17
	16	20	**67.91**	81.44	85.00	**62.94**	61.79	61.16	60.64	67.93	67.71	67.48	67.20
	32	41	**69.46**	83.01	86.30	**64.21**	62.96	62.22	61.58	69.18	68.87	68.54	68.17
	64	82	**70.17**	83.53	86.95	**64.69**	63.33	62.53	61.80	69.67	69.25	68.89	68.42
MRL	128	164	**70.52**	83.98	87.25	**64.94**	63.50	62.63	61.83	69.93	69.44	69.02	68.50
	256	328	**70.62**	84.17	87.38	**65.04**	63.56	62.66	61.81	70.02	69.52	69.07	68.50
	512	656	**70.82**	84.31	87.55	**65.14**	63.57	62.62	61.73	70.12	69.53	69.04	68.45
	1024	1312	**70.89**	84.44	87.68	**65.16**	63.58	62.60	61.68	70.14	69.54	69.01	68.41
	2048	2624	70.97	84.41	87.74	**65.20**	63.57	62.56	61.60	70.18	69.52	68.98	68.35
	12	15	65.89	80.04	83.68	60.84	59.66	58.98	58.37	65.94	65.72	65.45	65.08
	24	31	68.76	82.48	85.87	63.64	62.42	61.74	61.13	68.64	68.35	68.07	67.71
	48	61	69.96	83.40	86.65	64.58	63.2	62.42	61.72	69.53	69.10	68.75	68.32
MRL	96	123	70.40	83.83	87.04	64.86	63.46	62.62	61.84	69.82	69.38	68.98	68.48
Interpolated	192	246	70.64	84.09	87.37	65.00	63.53	62.66	61.83	69.98	69.49	69.05	68.50
	384	492	70.69	84.25	87.41	65.09	63.56	62.64	61.76	70.05	69.51	69.04	68.46
	768	984	70.84	84.40	87.63	65.16	63.59	62.62	61.71	70.14	69.55	69.03	68.44
	1536	1968	70.88	84.39	87.71	65.18	63.59	62.58	61.64	70.16	69.54	68.99	68.38

Table F.2: Retrieve a shortlist of 200-NN with D_s sized representations on ImageNetV2 via exact search with L2 distance metric. Top-1 and mAP@10 entries (%) where MRL–E outperforms FF are bolded. MRL outperforms FF at all D_s and is thus not bolded.

Config	D_s	MFLOPs	Top-1	Top-5	Top-10	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
FF	8	10	48.79	64.70	69.72	43.04	41.89	41.42	41.17	48.43	48.27	48.25	48.19
	16	20	55.08	69.50	74.08	49.63	48.53	48.06	47.75	54.76	54.64	54.53	54.39
	32	41	56.69	71.10	76.47	51.11	49.85	49.17	48.65	56.23	55.96	55.71	55.42
	64	82	57.37	72.71	77.48	51.28	49.75	48.85	47.99	56.65	56.14	55.71	55.15
	128	164	57.17	73.31	78.64	50.07	48.09	46.79	45.58	55.75	54.89	54.12	53.28
	256	328	57.09	74.04	79.24	49.11	46.66	44.99	43.35	55.02	53.77	52.74	51.53
	512	656	57.12	73.91	79.32	48.95	46.25	44.37	42.42	54.88	53.49	52.29	50.83
	1024	1312	57.53	74.17	79.55	48.27	45.41	43.36	41.26	54.31	52.84	51.49	49.87
	2048	2624	57.84	74.59	79.45	49.99	47.47	45.66	43.87	55.89	54.63	53.45	52.12
MRL–E	8	10	47.05	62.53	67.60	40.79	39.47	38.78	38.16	46.03	45.77	45.54	45.17
	16	20	**55.73**	70.54	74.86	**49.86**	48.57	47.84	47.26	54.97	54.71	54.44	54.10
	32	41	**57.33**	71.61	76.64	**51.26**	49.92	49.09	48.42	56.46	56.11	55.70	55.30
	64	82	**57.90**	72.55	77.44	**51.89**	50.29	49.34	48.53	57.06	56.45	55.97	55.43
	128	164	**57.73**	72.79	77.28	**52.02**	50.38	49.49	48.62	57.13	56.58	56.15	55.58
	256	328	**58.22**	72.77	77.67	**52.16**	50.61	49.67	48.81	57.30	56.79	56.33	55.77
	512	656	**58.46**	73.00	77.88	**52.52**	50.97	50.02	49.16	57.65	57.10	56.64	56.08
	1024	1312	**58.71**	73.29	78.00	**52.70**	51.13	50.17	49.30	57.83	57.26	56.77	56.20
	2048	2624	**58.86**	73.17	78.00	**52.88**	51.25	50.26	49.36	57.95	57.35	56.85	56.25
MRL	8	10	**50.41**	65.56	70.27	**45.51**	44.38	43.71	43.17	50.55	50.44	50.17	49.91
	16	20	**56.64**	70.19	74.61	**50.98**	49.76	49.16	48.69	55.90	55.66	55.52	55.29
	32	41	**57.96**	71.88	76.41	**52.06**	50.78	50.09	49.54	57.18	56.83	56.57	56.27
	64	82	**58.94**	72.74	77.17	**52.65**	51.24	50.44	49.76	57.72	57.29	56.94	56.52
	128	164	**59.13**	73.07	77.49	**52.94**	51.42	50.53	49.74	58.00	57.47	57.05	56.55
	256	328	**59.18**	73.64	77.75	**52.96**	51.45	50.52	49.70	58.01	57.53	57.06	56.54
	512	656	**59.40**	73.85	77.97	**53.01**	51.39	50.46	49.61	58.11	57.49	57.04	56.48
	1024	1312	**59.11**	73.77	77.92	**52.98**	51.37	50.40	49.54	58.13	57.51	57.00	56.45
	2048	2624	**59.63**	73.84	77.97	**52.96**	51.34	50.34	49.44	58.07	57.48	56.95	56.36

Table F.3: Retrieve a shortlist of 200-NN with D_s sized representations on ImageNet-4K via exact search with L2 distance metric. MRL–E and FF models are omitted for clarity and compute/inference time costs. All entries are in %.

Config	D_s	MFLOPs	Top-1	Top-5	Top-10	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
MRL	8	34	10.60	26.23	35.57	5.32	4.29	3.76	3.36	9.13	8.77	8.46	8.13
	16	67	16.74	36.91	47.28	8.64	6.83	5.84	5.05	13.82	12.79	12.04	13.27
	32	134	21.54	43.75	54.11	11.36	8.88	7.47	6.31	17.25	15.67	14.47	13.27
	64	269	25.00	47.97	58.25	13.38	10.40	8.67	7.23	19.68	17.64	16.14	14.65
	128	538	27.27	50.35	60.47	14.77	11.47	9.53	7.91	21.25	18.95	17.26	15.59
	256	1076	28.53	51.95	61.90	15.66	12.19	10.12	8.38	22.28	19.81	18.01	16.22
	512	2151	29.46	53.03	62.81	16.29	12.70	10.55	8.72	22.96	20.42	18.54	16.68
	1024	4303	30.23	53.72	63.45	16.76	13.08	10.86	8.97	23.48	20.88	18.93	17.00
	2048	8606	30.87	54.32	64.02	17.20	13.43	11.14	9.19	23.97	21.28	19.28	17.30
MRL-Interpolated	12	50	14.04	32.56	42.71	7.16	5.70	4.92	4.32	11.81	11.08	10.52	9.94
	24	101	19.49	40.82	51.26	10.17	7.98	6.75	5.75	15.76	14.43	13.42	12.40
	48	202	23.51	46.23	56.56	12.49	9.72	8.13	6.81	18.62	16.75	15.39	14.04
	96	403	26.25	49.32	59.48	14.15	11.00	9.15	7.61	20.55	18.36	16.78	15.17
	192	807	27.94	51.32	61.32	15.29	11.89	9.88	8.18	21.86	19.46	17.71	15.96
	384	1614	29.03	52.53	62.45	15.99	12.46	10.35	8.56	22.64	20.14	18.29	16.47
	768	3227	29.87	53.36	63.13	16.54	12.90	10.71	8.85	23.23	20.67	18.75	16.85
	1536	6454	30.52	54.02	63.79	16.99	13.27	11.01	9.08	23.73	21.09	19.12	17.16

Table F.4: Retrieve a shortlist of k-NN with D_s sized representations on ImageNet-1K with MRL representations, and then re-order the neighbors shortlist with L2 distances using D_r sized representations. Top-1 and mAP@10 entries (%) that are within 0.1% of the maximum value achievable without reranking on MRL representations, as seen in Table F.1, are bolded.

Shortlist Length = 200

D_s	D_r	MFLOPs	Top-1	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
8	16	10	68.21	63.35	62.25	61.70	61.19	68.32	68.14	67.96	67.65
	32		69.42	64.12	62.81	62.03	61.32	69.04	68.63	68.22	67.71
	64		70.05	64.46	63.03	62.14	61.29	69.37	68.83	68.32	67.66
	128		70.34	64.68	63.16	62.21	61.27	69.59	68.96	68.38	67.65
	256		70.40	64.77	63.21	62.23	61.26	69.66	69.02	68.41	67.65
	512		70.60	64.86	63.22	62.21	61.22	69.74	69.02	68.39	67.62
	1024		70.71	64.88	63.23	62.20	61.20	69.76	69.01	68.39	67.60
	2048		70.81	64.90	63.22	62.17	61.16	69.77	68.99	68.36	67.57
16	32	21	69.47	64.27	63.04	62.36	61.75	69.21	68.90	68.58	68.12
	64		70.16	64.74	63.42	62.66	61.94	69.66	69.22	68.81	68.22
	128		70.52	65.00	63.60	62.77	61.98	69.91	69.36	68.89	68.24
	256		70.55	**65.10**	63.67	62.82	62.01	69.98	69.43	68.92	68.25
	512		70.74	**65.21**	63.70	62.83	62.00	70.08	69.43	68.92	68.24
	1024		70.83	**65.23**	63.72	62.83	61.99	70.08	69.45	68.92	68.23
	2048		**70.90**	**65.27**	63.73	62.82	61.97	70.10	69.44	68.90	68.21
32	64	41	70.16	64.69	63.35	62.57	61.93	69.68	69.26	68.92	68.51
	128		70.52	64.97	63.54	62.73	62.04	69.95	69.47	69.06	68.59
	256		70.63	65.07	63.63	62.79	62.07	70.04	69.55	69.12	68.61
	512		70.82	**65.17**	63.66	62.80	62.06	70.11	69.57	69.12	68.60
	1024		**70.89**	**65.20**	63.68	62.80	62.04	70.15	69.59	69.12	68.59
	2048		**70.97**	**65.24**	63.70	62.79	62.02	70.19	69.59	69.10	68.56
64	128	82	70.51	64.94	63.50	62.64	61.88	69.94	69.44	69.02	68.54
	256		70.63	65.04	63.57	62.69	61.91	70.02	69.52	69.08	68.57
	512		70.83	**65.14**	63.59	62.67	61.87	70.12	69.54	69.06	68.54
	1024		**70.89**	**65.16**	63.59	62.65	61.85	70.15	69.54	69.05	68.52
	2048		**70.97**	**65.20**	63.59	62.63	61.82	70.18	69.53	69.03	68.49
128	256	164	70.63	65.04	63.56	62.66	61.82	70.02	69.52	69.07	68.51
	512		70.82	**65.14**	63.58	62.63	61.77	70.11	69.54	69.04	68.47
	1024		**70.89**	**65.16**	63.58	62.60	61.73	70.14	69.54	69.02	68.45
	2048		**70.97**	**65.20**	63.57	62.57	61.68	70.18	69.52	68.99	68.41
256	512	328	70.82	**65.14**	63.57	62.62	61.74	70.12	69.53	69.04	68.45
	1024		**70.88**	**65.16**	63.58	62.60	61.69	70.14	69.54	69.01	68.41
	2048		**70.97**	**65.20**	63.56	62.56	61.62	70.18	69.52	68.98	68.37
512	1024	656	**70.90**	**65.16**	63.58	62.60	61.68	70.14	69.54	69.01	68.41
	2048		**70.98**	**65.20**	63.57	62.56	61.60	70.18	69.52	68.98	68.35
1024	2048	1312	**70.97**	**65.20**	63.57	62.56	61.60	70.18	69.52	68.98	68.35

Table F.5: Retrieve a shortlist of k-NN with D_s sized representations on ImageNet-1K with MRL. This shortlist is then reranked with funnel retrieval, which uses a rerank cascade with a one-to-one mapping with a monotonically decreasing shortlist length as shown in the shortlist cascade. Top-1 and mAP@10 entries (%) within 0.1% of the maximum achievable without reranking on MRL representations, as seen in Table F.1, are bolded.

D_s	Rerank Cascade	Shortlist Cascade	MFLOPs	Top-1	Top-5	Top-10	mAP@10	P@10
8	16→32→64→128→2048	200→100→50→25→10	10.28	70.22	82.63	85.49	64.06	68.65
		400→200→50→25→10	10.29	70.46	83.13	86.08	64.43	69.10
		800→400→200→50→10	10.31	70.58	83.54	86.53	64.62	69.37
16	32→64→128→256→2048	200→100→50→25→10	20.54	**70.90**	83.96	86.85	**65.19**	69.97
		400→200→50→25→10	20.56	**70.95**	84.05	87.04	**65.18**	70.00
		800→400→200→50→10	20.61	**70.96**	84.18	87.22	**65.14**	70.01
32	64→128→256→512→2048	200→100→50→25→10	41.07	**70.96**	84.32	87.47	**65.21**	70.11
		400→200→50→25→10	41.09	**70.97**	84.32	87.47	**65.19**	70.11
		800→400→200→50→10	41.20	**70.97**	84.36	87.53	**65.18**	70.11

Table F.6: Retrieve a shortlist of k-NN with D_s sized representations on ImageNet-4K with MRL representations, and then re-order the neighbors shortlist with L2 distances using D_r sized representations. Top-1 and mAP@10 entries (%) that are within 0.1% of the maximum value achievable without reranking on MRL representations, as seen in Table F.3, are bolded.

Shortlist Length = 200

D_s	D_r	MFLOPs	Top-1	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
8	16	34	16.84	8.70	6.88	5.88	5.08	13.86	12.80	11.98	11.10
	32		20.73	10.66	8.19	6.77	5.61	16.18	14.39	13.02	11.61
	64		23.11	11.91	9.03	7.36	6.00	17.56	15.34	13.67	11.99
	128		24.63	12.71	9.59	7.76	6.25	18.42	15.94	14.08	12.22
	256		25.5	13.24	9.96	8.03	6.42	19.00	16.35	14.36	12.37
	512		26.07	13.59	10.21	8.20	6.53	19.37	16.62	14.54	12.46
	1024		26.52	13.85	10.40	8.34	6.61	19.65	16.80	14.68	12.53
	2048		26.94	14.11	10.57	8.45	6.68	19.92	16.98	14.79	12.58
16	32	67	21.44	11.24	8.72	7.26	6.02	17.02	15.30	13.92	12.41
	64		24.36	12.78	9.75	7.96	6.43	18.72	16.41	14.63	12.74
	128		26.08	13.70	10.39	8.39	6.69	19.68	17.07	15.05	12.94
	256		26.99	14.27	10.79	8.67	6.85	20.27	17.48	15.31	13.07
	512		27.60	14.66	11.06	8.86	6.97	20.67	17.75	15.50	13.16
	1024		28.12	14.94	11.26	8.99	7.05	20.96	17.95	15.62	13.22
	2048		28.56	15.21	11.43	9.11	7.12	21.23	18.13	15.73	13.27
32	64	134	24.99	13.35	10.35	8.59	7.09	19.61	17.52	15.92	14.21
	128		27.17	14.61	11.27	9.26	7.51	20.99	18.52	16.62	14.59
	256		28.33	15.37	11.83	9.67	7.77	21.80	19.12	17.05	14.81
	512		29.12	15.88	12.20	9.94	7.93	22.33	19.51	17.32	14.94
	1024		29.78	16.25	12.47	10.13	8.05	22.71	19.79	17.5	15.03
	2048		30.33	16.59	12.72	10.30	8.16	23.07	20.05	17.66	15.11
64	128	269	27.27	14.76	11.47	9.51	7.85	21.25	18.92	17.20	15.40
	256		28.54	15.64	12.15	10.05	8.21	22.24	19.71	17.81	15.76
	512		29.45	16.25	12.62	10.40	8.44	22.88	20.24	18.20	15.97
	1024		30.19	16.69	12.96	10.66	8.60	23.35	20.61	18.46	16.10
	2048		**30.81**	**17.10**	13.27	10.88	8.74	23.79	20.93	18.69	16.21
128	256	538	28.54	15.66	12.19	10.12	8.36	22.28	19.81	18.00	16.16
	512		29.45	16.29	12.69	10.53	8.66	22.96	20.41	18.50	16.48
	1024		30.22	16.76	13.07	10.83	8.86	23.47	20.84	18.83	16.68
	2048		**30.86**	**17.19**	13.41	11.09	9.03	23.95	21.22	19.12	16.84
256	512	1076	29.45	16.29	12.70	10.55	8.71	22.97	20.42	18.54	16.66
	1024		30.21	16.76	13.08	10.86	8.95	23.48	20.87	18.92	16.94
	2048		**30.85**	**17.20**	13.43	11.14	9.15	23.97	21.27	19.26	17.16
512	1024	2152	30.22	16.76	13.08	10.86	8.97	23.48	20.88	18.93	17.00
	2048		**30.87**	**17.20**	13.43	11.14	9.19	23.97	21.28	19.28	17.28
1024	2048	4303	**30.87**	**17.20**	13.43	11.15	9.19	23.97	21.28	19.28	17.29

Table F.7: Retrieve a shortlist of k-NN with D_s sized representations on ImageNet-4K with MRL. This shortlist is then reranked with funnel retrieval, which uses a rerank cascade with a one-to-one mapping with a monotonically decreasing shortlist length as shown in the shortlist cascade. Top-1 and mAP@10 entries (%) within 0.15% of the maximum achievable without reranking on MRL representations, as seen in Table F.3, are bolded.

D_s	Rerank Cascade	Shortlist Cascade	MFLOPs	Top-1	Top-5	Top-10	mAP@10	P@10
8	16→32→64→128→2048	200→100→50→25→10	33.65	26.20	46.45	54.12	12.79	17.85
		400→200→50→25→10	33.66	26.55	47.02	54.72	13.02	18.15
		800→400→200→50→10	33.68	26.83	47.54	55.35	13.24	18.44
16	32→64→128→256→2048	200→100→50→25→10	67.28	29.51	51.44	59.56	15.27	21.03
		400→200→50→25→10	67.29	29.66	51.71	59.88	15.42	21.22
		800→400→200→50→10	67.34	29.79	52.00	60.25	15.55	21.41
32	64→128→256→512→2048	200→100→50→25→10	134.54	30.64	53.52	62.16	16.45	22.64
		400→200→50→25→10	134.56	30.69	53.65	62.31	16.51	22.73
		800→400→200→50→10	134.66	**30.72**	53.78	62.43	16.55	22.79
64	128→256→512→1024→2048	200→100→50→25→10	269.05	**30.81**	54.06	63.15	16.87	23.34
		400→200→50→25→10	269.10	**30.84**	54.20	63.31	16.92	23.42
		800→400→200→50→10	269.31	**30.87**	54.27	63.42	16.95	23.46

Appendix G

ADANNS-**OPQ**

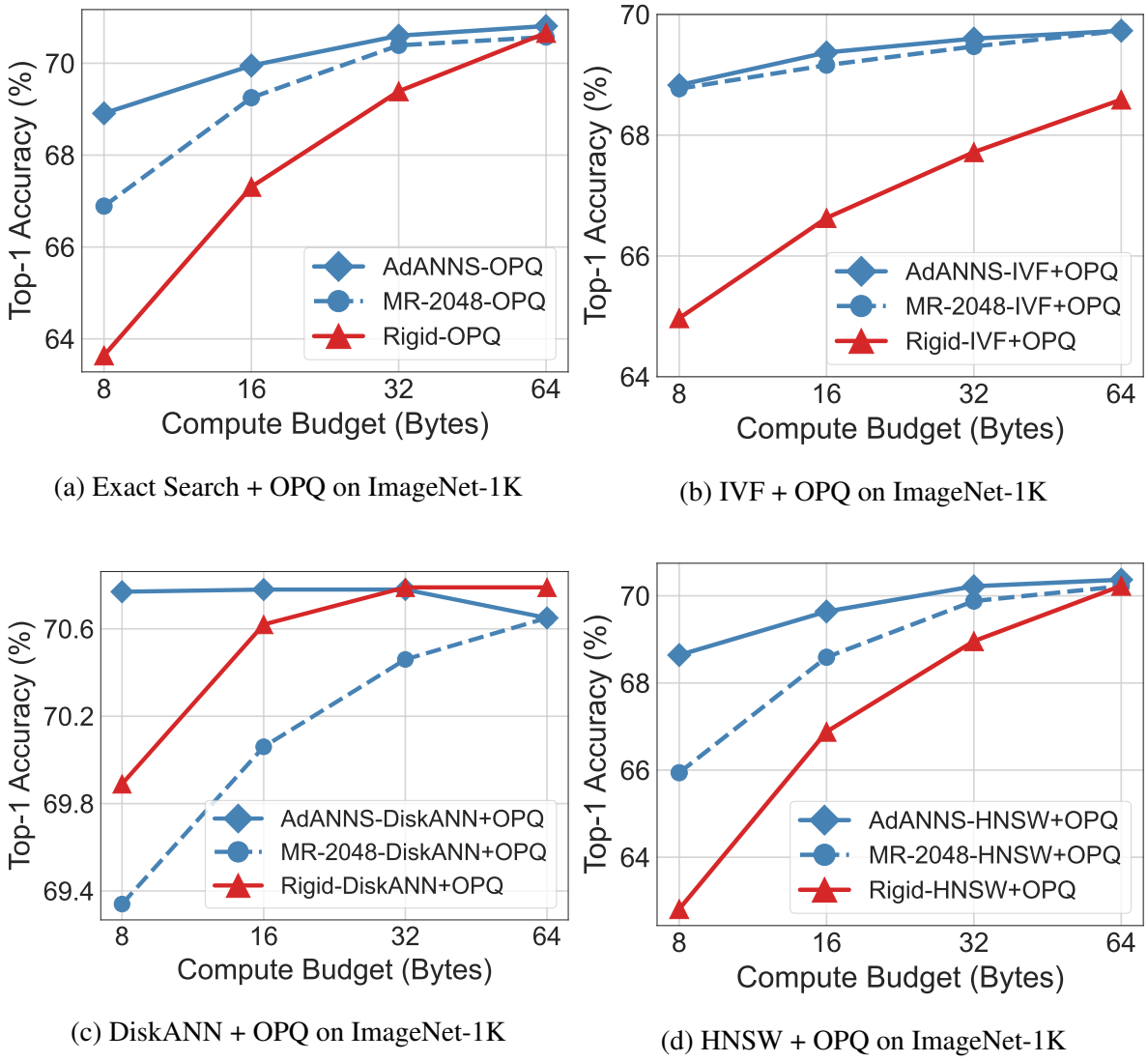
In this section, we take a deeper dive into the quantization characteristics of MR. In this work, we restrict our focus to optimized product quantization (OPQ) [29], which adds a learned space rotation and dimensionality permutation to PQ's sub-vector quantization to learn more optimal PQ codes.

We perform a study of composite OPQ $m \times b$ indices on ImageNet-1K across compression compute budgets m (where $b = 8$, i.e. 1 Byte), i.e. Exact Search with OPQ, IVF+OPQ, HNSW+OPQ, and DiskANN+OPQ, as seen in Figure G.1. It is evident from these results:

1. Learning OPQ codebooks with AdANNS (Figure G.1a) provides a 1-5% gain in top-1 accuracy over rigid representations at low compute budgets ($\leq$ 32 Bytes). AdANNS-OPQ saturates to Rigid-OPQ performance at low compression ($\geq$ 64 Bytes).

2. For IVF, learning clusters with MRs instead of RRs (Figure G.1b) provides substantial gains (1-4%). In contrast to Exact-OPQ, using AdANNS for learning OPQ codebooks does not provide substantial top-1 accuracy gains over MR with $d = 2048$ (highest), though it is still slightly better or equal to MR-2048 at all compute budgets. This further supports that IVF performance generally scales with embedding dimensionality, which is consistent with our findings on ImageNet across robustness variants and encoders (See Figures K.1 and M.6 respectively).

3. Note that in contrast to Exact, IVF, and HNSW coarse quantizers, DiskANN *inherently reranks* the retrieved shortlist with high-precision embeddings ($d = 2048$), which is reflected in its high top-1 accuracy. We find that AdANNS with 8-byte OPQ (Figure G.1c) matches the top-1 accuracy of rigid representations using 32-byte OPQ, for a $4\times$ cost reduction for the same accuracy. Also note that using AdANNS provides large gains over using MR-2048 at



(a) Exact Search + OPQ on ImageNet-1K

(b) IVF + OPQ on ImageNet-1K

(c) DiskANN + OPQ on ImageNet-1K

(d) HNSW + OPQ on ImageNet-1K

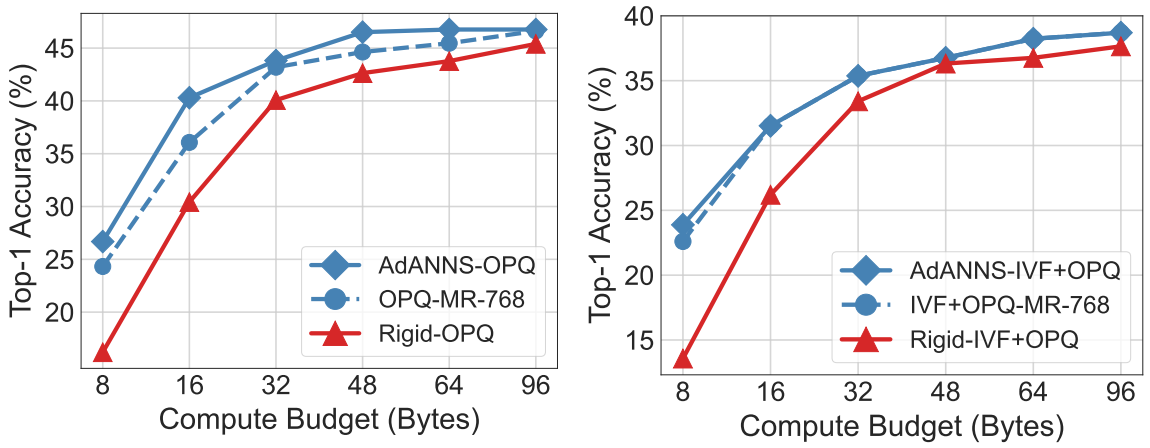
Figure G.1: Top-1 Accuracy of AdANNS composite indices with OPQ distance computation compared to MR and Rigid baselines models on ImageNet-1K and Natural Questions.

high compression (1.5%), highlighting the necessity of AdANNS's flexibility for high-precision retrieval at low compute budgets.

4. Our findings on the HNSW-OPQ composite index (Figure G.1d) are consistent with all other

indices, i.e. HNSW graphs constructed with AdANNS OPQ codebooks provide significant gains over RR and MR, especially at high compression (≤ 32 Bytes).

OPQ on NQ dataset



(a) Exact Search + OPQ on Natural Questions

(b) IVF + OPQ on Natural Questions

Figure G.2: Top-1 Accuracy of AdANNS composite indices with OPQ distance computation compared to MR and Rigid baselines models on Natural Questions.

Our observations on ImageNet with ResNet-50 MR across search structures also extend to the Natural Questions dataset with Dense Passage Retriever (DPR with BERT-Base MR embeddings). We note that AdANNS provides gains over RR-768 embeddings for both Exact Search and IVF with OPQ (Figure G.2a and G.2b). We find that similar to ImageNet (Figure M.6) IVF performance on Natural Questions generally scales with dimensionality. AdANNS thus reduces to MR-768 performance for $M \geq 16$. See Appendix J for a more in-depth discussion of AdANNS with DPR on Natural Questions.

Appendix H

ADANNS-**IVF**

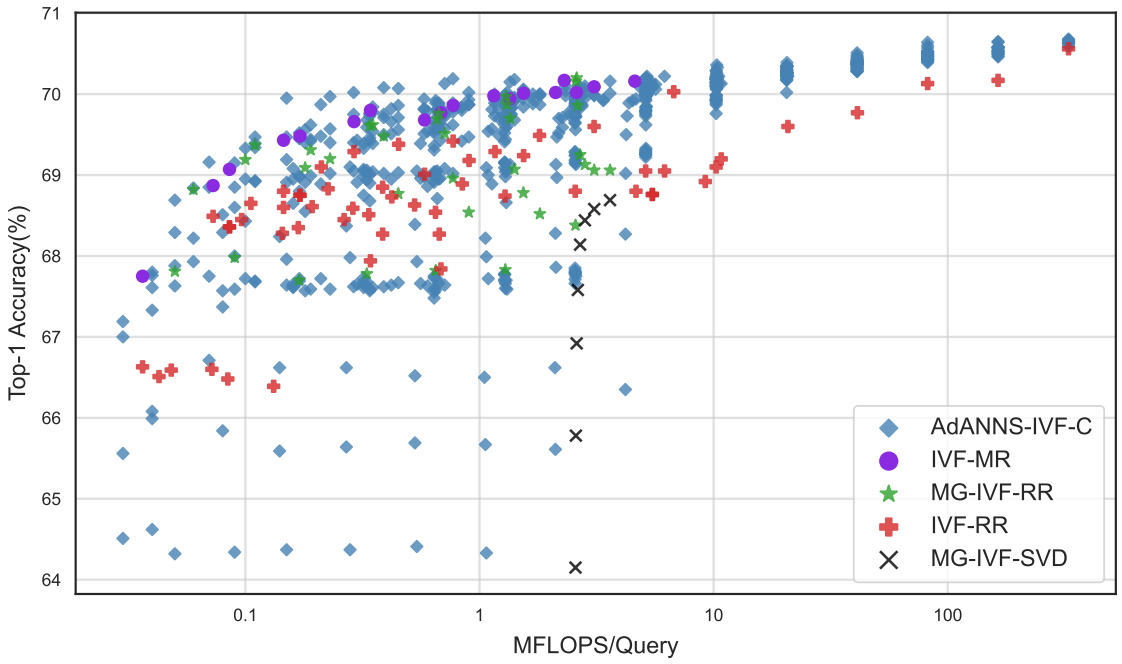


Figure H.1: Top-1 accuracy vs compute cost per query of AdANNS-IVF-C compared to IVF-MR, IVF-RR and MG-IVF-RR baselines on ImageNet-1K.

Inverted file index (IVF) [102] is a simple yet powerful ANNS data structure used in web-scale search systems [33]. IVF construction involves clustering (coarse quantization often through k-means) [76] on d-dimensional representation that results in an inverted file list [117] of all the data points in each cluster. During search, the d-dimensional query representation is first assigned to the closest clusters (# probes, typically set to 1) and then an exhaustive linear scan happens within each cluster to obtain the nearest neighbors.

Our proposed adaptive variant of IVF, AdANNS-IVF-C, decouples the clustering, with d_c dimensions, and the linear scan within each cluster, with d_s dimensions – setting $d_c = d_s$ results in non-adaptive vanilla IVF. This helps in the smooth search of design space for the optimal accuracy-compute trade-off. A naive instantiation yet strong baseline would be to use explicitly trained d_c and d_s dimensional rigid representations (called MG-IVF-RR, for multi-granular IVF with rigid representations). We also examine the setting of adaptively choosing low-dimensional MR to linear scan the shortlisted clusters built with high-dimensional MR, i.e. AdANNS-IVF-D, as seen in Table K.3. We discuss the inference compute for these settings in Appendix C.2.

As seen in Figure H.1, AdANNS-IVF-C provides pareto-optimal accuracy-compute tradeoff

Table H.1: Mathematical formulae of the retrieval phase across various methods built on IVF. See Section 3 for notations.

Method	Retrieval Formula during Inference
IVF-RR	$\arg\min_{j \in C_{h(q)}} \|\phi^{\text{RR}(d)}(q) - \phi^{\text{RR}(d)}(x_j)\|$, s.t. $h(q) = \arg\min_h \|\phi^{\text{RR}(d)}(q) - \mu_h^{\text{RR}(d)}\|$
IVF-MR	$\arg\min_{j \in C_{h(q)}} \|\phi^{\text{MR}(d)}(q) - \phi^{\text{MR}(d)}(x_j)\|$, s.t. $h(q) = \arg\min_h \|\phi^{\text{MR}(d)}(q) - \mu_h^{\text{MR}(d)}\|$
AdANNS-IVF-C	$\arg\min_{j \in C_{h(q)}} \|\phi^{\text{MR}(d_s)}(q) - \phi^{\text{MR}(d_s)}(x_j)\|$, s.t. $h(q) = \arg\min_h \|\phi^{\text{MR}(d_c)}(q) - \mu_h^{\text{MR}(d_c)}\|$
MG-IVF-RR	$\arg\min_{j \in C_{h(q)}} \|\phi^{\text{RR}(d_s)}(q) - \phi^{\text{RR}(d_s)}(x_j)\|$, s.t. $h(q) = \arg\min_h \|\phi^{\text{RR}(d_c)}(q) - \mu_h^{\text{RR}(d_c)}\|$
AdANNS-IVF-D	$\arg\min_{j \in C_{h(q)}} \|\phi^{\text{MR}(d)}(q)[1:\hat{d}] - \phi^{\text{MR}(d)}(x_j)[1:\hat{d}]\|$, s.t. $h(q) = \arg\min_h \|\phi^{\text{MR}(d)}(q)[1:\hat{d}] - \mu_h^{\text{MR}(d)}[1:\hat{d}]\|$
IVFOPQ	$\arg\min_{j \in C_{h(q)}} \|\phi^{\text{PQ}(m,b)}(q) - \phi^{\text{PQ}(m,b)}(x_j)\|$, s.t. $h(q) = \arg\min_h \|\phi(q) - \mu_h\|$

across inference compute. This figure is a more exhaustive indication of AdANNS-IVF-C behavior compared to baselines than Figures 5.2a and 5.1. AdANNS-IVF-C is evaluated for all possible tuples of $d_c, d_s, k = |C| \in \{8, 16, ..., 2048\}$. MG-IVF-RR configurations are evaluated for $d_c \in \{8, ..., d_s\}$, $d_s \in \{32, ..., 2048\}$ and $k = 1024$ clusters. A study over additional k values is omitted due to high compute cost. Finally, IVF-MR and IVF-RR configurations are evaluated for $d_c = d_s \in \{8, 16, ..., 2048\}$ and $k \in \{256, ..., 8192\}$. Note that for a fair comparison, we use $n_p = 1$ across all configurations.

Appendix I

ADANNS-**DISKANN**

Table I.1: Wall clock search latency (μs) of AdANNS-DiskANN across graph construction dimensionality $d \in \{8, 16, \ldots, 2048\}$ and compute budget in terms of OPQ budget $M \in \{8, 16, 32, 48, 64\}$. Search latency is fairly consistent across fixed embedding dimensionality D.

d	M=8	M=16	M=32	M=48	M=64
8	495	-	-	-	-
16	555	571	-	-	-
32	669	655	653	-	-
64	864	855	843	844	848
128	1182	1311	1156	1161	2011
256	1923	1779	1744	2849	1818
512	2802	3272	3423	2780	3171
1024	5127	5456	5724	4683	5087
2048	9907	9833	10205	10183	9329

DiskANN is a state-of-the-art graph-based ANNS index capable of serving queries from both RAM and SSD. DiskANN builds a greedy best-first graph with OPQ distance computation, with compressed vectors stored in memory. The index and full-precision vectors are stored on the SSD. During search, when a query's neighbor shortlist is fetched from the SSD, its full-precision vector is also fetched in a single disk read. This enables efficient and fast distance computation with PQ on a large initial shortlist of candidate nearest neighbors in RAM followed by a high-precision re-ranking with full-precision vectors fetched from the SSD on a much smaller shortlist. The

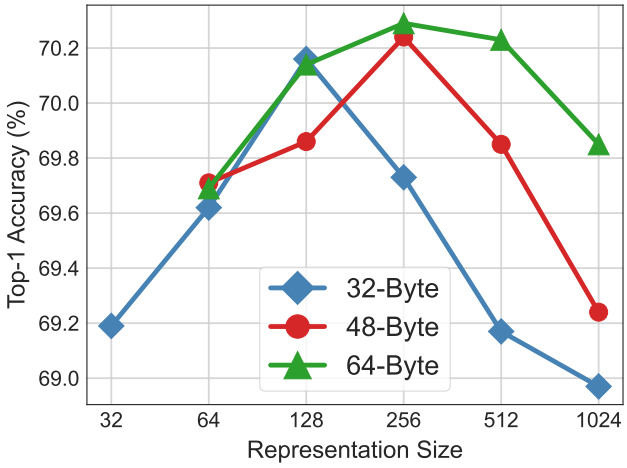


Figure I.1: DiskANN-MR with SSD indices for compute budgets $M_{disk} = M_{dc} \in \{32, 48, 64\}$ across graph construction and OPQ dimensionalities $d \in \{32, \ldots, 1024\}$. Note that this does not use any re-ranking after obtaining OPQ based shortlist.

experiments carried out in this work primarily utilize a DiskANN graph index built in-memory[1] with OPQ distance computation.

As with IVF, DiskANN is also well suited to the flexibility provided by AdANNS as we demonstrate on both ImageNet and NQ that the optimal PQ codebook for a given compute budget is learnt with a smaller embedding dimensionality d (see Figures G.1c and G.2a). We demonstrate the capability of AdANNS-DiskANN with a compute budget of $M \in \{32, 64\}$ in Table 5.1. We tabulate the search time latency of AdANNS-DiskANN in microseconds (μs) in Table I.1, which grows linearly with graph construction dimensionality d. We also examine DiskANN-MR with SSD graph indices across OPQ budgets for distance computation $M_{dc} \in \{32, 48, 64\}$, as seen in Figure I.1. With SSD indices, we store PQ-compressed vectors on disk with $M_{disk} = M_{dc}$, which essentially disables DiskANN's implicit high-precision re-ranking. We observe similar trends to other composite ANNS indices on ImageNet, where the *optimal* dim for fixed OPQ budget is not the highest dim ($d = 1024$ with fp32 embeddings is current highest dim supported by DiskANN which stores

[1] https://github.com/microsoft/DiskANN

vectors in 4KB sectors on disk). This provides further motivation for AdANNS-DiskANN, which leverages MRs to provide flexible access to the optimal dim for quantization and thus enables similar Top-1 accuracy to Rigid DiskANN for up to $1/4$ the cost (Figure G.1c).

Appendix J

ADANNS ON NATURAL QUESTIONS

In addition to image retrieval on ImageNet, we also experiment with dense passage retrieval (DPR) on Natural Questions. As shown in Figure G.1, MR representations are $1-10\%$ more accurate than their RR counterparts across PQ compute budgets with Exact Search + OPQ on NQ. We also demonstrate that IVF-MR is $1-2.5\%$ better than IVF-RR for Precision@k, $k \in \{1, 5, 20, 100, 200\}$. Note that on NQ, IVF loses $\sim 10\%$ accuracy compared to exact search, even with the RR-768 baseline. We hypothesize the weak performance of IVF owing to poor clusterability of the BERT-Base embeddings fine-tuned on the NQ dataset. A more thorough exploration of AdANNS-IVF on NQ is an immediate future work and is in progress.

Appendix K

ROBUSTNESS

Table K.1: Top-1 classification accuracy (%) on out-of-domain datasets (ImageNet-V2/R/A/Sketch) to examine robustness of Matryoshka Representation Learning. Note that these results are without any fine tuning on these datasets.

	ImageNet-V1			ImageNet-V2			ImageNet-R			ImageNet-A			ImageNet-Sketch		
Rep. Size	FF	MRL–E	MRL	FF	MRL–E	MRL	FF	MRL–E	MRL	FF	MRL–E	MRL	FF	MRL–E	MRL
8	65.86	56.92	67.46	54.05	47.40	55.59	24.60	22.98	23.57	2.92	3.63	3.39	17.73	15.07	17.98
16	73.10	72.38	73.80	60.52	60.48	61.71	28.51	28.45	28.85	3.00	3.55	3.59	21.70	20.38	21.77
32	74.68	74.80	75.26	62.24	62.23	63.05	31.28	30.79	31.47	2.60	3.65	3.57	22.03	21.87	22.48
64	75.45	75.48	76.17	63.51	63.15	63.99	32.96	32.13	33.39	2.87	3.99	3.76	22.13	22.56	23.43
128	75.47	76.05	76.46	63.67	63.52	64.69	33.93	33.48	34.54	2.81	3.71	3.73	22.73	22.73	23.70
256	75.78	76.31	76.66	64.13	63.80	64.71	34.80	33.91	34.85	2.77	3.65	3.60	22.63	22.88	23.59
512	76.30	76.48	76.82	64.11	64.09	64.78	35.53	34.20	34.97	2.37	3.57	3.59	23.41	22.89	23.67
1024	76.74	76.60	76.93	64.43	64.20	64.95	36.06	34.22	34.99	2.53	3.56	3.68	23.44	22.98	23.72
2048	77.10	76.65	76.95	64.69	64.17	64.93	37.10	34.29	35.07	2.93	3.49	3.59	24.05	23.01	23.70

K.1 MRL *models*

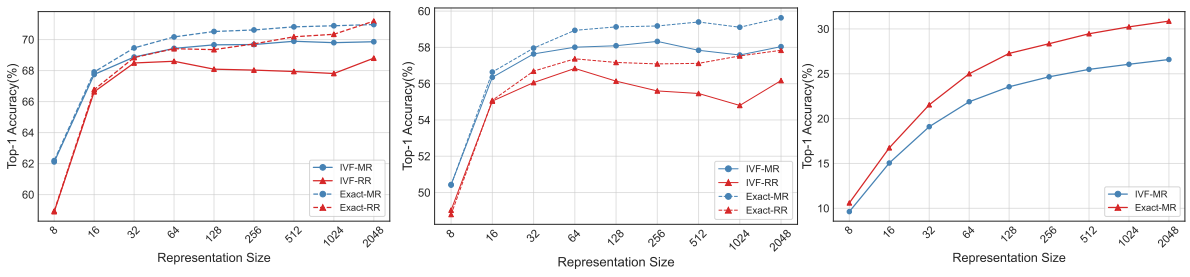
We evaluated the robustness of MRL models on out-of-domain datasets (ImageNetV2/R/A/Sketch) and compared them to the FF baseline. Each of these datasets is described in Appendix B. The results in Table K.1 demonstrate that learning Matryoshka Representations does not hurt out-of-domain generalization relative to FF models, and Matryoshka Representations in fact improve the performance on ImageNet-A. For a ALIGN–MRL model, we examine the the robustness via zero-shot retrieval on out-of-domain datasets, including ObjectNet, in Table K.2.

Table K.2: Zero-shot top-1 image classification accuracy (%) of a ALIGN-MRL model on ImageNet-V1/V2/R/A and ObjectNet.

Rep. Size	V1	V2	A	R	ObjectNet
12	30.57	23.98	14.59	24.24	25.52
24	45.64	37.71	22.75	46.40	35.89
48	53.84	46.16	28.88	60.71	42.76
96	58.31	51.34	33.21	70.12	45.20
192	60.95	53.56	36.10	74.41	48.24
384	62.06	54.77	37.95	76.51	49.10
768	62.26	55.15	37.84	76.73	49.26
Baseline	66.39	59.57	39.97	80.49	51.60

K.2 ***IVF-***MR

As shown in Figure K.1, we examined the clustering capabilities of MRs on both in-distribution (ID) queries via ImageNet-1K and out-of-distribution (OOD) queries via ImageNetV2 [91], as well



(a) ImageNet-1K (b) ImageNetV2 (c) ImageNet-4K

Figure K.1: Top-1 Accuracy variation of IVF-MR of ImageNet 1K, ImageNetV2 and ImageNet-4K. RR baselines are omitted on ImageNet-4K due to high compute cost.

as on larger-scale ImageNet-4K [64]. For ID queries on ImageNet-1K (Figure K.1a), IVF-MR is at least as accurate as Exact-RR for $d \leq 256$ with a single search probe, demonstrating the quality of in-distribution low-d clustering with MR. On OOD queries (Figure K.1b), we observe that IVF-MR is on average 2% more robust than IVF-RR across all cluster construction and linear scan dimensionalities d. It is also notable that clustering with MRs followed by linear scan with # probes $= 1$ is more robust than exact search with RR embeddings across all $d \leq 2048$, indicating the adaptability of MRs to distribution shifts during inference. As seen in Table K.3, on ImageNetV2 AdANNS-IVF-D is the best configuration for $d \leq 16$, and is similarly accurate to IVF-MR at all other d. AdANNS-IVF-D with $d = 128$ is able to match its own accuracy with $d = 2048$, a $16\times$ compute gain during inference. This demonstrates the potential of AdANNS to adaptively search pre-indexed clustering structures.

On 4-million scale ImageNet-4K (Figure K.1c), we observe similar accuracy trends of IVF-MR compared to Exact-MR as in ImageNet-1K (Figure K.1a) and ImageNetV2 (Figure K.1b). We omit baseline IVF-RR and Exact-RR experiments due to high compute cost at larger scale.

Table K.3: Top-1 Accuracy of AdANNS-IVF-D on out-of-distribution queries from ImageNetV2 compared to both IVF and Exact Search with MR and RR embeddings. Note that for AdANNS-IVF-D, the dimensionality used to build clusters $d_c = 2048$.

d	AdANNS-IVF-D	IVF-MR	Exact-MR	IVF-RR	Exact-RR
8	**53.51**	50.44	50.41	49.03	48.79
16	**57.32**	56.35	56.64	55.04	55.08
32	57.32	57.64	**57.96**	56.06	56.69
64	57.85	58.01	**58.94**	56.84	57.37
128	58.02	58.09	**59.13**	56.14	57.17
256	58.01	58.33	**59.18**	55.60	57.09
512	58.03	57.84	**59.40**	55.46	57.12
1024	57.66	57.58	**59.11**	54.80	57.53
2048	58.04	58.04	**59.63**	56.17	57.84

Appendix L
ANALYSIS OF MODEL DISAGREEMENT

Class Trends *Does increasing representation size necessarily help improve classification performance across all classes in ImageNet-1K?* We studied this question by examining trends in performance with increasing representation size from $d = 8, ...2048$. For MRL models, we observed that 244 classes showed a monotonic improvement in performance with increasing d, 177 classes first improved but then observed a slight dip (one or two misclassifications per class), 49 classes showed a decline first and then an improvement, and the remaining classes did not show a clear trend. When we repeated this experiment with independently trained FF models, we noticed that 950 classes did not show a clear trend. This motivated us to leverage the disagreement as well as gradual improvement of accuracy at different representation sizes by training $\mathrm{Matryoshka\ Representations}$. Figure L.1 showcases the progression of relative per-class accuracy distribution compared to the MRL-2048 dimensional model. This also showed that some instances and classes could benefit from lower-dimensional representations.

Discussion of Oracle Accuracy Based on our observed model disagreements for different representation sizes d, we defined an optimal *oracle* accuracy [71] for MRL. We labeled an image as correctly predicted if classification using any representation size was correct. The percentage of total samples of ImageNet-1K that were firstly correctly predicted using each representation size d is shown in Table L.1. This defined an upper bound on the performance of MRL models, as 18.46% of the ImageNet-1K validation set were incorrectly predicted $\forall d \in \{8, 16, \ldots, 2048\}$. We show the oracle performance on MRL models for ImageNet-1K/V2/A/R/Sketch datasets in Table L.2.

In an attempt to derive an optimal routing policy to emulate oracle accuracy, we designed the adaptive classification via cascading method as discussed in Appendix E.1. This led to an

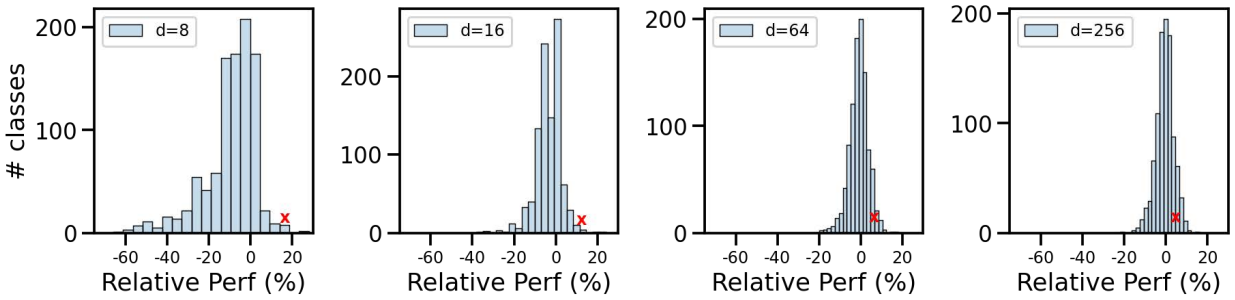


Figure L.1: Progression of relative per-class accuracy vs MRL-2048. As the dimensionality increases, the spread shrinks while the class marked (**x**) (Madagascar cat) loses accuracy.

interesting observation on the expected dimensionality for 76.30% top-1 classification accuracy being just $d \sim 37$. We leave the design and learning of a more optimal policy for future work.

Table L.1: Percentage of ImageNet-1K validation set that is first correctly predicted using each representation size d. We note that 18.46% of the samples cannot be correctly predicted by any representation size. The remaining 81.54% constitutes the oracle accuracy.

Table L.2: Oracle classification accuracy of various evaluation datasets for ResNet50–MRL model trained on ImageNet-1K.

Top-1	ImageNetV1	ImageNetV2	ImageNet-A	ImageNet-R	ImageNet-Sketch
FF–2048	76.9	64.9	3.6	35.1	23.7
MRL–Oracle	81.5	70.6	8.7	39.8	28.9

Grad-CAM Examples We analyzed the nature of model disagreement across representation sizes with MRL models with the help of Grad-CAM visualization [98]. We observed there were

certain classes in ImageNet-1K such as ”tools”, ”vegetables” and ”meat cutting knife” which were occasionally located around multiple objects and a cluttered environment. In such scenarios, we observed that smaller representation size models would often get confused due to other objects and fail to extract the object of interest which generated the correct label. We also observed a different nature of disagreement arising when the models got confused within the same superclass. For example, ImageNet-1K has multiple ”snake” classes, and models often confuse a snake image for an incorrect species of snake.

Superclass Performance We created a 30 superclass subset of the validation set based on wordnet hierarchy (Table L.3) to quantify the performance of MRL model on ImageNet-1K superclasses. Table L.4 quantifies the performance with different representation size.

Table L.3: 30 Superclasses in ImageNet-1K corresponding to the performance in Table L.4.

insect	motor vehicle	artiodactyl	vegetable	game equipment
terrier	serpent	machine	measuring device	sheepdog
protective covering	sporting dog	vessel, watercraft	building	lizard
garment	hound	monkey	home appliance	wind instrument
vessel	fish	nourishment	electronic equipment	oscine
furniture	wading bird	tool	canine	mechanism

Table L.4: Performance of MRL model on 31-way classification (1 extra class is for reject token) on ImageNet-1K superclasses.

Rep. Size	8	16	32	64	128	256	512	1024	2048
MRL	85.57	88.67	89.48	89.82	89.97	90.11	90.18	90.22	90.21

Appendix M

ABLATIONS

M.1 MRL *Training Paradigm*

Matryoshka Representations **via Finetuning.** To observe if nesting can be induced in models that were not explicitly trained with nesting from scratch, we loaded a pretrained FF-2048 ResNet50 model and initialized a new MRL layer, as defined in Algorithm 2, Appendix C. We then unfroze different layers of the backbone to observe how much non-linearity in the form of unfrozen conv layers needed to be present to enforce nesting into a pretrained FF model. A de-

Table M.1: Top-1 classification accuracy (%) on ImageNet-1K of various ResNet50 models which are finetuned on pretrained FF-2048 model. We observed that adding more non-linearities is able to induce nesting to a reasonable extent even if the model was not pretrained with nesting in mind.

Rep. Size	fc	4.2 conv3, fc	4.2 conv2, conv3, fc	4.2 full, fc	All (MRL)
8	5.15	36.11	54.78	60.02	66.63
16	13.79	58.42	67.26	70.10	73.53
32	32.52	67.81	71.62	72.84	75.03
64	52.66	72.42	73.61	74.29	75.82
128	64.60	74.41	74.67	75.03	76.30
256	69.29	75.30	75.23	75.38	76.47
512	70.51	75.96	75.47	75.64	76.65
1024	70.19	76.18	75.70	75.75	76.76
2048	69.72	76.44	75.96	75.97	76.80

scription of these layers can be found in the ResNet50 architecture [40]. All models were finetuned with the FFCV pipeline, with same training configuration as in the end-to-end training aside from changing lr $= 0.1$ and epochs $= 10$. We observed that finetuning the linear layer alone was insufficient to learn $\mathrm{Matryoshka\ Representations}$ at lower dimensionalities. Adding more and more non-linear conv+ReLU layers steadily improved classification accuracy of $d = 8$ from 5% to 60% after finetuning, which was only 6% less than training MRL end-to-end for 40 epochs. This difference was successively less pronounced as we increased dimensionality past $d = 64$, to within 1.5% for all larger dimensionalities. The full results of this ablation can be seen in Table M.1.

Table M.2: An ablation over boosting training loss at lower nesting dimensions, with top-1 and top-5 accuracy (%). The models are described in Appendix M.1.

Model	MRL		MRL-8boost		MRL-8+16boost	
Rep. Size	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
8	66.63	84.66	**69.53**	86.19	69.24	85.96
16	73.53	89.52	73.86	89.44	**73.91**	89.55
32	75.03	91.31	**75.28**	91.21	75.10	91.14
64	75.82	92.27	**75.84**	92.22	75.67	92.06
128	**76.30**	92.82	76.28	92.74	76.07	92.52
256	76.47	93.02	**76.48**	92.97	76.22	92.72
512	**76.65**	93.13	76.56	93.09	76.35	92.85
1024	**76.76**	93.22	76.71	93.21	76.39	92.98
2048	**76.80**	93.32	76.76	93.28	76.52	93.05

Relative Importance. We performed an ablation of MRL over the relative importance, c_m, of different nesting dimensions $m \in \mathcal{M}$, as defined in Sec. 3.1. In an attempt to improve performance at lower dimensionalities, we boosted the relative importance c_m of training loss at lower dimensions as in Eq. 3.1 with two models, MRL-8boost and MRL-8+16boost. The

MRL-8boost model had $c_{m\in\mathcal{M}} = [2, 1, 1, 1, 1, 1, 1, 1, 1]$ and the MRL-8+16boost model had $c_{m\in\mathcal{M}} = [2, 1.5, 1, 1, 1, 1, 1, 1, 1]$. The relative importance list $c_{m\in\mathcal{M}}$ had a 1-to-1 correspondence with nesting dimension set $\mathcal{M}$. In Table M.2, we observed that MRL-8boost improves top-1 accuracy by 3% at $d = 8$, and also improves top-1 accuracy of all representation scales from 16 to 256 over MRL, while only hurting the performance at 512 to 2048 representation scales by a maximum of 0.1%. This suggests that the relative importance c_m can be tuned/set for optimal accuracy for all $m \in \mathcal{M}$, but we leave this extension for future work.

Matryoshka Representations **at Arbitrary Granularities.** To train MRL, we used nested dimensions at logarithmic granularities $\mathcal{M} = \{8, 16, \ldots, 1024, 2048\}$ as detailed in Section 3.1. We made this choice for two empirically-driven reasons: a) The accuracy improvement with increasing representation size was more logarithmic than linear (as shown by FF models in Figure 4.1). This indicated that optimizing for granularities increasing in a non-logarithmic fashion would be sub-optimal both for maximum performance and expected efficiency; b) If we have m arbitrary granularities, the expected cost of the linear classifier to train MRL scales as $O(L * (m^2))$ while logarithmic granularities result in $O(L * 2log(d))$ space and compute costs.

To demonstrate this effect, we learned Matryoshka Representations with uniform (MRL-*Uniform*) nesting dimensions $m \in \mathcal{M} = \{8, 212, 416, 620, 824, 1028, 1232, 1436, 1640, 1844, 2048\}$. We evaluated this model at the standard (MRL-*log*) dimensions $m \in \mathcal{M} = \{8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ for ease of comparison to reported numbers using 1-NN accuracy (%). As shown in Table M.4, we observed that while performance interpolated, MRL-*Uniform* suffered at low dimensions as the logarithmic spacing of MRL-*log* resulted in tighter packing of information in these initial dimensions. The higher nesting dimensions of MRL-*Uniform* did not help in significant accuracy improvement due to accuracy saturation, which is often logarithmic in representation size as shown by FF models. Note that the slight improvement at dimensions higher than 512 for MRL-*Uniform* is due to multiple granularities around them compared to just three for MRL-*log*, which are not useful in practice for efficiency.

Lower Dimensionality. We experimented with training MRL with smaller nesting dimension than $m = 8$, as shown in Table M.3, with two models: MRL-4 and MRL-6. We found that using lower than 8-dimensions to train MRL, i.e. $m_0 \in \{4, 6\}$ for MRL-4 and MRL-6 respectively, did not affect the top-1 accuracy of other granularities significantly. However, granularities smaller than 8-dimensions had very low accuracy and were often unusable for deployment along with additional training difficulty. We also observed a small dip in accuracy at higher dimensions which

Table M.3: An ablation over training with smaller nesting dimensionalities in terms of Top-1 accuracy (%). MRL-4 and MRL-6 are variations of the original model (MRL-8) with $m_0 \in \{4, 6\}$, where $m \in \mathcal{M}$ is part of the nesting_list as seen in Alg 2.

Rep. Size	MRL-4	MRL-6	MRL-8
4	27.25	-	-
6	-	58.71	-
8	66.86	**67.55**	66.63
16	73.36	73.10	**73.53**
32	74.82	74.49	**75.03**
64	75.51	75.32	**75.82**
128	75.93	75.61	**76.30**
256	76.08	75.82	**76.47**
512	76.31	75.93	**76.65**
1024	76.38	76.04	**76.76**
2048	76.43	76.12	**76.80**

Table M.4: An ablation over training MRL with nesting list at uniformly distributed granularities. Entries in the MRL-Uniform column are evaluated at logarithmic dimensions for a fair comparison to MRL-Log (standard MRL) with 1-NN accuracy (%).

Rep. Size	MRL-Log	MRL-Uniform
8	**62.19**	58.44
16	**67.91**	61.11
32	**69.46**	63.82
64	**70.17**	66.44
128	**70.52**	68.71
256	**70.62**	70.06
512	70.82	**70.98**
1024	70.89	**71.37**
2048	70.97	**71.44**

Table M.5: Adaptive retrieval ablation over shortlist length k for $D_s = 16$, $D_r = 2048$ on ImageNet-1K with exact search. Entries with the highest P@1 and mAP@10 across all k are in bold.

Shortlist Length	P@1	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
100	70.88	65.19	63.62	62.59	61.24	69.96	69.24	68.53	67.20
200	70.90	**65.27**	63.73	62.82	61.97	70.10	69.44	68.90	68.21
400	70.94	65.26	63.71	62.81	62.03	70.15	69.51	69.02	68.47
800	70.96	65.23	63.64	62.69	61.85	70.16	69.52	69.02	68.45
1600	70.96	65.20	63.58	62.58	61.66	70.16	69.5	68.97	68.36
2048	**70.97**	65.20	63.57	62.58	61.64	70.16	69.5	68.97	68.35

we attribute to the joint loss that now also included the harder optimization of the smallest dimension. Lastly, we hypothesize the dimensionality of 8 is an empirically validated design choice due to the considerable accuracy it provided along with the ease of training.

M.2 MRL Retrieval

Adaptive Retrieval. To examine the effect of increasing shortlist lengths on search time, we performed a reranking ablation over shortlist lengths for D_s= 16 and D_r= 2048 over ImageNet-1K in Table M.5, and over ImageNet-4K in Table M.6. We observed that using a larger shortlist k saturated ImageNet-1K performance at k=200. But using larger shortlists until $k = 2048$, the maximum value supported by the FAISS framework, steadily improved performance on ImageNet-4K. This is likely due to the increased database size, but could also indicate a correlation with ImageNet-4K being slightly out-of-distribution making the task at hand harder.

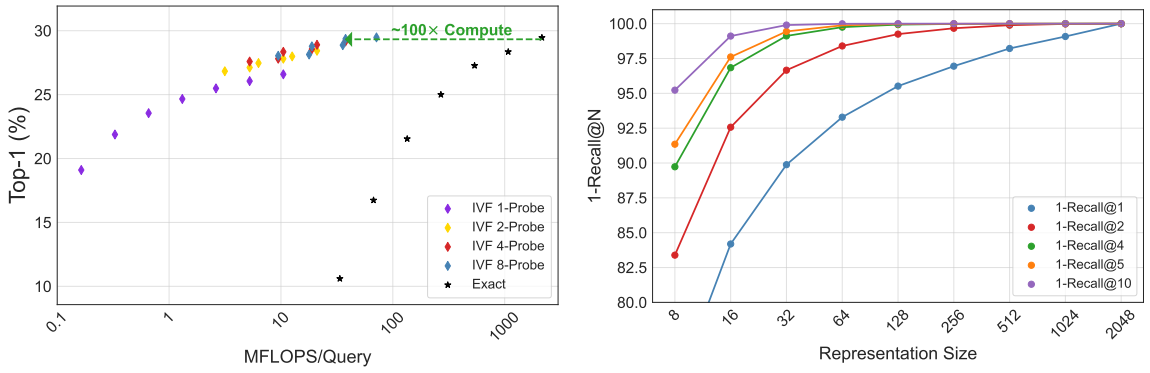
Table M.6: Adaptive retrieval ablation over shortlist length k for $D_s = 16$, $D_r = 2048$ on ImageNet-4K with exact search.

Shortlist Length	P@1	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
100	27.70	14.38	10.62	8.26	6.07	20.12	16.87	14.29	11.26
200	28.56	15.21	11.43	9.11	7.12	21.23	18.13	15.73	13.27
400	29.34	15.83	12.06	9.76	7.79	22.08	19.09	16.83	14.54
800	29.86	16.30	12.53	10.23	8.26	22.72	19.83	17.65	15.45
1600	30.24	16.63	12.86	10.56	8.60	23.18	20.36	18.23	16.11
2048	**30.35**	**16.73**	12.96	10.65	8.69	23.31	20.50	18.40	16.30

M.3 IVF

As seen in Figure M.1a, IVF-MR can match the accuracy of Exact Search on ImageNet-4K with $\sim 100\times$ less compute. We also explored the capability of MRs at retrieving cluster centroids with low-d compared to a ground truth of 2048-d with k-Recall@N, as seen in Figure M.1b. MRs were able to saturate to near-perfect 1-Recall@N for $d \geq 32$ and $N \geq 4$, indicating the potential of AdANNS at matching exact search performance with less than 10 search probes n_p.

Clustering Distribution We examined the distribution of learnt clusters across embedding dimensionalities d for both MR and RR models, as seen in Figure M.2. We observe IVF-MR to have less variance than IVF-RR at $d \in \{8, 16\}$, and slightly higher variance for $d \geq 32$, while IVF-MR outperforms IVF-RR in top-1 across all d (Figure K.1a). This indicates that although MR learns clusters that are less uniformly distributed than RR at high d, the quality of learnt clustering is superior to RR across all d. Note that a uniform distribution is N/k data points per cluster, i.e. ~ 1250 for ImageNet-1K with $k = 1024$. We quantitatively evaluate the proximity of the MR and RR clustering distributions with Total Variation Distance [72], which is defined over two discrete



(a) 4K Search Probes (n_p)

(b) Centroid Recall

Figure M.1: Ablations on IVF-MR Clustering: a) Analysis of accuracy-compute tradeoff with increasing IVF-MR search probes n_p on ImageNet-4K compared to Exact-MR and b) k-Recall@N on ImageNet-1K cluster centroids across representation sizes d. Cluster centroids retrieved with highest embedding dim $d = 2048$ were considered ground-truth centroids.

probability distributions p, q over $[n]$ as follows:

$$d_{TV}(p, q) = \frac{1}{2} \sum_{i \in [n]} |p_i - q_i|$$

We also compute $d_{TV,2048}(\text{MR-d}) = d_{TV}(\text{MR-d}, \text{RR-2048})$, which evaluates the total variation distance of a given low-d MR from high-d RR-2048. We observe a monotonically decreasing $d_{TV,2048}$ with increasing d, which demonstrates that MR clustering distributions get closer to RR-2048 as we increase the embedding dimensionality d. We observe in Figure M.2 that $d_{TV}(\text{MR-d}, \text{RR-d}) \sim 7e-4$ for $d \in \{8, 256, ..., 2048\}$ and $\sim 3e-4$ for $d \in \{16, 32, 64\}$. These findings agree with the top-1 improvement of MR over RR as shown in Figure K.1a, where there are smaller improvements for $d \in \{16, 32, 64\}$ (smaller d_{TV}) and larger improvements for $d \in \{8, 256, ..., 2048\}$ (larger d_{TV}). These results demonstrate a correlation between top-1 performance of IVF-MR and the quality of clusters learnt with MR.

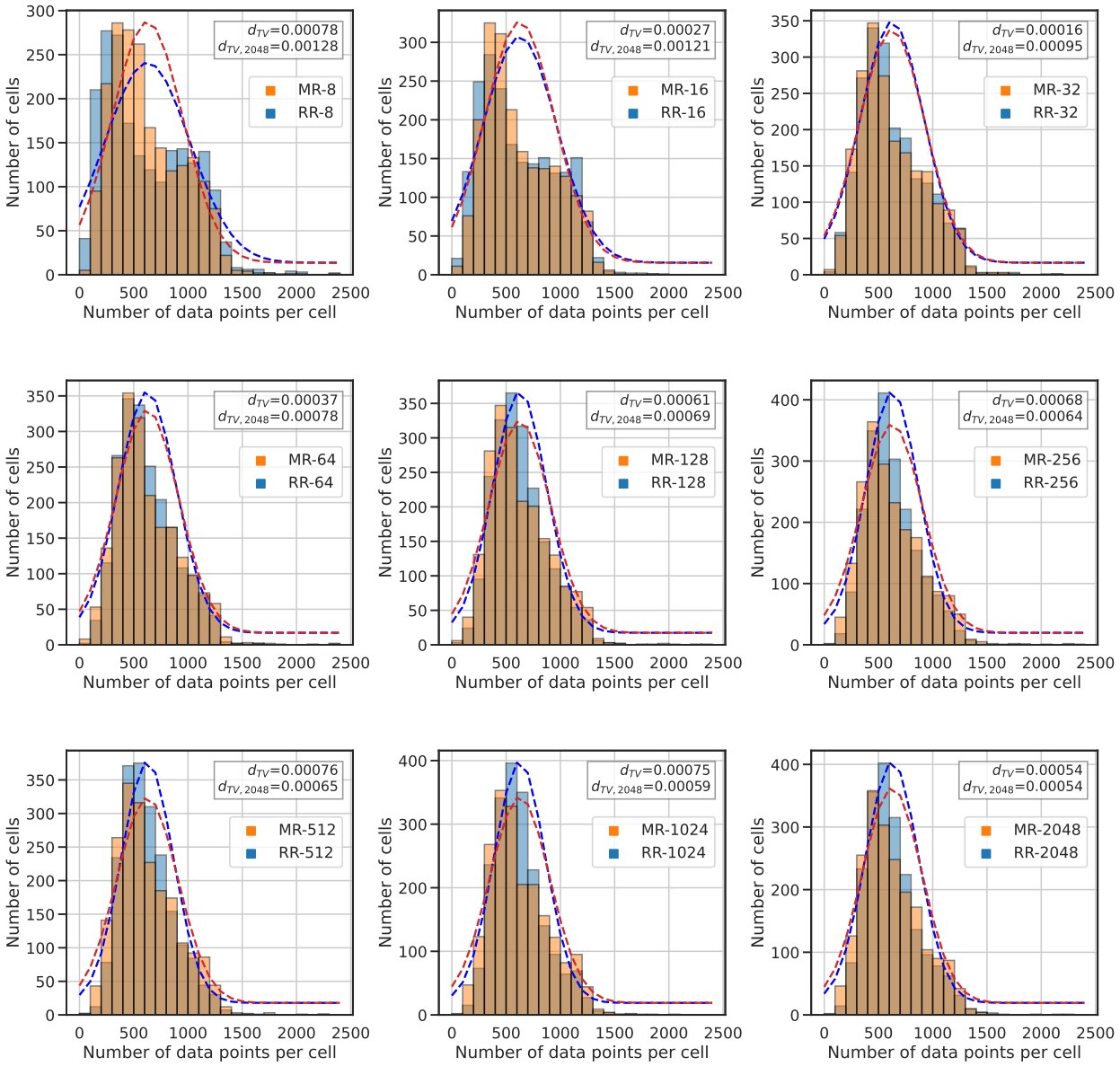


Figure M.2: Clustering distributions for IVF-MR and IVF-RR across embedding dimensionality d on ImageNet-1K. An IVF-MR and IVF-RR clustered with $d = 16$ embeddings is denoted by MR-16 and RR-16 respectively.

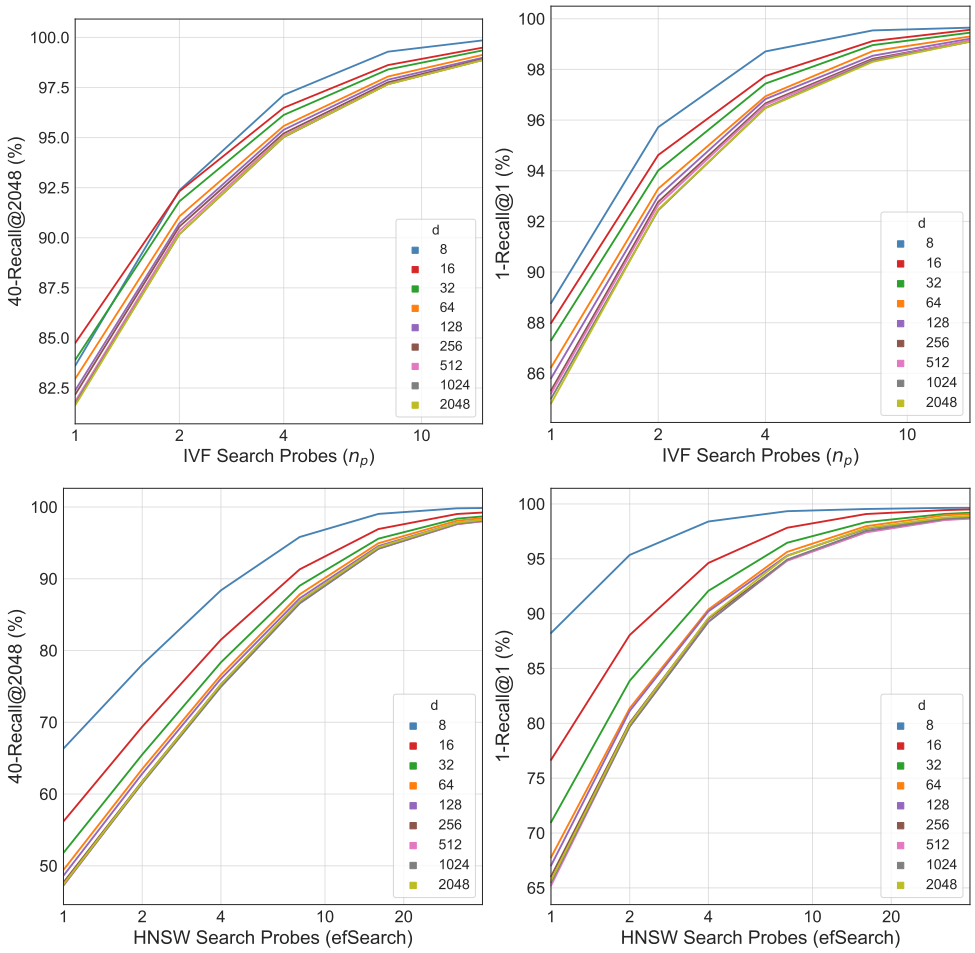


Figure M.3: k-Recall@N of d-dimensional MR for IVF and HNSW with increasing search probes n_p on ImageNet-1K.

M.4 Recall Score Analysis

In this section we also examine the variation of k-Recall@N with by probing a larger search space with IVF and HNSW indices. For IVF, search probes represent the number of clusters shortlisted for linear scan during inference. For HNSW, search quality is controlled by the $efSearch$ parameter [78], which represents the closest neighbors to query q at level l_c of the graph and is analogous to number of search probes in IVF. As seen in Figure M.3, general trends show a) an intuitive increase in recall with increasing search probes n_p) for fixed search probes, a decrease in recall

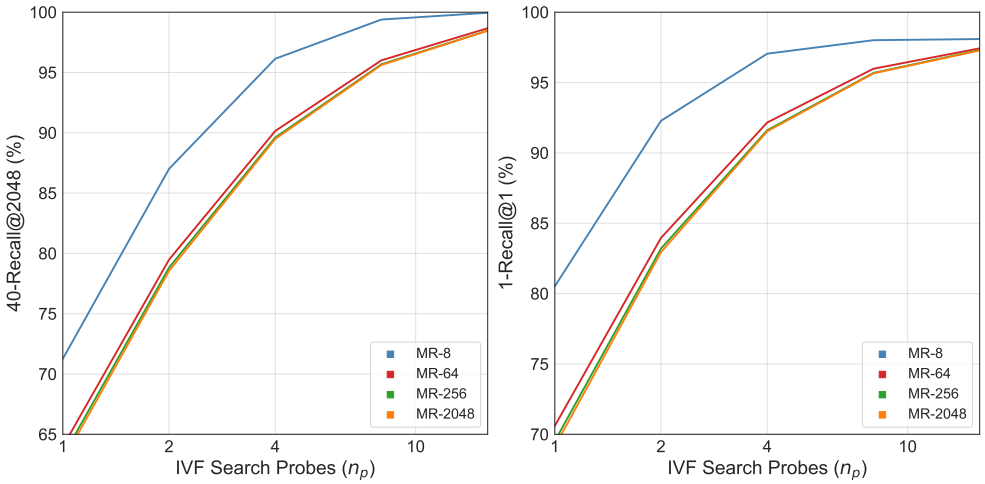


Figure M.4: k-Recall@N for IVF-MR-d on ImageNet-4K for $d \in \{8, 64, 256, 2048\}$. Other embedding dimensionalities, HNSW-MR and RR baselines are omitted due to high compute cost. We observe that trends from ImageNet-1K with increasing d and n_p extend to ImageNet-4K, which is $4\times$ larger.

with increasing search dimensionality d. These trends extend from ImageNet-1K to $4\times$ larger ImageNet-4K, as seen in Figure M.4.

M.5 Relative Contrast

We utilize Relative Contrast [37] to capture the difficulty of nearest neighbors search with IVF-MR compared to IVF-RR. For a given database $X = \{x_i \in \mathbb{R}^d, i = 1, ..., N_D\}$, a query $q \in \mathbb{R}^d$, and a distance metric $D(.,.)$ we compute relative contrast C_r as a measure of the difficulty in finding the 1-nearest neighbor (1-NN) for a query q in database X as follows:

1. Compute $D^q_{min} = \min_{i=1...n} D(q, x_i)$, i.e. the distance of query q to its nearest neighbor $x^q_{nn} \in X$
2. Compute $D^q_{mean} = E_x[D(q, x)]$ as the average distance of query q from all database points $x \in X$
3. Relative Contrast of a given query $C^q_r = \dfrac{D^q_{mean}}{D^q_{min}}$, which is a measure of how *separable* the

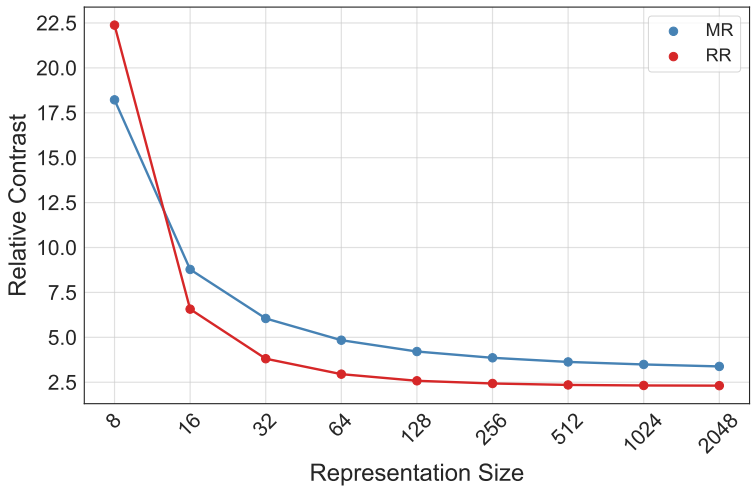


Figure M.5: Relative contrast of varying capacity MRs and RRs on ImageNet-1K corroborating the findings of He at al. [37].

query's nearest neighbor x_{nn}^q is from an average point in the database x

4. Compute an expectation over all queries for Relative Contrast over the entire database as

$$C_r = \frac{E_q[D_{mean}^q]}{E_q[D_{min}^q]}$$

It is evident that C_r captures the difficulty of Nearest Neighbor Search in database X, as a $C_r \sim 1$ indicates that for an average query, its nearest neighbor is almost equidistant from a random point in the database. As demonstrated in Figure M.5, MRs have higher R_c than RR Embeddings for an Exact Search on ImageNet-1K for all $d \geq 16$. This result implies that a portion of MR's improvement over RR for 1-NN retrieval across all embedding dimensionalities d [64] is due to a higher average separability of the MR 1-NN from a random database point.

M.6 Generality across Encoders

We perform an ablation over the representation function $\phi : X \to \mathbb{R}^d$ learnt via a backbone neural network (primarily ResNet50 in this work), as detailed in Section 3. We also train MRL models [64] $\phi^{MR(d)}$ on ResNet18/34/101 [40] that are as accurate as their independently trained

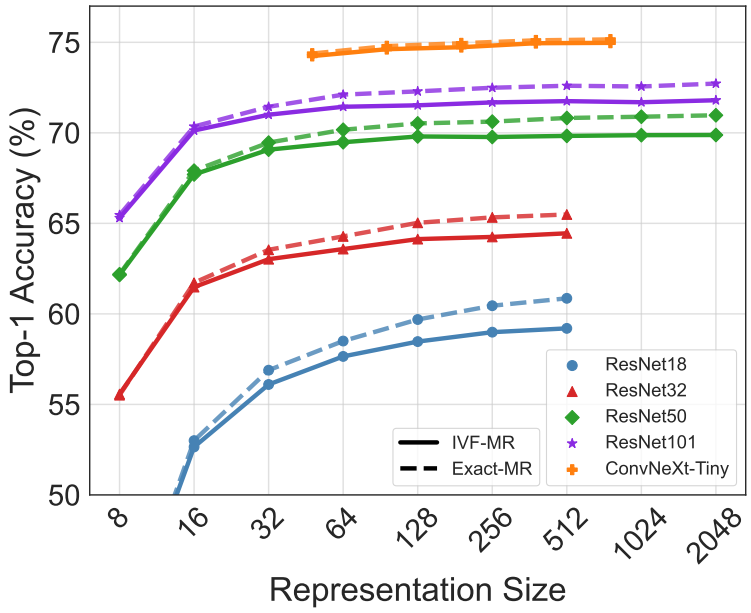


Figure M.6: Top-1 Accuracy variation of IVF-MR on ImageNet-1K with different embedding representation function $\phi^{MR(d)}$ (see Section 3), where $\phi \in \{\text{ResNet18/34/101, ConvNeXt-Tiny}\}$. We observe similar trends between IVF-MR and Exact-MR on ResNet18/34/101 when compared to ResNet50 (Figure K.1a) which is the default in all experiments in this work.

RR baseline models $\phi^{RR(d)}$, where d is the default max representation size of each architecture. We also train MRL with a ConvNeXt-Tiny[1] backbone with $[d] = \{48, 96, 192, 384, 786\}$. MR-768 has a top-1 accuracy of 79.45% compared to independently trained publicly available RR-768 baseline with top-1 accuracy 82.1%. We note that this training had no hyperparameter tuning whatsoever, and this gap can be closed with additional model training effort. We then compare clustering the MRs via IVF-MR with $k = 2048, n_p = 1$ on ImageNet-1K to Exact-MR, which is shown in Figure M.6. IVF-MR shows similar trends across backbones compared to Exact-MR, i.e. a maximum top-1 accuracy drop of $\sim 1.6\%$ for a single search probe. This suggests the clustering capabilities of MR extend beyond an inductive bias of $\phi^{MR(d)} \in$ ResNet50, though we leave a detailed exploration for future work.

[1] https://github.com/facebookresearch/ConvNeXt