Matryoshka Representations for Adaptive Deployment

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learned representations are a central component in modern ML systems, serving a multitude of downstream tasks. When training such representations, it is often the case that computational and statistical constraints for each downstream task are unknown. In this context, rigid fixed-capacity representations can be either over or under-accommodating to the task at hand. This leads us to ask: *can we design a flexible representation that can adapt to multiple downstream tasks with varying computational resources?* Our main contribution is Matryoshka Representation Learning (MRL) which encodes information at different granularities and allows a single embedding to adapt to the computational constraints of downstream tasks. MRL minimally modifies existing representation learning pipelines and imposes no additional cost during inference and deployment. MRL learns coarse-to-fine representations that are at least as accurate and rich as independently trained low-dimensional representations. The flexibility within the learned Matryoshka Representations offer: (a) up to $\mathbf{14}\times$ smaller embedding size for ImageNet-1K classification at the same level of accuracy; (b) up to $\mathbf{14}\times$ real-world speed-ups for large-scale retrieval on ImageNet-1K and 4K; and (c) up to $\mathbf{2}\%$ accuracy improvements for long-tail few-shot classification, all while being as robust as the original representations. Finally, we show that MRL extends seamlessly to web-scale datasets (ImageNet, JFT) across various modalities – vision (ViT, ResNet), vision + language (ALIGN) and language (BERT).

1 Introduction

Learned representations [51] are fundamental building blocks of real-world ML systems [58, 82]. Trained once and frozen, d-dimensional representations encode rich information and can be used to perform multiple downstream tasks [4]. The deployment of deep representations has two steps (1) an expensive yet constant-cost forward pass to compute the representation [27] and (2) utilization of the representation for downstream applications [44, 80]. Compute costs for the latter part of the pipeline scale with the the embedding dimensionality as well as the data size (N) and label space (L).At web-scale [15, 76] 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 [26, 30]. However, perhaps due to the inductive bias of gradient-based training [75], 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 [27], jointly optimizing sub-networks of varying capacity [9, 91] or post-hoc compression [34, 53]. 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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

or 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 learned representation. MRL learns representations of varying capacities within the same high-dimensional vector through explicit optimization of $O(\log(d))$-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 illustrates the core idea of Matryoshka Representation Learning (MRL) and the adaptive deployment settings of the learned Matryoshka Representations.

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 additional training or 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.

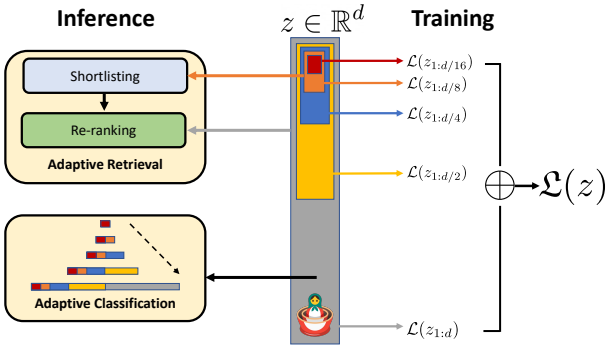


Figure 1: Matryoshka Representation Learning is adaptable to any representation learning setup and begets a Matryoshka Representation z by optimizing the original loss $\mathcal{L}(.)$ at $O(\log(d))$ chosen representation sizes. Matryoshka Representation can be utilized effectively for adaptive deployment across environments and downstream tasks.

The Matryoshka Representations improve efficiency for large-scale classification and retrieval without any significant loss of accuracy. While there are potentially several applications of coarse-to-fine Matryoshka Representations, 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 4.2.1). Similarly, we use MRL in an adaptive retrieval system. Given a query, we shortlist retrieval candidates using the first 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 4.3.1). Finally, as MRL explicitly learns coarse-to-fine representation vectors, intuitively it should share more semantic information among its various dimensions (Figure 5). This is reflected in up to 2% accuracy gains in long-tail continual learning settings while being as robust as the original embeddings. Furthermore, due to its coarse-to-fine grained nature, MRL can also be used as method to analyze hardness of classification among instances and information bottlenecks.

We make the following key contributions:

1. We introduce Matryoshka Representation Learning (MRL) to obtain flexible representations (Matryoshka Representations) for adaptive deployment (Section 3).

2. Up to $14\times$ faster yet accurate large-scale classification and retrieval using MRL (Section 4).

3. 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).

4. Further analysis of MRL's representations in the context of other downstream tasks (Section 5).

2 Related Work

Representation Learning. Large-scale datasets like ImageNet [16, 67] and JFT [76] enabled the learning of general purpose representations for computer vision [4, 89]. These representations are typically learned through supervised and un/self-supervised learning paradigms. Supervised pretraining [27, 45, 73] casts representation learning as a multi-class/label classification problem, while un/self-supervised learning learns representation via proxy tasks like instance classification [88] and reconstruction [29, 56]. Recent advances [12, 28] in contrastive learning [25] enabled learning from web-scale data [21] that powers large-capacity cross-modal models [18, 41, 63, 92]. Similarly, natural language applications are built [36] on large language models [8] that are pretrained [60, 66] in a un/self-supervised fashion with masked language modelling [19] or autoregressive training [62].

Matryoshka Representation Learning (MRL) is complementary to all these setups and can be adapted with minimal overhead (Section 3). MRL equips representations with multifidelity at no additional cost which enables adaptive deployment based on the data and task (Section 4).

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 [24, 48] and design choices [35, 47, 78]. 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 [3, 38, 61] and memory [20] using Approximate Nearest Neighbor Search (ANNS) [55] or leveraging the underlying hierarchy [17, 49]. In case of the representation size, often dimensionality reduction [68, 79], hashing techniques [14, 46, 69] and feature selection [57] help in alleviating selective aspects of the $O(d)$ scaling at a cost of significant drops in accuracy. Lastly, most real-world search systems [11, 15] are often powered by large-scale embedding based retrieval [10, 58] that scales in cost with the ever increasing web-data. While categorization [80, 90] clusters similar things together, it is imperative to be equipped with retrieval capabilities that can bring forward every instance [7]. Approximate Nearest Neighbor Search (ANNS) [37] makes it feasible with efficient indexing [14] and traversal [5, 6] to present the users with the most similar documents/images from the database for a requested query. Widely adopted HNSW [55] ($O(d \log(N))$) is as accurate as exact retrieval ($O(dN)$) at the cost of a graph-based index overhead for RAM and disk [39].

MRL tackles the linear dependence on embedding size, d, by learning multifidelity Matryoshka Representations. Lower-dimensional Matryoshka Representations are as accurate as independently trained counterparts without the multiple expensive forward passes. Matryoshka Representations provide an *intermediate abstraction* between high-dimensional vectors and their efficient ANNS indices through the adaptive embeddings nested within the original representation vector (Section 4). All other aforementioned efficiency techniques are complementary and can be readily applied to the learned Matryoshka Representations obtained from MRL.

Several works in efficient neural network literature [9, 84, 91] 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. [65] 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 5); making web-scale feasible.

3 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 learning 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 [27] on ImageNet-1K [67] 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

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

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 Section 5 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 [80, 90]. 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 Matryoshka Representation Learning (MRL).

Adaptation to Learning Frameworks. MRL can 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).

4 Applications

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).

4.1 Representation Learning

We adapt Matryoshka Representation Learning (MRL) to various representation learning setups – a) Supervised learning for vision: ResNet50 [27] on ImageNet-1K [67] and ViT-B/16 [22] on JFT-300M [76], b) Contrastive learning for vision + language: ALIGN model with ViT-B/16 vision encoder and BERT language encoder on ALIGN data [41] and c) Masked language modelling: BERT [19] on English Wikipedia and BooksCorpus [93]. 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-dim. represen-

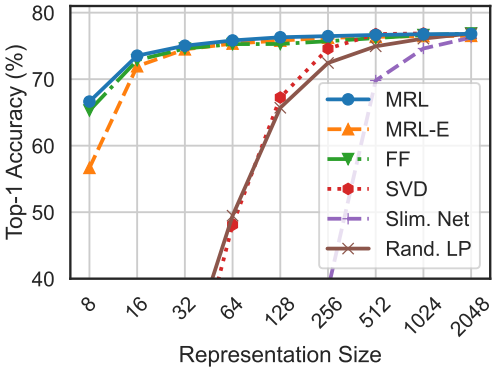


Figure 2: ImageNet-1K linear classification accuracy of ResNet50 models. MRL is as accurate as the independently trained FF models for every representation size.

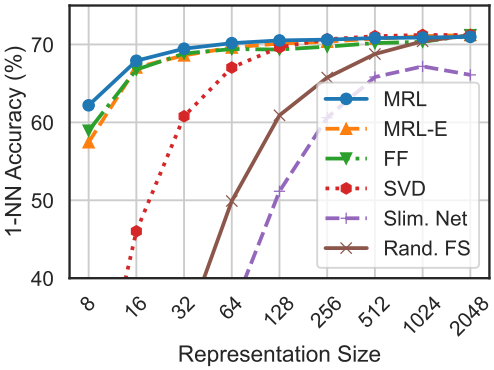


Figure 3: ImageNet-1K 1-NN accuracy of ResNet50 models measuring the representation quality for downstream task. MRL outperforms all the baselines across all representation sizes.

tation while ViT-B/16 and BERT output 768-dim. 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 [91]) 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 2 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 3 showcases the comparison of learned representation quality through 1-NN accuracy on ImageNet-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, 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 D.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 5 shows that, while MRL explicitly optimizes $O(\log(d))$ nested representations – removing the $O(d)$ dependence [65] –, the coarse-to-fine grained information is interpolated across all d dimensions providing highest flexibility for adaptive deployment.

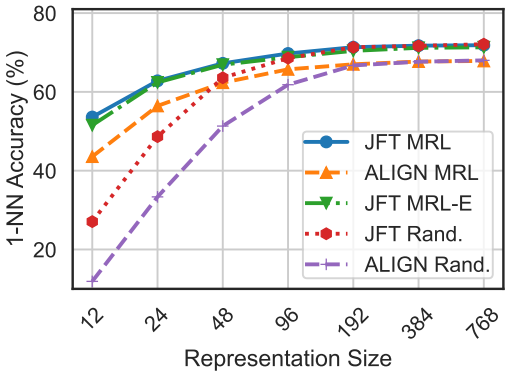


Figure 4: ImageNet-1K 1-NN accuracy for ViT-B/16 models trained on JFT-300M & as part of ALIGN. MRL scales seamlessly to web-scale with minimal training overhead.

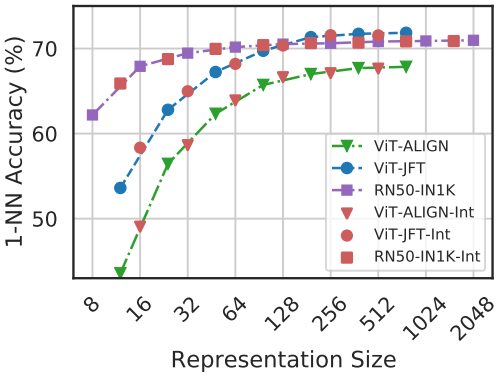


Figure 5: Despite optimizing MRL only for $O(\log(d))$ dimensions for ResNet50 and ViT-B/16 models; the accuracy in the intermediate dimensions shows interpolating behaviour.

4.2.1 Adaptive Classification

The flexibility and coarse-to-fine granularity within Matryoshka Representations allows model cascades [81] for Adaptive Classification (AC) [26]. Unlike standard model cascades [86], 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 [31] 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 D.1 discusses the learning of thresholds for cascades used for adaptive classification in detail.

Figure 6 shows the comparison between cascaded MRL representations (MRL–AC) and independently trained fixed feature (FF) models on ImageNet-1K with ResNet50. We compute the expected representation size for MRL–AC based on the final dimensionality used in the cascade. We observe that MRL–AC is as accurate, 76.30%, as a 512-dimensional FF model but requires an expected dimensionality of ~ 37 while being only 0.5% 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 [80]. Lastly, our results with MRL–AC indicate that instances and classes vary in difficulty which we analyze in Section 5 and Appendix J.

4.3 Retrieval

Nearest neighbour search with learned representations powers a plethora of retrieval and search applications [15, 82, 11, 58]. In this section, we discuss the image retrieval performance of the pretrained ResNet50 models (Section 4.1) on two large-scale datasets ImageNet-1K [67] and ImageNet-4K. ImageNet-1K has a database size of ~1.3M and a query set of 50K samples uniformly spanning 1000 classes. We also introduce ImageNet-4K which has a database size of ~4.2M and query set of ~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 [55], $O(d\log(N))$ 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

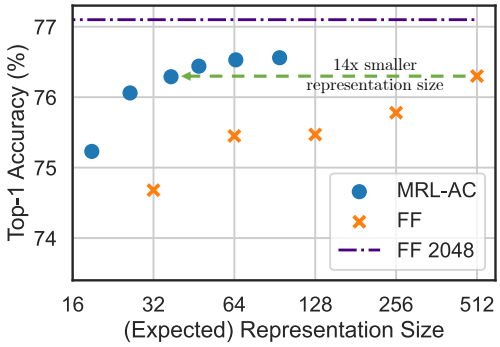


Figure 6: Adaptive classification on MRL ResNet50 using cascades results in $14\times$ smaller representation size for the same level of accuracy on ImageNet-1K (~ 37 vs 512 dims for 76.3%).

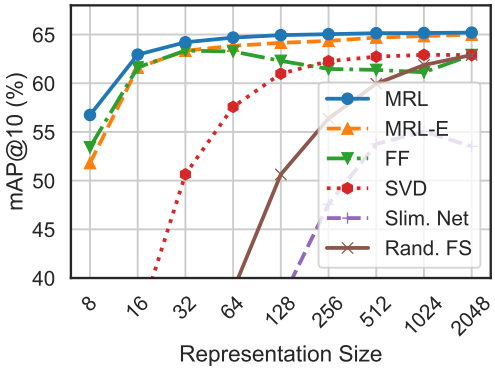


Figure 7: mAP@10 for Image Retrieval on ImageNet-1K with ResNet50. MRL consistently produces better retrieval performance over the baselines across all the representation sizes.

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 Appendix F for more implementation details.

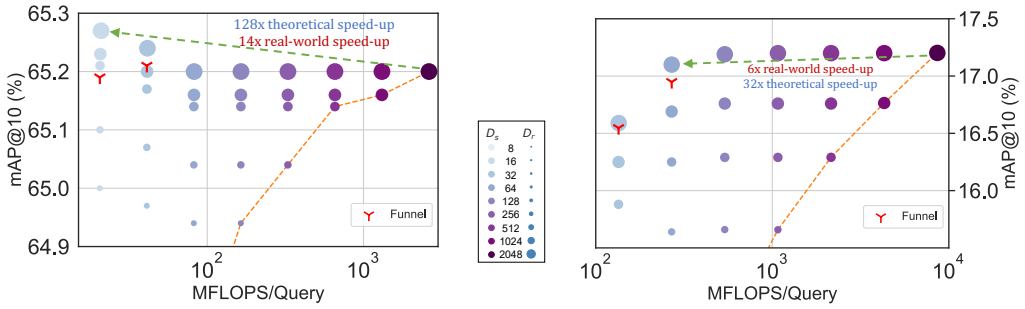
Figure 7 compares the mAP@10 performance of ResNet50 representations on ImageNet-1K across dimensionalities for MRL, MRL–E, FF, slimmable networks along with post-hoc compression of vectors using SVD and random feature selection. Matryoshka Representations are often the most accurate while being up to 3% better than the FF baselines. Similar to classification, post-hoc compression and slimmable network baselines suffer from significant drop-off in retrieval mAP@10 with ≤ 256 dimensions. Appendix E 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 Representation enables 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 [53, 40] used as part of the ANNS pipelines are complimentary to Matryoshka Representations and can further improve efficiency-vs-accuracy trade-off.

4.3.1 Adaptive Retrieval

We benchmark MRL in the adaptive retrieval setting (AR) [44]. For a given query image, we obtain a shortlist, $K = 200$, of images from the database using a lower-dimensional representation, eg., $D_s = 16$ followed by reranking with a higher capacity representation, eg., $D_r = 2048$. 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 of the AR, the shortlisting component of the pipeline can be sped-up using ANNS (HNSW). Appendix I 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 8 showcases the compute-vs-accuracy trade-off for adaptive retrieval using Matryoshka Representations compared to single-shot using fixed features with ResNet50 on ImageNet-1K. We observe that all AR settings lie above the Pareto frontier of the 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



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

Figure 8: 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.

ablate over the shortlist size, k in Appendix K.2 to find that the accuracy gains stop after a point further strengthening the use-case for 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 double 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 Matryoshka Representation is as accurate as the single-shot 2048-dim retrieval while being $\sim 128\times$ more efficient theoretically (see Appendix F for more results). All these results showcase the potential of MRL and AR for large-scale multi-stage search systems [15].

5 Further Analysis and Ablations

Robustness We evaluate the robustness of the MRL models trained on ImageNet-1K on out-of-domain datasets, ImageNetV2/R/A/Sketch [64, 32, 33, 85], and compare them to the FF baselines. Table 17 in Appendix H 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 9 in Appendix E shows that MRL models have more robust retrieval compared to the FF baselines by having up to 3% higher mAP@10 performance. Lastly, we also find that the zero-shot robustness of ALIGN-MRL (Table 18 in Appendix H) agree with the observations made by Wortsman et al. [87].

Few-shot and Long-tail Learning. We exhaustively evaluate few-shot learning on MRL models using nearest class mean (NCM) [70]. Table 15 in Appendix G shows that that representations learned through MRL perform comparably to independently trained representations across varying shots and number of classes.

Matryoshka Representations realize a unique pattern while evaluating on FLUID [83], a long-tail sequential learning framework. We observe that MRL provides up to 2% accuracy higher on novel classes in the tail of the distribution, without sacrificing accuracy on other classes (Table 16 in Appendix G). Additionally we find the accuracy between low-dimensional and high-dimensional representations is marginal for pretrain classes. We hypothesize that the higher-dimensional representations are required to differentiate the classes when few training examples of each are known. This results provides further evidence that different tasks require varying capacity based on their difficulty.

Disagreement across Dimensions The information packing in Matryoshka Representations often results in gradual increase of accuracy with increase in capacity. However, we observe that this trend is not ubiquitous and certain instances and classes are more accurate when evaluated with lower-dimensions (Figure 10 in Appendix J). With perfect routing of instances to appropriate dimension,

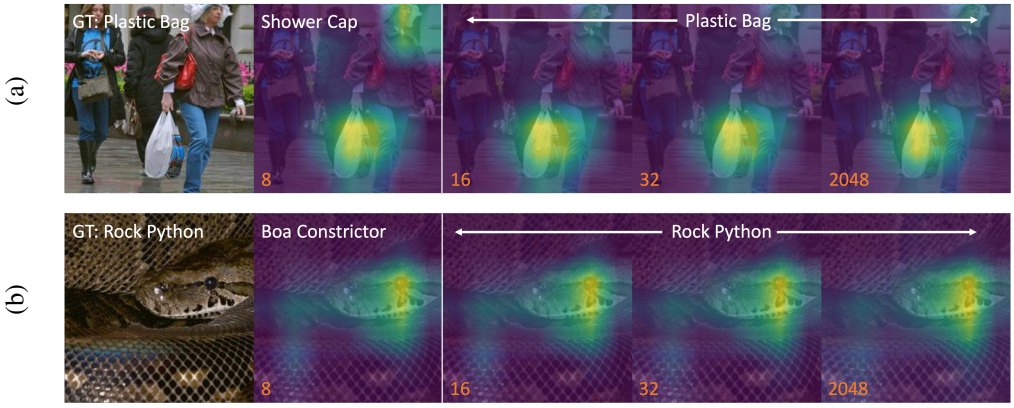


Figure 9: Grad-CAM [71] progression of predictions in MRL model across $8, 16, 32, 2048$ dimensions. (a) 8-dimensional representation confuses due to presence of other relevant objects (with a larger field of view) in the scene and predicts "shower cap" & (b) 8-dim model confuses within the same super-class of "boa"; thus failing gracefully in both these scenarios.

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 [23] of the ImageNet hierarchy or presence of multiple objects of interest. Figure 9 showcases 2 such examples for 8-dimensional representation. These results along with Appendix J put forward the potential for MRL to be a systematic framework for analyzing the utility and efficiency of information bottlenecks.

5.1 Ablations

Table 26 in Appendix K 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 27 in Appendix C indicates that with optimal weighting of the nested losses we could improve accuracy of lower-dimensions representations without accuracy loss. Lastly, Tables 28 and 29 in Appendix K.2 show that the retrieval performance saturates after a certain shortlist dimension and length depending on the complexity of the dataset.

6 Discussion and Conclusions

The results in Section 5.1 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. (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.

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 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. Finally, most of the efficiency techniques for model inference and vector search are complementary to MRL 🪆 further assisting in deployment at the compute-extreme environments.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

[3] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. *Advances in Neural Information Processing Systems*, 23, 2010.

[4] Y. 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.

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

[6] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104, 2006.

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

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[9] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[10] W.-C. Chang, F. X. Yu, Y.-W. Chang, Y. Yang, and S. Kumar. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932*, 2020.

[11] W.-C. Chang, D. Jiang, H.-F. Yu, C. H. Teo, J. Zhang, K. Zhong, K. Kolluri, Q. Hu, N. Shandilya, V. 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.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[13] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang. Meta-baseline: exploring simple meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9062–9071, 2021.

[14] M. Datar, N. Immorlica, P. Indyk, and V. 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.

[15] J. 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.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] J. Deng, A. C. Berg, and L. Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *CVPR 2011*, pages 785–792. IEEE, 2011.

[18] K. Desai and J. 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.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[21] S. K. Divvala, A. Farhadi, and C. 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.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[23] L. Engstrom, A. Ilyas, H. Salman, S. Santurkar, and D. Tsipras. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

[24] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[25] M. Gutmann and A. 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.

[26] M. G. Harris and C. 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.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[28] K. He, H. Fan, Y. Wu, S. Xie, and R. 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.

[29] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

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

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

[32] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. 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.

[33] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021.

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

[35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[36] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[37] P. Indyk and R. 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.

[38] H. Jain, V. Balasubramanian, B. Chunduri, and M. 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.

[39] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems*, 32, 2019.

[40] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[41] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[42] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[43] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. 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.

[44] T. C. Kaz Sato. Vertex ai matching engine. *Microsoft AI Blog*, 2021. URL https://cloud.google.com/blog/topics/developers-practitioners/find-anything-blazingly-fast-googles-vector-search-technology.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[46] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2143–2157, 2009.

[47] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. *Advances in Neural Information Processing Systems*, 31, 2018.

[48] A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. Kakade, and A. Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

[49] A. Kusupati, M. Wallingford, V. Ramanujan, R. Somani, J. S. Park, K. Pillutla, P. Jain, S. Kakade, and A. Farhadi. Llc: Accurate, multi-purpose learnt low-dimensional binary codes. *Advances in Neural Information Processing Systems*, 34, 2021.

[50] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry. ffcv. https://github.com/libffcv/ffcv/, 2022. commit 607d117.

[51] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[52] S. Lee, S. Purushwalkam Shiva Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra. Stochastic multiple choice learning for training diverse deep ensembles. *Advances in Neural Information Processing Systems*, 29, 2016.

[53] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on communications*, 28(1):84–95, 1980.

[54] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

[56] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[57] P. Mitra, C. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[58] P. Nayak. Understanding searches better than ever before. *Google AI Blog*, 2019. URL https://blog.google/products/search/search-language-understanding-bert/.

[59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[60] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. 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. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

[61] Y. Prabhu, A. Kusupati, N. Gupta, and M. Varma. Extreme regression for dynamic search advertising. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 456–464, 2020.

[62] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018. URL https://openai.com/blog/language-unsupervised/.

[63] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[64] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[65] O. Rippel, M. Gelbart, and R. Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014.

[66] S. Ruder, M. E. Peters, S. Swayamdipta, and T. 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.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

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

[70] J. S. Sánchez, F. Pla, and F. J. Ferri. On the use of neighbourhood-based non-parametric classifiers. *Pattern Recognition Letters*, 18(11-13):1179–1186, 1997.

[71] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. 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.

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

[73] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[74] L. 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.

[75] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

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

[77] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

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

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

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

[81] P. Viola and M. 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.

[82] C. Waldburger. As search needs evolve, microsoft makes ai tools for better search available to researchers and developers. *Microsoft AI Blog*, 2019. URL https://blogs.microsoft.com/ai/bing-vector-search/.

[83] M. Wallingford, A. Kusupati, K. Alizadeh-Vahid, A. Walsman, A. Kembhavi, and A. Farhadi. Are we overfitting to experimental setups in recognition? *arXiv preprint arXiv:2007.02519*, 2020.

[84] M. Wallingford, H. Li, A. Achille, A. Ravichandran, C. Fowlkes, R. Bhotika, and S. Soatto. Task adaptive parameter sharing for multi-task learning. *arXiv preprint arXiv:2203.16708*, 2022.

[85] H. Wang, S. Ge, Z. Lipton, and E. P. Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

[86] X. Wang, D. Kondratyuk, K. M. Kitani, Y. Movshovitz-Attias, and E. Eban. Multiple networks are more efficient than one: Fast and accurate models via ensembles and cascades. *arXiv preprint arXiv:2012.01988*, 2020.

[87] M. Wortsman, G. Ilharco, M. Li, J. W. Kim, H. Hajishirzi, A. Farhadi, H. Namkoong, and L. Schmidt. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*, 2021.

[88] Z. Wu, Y. Xiong, S. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *arXiv preprint arXiv:1805.01978*, 2018.

[89] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[90] H.-F. Yu, K. Zhong, J. Zhang, W.-C. Chang, and I. S. Dhillon. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23(98):1–32, 2022.

[91] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[92] R. Zellers, J. Lu, X. Lu, Y. Yu, Y. Zhao, M. Salehi, A. Kusupati, J. Hessel, A. Farhadi, and Y. Choi. Merlot reserve: Neural script knowledge through vision and language and sound. *arXiv preprint arXiv:2201.02639*, 2022.

[93] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] To be Filled
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material and Appendix A. All the code and public models will be open sourced.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We benchmarked on large-scale datasets like ImageNet-1K, JFT-300M and ALIGN data with models like ResNet and ViT making it extremely expensive to run things multiple times.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C and Appendix I.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] All the non-proprietary datasets and code used are public under MIT, BSD or CC licenses.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We created a new subset of ImageNet-21K for downstream evaluation of retrieval performance at scale. See Section 4.3 and Appendix B
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Code for Matryoshka Representation Learning (MRL)

We use Alg 1 and 2 provided below to train supervised ResNet50–MRL models on ImageNet-1K. We provide this code as a template to extend MRL to any domain. The full code for MRL training and evaluation will be open-sourced.

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 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:
                self.nesting_list = self.nesting_list[-1] # Highest
                    dimensionality

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.
                            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
```

B Datasets

ImageNet-1K [67] 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 [50].

ImageNet-4K dataset was constructed by selecting 4,202 classes, non-overlapping with ImageNet-1K, from ImageNet-21K [16] 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. We will release the list of images curated together to construct ImageNet-4K.

JFT-300M [76] is a large-scale multi-label dataset with 300M images labelled across 18,291 categories.

ALIGN [41] utilizes a large scale noisy image-text dataset containing 1.8B image-text pairs.

ImageNet Robustness Datasets We experimented on the following datasets to examine the robustness of MRL models:

ImageNetV2 [64] is a collection of 10K images sampled a decade after the original construction of ImageNet [16]. ImageNetV2 contains 10 examples each from the 1,000 classes of ImageNet-1K.

ImageNet-A [33] contains 7.5K real-world adversarially filtered images from 200 ImageNet-1K classes.

ImageNet-R [32] contains 30K artistic image renditions for 200 of the original ImageNet-1K classes.

ImageNet-Sketch [85] contains 50K sketches, evenly distributed over all 1,000 ImageNet-1K classes.

ObjectNet [2] contains 50K images across 313 object classes, each containing $\sim$160 images each.

C Matryoshka Representation Learning **Model Training**

We trained all ResNet50–MRL models using the efficient dataloaders of FFCV [50]. 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 [74]. 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 [77] 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.

We trained ViT-B/16 models for JFT-300M on a 8x8 cloud TPU pod [43] 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 [72] with a linear learning rate decay starting at 1e-3.

Lastly, we trained BERT-base model on English Wikipedia and BookCorpus. We trained our models in Tensorflow used a 4x4 cloud TPU pod with a total batchsize of 1024. We used an AdamW [54] with a linear learning rate decay starting at 1e-4 and trained for 450K steps.

D Classification Results

We show the Top-1 classification accuracy of ResNet50–MRL models on ImageNet-1K in Table 1 and Figure 2. We compare the performance of MRL models (MRL, MRL–E) to several baselines:

Table 1: Top-1 classification accuracy (%) for ResNet50 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**	76.26	76.80	76.51

- **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 linear classifier fit on randomly selected features [28].
- **Slim. Net**: We take the pretrained Slimmable neural networks [91] which are trained with a flexible width backbone (25%, 50%, 75% and full width). For each rep. size, we take 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 (Rep. Size ≤ 128), MRL outperforms all baselines significantly, which indicates that pretrained models lack the multifidelity of Matryoshka Representations and are incapable of fitting accurate linear classifier for low-dimensions.

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 2 and Figure 3. We provide detailed information regarding the k-NN search pipeline in Appendix E. 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. We also compared against the 1-NN accuracy of slimmable neural nets [91] 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.

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

Rep. Size	FF	Rand. FS	FF+SVD	Slimmable	MRL	MRL–E
8	58.93	2.36	19.14	1.00	**62.19**	57.45
16	66.77	12.06	46.02	5.12	**67.91**	67.05
32	68.84	32.91	60.78	16.95	**69.46**	68.60
64	69.41	49.91	67.04	35.60	**70.17**	69.61
128	69.35	60.91	69.63	51.16	**70.52**	70.12
256	69.72	65.75	**70.67**	60.61	70.62	70.36
512	70.18	68.77	**71.06**	65.82	70.82	70.74
1024	70.34	70.41	**71.22**	67.19	70.89	71.07
2048	71.19	71.19	71.21	66.10	70.97	**71.21**

D.1 Adaptive Classification (MRL–AC)

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 rep. size m_i exceeds a learned threshold t_i^*. If $p_i \geq t_i^*$, we used predictions from rep. size m_i otherwise, we increased to rep. 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, we computed the classification accuracy over our 10K image subset. 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., $\{t_j^* \mid j \in \{8, 16, 32, 64, \ldots, 2048\}\}$

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 3 and Figure 6, we observed that in expectation, we only needed a ~ 37 sized representation to achieve 76.3% classification accuracy on ImageNet-1K, which is roughly $14\times$ smaller than the FF–512 baseline. Even if we compute the expectation as weighted average over cumulative sum of representation size $\{8, 24, 56, \ldots\}$, due to the nature of multiple linear heads for MRL, we end up with an expected size of 62 that still provides roughly $8.2\times$ efficient representation than FF–512 baseline.

Table 3: Threshold-based adaptive classification performance of ResNet50 MRL on a 40K sized held-out subset of ImageNet-1K validation set.

Expected Rep. Size	Accuracy
13.57 ± 0.95	73.79 ± 0.11
18.86 ± 1.53	75.23 ± 0.11
26.18 ± 2.02	76.06 ± 0.13
37.19 ± 5.02	76.29 ± 0.18
47.21 ± 6.21	76.44 ± 0.19
64.79 ± 9.00	76.53 ± 0.18
94.32 ± 15.87	76.56 ± 0.19
137.04 ± 31.01	76.57 ± 0.19

D.2 JFT, ALIGN, BERT

We examine the k-NN classification accuracy of learned Matryoshka Representations via ALIGN–MRL and JFT-ViT–MRL in Table 4. For ALIGN [41], we observed that learning 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 [22] for JFT-300M [76] classification, where learning 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 Matryoshka Representations is feasible and extendable even for extremely large scale datasets. We also demonstrate that Matryoshka Representations are learned at interpolated dimensions for both ALIGN and JFT-ViT, as shown in Table 5, despite not being trained explicitly at these dimensions.

Table 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 [19], whose results are tabulated in Table 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 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-NRL		JFT-ViT-NRL	
	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 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

E 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 [59] 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) [49] and precision (P@k): $P@k = \frac{correct_pred}{k}$ where $correct_pred$ is the number of retrieved NN with the correct label over the entire query set using a shortlist of length k.

We performed retrieval with FAISS [42], 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 [42] 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 20. 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 K). 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 8. MRL outperforms FF models for nearly all rep. size for both Top-1 and mAP@10, and especially at low rep. 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 rep. size over models explicitly trained at those rep. 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 rep. 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 rep. 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. We observed that performance scaled nearly monotonically between the original rep. size and the interpolated rep. size as we increase D_s, which demonstrates that MRL is able to

Table 7: BERT [19] Masked-Language Modelling with FF and MRL models.

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

Table 8: Retrieve a shortlist of 200-NN with D_s sized representations on ImageNet-1K via exact search with L2 distance metric. The Top-1 and mAP@10 entries where MRL–E and MRL outperform FF 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
FF	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
	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	66.90	65.52	63.83
MRL–E	8	10	57.45	57.68	57.50	51.80	50.41	49.6	48.86	57.50	57.16	56.81	56.36
	16	20	**67.05**	66.94	66.79	61.60	60.36	59.66	59.04	66.79	66.53	66.24	65.87
	32	41	68.60	68.74	68.49	63.34	61.97	61.14	60.39	68.49	68.06	67.65	67.17
	64	82	**69.61**	69.28	68.93	**63.84**	62.33	61.43	60.57	68.93	68.40	67.96	67.38
	128	164	**70.12**	69.60	69.19	**64.15**	62.58	61.61	60.70	69.19	68.62	68.11	67.50
	256	328	**70.36**	69.83	69.36	**64.35**	62.76	61.76	60.82	69.36	68.79	68.26	67.63
	512	656	**70.74**	70.09	69.63	**64.69**	63.05	62.06	61.14	69.63	69.00	68.50	67.88
	1024	1312	**71.07**	70.24	69.78	**64.85**	63.22	62.19	61.26	69.78	69.16	68.60	67.99
	2048	2624	**71.21**	70.41	69.90	**64.99**	63.33	62.29	61.33	69.90	69.24	68.68	68.05
MRL	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
	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
MRL-Interpolated	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	63.20	62.42	61.72	69.53	69.10	68.75	68.32
	96	123	70.40	83.83	87.04	64.86	63.46	62.62	61.84	69.82	69.38	68.98	68.48
	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

learn Matryoshka Representations at nearly all rep. 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 9 and Table 10 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 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 I. 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 rep. size D_s, and then reranking the shortlist with representations of rep. size = D_r. Adaptive retrieval for a shortlist length $k = 200$ is shown in Table 11 for ImageNet-1K, and in Table 12 for ImageNet-4K. On ImageNet-1K, we are able to achieve comparable performance to retrieval with

Table 9: 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 10: 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.

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

$D_s = 2048$ (from Table 8) with $D_s = 16$ at $128\times$ less MFLOPs/Query (interchangeably used with MFLOPs). Similarly, on ImageNet-4K, we are able to achieve comparable performance to retrieval with $D_s = 2048$ (from Table 10) 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 Matryoshka Representations for retrieval.

Table 11: 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 8, are bolded.

	D_s	D_r	MFLOPs	Top-1	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
Shortlist Length = 200	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

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 is 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(shortlist cascade) and decreasing the shortlist length (cascade list), which resembles a funnel structure. We tabulate the performance of funnel retrieval in various configurations in Table 13 on ImageNet-1K, and in Table 14 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 8) 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 10) 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.

G Few-shot and Sample Efficiency

We compare MRL, MRL–E, and FF on various benchmarks to observe the effect of representation size on sample efficiency. We use Nearest Class Means [70] for classification which has been shown to be effective in the few-shot regime [13].

ImageNetV2. Representations are evaluated on ImageNetV2 with the n-shot k-way setup. ImageNetV2 is a dataset traditionally used to evaluate the robustness of models to natural distribution shifts. For our experiments we evaluate accuracy of the model given n examples from the ImageNetV2 distribution. We benchmark representations in the traditional small-scale (10-way) and large-scale (1000-way) setting. We evaluate for $n \in 1, 3, 5, 7, 9$ with 9 being the maximum value for n because there are 10 images per class.

Table 12: 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 10, are bolded.

	D_s	D_r	MFLOPs	Top-1	mAP@10	mAP@25	mAP@50	mAP@100	P@10	P@25	P@50	P@100
Shortlist Length = 200	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 13: 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 8, are bolded.

D_s	Rerank Cascade	Cascade List	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

We observe that MRL has equal performance to FF across all representation sizes and shot numbers. We also find that for both MRL and FF as the shot number decreases, the required representation size to reach optimal accuracy decreases (Table 15). For example, we observe that 1-shot performance at 32 rep. size has equal accuracy to 2048 rep. size.

FLUID. For the long-tailed setting we evaluate MRL on the FLUID benchmark [83] which contains a mixture of pretrain and new classes. Table 16 shows the evalatuation of the learned representation on FLUID. We observe that MRL provides up to 2% accuracy higher on novel classes in the tail of the distribution, without sacrificing accuracy on other classes. Additionally we find the accuracy between low-dimensional and high-dimensional representations is marginal for pretrain classes.

Table 14: 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 8, are bolded.

D_s	Rerank Cascade	Cascade List	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

Table 15: Few-shot accuracy on ImageNetV2 for 1000 way classification. MRL performs equally to FF across all shots and representation sizes. We also observe that accuracy saturates at a lower dimension for lower shot numbers. For example, at 1-shot a rep. size of 32 performs equally to 2048.

Rep. Size	Method	1-Shot	3-Shot	5-Shot	7-Shot	9-Shot
8	FF	35.41	45.73	49.23	50.89	51.72
	MH	35.37	45.69	49.25	50.85	51.73
16	FF	40.88	53.96	57.36	58.72	59.39
	MH	40.90	53.94	57.37	58.65	59.29
32	FF	41.41	54.88	58.28	59.63	60.40
	MH	41.40	54.91	58.30	59.65	60.45
64	FF	41.25	54.83	58.29	59.82	60.61
	MH	41.28	54.80	58.32	59.77	60.69
128	FF	41.36	54.90	58.50	60.05	60.90
	MH	41.38	54.95	58.50	60.06	60.83
256	FF	41.36	54.90	58.50	60.05	60.90
	MH	41.38	54.95	58.50	60.06	60.83
512	FF	41.36	55.05	58.70	60.19	61.02
	MH	41.34	55.14	58.78	60.40	61.18
1024	FF	41.32	55.20	58.85	60.46	61.38
	MH	41.31	55.24	58.86	60.42	61.34
2048	FF	41.18	55.09	58.77	60.38	61.34
	MH	41.16	55.10	58.77	60.40	61.28

For example, the 64-dimensional MRL performs ~ 1 lower in accuracy compared to the 2048-dimensional counterpart on pretrain-head classes (84.46% vs 85.60%). However for novel-tail classes the gap is far larger (6.22% vs 12.88%). We hypothesize that the higher-dimensional representations are required to differentiate the classes when few training examples of each are known. This results provides further evidence that different tasks require varying capacity based on their difficulty.

H Robustness Experiments

We evaluate the robustness of MRL models on out-of-domain datasets (ImageNetV2/R/A/Sketch) and compare them to the FF baseline. Each of these datasets is described in Appendix B. The results in Table 17 demonstrate that learning Matryoshka Representations does not hurt out-of-

Table 16: Accuracy for all methods with varying feature dimension size on the FLUID [83] benchmark. The novel/pretrain categories indicates whether classes were present during ImageNet pretraining and head/tail indicates classes that have greater/less than 50 examples in the streaming test set. We observe that MRL performs better than the baseline on novel tail classes by $\sim 2\%$ on average.

Rep. Size	Method	Pretrain - Head (>50)	Novel - Head (>50)	Pretrain - Tail (<50)	Novel - Tail (<50)	Mean Per Class Acc.	Acc.
8	FF	68.04	**11.30**	33.18	**0.36**	16.29	28.47
	MRL	**71.75**	10.70	**38.29**	0.19	**17.15**	**29.34**
	MRL–E	57.40	6.25	23.14	0.04	11.78	22.81
16	FF	80.74	**19.12**	**63.29**	**2.78**	**25.65**	**37.61**
	MRL	**81.79**	17.90	61.39	1.95	24.73	37.59
	MRL–E	79.08	9.15	60.33	0.08	20.45	30.24
32	FF	**83.67**	**24.30**	**66.66**	**4.23**	**28.86**	**42.40**
	MRL	83.46	23.26	65.82	3.75	28.16	41.90
	MRL–E	81.42	10.47	68.01	0.23	22.31	32.17
64	FF	84.12	27.49	68.20	5.17	30.64	45.18
	MRL	84.46	27.61	67.59	6.22	31.03	45.35
	MRL–E	82.57	13.23	70.18	0.52	23.83	34.74
128	FF	84.87	29.96	68.79	5.54	31.84	47.06
	MRL	84.88	30.86	68.58	8.41	33.23	47.79
	MRL–E	82.76	18.93	64.46	2.22	25.75	39.19
256	FF	84.77	32.78	69.96	7.21	33.65	49.15
	MRL	85.10	32.91	69.39	9.99	34.74	49.39
	MRL–E	82.96	22.63	64.55	3.59	27.64	41.96
512	FF	85.62	35.27	70.27	9.05	35.42	51.14
	MRL	85.62	34.67	70.24	11.43	36.11	50.79
	MRL–E	82.86	25.62	64.34	4.99	29.22	44.20
1024	FF	86.30	37.49	71.12	10.92	37.14	52.88
	MRL	85.64	35.88	70.02	12.19	36.80	51.58
	MRL–E	83.03	27.78	64.58	6.32	30.57	45.71
2048	FF	86.40	37.09	71.74	10.77	37.04	52.67
	MRL	85.60	36.83	70.34	12.88	37.46	52.18
	MRL–E	83.01	29.99	65.37	7.60	31.97	47.16

Table 17: 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

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 18.

I In Practice Costs

All approximate NN search experiments via HNSW32 were run on an Intel Xeon 2.20GHz CPU with 24 cores. All exact search experiments were run with CUDA 11.0 on 2xA100-SXM4 NVIDIA GPUs with 40G RAM each.

Table 18: Zero-shot image classification 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

MRL models 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.

Table 19: Retrieval k-NN wall clock search times (s) over entire valset (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

Table 20: Faiss [42] index size and build times for exact k-NN search with L2 Distance metric and approximate k-NN search with HNSW32 [55].

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

Table 21: Retrieval k-NN wall clock search times (s) over entire valset (query set) of ImageNet-1K over various shortlist lengths.

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

Retrieval 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 tabulate wall clock search time for

every query in the ImageNet-1K and ImageNet-4K validation sets over all rep. size d in Table 19 for both Exact Search and HNSW32, and ablate wall clock query time over shortlist length k on the ImageNet-1K validation set in Table 21. The wall clock time to build the index and the index size is also shown in Table 20.

J Analysis of Model Disagreement

Class Trends *Does increasing representation size necessarily help improve the classification performance of 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), 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 FFmodels, 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 rep. sizes by training Matryoshka Representations. Figure 10 also showcases the progression of relative per-class accuracy distribution compared to the 2048 dimensional model. This also shows that some instances and classes could benefit from the lower-dimensional representations.

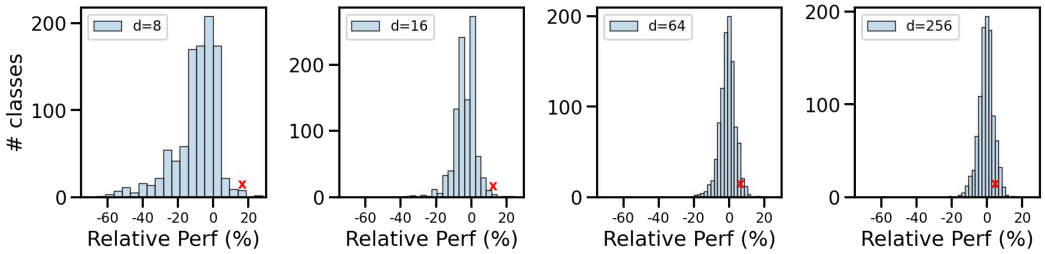


Figure 10: Progression of relative per-class accuracy vs MRL-2048. As the dimensionality increases, the spread shrinks while the class marked (**x**) (Madagascar cat) loses accuracy.

Discussion of Oracle Accuracy Based on our observed model disagreements for different rep. sizes d, we defined an optimal *oracle* accuracy [52] for MRL. We labeled an image as correctly predicted if classification using any rep. size was right. The percentage of total samples of ImageNet-1K that were correctly predicted for each rep. size d is shown in Table 22. This defined an upper bound on the performance of MRL models, as 18.52% of the ImageNet-1K validation set were incorrectly predicted $\forall d \in \{8, 2048\}$. We show the oracle performance on MRL models for ImageNet-1K/V2/A/R/Sketch datasets in Table 23.

Table 22: Percentage of ImageNet-1K validation set that is correctly predicted by each representation size. We note that 18.46% of the samples cannot be correctly predicted by any rep. size. The remaining 81.54% constitutes the oracle accuracy.

Rep. Size	8	16	32	64	128	256	512	1024	2048	Always Wrong
Correctly Predicted	67.46	8.78	2.58	1.35	0.64	0.31	0.20	0.12	0.06	18.52

Table 23: 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

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 D.1. This led to an 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 accurate policy for future work.

Grad-CAM Examples We analyzed the nature of model disagreement across representation sizes with MRL models with the help of Grad-CAM visualization [71]. We observe there are certain classes in ImageNet-1K such as "tools", "vegetables" and "meat cutting knife" which are at times around multiple objects and a cluttered environment. In such scenarios, we observed that smaller rep. 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 To quantify the performance of MRL model on ImageNet-1K superclasses, we create a 30 superclass subset of the validation set based on wordnet hierarchy (Table 24). These 30 superclasses contain 467 out of 1000 classes, and we have an additional class named reject, when an image does not belong to any of the sueprclass. We see that there is a jump in performance from Rep. Size 8 to 16 because 8 dimensional model was confusing images from superclass vegetable with reject token. We further show 8 of those 30 superclasses and plot their improvement with Rep. Size in Figure 11.

Table 24: 30 Superclasses in ImageNet-1K corresponding to the performance in Table 25.

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 25: Performance of MRL model on 31-way classification (1 extra class is for reject token). We see that there is a jump in performance from Rep. Size 8 to 16 because 8 dimensional model was confusing images from superclass vegetable with reject token.

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

K Ablation Studies

K.1 MRL Training Paradigm

Nesting as Finetuning. To observe if nesting can be induced in models that were not explicitly trained with nesting from scratch, we load a pretrained FF2048 ResNet50 model and initialize a new MRL layer, as defined in Algorithm 2, Appendix C. Different layers of the backbone are then unfrozen to observe how much non-linearity in the form of unfrozen conv layers needs to be present to enforce nesting into a pretrained FF model. A description of these layers can be found in the ResNet50 architecture [27]. All models are 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 observe that finetuning the linear layer alone is insufficient for learning Matryoshka Representations at lower dimensionalities. Adding more and more non-linear conv+ReLU layers steadily improves classification accuracy of Rep. Size $= 8$ from 5% to 60% after finetuning, which is only 6% less than training MRL end-to-end for 40 epochs. This difference is successively less pronounced as we increase dimensionality past Rep. Size $= 64$, to within 1.5% for all larger dimensionalities. The full results of this ablation can be seen in Table 26.

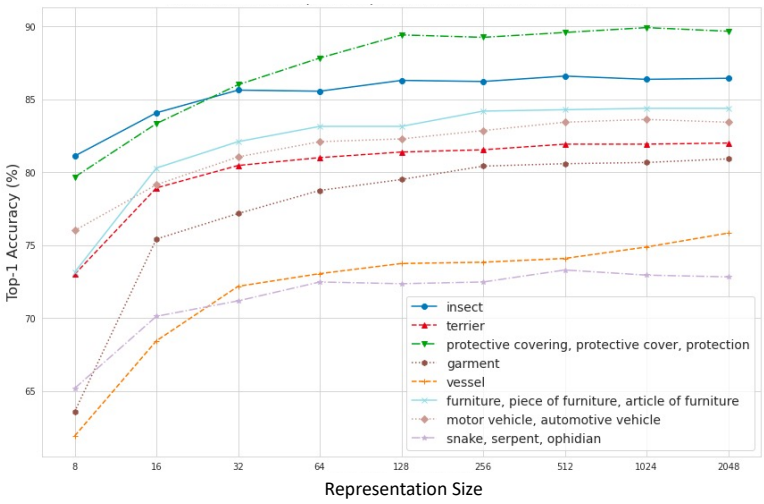


Figure 11: Improvement within a particular superclass classification with increasing Rep. Size ImageNet-1K.

Table 26: Top-1 Classification Accuracy (%) on ImageNet-1K of various ResNet50 models which are finetuned on pretrained FF2048 model. Different layers of the backbone are then unfrozen to observe how much non-linearity in the form of unfrozen conv layers needs to be present to enforce nesting into a pretrained FF model. We observed that adding more and 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	4.2 conv3, fc	4.2 conv3, conv2, fc	4.2 full, fc	fc	All (MRL)
8	36.11	54.78	60.02	5.15	66.63
16	58.42	67.26	70.10	13.79	73.53
32	67.81	71.62	72.84	32.52	75.03
64	72.42	73.61	74.29	52.66	75.82
128	74.41	74.67	75.03	64.60	76.30
256	75.30	75.23	75.38	69.29	76.47
512	75.96	75.47	75.64	70.51	76.65
1024	76.18	75.70	75.75	70.19	76.76
2048	76.44	75.96	75.97	69.72	76.80

Relative Importance. We perform an ablation of MRL over the relative importance, c_m, of different nesting dimensions $m \in \mathcal{M}$, as defined in Sec. 3. In an attempt to improve performance at lower dimensionalities, we boost the relative importance c_m of training loss at lower dimensions as in Eq. 1 with two models, MRL-8boost and MRL-8+16boost. The MRL-8boost model has $c_{m\in\mathcal{M}} = [2, 1, 1, 1, 1, 1, 1, 1, 1]$ and the MRL-8+16boost model has $c_{m\in\mathcal{M}} = [2, 1.5, 1, 1, 1, 1, 1, 1, 1]$. The relative importance list $c_{m\in\mathcal{M}}$ has a 1-to-1 correspondence with nesting dimension set $\mathcal{M}$. In Table 27, we observe that MRL-8boost improves Top-1 accuracy by 3% at Rep. Size $= 8$, and also improves Top-1 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.

K.2 Retrieval

Adaptive Retrieval. To examine the effect of increasing shortlist lengths on search time, a reranking ablation over shortlist lengths is also performed for D_s= 16 and D_r= 2048 over ImageNet-1K in Table 28, and over ImageNet-4K in Table 29.

Table 27: An ablation over boosting training loss at lower nesting dimensions, with Top-1 and Top-5 accuracy (%). The models are described in Appendix K.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

Table 28: Adaptive retrieval ablation over shortlist length k for $D_s = 16$, $D_r = 2048$ on ImageNet-1K with exact search.

Shortlist Length	Top-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

Table 29: 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