Starbucks: Improved Training for 2D Matryoshka Embeddings

Anonymous ACL submission

Abstract

Effective approaches that can scale embedding model depth (i.e. layers) and embedding size allow for the creation of models that are highly scalable across different computational resources and task requirements. While the recently proposed 2D Matryoshka training approach can efficiently produce a single embedding model such that its sub-layers and sub-dimensions can measure text similarity, its effectiveness is significantly worse than if smaller models were trained separately. To address this issue, we propose Starbucks, a new training strategy for Matryoshka-like embedding models, which encompasses both the fine-tuning and pre-training phases. For the fine-tuning phase, we discover that, rather than sampling a random sub-layer and sub-dimensions for each training steps, providing a fixed list of layer-dimension pairs, from small size to large sizes, and computing the loss across all pairs significantly improves the effectiveness of 2D Matryoshka embedding models, bringing them on par with their separately trained counterparts. To further enhance performance, we introduce a new pre-training strategy, which applies masked autoencoder language modelling to sub-layers and sub-dimensions during pre-training, resulting in a stronger backbone for subsequent fine-tuning of the embedding model. Experimental results on both semantic text similarity and retrieval benchmarks demonstrate that the proposed pre-training and fine-tuning strategies significantly improved the effectiveness over 2D Matryoshka models, enabling Starbucks models to perform more efficiently and effectively than separately trained models.[1]

1 Introduction

Matryoshka Representation Learning (MRL) (Kusupati et al., 2022) is an embedding model training technique that enables models to encode information at different granularities, i.e., using different embedding dimension sizes, allowing them to adapt to the computational constraints of downstream tasks. Recently, the 2D Matryoshka Sentence Embeddings (2DMSE)[2] fine-tuning method was proposed to further enhance the flexibility of embedding models (Li et al., 2024). 2DMSE enables adaptability not only in embedding dimensionality but also across model layers, providing elastic embedding generation from a single embedding model training process. This method produces efficiency gains because a single embedding model needs to be created to serve diverse requirements in terms of embedding size and model layers, in place of needing to train separate individual embedding models for different combinations of embedding size and layers number.

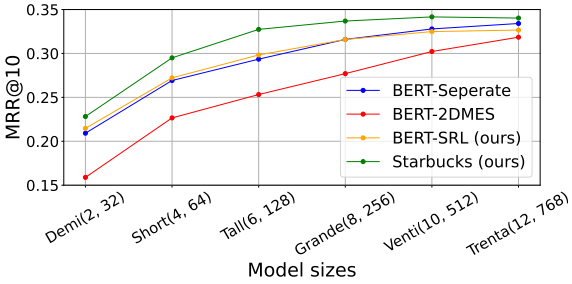


Figure 1: MRR@10 on MS MARCO obtained by different 2D Matryoshka training methods. The x axis refers to model sizes in the format *SizeName(number of layers, embedding dimension)*.

In this paper, however, we identify that the effectiveness of embeddings generated by 2DMSE's sub-layers and sub-dimensions is significantly lower than if separate embedding models were trained each with the different target number of layers and dimensions. For instance, in Figure 1, we plot the effectiveness of BERT-based embedding models trained on the MS MARCO dataset for the passage

[1] Code available at Anonymous github link.

[2] Also known as Espresso Sentence Embeddings (ESE).

retrieval task: The MRR@10 of embeddings generated by 2DMSE is significantly lower than that achieved by embedding models trained separately.

This observation suggests that, although training multiple embedding models with different layer and dimension sizes is computationally more expensive than 2DMSE, separate training remains the preferred approach because (1) embedding models trained this way are more effective, (2) model training is a one-time process, and (3) in practice, rather than all possible combinations, only a few target model and dimension size variations (from small to large) are typically required, making the computational overhead acceptable.

To make 2D Matryoshka representation learning more practical and effective, we propose Starbucks: a new MRL fine-tuning and pre-training method. Starbucks is composed of two key processes: the Starbucks Masked Autoencoding (SMAE) pre-training and the Starbucks Representation Learning (SRL) fine-tuning processes. In Starbucks, the model loss is computed based on a limited target list of layer-dimension pairs, ranging from smaller to larger sizes.

Once SRL training is completed, embeddings can be generated with targeted layer-dimension sizes, from small to large, much like how the coffeehouses chain Starbucks offers coffee in different cup sizes, from Demi to Trenta, as shown in Figure 2.[3] Our empirical results show that with this simple adjustment in the loss function, the effectiveness of small embedding models trained using Starbucks matches that of separately trained models with equivalent model sizes and dimensions, making 2D Matryoshka representation learning practical (see Figure 1).

For the SMAE pre-training task, we draw inspiration from MatFormer (Devvrit et al., 2023), which introduces Matryoshka-style unsupervised pre-training for transformer models, and Masked Autoencoding (MAE) language modeling (Gao and Callan, 2021; Xiao et al., 2022; Wang et al., 2023; Zhuang et al., 2023), a CLS token embedding enhancement pre-training approach for embedding models. SMAE combines the benefits of these two approaches by applying MAE to embeddings from a list of various layer-dimension pairs, tailoring the model's targeted layers and dimensions for downstream SRL.

[3]Starbucks coffee cup sizes from small to large are: Demi, Short, Tall, Grande, Venti, and Trenta.

We comprehensively compare Starbucks with 2DMSE and separately trained embedding models on both STS and retrieval datasets. Our results demonstrate that the combination of SMAE pre-training and SRL fine-tuning significantly enhances the effectiveness of 2D Matryoshka models.

2 Related Work

Transformer-based text embedding models are highly effective across various NLP tasks, such as semantic text similarity (Muennighoff et al., 2023; Reimers and Gurevych, 2019; Gao et al., 2021; Li and Li, 2024) and information retrieval (Thakur et al., 2021; Karpukhin et al., 2020; Zhao et al., 2024; Zhuang et al., 2024) The Matryoshka Representation Learning (MRL) approach aims to make embedding models more efficient and practical by utilising sub-dimensions of the embeddings to calculate the training loss (Kusupati et al., 2022). After MRL training, all corresponding embeddings sub-dimensions can effectively represent the text, enabling more efficient inference for applications (Lee et al., 2024) with adaptive algorithms (Rege et al., 2023).

The original MRL method provided adaptable embeddings only across dimensions. The 2D Matryoshka Sentence Embeddings (2DMSE) extends MRL by incorporating embeddings from multiple model layers, each associated with different embedding sizes, into the training objective (Li et al., 2024). As a result, embedding models trained with 2DMSE can adapt not only to varying embedding dimensions but also to different model sizes. Our proposed Starbucks also provides adaptive embedding dimensions and model sizes but is targeted at a set of fixed dimension-layer pairs, ranging from small to large. We demonstrate that our Matryoshka-like training strategy is more effective than 2DMSE. Figure 2 illustrates the difference between Starbucks and previous MRL methods.

In addition to MRL for embedding models, other works have explored Matryoshka-like training within unsupervised language modelling tasks. MatFormer introduced a nested transformer block for decoder-only language models (Devvrit et al., 2023). Inspired by MatFormer, we devised Starbucks Masked Autoencoding (SMAE) language modelling (Gao and Callan, 2021; Xiao et al., 2022; Wang et al., 2023; Zhuang et al., 2023), an unsupervised Matryoshka pre-training task specifically designed for embedding models.

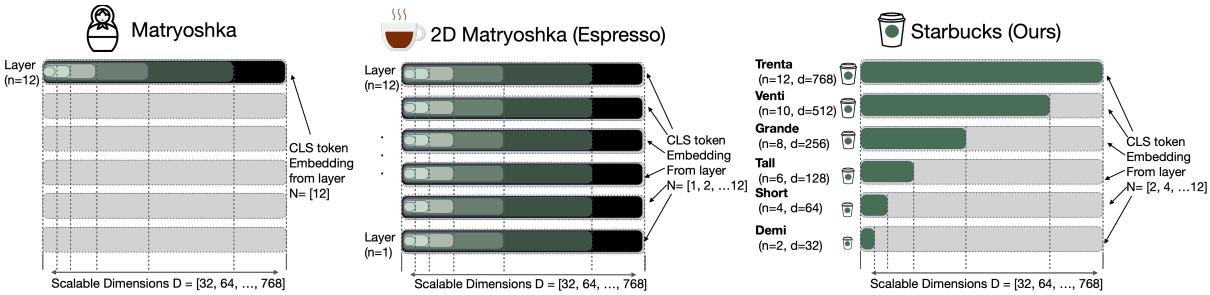


Figure 2: Starbucks (right) provides a single model with flexible configurations in terms of embedding dimensionality and network layer. Also shown is the original Matryoshka model (left) and extension 2D Matryoshka (center).

3 Method

There are two key components of Starbucks training: the Starbucks Representation Learning (SRL) fine-tuning process and the Starbucks Masked Autoencoding (SMAE) pre-training process.

3.1 Starbucks Representation Learning (SLR)

Previous methods for fine-tuning 2D Matryoshka embedding models attempt to train all combinations of sub-layers and sub-dimensions of the embedding model simultaneously during the fine-tuning process. Formally, in 2D Matryoshka embedding learning, given a transformer-based encoder model, an input text x is encoded into vector representations:

$$\begin{aligned} V_n^d &= \text{encoder}_{1:n}(x)_{1:d} \in \mathbb{R}^{|x| \times d} \\ v_n^d &= \text{pool}(V_n^d) \in \mathbb{R}^{1 \times d}, \end{aligned} \tag{1}$$

where V_n^d is the contextualized embedding matrix outputted by the encoder model's nth layer and d sub-dimensions, and pool$(\cdot)$ is the pooling function, e.g., mean pooling or CLS token pooling, and v_n^d is the final embedding representation for the input text x. The fine-tuning objective of 2D Matryoshka embedding learning is as follows: given a target set of model layers $\mathcal{N} = [1, 2, ..., N]$ and embedding dimensions $\mathcal{D} = [1, 2, ..., D]$, the training loss is defined as:

$$\mathcal{L}_{\mathcal{N}}^{\mathcal{D}} = \sum_{n \in \mathcal{N}} \sum_{d \in \mathcal{D}} \text{loss}(v_n^d; A), \tag{2}$$

where loss$(\cdot)$ is the loss function for embedding learning, e.g., contrastive loss, and A represents the training data required to compute the loss.

Since the encoder model could be a large language model with many transformer layers and large embedding dimensions, in practice, a target sub-layer and sub-dimension are often sampled, along with the full model and dimension, to compute the loss at each training step (Li et al., 2024):

$$\begin{aligned} n &\sim \mathcal{U}(1, N-1) \\ d &\sim \mathcal{U}(1, D-1) \\ \mathcal{L}_n^d &= \text{loss}(v_n^d; A) \\ \mathcal{L}_{\text{2D}} &= \mathcal{L}_n^d + \mathcal{L}_n^D + \mathcal{L}_N^d + \mathcal{L}_N^D, \end{aligned} \tag{3}$$

thus ensuring that training stochastically covers all sub-layers and sub-dimensions of the model.

However, we argue that the training objective described above, which attempts to cover all combinations of possible sub-layers and sub-dimensions, is unnecessarily complex for 2D Matryoshka representation learning. As we demonstrate in both the introduction and the results section, while this objective allows each combination of sub-layers and sub-dimensions to function in STS and retrieval tasks, the effectiveness of the learned representations is significantly worse than when training smaller models independently. Moreover, in real-world scenarios, it is often unnecessary to offer all possible model sizes and sub-dimensions at inference. Instead, a pre-defined list of model sizes and dimensions is typically chosen based on the available computational budget and required task.

Conversely, our Starbucks Representation Learning computes the loss only for a limited list of layer-dimension pairs, ranging from smaller to larger, to form the final loss. That is, given a list of layer-dimension pairs $S = [(n_1, d_1), (n_2, d_2), \ldots, (N, D)]$, where $n_1 < n_2 < \cdots < N$ and $d_1 < d_2 < \cdots < D$, the loss function is calculated as follows:

$$\mathcal{L}_S = \frac{1}{|S|} \sum_{(n,d) \in S} \text{loss}(v_n^d; A). \tag{4}$$

For example, in our experiments, we use BERT (Devlin et al., 2019) as the text encoder model, which has $N = 12$ layers and

$D = 768$ embedding dimensions. We set $S = [(2, 32), (4, 64), (6, 128), (8, 256), (10, 512), (12, 768)]$ as the pre-defined list of model sizes and dimensions. Thus, only 6 losses are computed to form the final loss.

Following prior 2D Matryoshka representation learning methods, we also incorporate a KL divergence loss to align the embeddings of the full-layer, full-dimension size with those of smaller layer-dimension sizes:

$$\mathcal{L}_{\text{KL}} = \frac{1}{|S|} \sum_{(n,d)\in S} \text{KLD}(s_n^d, s_N^D), \tag{5}$$

where $\text{KLD}(\cdot)$ represents the Kullback-Leibler divergence, and s_n^d is the logits score distribution produced by the nth layer and the first d sub-dimension embedding. The final loss for our Starbucks Representation Learning is given by $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_{\text{KL}}$.

3.2 Starbucks Masked Autoencoding (SMAE)

Previous Matryoshka representation learning methods focused only on the fine-tuning stage. In this section, we introduce our Starbucks MAE, which is designed to obtain a better backbone model for the downstream Starbucks Representation Learning.

SMAE adapts the idea of Matryoshka training to the pre-training phase, and it combines it with MAE language modeling training. Specifically, the model architecture for our pre-training method consists of an encoder and a decoder, both BERT-like transformer models, where the decoder is much smaller than the encoder. The pre-training task is masked language modeling (MLM), where a certain ratio of input text tokens x are randomly replaced with mask tokens:

$$\begin{aligned} x_{\text{enc}} &= \text{Mask}(x, p_{\text{enc}}) \\ x_{\text{dec}} &= \text{Mask}(x, p_{\text{dec}}), \end{aligned} \tag{6}$$

where p_{enc} and p_{dec} are the masking ratios for the encoder and decoder inputs, respectively, and x_{enc} and x_{dec} represent the input tokens with masked tokens for the encoder and decoder. Typically, p_{dec} is larger than p_{enc}. The embedding associated with the CLS token in x_{dec} is replaced by the CLS token output embedding from a target encoder layer and dimension. Let $\vec{x}_{\text{dec}}^{\text{cls}} \in \mathbb{R}^{1\times D}$ be the CLS token embedding of the decoder input embeddings. Given a target layer n and dimension d, the CLS embedding is computed as follows:

$$\begin{aligned} V_n^d &= \text{encoder}_{1:n}(x_{\text{enc}})_{1:d} \\ \vec{x}_{\text{dec}}^{\text{cls}} &= \text{pool}(V_n^d \times W_{1:d}), \end{aligned} \tag{7}$$

where $W \in \mathbb{R}^{D\times D}$ is a projection matrix and $W_{1:d} \in \mathbb{R}^{d\times D}$ represents the first d rows of the matrix. The matrix W projects the encoder output embeddings from sub-dimension d into the full dimension size, ensuring compatibility with the decoder input. The same projection is used to compute the masked token logits for the MLM head, as the full dimension size is required.

Next, we compute the MLM loss for both the decoder and the targeted encoder layer and dimension:

$$\begin{aligned} \mathcal{L}_{\text{enc}}(n, d) &= -\frac{1}{|M_{\text{enc}}|} \sum_{i\in M_{\text{enc}}} \log P(x_i | V_n^d) \\ \mathcal{L}_{\text{dec}}(n, d) &= -\frac{1}{|M_{\text{dec}}|} \sum_{i\in M_{\text{dec}}} \log P(x_i | \vec{x}_{\text{dec}}^{\text{cls}}, x_{\text{dec}}) \\ \mathcal{L}_{mlm}(n, d) &= \mathcal{L}_{\text{enc}}(n, d) + \mathcal{L}_{\text{dec}}(n, d) \end{aligned} \tag{8}$$

where M_{enc} and M_{dec} are masked token set for encoder and decoder respectively. Intuitively, the encoder needs to learn to compress useful information about the input text into its representations for the weaker decoder to recover the masked tokens.

Finally, similar to the SRL fine-tuning, we predefine a fixed list of sub-layer and sub-dimension pairs, summing the MLM loss for all pairs during pre-training:

$$\mathcal{L}_{\text{SMAE}} = \frac{1}{|S|} \sum_{(n,d)\in S} \mathcal{L}_{mlm}(n, d). \tag{9}$$

After the pre-training, the decoder is discarded and the encoder is used for the downstream tasks.

Starbucks pre-training objective covers various model sizes and dimensions, explicitly enhancing the text representations outputted by each model layer. As a result, it can potentially yield a backbone model that is more tolerant to the downstream Starbucks fine-tuning.

4 Experiment Setup

4.1 Starbucks Masked Autoencoding

We pre-train our SMAE model using the BERT base architecture and weights (Devlin et al., 2019), initialized from the model available on Hugging Face.[4] We then continue pre-training on the Fineweb dataset (Penedo et al., 2024), utilizing

[4] https://huggingface.co/google-bert/bert-base-uncased

its 10-billion-token subset.[5] The model is trained for one epoch with a batch size of 512, using the AdamW optimizer with a learning rate of 1e-4, a weight decay of 0.05, and cosine learning rate scheduling with a warmup ratio of 0.05. Our training was implemented using the Hugging Face Transformers library (Wolf et al., 2020). We open-source our pre-trained SMAE model checkpoint on Huggingface at `anonymized-for-peer-review`.

4.2 Evaluation

We evaluated our trained Starbucks models on two tasks: Semantic Text Similarity (STS) and passage retrieval. To assess the effectiveness of our proposed pre-training and fine-tuning methods, we fine-tuned 2D Matryoshka models using both our SRL approach and the existing 2DMSE baselines for both tasks. Additionally, we compared models fine-tuned separately from the original BERT checkpoint and our SMAE model checkpoint.

4.2.1 Semantic Text Similarity

For STS, we followed previous work by training an embedding model using combined training data from the Multi-Genre Natural Language Inference (MultiNLI) (Williams et al., 2018) and the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) corpora. We evaluate models' effectiveness using STS benchmarks, which comprise seven STS datasets from 2012 to 2016 (Agirre et al., 2016), SICK-Relatedness (SICK-R) (Marelli et al., 2014), and the STS Benchmark (STS-B) (Cer et al., 2017). We reported Spearman's rank correlation coefficient, following previous work, to measure the strength and direction of the association between the predicted and actual similarity scores. For consistency, we implemented using the sentence transformers library (Reimers and Gurevych, 2019), which already included 2DMSE baseline implementations, training data, evaluation and example tuning guide. We further modify the loss computation in the library for our SRL fine-tuning.

4.2.2 Passage Retrieval

For the passage retrieval task, we implemented our models and baselines using the Tevatron library (Gao et al., 2023). All models were trained on the MS MARCO Passage Ranking dataset (Bajaj et al., 2018). We evaluated the models on the MS MARCO development set using Mean Reciprocal Rank at 10 (MRR@10). Additionally, we assessed their performance on the TREC Deep Learning 2019 (DL19)(Craswell et al., 2020) and 2020 (DL20)(Craswell et al., 2021) datasets, which share the same corpus as MS MARCO, using normalized Discounted Cumulative Gain at 10 (nDCG@10) to account for the position of relevant documents in the rankings and graded relevance labels.

In Appendix A.4, we list the hyperparameters used to fine-tune our models and baselines for the STS and retrieval tasks.

5 Results

Table 1 presents the main comparison between Starbucks and all baseline approaches on STS and retrieval tasks, with results averaged across different Starbucks layer-dimension sizes $S = [(2, 32), (4, 64), (6, 128), (8, 256), (10, 512), (12, 768)]$. In the table, BERT refers to the average effectiveness of full-size BERT checkpoints fine-tuned on the corresponding task and evaluated on different Starbucks sizes. BERT-Separate refers to the average performance of truncated BERT checkpoints (one checkpoint for each of the Starbucks layer-dimension pairs), separately fine-tuned on the corresponding task. BERT-2DMSE refers to average results obtained with the 2D Matryoshka method. BERT-SRL refers to average results obtained with the SRL fine-tuning method. SMAE-SRL refers to the average results obtained using our full Starbucks method, consisting of the SMAE pre-training followed by SRL fine-tuning. Table 4 in the Appendix reports a breakdown of the results across individual Starbucks sizes, i.e. not averaged.

Overall, Starbucks outperforms most baseline methods, with the exception being the SICK-R dataset for STS and DL20 for retrieval. When only using the SRL component of Starbucks (BERT-SRL), i.e. when BERT is used as the checkpoint from which fine-tuning commences, SRL tuning demonstrates higher effectiveness on most datasets, including all datasets for retrieval and most for STS, exceptions occurring in STS13, STS14. Our main results suggest that while our proposed SRL tuning strategy already yields higher effectiveness than other Matryoshka-based methods, coupling it with SMAE pretraining (i.e. the full Starbucks stack) can further enhance this effectiveness gain.

In the following sections, we present the detailed result analysis of SRL and SMAE in comparison to

[5] https://huggingface.co/datasets/HuggingFaceFW/fineweb

Table 1: Starbucks (right) provides a single model with flexible configurations in terms of embedding dimensionality and network layer. Also shown is the original Matryoshka model (left) and extension 2D Matryoshka (center).

	Semantic Text Similarity Tasks								Retrieval Tasks		
Method	**STS-B**	**STS12**	**STS13**	**STS14**	**STS15**	**STS16**	**SICK-R**	**Average**	**MS MARCO**	**DL19**	**DL20**
BERT	0.6707	0.5525	0.6607	0.6036	0.7219	0.6902	0.6656	0.6522	0.0559	0.1084	0.1076
BERT-Separate	0.7872	0.7128	0.7742	0.7269	0.8075	0.7722	0.7702	0.7644	0.2917	0.5859	0.5857
BERT-2DMSE	0.7629	0.6691	0.7282	0.6793	0.7883	0.7501	0.7584	0.7338	0.2560	0.5218	0.5408
BERT-SRL	0.7906	0.7197	0.7742	0.7252	0.8126	0.7784	**0.7766**	0.7682	0.2921	0.6009	**0.6126**
SMAE-SRL (Starbucks)	**0.8073**	**0.7286**	**0.7868**	**0.7558**	**0.8362**	**0.7948**	0.7764	**0.7837**	**0.3116**	**0.6135**	0.6039

all baselines on the STS task (results for retrieval are reported in Appendix A.2), with different layer and dimension setups. We begin with comparing the use of embedding models trained separately vs. using the 2DMSE approach in Section 5.1. Subsequently, in Section 5.2, we discuss how SRL performs relative to all baseline approaches. Finally, we investigate how SMAE pre-training helps with embeddings in comparison with other pretrained models (Section 5.3).

5.1 2DMSE vs. Separate Trained Models

We next compare using embedding models trained separately to the 2DMSE approach for STS tasks, see Figure 3 (see retrieval results in Figure 6 in Appendix). Our findings demonstrate that separately trained models surpass 2DMSE across dimensions and layer configurations (the only exception being when layer=12 and dimension=32). Furthermore, we note that the number of layers has a larger impact than changes in the number of layers in terms of effectiveness degradation by 2DMSE, i.e. variations in colour intensity in the figure are larger across individual columns (changes in layer sizes) than across individual rows (changes in dimensions).

5.2 Starbucks Representation Learning (SLR)

To evaluate SRL, we first compare it with 2DMSE across various layers and dimensions, as depicted in Figure 4a. For this, we use the BERT-SRL results, i.e. we do not use SMAE pre-training. We name the diagonal in the figure composed by entries that match the Starbucks sizes, i.e. $(2, 32), (4, 64), \ldots$, as the Starbucks diagonal. Surprisingly, despite being trained solely on Starbucks sizes, SRL outperforms 2DMSE in most configurations. This includes scenarios involving layer and dimension sizes that were not explicitly targeted in the fine-tuning objectives (non-diagonal entries). As anticipated, the greatest effectiveness improvements are observed in configurations that align with

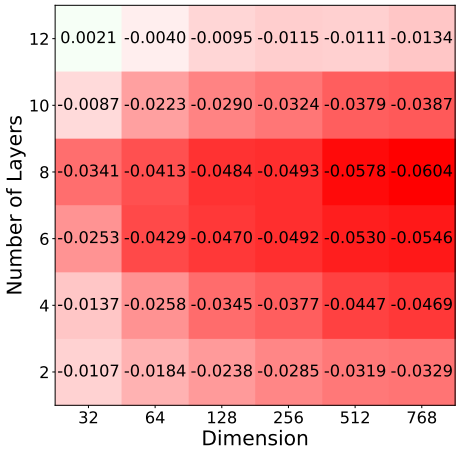
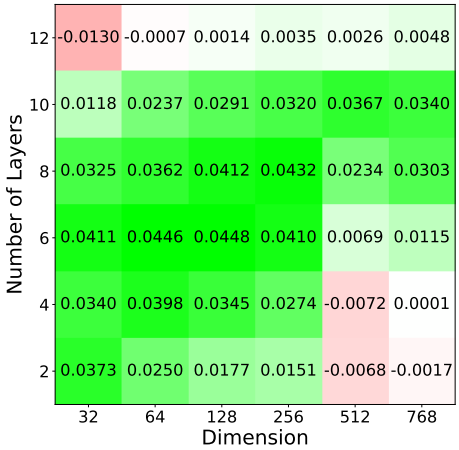


Figure 3: Average Spearman's Correlation deltas between 2DMSE and separately trained models (i.e. 2DMSE - BERT-Separate) on seven STS datasets for different layer and dimensionality sizes. Negative values (red) indicate the separate trained model was better.

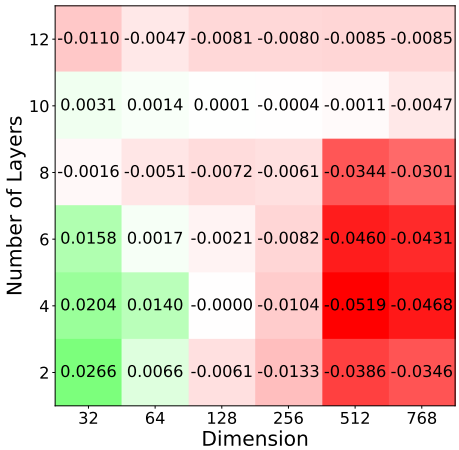
the Starbucks sizes. However, in instances significantly divergent from these objectives – such as at layer 12 with 32 dimensions, where instead the training was focused on dimension 768 (i.e. the further away from the Starbucks diagonal) – SRL tends to underperform compared to 2DMSE.

In comparison with separately trained models (Figure 4b), SRL exhibits marginal differences in most cases. Results obtained with settings that diverge more significantly from the Starbucks training objectives (off the Starbucks diagonal) generally perform worse than those within its targeted range. Nevertheless, in scenarios involving shallow layers and smaller dimensions, SRL often achieves better effectiveness than the separately trained models. It is important to note that SRL requires only a single training session and can adapt to a wide range of layer and dimension setups. In contrast, separately trained models require multiple training sessions (thus involving higher training costs), and are only effective in their specifically targeted layer and dimension configurations.

We also report the same comparisons on the re-



(a) SRL – 2DMSE



(b) SRL – Seperate trained

Figure 4: Average Spearman's Correlation deltas between SRL fine-tuning and (a) 2DMSE and (b) Separate trained models on seven STS datasets. Original BERT model is always used to start fine-tuning. Positive (green) values indicate where SRL was more effective.

trieval task in Appendix A.2.1. The conclusion still holds: Starbucks performs the best or is on par with separately trained models across all settings in the Starbucks diagonal.

5.3 Starbucks Masked Autoencoding (SMAE)

Finally, we evaluate the effectiveness of SMAE pre-training compared to the original BERT baseline. We employ two fine-tuning approaches: SRL and the fine-tuning of separate models. As illustrated in Figure 5 (Figure 8 for retrieval), SMAE pre-training consistently surpasses the original BERT checkpoint when SRL fine-tuning is applied, with the only exceptions being the full-size model, which shows a marginal decrease in effectiveness, and the 8-layer model with full dimensions. We also observe that the largest improvements are found in configurations involving shallow layers

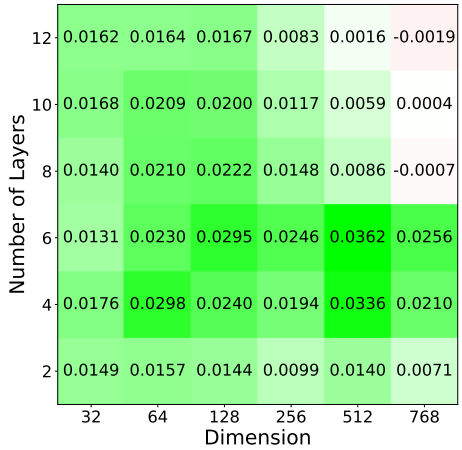


Figure 5: Average Spearman's Correlation deltas between SMAE and BERT base model when used for pre-training and SRL used for fine-tuning. Positive (green) values indicate where SMAE was more effective.

and smaller dimensions. Results obtained with separately trained models show similar trends (Figure 9). This outcome is intuitive, considering that the primary distinction between SMAE pre-training and the standard MLM pre-training of BERT lies in the focus on shallow layers and smaller dimensions for learning representations.

6 Impact of Starbucks Modelling Choices

In this section, we investigate variations in the modelling approach and their impact on the quality of the final embeddings. We detail two critical ablations conducted in our experiments: First we explore the role of KL divergence during fine-tuning and its influence on final model effectiveness (Section 6.1). Then we examine the effect of different pre-training strategies on the final effectiveness of the embedding model (Section 6.2). For the purpose of these ablation studies, we selected a single dataset for each task: STAB for the STS task, and MSMARCO-dev for the retrieval task.

6.1 Impact of KL Divergence

We investigate the influence of KL divergence within the SRL loss $\mathcal{L} = \mathcal{L}_S + \mathcal{L}_{\text{KL}}$. This loss is used to align the embeddings of the full-layer, full-dimension size with those of smaller layer-dimension sizes. The results indicate that incorporating $\mathcal{L}_{\text{KL}}$ during training benefits nearly all Starbucks sizes (Table 2). Exceptions include the Demi size for the STS task and the Trenta for retrieval. Overall, we observe that $\mathcal{L}_{\text{KL}}$ has a more pronounced impact on retrieval effectiveness, while gains for STS are generally marginal for most sizes.

Table 2: Impact of KL divergence loss $\mathcal{L}_{\text{KL}}$. STAB Spearman's Correlation is used for STS task, and MSMARCO dev MRR@10 is used for Retrieval.

		Demi (2, 32)	Short (4, 64)	Tall (6, 128)	Grande (8, 256)	Venti (10. 512)	Trenta (12, 768)	Average
STS	w. $\mathcal{L}_{\text{KL}}$	0.7455	**0.7933**	**0.8202**	**0.8278**	**0.8292**	**0.8274**	**0.8073**
	w/o. $\mathcal{L}_{\text{KL}}$	**0.7463**	0.7925	0.8190	0.8274	0.8283	0.8269	0.8067
Retrieval	w. $\mathcal{L}_{\text{KL}}$	**0.2282**	**0.2950**	**0.3274**	**0.3369**	**0.3416**	0.3403	**0.3116**
	w/o. $\mathcal{L}_{\text{KL}}$	0.2205	0.2885	0.3162	0.3295	0.3372	**0.3405**	0.3054

Table 3: Impact of pre-training components. STAB Spearman's Correlation is used for the STS task, and MSMARCO dev MRR@10 is used for the retrieval task.

		Demi (2, 32)	Short (4, 64)	Tall (6, 128)	Grande (8, 256)	Venti (10. 512)	Trenta (12, 768)	Average
STS	SMAE	0.7455	0.7933	0.8202	**0.8278**	**0.8292**	0.8274	**0.8073**
	- MAE	**0.7475**	**0.7968**	**0.8211**	0.8257	0.8288	0.8239	**0.8073**
	- Starbucks	0.7399	0.7675	0.7902	0.8134	0.8238	**0.8360**	0.7951
Retrieval	SMAE	0.2282	0.2950	**0.3274**	**0.3369**	**0.3416**	**0.3403**	**0.3116**
	- MAE	**0.2349**	**0.2972**	0.3237	0.3282	0.3339	0.335	0.3088
	- Starbucks	0.2255	0.2859	0.3098	0.3277	0.3375	**0.3403**	0.3044

6.2 Impact of Pre-training Approaches

To explore how various pre-training strategies impact effectiveness, we incrementally modified the pre-training components as detailed in Section 3.2. We conducted experiments with fixed SLR fine-tuning but two alternative further pretrained backbone models on the same dataset. The first model excludes the Masked Autoencoder (MAE) component yet retains the Starbucks training regime; the second model eliminates both the MAE and Starbucks pre-training, effectively reverting to the original BERT model pre-trained using its standard MLM objective.

Table 3 shows that Starbucks pre-training consistently enhances effectiveness across all Starbucks sizes, with the exception of Trenta (full). This is expected since the further pre-trained models only used the full-size model as their training objective. When assessing the impact of the MAE component in pre-training, results differ across Starbucks sizes and evaluation tasks. Specifically, smaller-sized models, such as Demi and Short, exhibit improved performance when pre-trained without the MAE. Conversely, for larger sizes – including Grande, Venti, and Trenta – models pre-trained with the MAE demonstrate higher effectiveness. Notably, the disparity in effectiveness is more pronounced in the retrieval task than in the STS; we posit this is due to the inherently greater challenge posed by retrieval compared to text similarity.

7 Conclusion

In this paper we highlight a key drawback of 2D Matryoshka Sentence Embedding models: While they provide elastic embedding generation because a single embedding model serves the diverse requirements in terms of embedding size and model layers, empirically we show their effective is significantly lower than if separate models were trained.

To address this, we introduced Starbucks, an enhancement over the previous 2D Matryoshka approach, designed to train embedding models of varying sizes simultaneously. Our empirical analysis across semantic text similarity and retrieval tasks demonstrates that Starbucks significantly outperforms the existing 2D Matryoshka and achieves comparable or even higher effectiveness than separately trained models.

Furthermore, Starbucks can be used to produce a BERT-based backbone, trained using Starbucks' Masked Autoencoding method, which serves as a robust checkpoint for the further training of more effective sub-models. This checkpoint proves advantageous regardless of whether Starbucks Representation Learning is applied or if the sub-models are then fine-tuned independently.

Our findings suggest that Starbucks not only preserves the adaptability of previous 2D Matryoshka models, but also improves model effectiveness making it a promising approach for future research in embedding model training.

Starbucks opens new avenues for investigating the automatic selection of optimal layer-dimension configurations. This could either enhance overall effectiveness—leveraging early layers that may provide better embeddings—or cater to budget-conscious scenarios, such as companies offering API search services, where a slight reduction in effectiveness could be exchanged for increased efficiency. The feasibility of this exploration is underscored by the now minimal gap between different layer-dimension performances.

8 Limitations

Our experiments are confined to the use of a BERT-based model architecture, including further pre-trained BERT checkpoints, as our principal embedding framework. Consequently, this excludes consideration of newer, larger, and potentially more effective decoder-based models, in particular methods such as RepLlama (Ma et al., 2024), PromptReps (Zhuang et al., 2024), etc. for the retrieval task. While these advanced models might offer better performance, training/fine-tuning on the same set of experiments we considered here takes much longer and requires much more computational resources. In this paper, we aimed to prove the viability and effectiveness of the Starbucks approach, and we left this extension on larger, more expensive backbones to future work.

The Starbucks sizes we considered included only six configurations (i.e., $(2, 32), (4, 64), \ldots$). We did not investigate other potential combinations due to the extensive variety of possible variations. Our results show that increasing dimension and layer numbers always led to improvements in effectiveness; however, we still do not know if there are specific combinations of layers and dimensions that would be more effective. Additionally, our study did not explore scenarios where a fixed dimension size is maintained while varying all layers, which we identify as an area for future experimentation.

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *Preprint*, arXiv:2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *Preprint*, arXiv:2003.07820.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. 2023. MatFormer: Nested transformer for elastic inference. *Preprint*, arXiv:2310.07707.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Tevatron: An efficient and flexible toolkit for neural retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pages 3120–3124, New York, NY, USA. Association for Computing Machinery.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2022. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 30233–30249. Curran Associates, Inc.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. 2024. Gecko: Versatile text embeddings distilled from large language models. *Preprint*, arXiv:2403.20327.

Xianming Li and Jing Li. 2024. AoE: Angle-optimized embeddings for semantic textual similarity. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1839, Bangkok, Thailand. Association for Computational Linguistics.

Xianming Li, Zongxi Li, Jing Li, Haoran Xie, and Qing Li. 2024. ESE: Espresso sentence embeddings. *Preprint*, arXiv:2402.14776.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pages 2421–2425, New York, NY, USA. Association for Computing Machinery.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Preprint*, arXiv:2406.17557.

Aniket Rege, Aditya Kusupati, Sharan Ranjit S, Alan Fan, Qingqing Cao, Sham Kakade, Prateek Jain, and Ali Farhadi. 2023. AdANNS: A framework for adaptive semantic search. In *Advances in Neural Information Processing Systems*, volume 36, pages 76311–76335. Curran Associates, Inc.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. SimLM: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2244–2258, Toronto, Canada. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Trans. Inf. Syst.*, 42(4).

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2024. PromptReps: Prompting large language models to generate dense and sparse representations for zero-shot document retrieval. *Preprint*, arXiv:2404.18424.

Shengyao Zhuang, Linjun Shou, Jian Pei, Ming Gong, Houxing Ren, Guido Zuccon, and Daxin Jiang. 2023. Typos-aware bottlenecked pre-training for robust dense retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP '23, pages 212–222, New York, NY, USA. Association for Computing Machinery.

A Appendix

A.1 Full Results across Starbucks Sizes

In Table 4 we report the full results of methods across various Starbucks sizes. We report whether statistical significance differences ($p < 0.05$) for each dataset were found between each of the Starbucks setting and the corresponding baselines. For statistical analysis, we used a two-tail paired Student's t-test for the retrieval task; for the STS task, we transformed the correlation coefficient values into z scores (Fisher's r to z transformation), and then use the z test statistic.

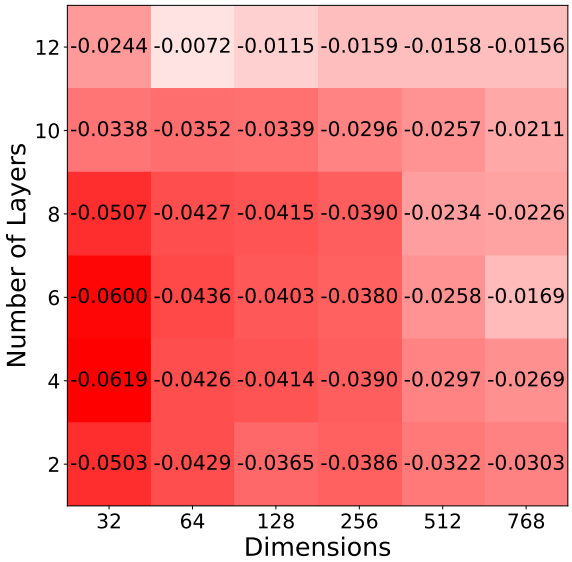


Figure 6: Average MRR@10 between 2DMSE and separately trained models (i.e. 2DMSE - BERT-Separate) on MS MARCO for different layer and dimensionality sizes. Negative values (red) indicate the separate trained model was better.

As the results demonstrate, Starbucks is the most effective method across most of the smaller sizes. For the full size setting, the use of separately trained models (BERT Separate) often performs best on STS tasks; nevertheless, Starbucks is still generally the best Matryoshka-like training method in this setting.

A.2 Retrieval Results

A.2.1 2DMSE vs. Separate Trained Models

Figure 6 shows the difference between 2DMSE and BERT models, each trained separately on the MS MARCO passage retrieval task. Similarly to what observed on the STS tasks, 2DMSE performs worse than the BERT models across all investigated sizes.

A.2.2 Starbucks Representation Learning (SRL)

Figure 7 shows the results of our SRL fine-tuning on top of the BERT base model on the MS MARCO passage retrieval task. We observe the best results across the Starbucks diagonal, compared to separately trained models. However, Starbucks is less effective on off-diagonal settings of embedding dimensions and layers: this is to be expected since our training objective focuses on settings in the Starbucks diagonal.

Table 4: Complete results across all Starbucks sizes. Spearman's Correlation is used to evaluate the STS task. For retrieval task, MRR@10 is used to evaluate on MS MARCO dev set, while nDCG@10 is used to evaluate on DL19 and DL20. * denotes statistical significance differences (p<0.05) between the corresponding method and SMAE-SRL (Starbucks).

		Semantic Text Similarity Tasks								Retrieval Tasks		
	Method	**STS-B**	**STS12**	**STS13**	**STS14**	**STS15**	**STS16**	**SICK-R**	**Average**	**MS MARCO**	**DL19**	**DL20**
Demi (n=2, d=32)	BERT	0.5919*	0.4700*	0.6052*	0.5424*	0.6530*	0.6397*	0.5996*	0.5860	0.0001*	0.0000*	0.0000*
	BERT-Separate	0.7134	0.6637	0.6770*	0.6374*	0.7451*	0.7059*	0.7210*	0.6948	0.2092*	0.4599*	0.4826
	BERT-2DMSE	0.7197	0.6204*	0.6747*	0.6278*	0.7336*	0.7058*	0.7066*	0.6841	0.1589*	0.3459*	0.3828*
	BERT-SRL	0.7413	0.6789	0.7190	0.6684*	0.7655*	0.7309	0.7455	0.7214	0.2147*	0.5003	**0.5294**
	SMAE-SRL (Starbucks)	**0.7455**	**0.6854**	**0.7368**	**0.6955**	**0.7972**	**0.7459**	**0.7473**	**0.7362**	**0.2282**	**0.5122**	0.5042
Short (n=4, d=64)	BERT	0.6021*	0.4478*	0.5881*	0.5230*	0.6599*	0.6455*	0.6083*	0.5821	0.0000*	0.0000*	0.0000*
	BERT-Separate	0.7399*	0.6882*	0.7387*	0.6800*	0.7819*	0.7419*	0.7579*	0.7326	0.2692*	0.5527	0.5487*
	BERT-2DMSE	0.7311*	0.6364*	0.6977*	0.6551*	0.7637*	0.7312*	0.7323*	0.7068	0.2266*	0.4988*	0.5179*
	BERT-SRL	0.7621*	0.7126	0.7400*	0.6900*	0.7924*	0.7613	0.7678	0.7466	0.2722*	0.5653	0.5853
	SMAE-SRL (Starbucks)	**0.7933**	**0.7296**	**0.7824**	**0.7444**	**0.8308**	**0.7840**	**0.7700**	**0.7764**	**0.2950**	**0.5766**	**0.5920**
Tall (n=6, d=128)	BERT	0.6128*	0.4689*	0.6037*	0.5464*	0.6728*	0.6624*	0.6271*	0.5991	0.0000*	0.0000*	0.0000*
	BERT-Separate	0.7853*	0.7150	0.7794	0.7261*	0.7959*	0.7778	0.7802	0.7657	0.2935*	0.5939	0.6028
	BERT-2DMSE	0.7501*	0.6434*	0.7061*	0.6630*	0.7739*	0.7421*	0.7522*	0.7187	0.2532*	0.5464*	0.5503*
	BERT-SRL	0.7838*	0.7158	0.7685*	0.7146*	0.8066*	0.7762	0.7792	0.7635	0.2983*	0.6102	0.6247
	SMAE-SRL (Starbucks)	**0.8202**	**0.7374**	**0.7998**	**0.7654**	**0.8426**	**0.8029**	**0.7831**	**0.7931**	**0.3274**	**0.6346**	**0.6319**
Grande (n=8, d=256)	BERT	0.6433*	0.5121*	0.6370*	0.5712*	0.7062*	0.6737*	0.6504*	0.6277	0.0009*	0.0000*	0.0000*
	BERT-Separate	0.8174	0.7304	**0.8097**	0.7588	0.8289*	0.7998	0.7807	0.7894	0.3160*	0.6185	0.6259
	BERT-2DMSE	0.7721*	0.6635*	0.7400*	0.6756*	0.7949*	0.7570*	0.7774	0.7401	0.2770*	0.5604*	0.5856*
	BERT-SRL	0.8112	0.7310	0.7925	0.7397*	0.8246*	0.7959	**0.7879**	0.7833	0.3159*	0.6342	**0.6344**
	SMAE-SRL (Starbucks)	**0.8278**	**0.7382**	0.8033	**0.7736**	**0.8469**	**0.8092**	0.7874	**0.7981**	**0.3369**	**0.6525**	0.6292
Venti (n=10, d=512)	BERT	0.7331*	0.6748*	0.6983*	0.6488*	0.7876*	0.7047*	0.7133*	0.7087	0.0000*	0.0100*	0.0047*
	BERT-Separate	0.8259	0.7379	**0.8088**	0.7691	0.8412	0.7931	0.7866	0.7946	0.3279*	**0.6502**	0.6133
	BERT-2DMSE	0.7850*	0.7097*	0.7422*	0.6927*	0.8218*	0.7626*	0.7832	0.7568	0.3022*	0.5769*	0.6082
	BERT-SRL	0.8196	0.7346	0.8070	0.7635	0.8391	0.8003	**0.7903**	0.7935*	0.3249*	0.6405	**0.6499**
	SMAE-SRL (Starbucks)	**0.8292**	**0.7405**	0.7989	**0.7772**	**0.8503**	**0.8138**	0.7860	**0.7994**	**0.3416**	0.6493	0.6310
Trenta (n=12, d=768)	BERT	**0.8411**	0.7413	**0.8319***	**0.7902**	**0.8521**	**0.8151**	0.7949*	**0.8095**	0.3341	0.6404	0.6410
	BERT-Separate	**0.8411**	0.7413	**0.8319***	**0.7902**	**0.8521**	**0.8151**	0.7949*	**0.8095**	0.3341	0.6404	0.6410
	BERT-2DMSE	0.8196	0.7414	0.8087	0.7616	0.8417	0.8017	**0.7986***	0.7962	0.3185*	0.6021	0.6001*
	BERT-SRL	0.8258	**0.7456**	0.8185	0.7752	0.8474	0.8059	0.7889	0.8010	0.3266*	0.6550	**0.6518**
	SMAE-SRL (Starbucks)	0.8274	0.7404	0.7998	0.7789	0.8497	0.8131	0.7844	0.7991	**0.3403**	**0.6558**	0.6358

A.2.3 Starbucks Masked Autoencoding (SMAE)

Figure 8 shows the comparison between SRL trained with our SMAE and the BERT base model on the MS MARCO passage retrieval task. SRL with SMAE achieves a higher MRR@10 across most sizes, similar to what observed across STS tasks. This suggests that SMAE is a better backbone model for SRL training.

A.3 Separate Trained Models with SMAE

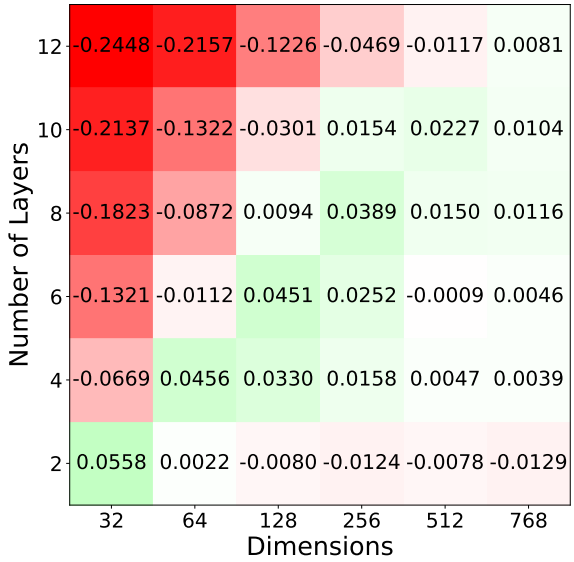
Figure 9 shows the comparison of separately trained models across different sizes using either SMAE or BERT on STS tasks. Although SMAE pre-training specifically targets settings in the Starbucks diagonal, it also benefits other combinations of dimensions and layers.

A.4 Fine-tuning Hyperparameters

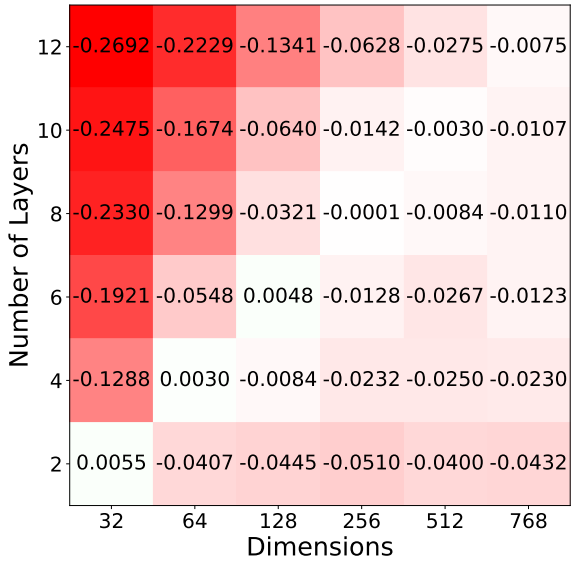
We report the fine-tuning hyperparameters used for the STS tasks and retrieval tasks in Tables 5 and 6, respectively.

Table 5: Hyperparameters: fine-tuning for the STS task.

Hyperparameter	**Assignment**
learning Rate	5e-5
lr scheduler type	linear
warmup ratio	0.1
overall batch size	128
optimizer	AdamW
epochs	1
fp16	True
kl_temperature	0.3
distance metric	cosine similarity
Inner loss function	MultipleNegativesRankingLoss
GPU	1 x H100 80GB



(a) SRL – 2DMSE



(b) SRL – Seperate trained

Figure 7: Average MRR@10 deltas between SRL fine-tuning and (a) 2DMSE and (b) Separate trained models on seven MS MARCO datasets. Original BERT model is always used to start fine-tuning. Positive (green) values indicate where SRL was more effective.

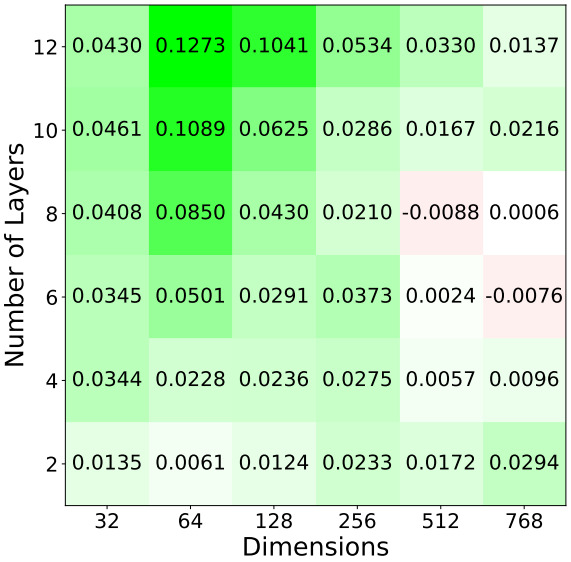


Figure 8: Average MRR@10 deltas between SMAE and BERT base model when used for pretraining and SRL used for fine-tuning. Positive (green) values indicate where SMAE was more effective.

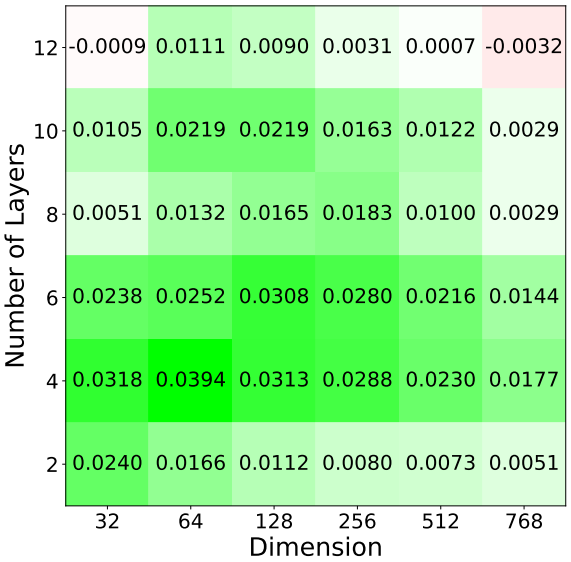


Figure 9: Average Spearman's Correlation deltas between separately trained models with SMAE and BERT (i.e. SMAE-Separate - BERT-Separate) on seven STS datasets for different layer and dimensionality sizes. Negative values (red) indicate the separate trained model was better.

Table 6: Hyperparameters: fine-tuning for the retrieval task.

Hyperparameter	**Assignment**
learning Rate	1e-4
lr scheduler type	linear
warmup ratio	0.1
overall batch size	128
optimizer	AdamW
epochs	3
fp16	True
kl_temperature	1.0
Inner loss function	InfoNCE
distance metric	dot product
# hard negative	7
in-batch negatives	True
GPU	1 x H100 80GB