Matryoshka Representation Learning

Aditya Kusupati[*†◇], **Gantavya Bhatt**[*†], **Aniket Rege**[*†],
Matthew Wallingford[†], **Aditya Sinha**[◇], **Vivek Ramanujan**[†], **William Howard-Snyder**[†],
Kaifeng Chen[◇], **Sham Kakade**[‡], **Prateek Jain**[◇] **and Ali Farhadi**[†]
[†]University of Washington, [◇]Google Research, [‡]Harvard University
{kusupati,ali}@cs.washington.edu, prajain@google.com

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. MRL is task agnostic and can be easily adapted to self-supervised learning setups even at web-scale. The flexibility within the learned Matryoshka Representations offer: (a) up to $\mathbf{8}\times$ smaller embedding size for web-scale ALIGN model at the same level of accuracy; (b) Within 0.5% accuracy of pretrained masked-language model (BERT) with no hyperparameter tuning. MRL code and pretrained models will be open-sourced.

1 Introduction

Learned representations [28] are fundamental building blocks of real-world ML systems [32, 46]. Trained once and frozen, d-dimensional representations encode rich information and can be used to perform multiple downstream tasks [3]. The deployment of deep representations has two steps: (1) an expensive yet constant-cost forward pass to compute the representation [15] and (2) utilization of the representation for downstream applications [25, 45]. Compute costs for the latter part of the pipeline scale with the embedding dimensionality as well as the data size (N) and label space (L). At web-scale [7, 43] 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 [14, 18]. However, perhaps due to the inductive bias of gradient-based training [42], 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 [15], jointly optimizing sub-networks of varying capacity [5, 52] or post-hoc compression [21, 29]. Each of these techniques struggle to meet the requirements for adaptive large-scale deployment either due to training/maintenance overhead, numerous expensive forward passes through all of the data, storage and memory cost for multiple copies of encoded data, expensive on-the-fly feature selection or a significant drop in accuracy.

[*]Equal contribution – AK led the project with extensive support from GB and AR for experimentation.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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 $O(\log(d))$ lower-dimensional vectors in a nested fashion, hence the name Matryoshka. MRL can be adapted to any existing representation pipeline and is easily extended to many standard tasks in computer vision and natural language processing. Figure 1 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 significant training or additional deployment overhead. MRL equips the representation vector with the desired flexibility and multi-fidelity that can ensure a near-optimal accuracy-vs-compute trade-off. With these advantages, MRL enables adaptive deployment based on accuracy and compute constraints.

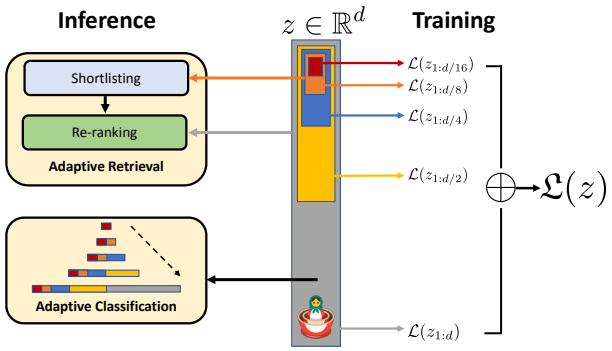

Figure 1: Matryoshka Representation Learning is adaptable to any representation learning setup and begets a Matryoshka Representation z by optimizing the original loss $\mathcal{L}(.)$ at $O(\log(d))$ chosen representation sizes. Matryoshka Representation can be utilized effectively for adaptive deployment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

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 $8\times$ smaller embedding for web-scale classification using MRL (Section 4.1).
3. Seamless adaptation of MRL across representation learning setups, modalities (vision - ViT, vision + language - ALIGN, language - BERT) and to web-scale data (JFT-300M & ALIGN data).

2 Related Work

Representation Learning. Large-scale datasets like ImageNet [8, 38] and JFT [43] enabled the learning of general purpose representations for computer vision [3, 50]. These representations are typically learned through supervised and un/self-supervised learning paradigms. Supervised pretraining [15, 26, 40] casts representation learning as a multi-class/label classification problem, while un/self-supervised learning learns representation via proxy tasks like instance classification [49] and reconstruction [17, 31]. Recent advances [6, 16] in contrastive learning [13] enabled learning from web-scale data [11] that powers large-capacity cross-modal models [9, 23, 35, 53]. Similarly, natural language applications are built [22] on large language models [4] that are pretrained [33, 37] in a un/self-supervised fashion with masked language modeling [10] or autoregressive training [34].

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.

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

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

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 $m \in \mathcal{M}$.

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 [45, 51]. 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 Self-Supervised Learning Frameworks. MRL can be adapted seamlessly to most representation learning frameworks at web-scale with minimal modifications (Section 4.1). For example, MRL's adaptation to masked language modeling 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.

4 Applications

4.1 Representation Learning

We adapt Matryoshka Representation Learning (MRL) to various self-supervised representation learning setups (a) Contrastive learning for vision + language: ALIGN model with ViT-B/16 vision encoder and BERT language encoder on ALIGN data [23] and (b) Masked language modeling: BERT [10] on English Wikipedia and BooksCorpus [54]. Please refer to Appendices B and D 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. ViT-B/16 and BERT-Base output 768-dimensional embeddings for each data point. We use $\mathcal{M} = \{12, 24, 48, 96, 192, 384, 768\}$ as the explicitly optimized nested dimensions.

In section 4.2, we evaluate the quality and capacity of the learned representations through 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

We 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 2, 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.

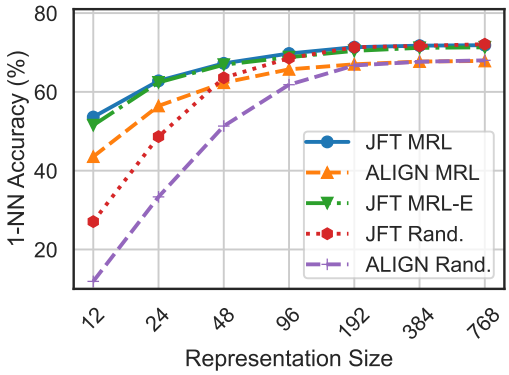

Figure 2: ImageNet-1K 1-NN accuracy for ViT-B/16 models trained on JFT-300M & as part of ALIGN. MRL scales seamlessly to web-scale with minimal training overhead.

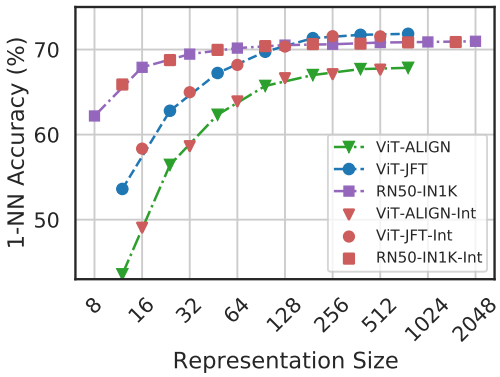

Figure 3: Despite optimizing MRL only for $O(\log(d))$ dimensions for ResNet50 and ViT-B/16 models; the accuracy in the intermediate dimensions shows interpolating behaviour.

We also evaluated the capability of Matryoshka Representations to extend to other natural language processing via masked language modeling (MLM) with BERT [10], whose results are tabulated in Table 1. 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 1: Masked Language Modeling (MLM) accuracy (%).

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

4.3 Robustness

We find that the zero-shot robustness of ALIGN-MRL (Table 8 in Appendix G) agrees with the observations made by Wortsman et al. [48]. Table 7 in Appendix F shows that MRL also improves the cosine similarity span between positive and random image-text pairs.

5 Conclusions

In conclusion, we presented Matryoshka Representation Learning (MRL), a flexible representation learning approach that encodes information at multiple granularities in a single embedding vector. This enables the MRL to adapt to a downstream task's statistical complexity as well as the available compute resources. We show that MRL is extremely easy to adapt to self-supervised learning setups like ALIGN and BERT – making it a strong alternative to simple dense and rigid representations. Finally, we demonstrate that Matryoshka Representations provide excellent accuracy-compute tradeoff at lower dimensions, even at web-scale (ALIGN and JFT).

Acknowledgments

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

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

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

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

[23] 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.

[24] 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.

[25] 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.

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

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

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

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

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

[31] 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.

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

[33] 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.

[34] 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/.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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/.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

Contents

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.

Algorithm 1 Pytorch code for Matryoshka Cross-Entropy Loss

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

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

Algorithm 2 Pytorch code for MRL Linear Layer

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

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

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

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

               return nesting_logits
```

B Datasets

ImageNet-1K [38] 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 [27].

JFT-300M [43] is a large-scale multi-label dataset with 300M images labelled across 18,291 categories.

ALIGN [23] 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 [36] is a collection of 10K images sampled a decade after the original construction of ImageNet [8]. ImageNetV2 contains 10 examples each from the 1,000 classes of ImageNet-1K.

ImageNet-A [20] contains 7.5K real-world adversarially filtered images from 200 ImageNet-1K classes.

ImageNet-R [19] contains 30K artistic image renditions for 200 of the original ImageNet-1K classes.

ImageNet-Sketch [47] 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 Supervised Learning

To demonstrate the flexibility of MRL, we also present its 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 [15] on ImageNet-1K [38] 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. Note that we do not search for best hyper-parameters for all MRL experiments but use the same hyper-parameters as the independently trained baselines. 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 [52]) and randomly selected features of the highest capacity FF model. We discuss the implementation and results of supervised classification in Appendix D and E respectively. We also discuss in-practice costs of MRL and ablation studies over training methodologies in Appendix H and I.

D Matryoshka Representation Learning Model Training

We trained all ResNet50–MRL models using the efficient dataloaders of FFCV [27]. 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 [41]. 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 [44] 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 [24] 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 [39] with a linear learning rate decay starting at 1e-3.

Lastly, we trained a BERT-Base model on English Wikipedia and BookCorpus. We trained our models in Tensorflow using a 4x4 cloud TPU pod with a total batchsize of 1024. We used AdamW [30] optimizer with a linear learning rate decay starting at 1e-4 and trained for 450K steps.

In each configuration/case, if the final representation was normalized in the FF implementation, MRL models adopted the same for each nested dimension for a fair comparison.

E Classification Results

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

- FF: We utilize the FF-k models described in Appendix D for $k \in \{8, ...2048\}$.
- **SVD**: We performed a low rank approximation of the 1000-way classification layer of FF-2048, with rank = 1000.
- **Rand. LP**: We compared against a linear classifier fit on randomly selected features [16].
- **Slim. Net**: We take pretrained slimmable neural networks [52] which are trained with a flexible width backbone (25%, 50%, 75% and full width). For each representation size, we consider the first k dimensions for classification. Note that training of slimmable neural networks becomes unstable when trained below 25% width due to the hardness in optimization and low complexity of the model.

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

Table 2: Top-1 classification accuracy (%) for ResNet50 MRL and baseline models on ImageNet-1K.

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

We compared the performance of MRL models at various representation sizes via 1-nearest neighbors (1-NN) image classification accuracy on ImageNet-1K in Table 3. 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 [52] as an additional baseline. We observed these baseline models to perform very poorly at lower dimensions, as they were not explicitly trained to learn Matryoshka Representations.

E.1 Adaptive Classification (MRL–AC)

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

Table 3: 1-NN accuracy (%) on ImageNet-1K for various ResNet50 models.

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

the prediction confidence p_i using representation size m_i exceeds a learned threshold t_i^*. If $p_i \geq t_i^*$, we used predictions from representation size m_i otherwise, we increased to representation size m_{i+1}. To learn the optimal threshold t_i^*, we performed a grid search between 0 and 1 (100 samples). For each threshold t_k, we computed the classification accuracy over our 10K image 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., $\{t_j^* \mid j \in \{8, 16, 32, 64, \ldots, 2048\}\}$. To improve reliability of threshold based greedy policy, we use test time augmentation which has been used successfully in the past [40]. For inference, we used the remaining held-out 40K samples from the ImageNet-1K

Table 4: Threshold-based adaptive classification performance of ResNet50 MRL on a 40K sized held-out subset of the ImageNet-1K validation set. Results are averaged over 30 random held-out subsets.

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

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 4, we observed that in expectation, we only needed a ~ 37 sized representation to achieve 76.3% classification accuracy on ImageNet-1K, which was roughly $14\times$ smaller than the FF–512 baseline. Even if we computed the expectation as a weighted average over the cumulative sum of representation sizes $\{8, 24, 56, \ldots\}$, due to the nature of multiple linear heads for MRL, we ended up with an expected size of 62 that still provided a roughly $8.2\times$ efficient representation than the FF–512 baseline. However, MRL–E alleviates this extra compute with a minimal drop in accuracy.

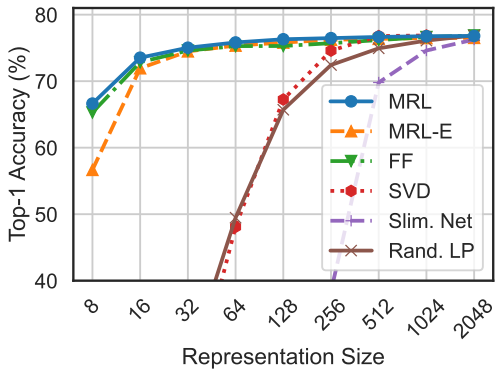

Figure 4: ImageNet-1K linear classification accuracy of ResNet50 models. MRL is as accurate as the independently trained FF models for every representation size.

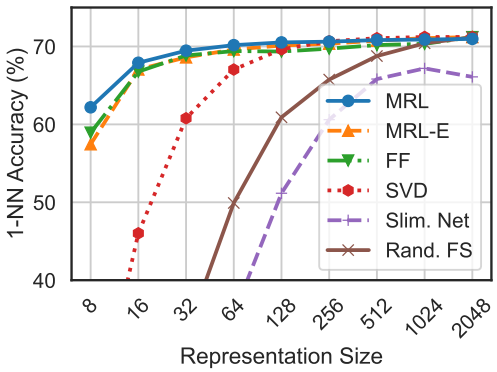

Figure 5: ImageNet-1K 1-NN accuracy of ResNet50 models measuring the representation quality for downstream task. MRL outperforms all the baselines across all representation sizes.

F JFT, ALIGN and BERT

Table 5: 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

Table 6: Examining top-1 and top-5 k-NN accuracy (%) at interpolated hidden dimensions for ALIGN and JFT. This indicates that MRL is able to scale classification accuracy as hidden dimensions increase even at dimensions that were not explicitly considered during training.

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

Table 7: 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

We examine the k-NN classification accuracy of learned Matryoshka Representations via ALIGN–MRL and JFT-ViT–MRL in Table 5. For ALIGN [23], 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 [12] for JFT-300M [43] 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 6, despite not being trained explicitly at these dimensions. Lastly, Table 7 shows that MRL training leads to a increase in the cosine similarity span between positive and random image-text pairs.

G Robustness Experiments

We evaluate the robustness of MRL models on out-of-domain datasets (ImageNetV2/R/A/Sketch), each of which is described in Appendix B. For an ALIGN–MRL model, we examine the the robustness via zero-shot retrieval on out-of-domain datasets, including ObjectNet, in Table 8.

Table 8: Zero-shot top-1 image classification accuracy (%) of a ALIGN-MRL model on ImageNet-V1/V2/R/A and ObjectNet.

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

H In Practice Costs

MRL **Memory Cost** 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.

MRL **Inference Cost** Our FF-2048 baseline follows the original ResNet-50 model architecture [15], which has 3.8 billion floating-point multiply-adds (FLOPs). MRL–E adds zero overhead to FF-2048 due to its shared logit head. MRL adds a cost of $2m * n$ FLOPs for each additional dimension $m \in \mathcal{M}$, where n is the number of classes, i.e. 1000 for ImageNet-1K. With nesting dimensions $\mathcal{M} = \{8, 16, \ldots, 1024, 2048\}$, MRL thus adds 4.08 million FLOPs to FF-2048, which is an increase of merely 0.1%.

I Ablation Studies

I.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 loaded a pretrained FF-2048 ResNet50 model and initialized a new MRL layer, as defined in Algorithm 2, Appendix D. We then unfroze different layers of the backbone to observe how much non-linearity in the form of unfrozen conv layers needed to be present to enforce nesting into a pretrained FF model. A description of these layers can be found in the ResNet50 architecture [15]. All models were finetuned with the FFCV pipeline, with same training configuration as in the end-to-end training aside from changing lr $= 0.1$ and epochs $= 10$. We

Table 9: Top-1 classification accuracy (%) on ImageNet-1K of various ResNet50 models which are finetuned on pretrained FF-2048 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	fc	4.2 conv3, fc	4.2 conv2, conv3, fc	4.2 full, fc	All (MRL)
8	5.15	36.11	54.78	60.02	66.63
16	13.79	58.42	67.26	70.10	73.53
32	32.52	67.81	71.62	72.84	75.03
64	52.66	72.42	73.61	74.29	75.82
128	64.60	74.41	74.67	75.03	76.30
256	69.29	75.30	75.23	75.38	76.47
512	70.51	75.96	75.47	75.64	76.65
1024	70.19	76.18	75.70	75.75	76.76
2048	69.72	76.44	75.96	75.97	76.80

Table 10: An ablation over boosting training loss at lower nesting dimensions, with top-1 and top-5 accuracy (%). The models are described in Appendix I.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

observed that finetuning the linear layer alone was insufficient to learn Matryoshka Representations at lower dimensionalities. Adding more and more non-linear conv+ReLU layers steadily improved classification accuracy of $d = 8$ from 5% to 60% after finetuning, which was only 6% less than training MRL end-to-end for 40 epochs. This difference was successively less pronounced as we increased dimensionality past $d = 64$, to within 1.5% for all larger dimensionalities. The full results of this ablation can be seen in Table 9.

Relative Importance. We performed an ablation of MRL over the relative importance, c_m, of different nesting dimensions $m \in \mathcal{M}$, as defined in Sec. 3. In an attempt to improve performance at lower dimensionalities, we boosted the relative importance c_m of training loss at lower dimensions as in Eq. 1 with two models, MRL-8boost and MRL-8+16boost. The MRL-8boost model had $c_{m\in\mathcal{M}} = [2, 1, 1, 1, 1, 1, 1, 1, 1]$ and the MRL-8+16boost model had $c_{m\in\mathcal{M}} = [2, 1.5, 1, 1, 1, 1, 1, 1, 1]$. The relative importance list $c_{m\in\mathcal{M}}$ had a 1-to-1 correspondence with nesting dimension set $\mathcal{M}$. In Table 10, we observed that MRL-8boost improves top-1 accuracy by 3% at $d = 8$, 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.