[![Logo](../../../_static/logo.png)](../../../index.html)

Getting Started

* [Installation](../../../docs/installation.html)
  * [Install with pip](../../../docs/installation.html#install-with-pip)
  * [Install with Conda](../../../docs/installation.html#install-with-conda)
  * [Install from Source](../../../docs/installation.html#install-from-source)
  * [Editable Install](../../../docs/installation.html#editable-install)
  * [Install PyTorch with CUDA support](../../../docs/installation.html#install-pytorch-with-cuda-support)
* [Quickstart](../../../docs/quickstart.html)
  * [Sentence Transformer](../../../docs/quickstart.html#sentence-transformer)
  * [Cross Encoder](../../../docs/quickstart.html#cross-encoder)
  * [Next Steps](../../../docs/quickstart.html#next-steps)

Sentence Transformer

* [Usage](../../../docs/sentence_transformer/usage/usage.html)
  * [Computing Embeddings](../../applications/computing-embeddings/README.html)
  * [Initializing a Sentence Transformer Model](../../applications/computing-embeddings/README.html#initializing-a-sentence-transformer-model)
  * [Calculating Embeddings](../../applications/computing-embeddings/README.html#calculating-embeddings)
    * [Prompt Templates](../../applications/computing-embeddings/README.html#prompt-templates)
    * [Input Sequence Length](../../applications/computing-embeddings/README.html#id1)

* [Multi-Process / Multi-GPU Encoding](../../applications/computing-embeddings/README.html#multi-process-multi-gpu-encoding)
* [Semantic Textual Similarity](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html)
* [Similarity Calculation](../../../docs/sentence_transformer/usage/semantic_textual_similarity.html#similarity-calculation)
* [Semantic Search](../../applications/semantic-search/README.html)
* [Background](../../applications/semantic-search/README.html#background)
* [Symmetric vs. Asymmetric Semantic Search](../../applications/semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
* [Manual Implementation](../../applications/semantic-search/README.html#manual-implementation)
* [Optimized Implementation](../../applications/semantic-search/README.html#optimized-implementation)
* [Speed Optimization](../../applications/semantic-search/README.html#speed-optimization)
* [Elasticsearch](../../applications/semantic-search/README.html#elasticsearch)
* [Approximate Nearest Neighbor](../../applications/semantic-search/README.html#approximate-nearest-neighbor)
* [Retrieve & Re-Rank](../../applications/semantic-search/README.html#retrieve-re-rank)
* [Examples](../../applications/semantic-search/README.html#examples)
* [Retrieve & Re-Rank](../../applications/retrieve_rerank/README.html)
* [Retrieve & Re-Rank Pipeline](../../applications/retrieve_rerank/README.html#retrieve-re-rank-pipeline)
* [Retrieval: Bi-Encoder](../../applications/retrieve_rerank/README.html#retrieval-bi-encoder)

* [Re-Ranker: Cross-Encoder](../../applications/retrieve_rerank/README.html#re-ranker-cross-encoder)
  * [Example Scripts](../../applications/retrieve_rerank/README.html#example-scripts)
* [Pre-trained Bi-Encoders (Retrieval)](../../applications/retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
* [Pre-trained Cross-Encoders (Re-Ranker)](../../applications/retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)
* [Clustering](../../applications/clustering/README.html)
  * [k-Means](../../applications/clustering/README.html#k-means)
  * [Agglomerative Clustering](../../applications/clustering/README.html#agglomerative-clustering)
  * [Fast Clustering](../../applications/clustering/README.html#fast-clustering)
  * [Topic Modeling](../../applications/clustering/README.html#topic-modeling)
* [Paraphrase Mining](../../applications/paraphrase-mining/README.html)
* [`paraphrase_mining()`](../../applications/paraphrase-mining/README.html#sentence_transformers.util.paraphrase_mining)
* [Translated Sentence Mining](../../applications/parallel-sentence-mining/README.html)
* [Margin Based Mining](../../applications/parallel-sentence-mining/README.html#margin-based-mining)
  * [Examples](../../applications/parallel-sentence-mining/README.html#examples)
* [Image Search](../../applications/image-search/README.html)
  * [Installation](../../applications/image-search/README.html#installation)
  * [Usage](../../applications/image-search/README.html#usage)
  * [Examples](../../applications/image-search/README.html#examples)
* [Embedding Quantization](../../applications/embedding-quantization/README.html)
* [Binary Quantization](../../applications/embedding-quantization/README.html#binary-quantization)

* [Scalar (int8) Quantization](../../applications/embedding-quantization/README.html#scalar-int8-quantization)
* [Additional extensions](../../applications/embedding-quantization/README.html#additional-extensions)
* [Demo](../../applications/embedding-quantization/README.html#demo)
* [Try it yourself](../../applications/embedding-quantization/README.html#try-it-yourself)
* [Speeding up Inference](../../../docs/sentence_transformer/usage/efficiency.html)
* [PyTorch](../../../docs/sentence_transformer/usage/efficiency.html#pytorch)
* [ONNX](../../../docs/sentence_transformer/usage/efficiency.html#onnx)
* [OpenVINO](../../../docs/sentence_transformer/usage/efficiency.html#openvino)
* [Benchmarks](../../../docs/sentence_transformer/usage/efficiency.html#benchmarks)
* [Creating Custom Models](../../../docs/sentence_transformer/usage/custom_models.html)
* [Structure of Sentence Transformer Models](../../../docs/sentence_transformer/usage/custom_models.html#structure-of-sentence-transformer-models)
* [Sentence Transformer Model from a Transformers Model](../../../docs/sentence_transformer/usage/custom_models.html#sentence-transformer-model-from-a-transformers-model)
* [Pretrained Models](../../../docs/sentence_transformer/pretrained_models.html)
* [Original Models](../../../docs/sentence_transformer/pretrained_models.html#original-models)
* [Semantic Search Models](../../../docs/sentence_transformer/pretrained_models.html#semantic-search-models)
* [Multi-QA Models](../../../docs/sentence_transformer/pretrained_models.html#multi-qa-models)
* [MSMARCO Passage Models](../../../docs/sentence_transformer/pretrained_models.html#msmarco-passage-models)
* [Multilingual Models](../../../docs/sentence_transformer/pretrained_models.html#multilingual-models)

* [Dataset Overview](../../../docs/sentence_transformer/dataset_overview.html)
* [Datasets on the Hugging Face Hub](../../../docs/sentence_transformer/dataset_overview.html#datasets-on-the-hugging-face-hub)
* [Pre-existing Datasets](../../../docs/sentence_transformer/dataset_overview.html#pre-existing-datasets)
* [Loss Overview](../../../docs/sentence_transformer/loss_overview.html)
* [Loss modifiers](../../../docs/sentence_transformer/loss_overview.html#loss-modifiers)
* [Distillation](../../../docs/sentence_transformer/loss_overview.html#distillation)
* [Commonly used Loss Functions](../../../docs/sentence_transformer/loss_overview.html#commonly-used-loss-functions)
* [Custom Loss Functions](../../../docs/sentence_transformer/loss_overview.html#custom-loss-functions)
* [Training Examples](../../../docs/sentence_transformer/training/examples.html)
* [Semantic Textual Similarity](../sts/README.html)
* [Training data](../sts/README.html#training-data)
* [Loss Function](../sts/README.html#loss-function)
* [Natural Language Inference](../nli/README.html)
* [Data](../nli/README.html#data)
* [SoftmaxLoss](../nli/README.html#softmaxloss)
* [MultipleNegativesRankingLoss](../nli/README.html#multiplenegativesrankingloss)
* [Paraphrase Data](../paraphrases/README.html)
* [Pre-Trained Models](../paraphrases/README.html#pre-trained-models)
* [Quora Duplicate Questions](../quora_duplicate_questions/README.html)
* [Training](../quora_duplicate_questions/README.html#training)
* [MultipleNegativesRankingLoss](../quora_duplicate_questions/README.html#multiplenegativesrankingloss)

  * [Pretrained Models](../quora_duplicate_questions/README.html#pretrained-models)
* [MS MARCO](../ms_marco/README.html)
  * [Bi-Encoder](../ms_marco/README.html#bi-encoder)
* Matryoshka Embeddings
  * Use Cases
  * Results
  * Training
  * Inference
  * Code Examples
* [Adaptive Layers](../adaptive_layer/README.html)
  * [Use Cases](../adaptive_layer/README.html#use-cases)
  * [Results](../adaptive_layer/README.html#results)
  * [Training](../adaptive_layer/README.html#training)
  * [Inference](../adaptive_layer/README.html#inference)
  * [Code Examples](../adaptive_layer/README.html#code-examples)
* [Multilingual Models](../multilingual/README.html)
  * [Extend your own models](../multilingual/README.html#extend-your-own-models)
  * [Training](../multilingual/README.html#training)
  * [Datasets](../multilingual/README.html#datasets)
  * [Sources for Training Data](../multilingual/README.html#sources-for-training-data)
  * [Evaluation](../multilingual/README.html#evaluation)
  * [Available Pre-trained Models](../multilingual/README.html#available-pre-trained-models)
  * [Usage](../multilingual/README.html#usage)
  * [Performance](../multilingual/README.html#performance)
  * [Citation](../multilingual/README.html#citation)
* [Model Distillation](../distillation/README.html)
  * [Knowledge Distillation](../distillation/README.html#knowledge-distillation)

    * [Speed - Performance Trade-Off](../distillation/README.html#speed-performance-trade-off)
    * [Dimensionality Reduction](../distillation/README.html#dimensionality-reduction)
    * [Quantization](../distillation/README.html#quantization)
  * [Augmented SBERT](../data_augmentation/README.html)
    * [Motivation](../data_augmentation/README.html#motivation)
    * [Extend to your own datasets](../data_augmentation/README.html#extend-to-your-own-datasets)
    * [Methodology](../data_augmentation/README.html#methodology)
      * [Scenario 1: Limited or small annotated datasets (few labeled sentence-pairs)](../data_augmentation/README.html#scenario-1-limited-or-small-annotated-datasets-few-labeled-sentence-pairs)
      * [Scenario 2: No annotated datasets (Only unlabeled sentence-pairs)](../data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)
    * [Training](../data_augmentation/README.html#training)
    * [Citation](../data_augmentation/README.html#citation)
  * [Training with Prompts](../prompts/README.html)
    * [What are Prompts?](../prompts/README.html#what-are-prompts)
    * [Why would we train with Prompts?](../prompts/README.html#why-would-we-train-with-prompts)
    * [How do we train with Prompts?](../prompts/README.html#how-do-we-train-with-prompts)
  * [Training with PEFT Adapters](../peft/README.html)
    * [Compatibility Methods](../peft/README.html#compatibility-methods)
    * [Adding a New Adapter](../peft/README.html#adding-a-new-adapter)
    * [Loading a Pretrained Adapter](../peft/README.html#loading-a-pretrained-adapter)
    * [Training Script](../peft/README.html#training-script)
  * [Unsupervised Learning](../../unsupervised_learning/README.html)

Cross Encoder

Package Reference

* [Sentence Transformer](../../../docs/package_reference/sentence_transformer/index.html)
* [SentenceTransformer](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html)
* [SentenceTransformer](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#id1)
* [SentenceTransformerModelCardData](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#sentencetransformermodelcarddata)
* [SimilarityFunction](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#similarityfunction)
* [Trainer](../../../docs/package_reference/sentence_transformer/trainer.html)
* [SentenceTransformerTrainer](../../../docs/package_reference/sentence_transformer/trainer.html#sentencetransformertrainer)
* [Training Arguments](../../../docs/package_reference/sentence_transformer/training_args.html)
* [SentenceTransformerTrainingArguments](../../../docs/package_reference/sentence_transformer/training_args.html#sentencetransformertrainingarguments)
* [Losses](../../../docs/package_reference/sentence_transformer/losses.html)
* [BatchAllTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchalltripletloss)

*
[BatchHardSoftMarginTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchhardsoftmargintripletloss)
*
[BatchHardTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchhardtripletloss)
*
[BatchSemiHardTripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#batchsemihardtripletloss)
*
[ContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#contrastiveloss)
*
[OnlineContrastiveLoss](../../../docs/package_reference/sentence_transformer/losses.html#onlinecontrastiveloss)
*
[ContrastiveTensionLoss](../../../docs/package_reference/sentence_transformer/losses.html#contrastivetensionloss)
*
[ContrastiveTensionLossInBatchNegatives](../../../docs/package_reference/sentence_transformer/losses.html#contrastivetensionlossinbatchnegatives)
    * [CoSENTLoss](../../../docs/package_reference/sentence_transformer/losses.html#cosentloss)
    * [AnglELoss](../../../docs/package_reference/sentence_transformer/losses.html#angleloss)
*
[CosineSimilarityLoss](../../../docs/package_reference/sentence_transformer/losses.html#cosinesimilarityloss)
*
[DenoisingAutoEncoderLoss](../../../docs/package_reference/sentence_transformer/losses.html#den

oisingautoencoderloss)

*

[GISTEmbedLoss](../../../docs/package_reference/sentence_transformer/losses.html#gistembedloss

)

*

[CachedGISTEmbedLoss](../../../docs/package_reference/sentence_transformer/losses.html#cache

dgistembedloss)

* [MSELoss](../../../docs/package_reference/sentence_transformer/losses.html#mseloss)

*

[MarginMSELoss](../../../docs/package_reference/sentence_transformer/losses.html#marginmseloss

)

*

[MatryoshkaLoss](../../../docs/package_reference/sentence_transformer/losses.html#matryoshkaloss

)

*

[Matryoshka2dLoss](../../../docs/package_reference/sentence_transformer/losses.html#matryoshka2

dloss)

*

[AdaptiveLayerLoss](../../../docs/package_reference/sentence_transformer/losses.html#adaptivelaye

rloss)

*

[MegaBatchMarginLoss](../../../docs/package_reference/sentence_transformer/losses.html#megabat

chmarginloss)

*

[MultipleNegativesRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#

multiplenegativesrankingloss)

*

[CachedMultipleNegativesRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedmultiplenegativesrankingloss)

*

[MultipleNegativesSymmetricRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#multiplenegativessymmetricrankingloss)

*

[CachedMultipleNegativesSymmetricRankingLoss](../../../docs/package_reference/sentence_transformer/losses.html#cachedmultiplenegativessymmetricrankingloss)

  * [SoftmaxLoss](../../../docs/package_reference/sentence_transformer/losses.html#softmaxloss)
  * [TripletLoss](../../../docs/package_reference/sentence_transformer/losses.html#tripletloss)

* [Samplers](../../../docs/package_reference/sentence_transformer/sampler.html)

*

[BatchSamplers](../../../docs/package_reference/sentence_transformer/sampler.html#batchsamplers)

*

[MultiDatasetBatchSamplers](../../../docs/package_reference/sentence_transformer/sampler.html#multidatasetbatchsamplers)

* [Evaluation](../../../docs/package_reference/sentence_transformer/evaluation.html)

*

[BinaryClassificationEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#binaryclassificationevaluator)

*

[EmbeddingSimilarityEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#embeddingsimilarityevaluator)

*

[InformationRetrievalEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#informationretrievalevaluator)

*
[NanoBEIREvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#nanobeirevaluator)
*
[MSEEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#mseevaluator)
*
[ParaphraseMiningEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#paraphraseminingevaluator)
*
[RerankingEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#rerankingevaluator)
*
[SentenceEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#sentenceevaluator)
*
[SequentialEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#sequentialevaluator)
*
[TranslationEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#translationevaluator)
*
[TripletEvaluator](../../../docs/package_reference/sentence_transformer/evaluation.html#tripletevaluator)
    * [Datasets](../../../docs/package_reference/sentence_transformer/datasets.html)
*
[ParallelSentencesDataset](../../../docs/package_reference/sentence_transformer/datasets.html#par

allelsentencesdataset)

*

[SentenceLabelDataset](../../../docs/package_reference/sentence_transformer/datasets.html#sentencelabeldataset)

*

[DenoisingAutoEncoderDataset](../../../docs/package_reference/sentence_transformer/datasets.html#denoisingautoencoderdataset)

*

[NoDuplicatesDataLoader](../../../docs/package_reference/sentence_transformer/datasets.html#noduplicatesdataloader)

* [Models](../../../docs/package_reference/sentence_transformer/models.html)

* [Main Classes](../../../docs/package_reference/sentence_transformer/models.html#main-classes)

* [Further Classes](../../../docs/package_reference/sentence_transformer/models.html#further-classes)

* [quantization](../../../docs/package_reference/sentence_transformer/quantization.html)

*

[`quantize_embeddings()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.quantize_embeddings)

*

[`semantic_search_faiss()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_faiss)

*

[`semantic_search_usearch()`](../../../docs/package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_usearch)

* [Cross Encoder](../../../docs/package_reference/cross_encoder/index.html)

* [CrossEncoder](../../../docs/package_reference/cross_encoder/cross_encoder.html)

* [CrossEncoder](../../../docs/package_reference/cross_encoder/cross_encoder.html#id1)
* [Training Inputs](../../../docs/package_reference/cross_encoder/cross_encoder.html#training-inputs)
* [Evaluation](../../../docs/package_reference/cross_encoder/evaluation.html)
* [CEBinaryAccuracyEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cebinaryaccuracyevaluator)
* [CEBinaryClassificationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cebinaryclassificationevaluator)
* [CECorrelationEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cecorrelationevaluator)
* [CEF1Evaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cef1evaluator)
* [CESoftmaxAccuracyEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cesoftmaxaccuracyevaluator)
* [CERerankingEvaluator](../../../docs/package_reference/cross_encoder/evaluation.html#cererankingevaluator)
* [util](../../../docs/package_reference/util.html)
* [Helper Functions](../../../docs/package_reference/util.html#module-sentence_transformers.util)
* [`community_detection()`](../../../docs/package_reference/util.html#sentence_transformers.util.community_detection)
* [`http_get()`](../../../docs/package_reference/util.html#sentence_transformers.util.http_get)
*

[`is_training_available()`](../../../docs/package_reference/util.html#sentence_transformers.util.is_training_available)

*

[`mine_hard_negatives()`](../../../docs/package_reference/util.html#sentence_transformers.util.mine_hard_negatives)

*

[`normalize_embeddings()`](../../../docs/package_reference/util.html#sentence_transformers.util.normalize_embeddings)

*

[`paraphrase_mining()`](../../../docs/package_reference/util.html#sentence_transformers.util.paraphrase_mining)

*

[`semantic_search()`](../../../docs/package_reference/util.html#sentence_transformers.util.semantic_search)

*

[`truncate_embeddings()`](../../../docs/package_reference/util.html#sentence_transformers.util.truncate_embeddings)

* [Model Optimization](../../../docs/package_reference/util.html#module-sentence_transformers.backend)

*

[`export_dynamic_quantized_onnx_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model)

*

[`export_optimized_onnx_model()`](../../../docs/package_reference/util.html#sentence_transformers.backend.export_optimized_onnx_model)

*

[`export_static_quantized_openvino_model()`](../../../docs/package_reference/util.html#sentence_tra

nsformers.backend.export_static_quantized_openvino_model)

* [Similarity Metrics](../../../docs/package_reference/util.html#module-sentence_transformers.util)
  * [`cos_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.cos_sim)
  * [`dot_score()`](../../../docs/package_reference/util.html#sentence_transformers.util.dot_score)
  * [`euclidean_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.euclidean_sim)
  * [`manhattan_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.manhattan_sim)
  * [`pairwise_cos_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_cos_sim)
  * [`pairwise_dot_score()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_dot_score)
  * [`pairwise_euclidean_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_euclidean_sim)
  * [`pairwise_manhattan_sim()`](../../../docs/package_reference/util.html#sentence_transformers.util.pairwise_manhattan_sim)

__[Sentence Transformers](../../../index.html)

* [](../../../index.html)
* [Training Examples](../../../docs/sentence_transformer/training/examples.html)

* Matryoshka Embeddings

* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/matryoshka/README.md)

* * *

Matryoshka Embeddingsïƒ•

Dense embedding models typically produce embeddings with a fixed size, such as 768 or 1024. All further computations (clustering, classification, semantic search, retrieval, reranking, etc.) must then be done on these full embeddings. [Matryoshka Representation Learning](https://arxiv.org/abs/2205.13147) revisits this idea, and proposes a solution to train embedding models whose embeddings are still useful after truncation to much smaller sizes. This allows for considerably faster (bulk) processing.

Use Casesïƒ•

A particularly interesting use case is to split up processing into two steps: 1) pre-processing with much smaller vectors and then 2) processing the remaining vectors as full size (also called â€œshortlisting and rerankingâ€•). Additionally, Matryoshka models will allow you to scale your embedding solutions to your desired storage cost, processing speed and performance.

Resultsïƒ•

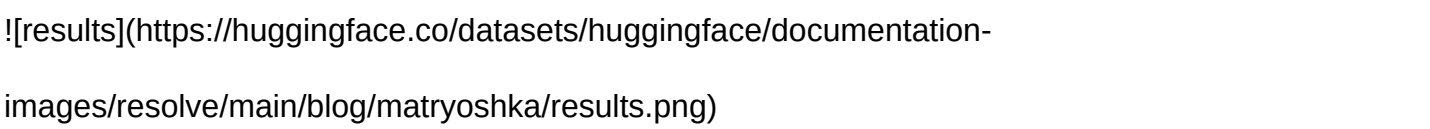
Letâ€™s look at the actual performance that we may be able to expect from a Matryoshka embedding model versus a regular embedding model. For this experiment, I have trained two models:

* [tomaarsen/mpnet-base-nli-matryoshka](https://huggingface.co/tomaarsen/mpnet-base-nli-matryoshka): Trained by running [matryoshka_nli.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/matryoshka_nli.py) with [microsoft/mpnet-base](https://huggingface.co/microsoft/mpnet-base).

* [tomaarsen/mpnet-base-nli](https://huggingface.co/tomaarsen/mpnet-base-nli): Trained by running a modified version of [matryoshka_nli.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/matryoshka_nli.py) where the training loss is only `MultipleNegativesRankingLoss` rather than `MatryoshkaLoss` on top of `MultipleNegativesRankingLoss`. I also use [microsoft/mpnet-base](https://huggingface.co/microsoft/mpnet-base) as the base model.

Both of these models were trained on the AllNLI dataset, which is a concatenation of the [SNLI](https://huggingface.co/datasets/snli) and [MultiNLI](https://huggingface.co/datasets/multi_nli) datasets. I have evaluated these models on the [STSBenchmark](https://huggingface.co/datasets/mteb/stsbenchmark-sts) test set using multiple different embedding dimensions. The results, obtained by running [matryoshka_eval_stsb.py](https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/matryoshka/matryoshka_eval_stsb.py),

are plotted in the following figure:



In the top figure, you can see that the Matryoshka model reaches a higher Spearman similarity than the standard model at all dimensionalities, indicative that the Matryoshka model is superior in this task.

Furthermore, the performance of the Matryoshka model falls off much less quickly than the standard model. This is shown clearly in the second figure, which shows the performance at the embedding dimension relative to the maximum performance. **Even at 8.3% of the embedding size, the Matryoshka model preserves 98.37% of the performance** , much higher than the 96.46% by the standard model.

These findings are indicative that truncating embeddings by a Matryoshka model could: 1) significantly speed up downstream tasks such as retrieval and 2) significantly save on storage space, all without a notable hit in performance.

Trainingïƒ•

Training using Matryoshka Representation Learning (MRL) is quite elementary: rather than applying some loss function on only the full-size embeddings, we also apply that same loss function on truncated portions of the embeddings. For example, if a model has an embedding dimension of 768 by default, it can now be trained on 768, 512, 256, 128, 64 and 32. Each of these losses will be

added together, optionally with some weight:

```
from sentence_transformers import SentenceTransformer
from sentence_transformers.losses import CoSENTLoss, MatryoshkaLoss

model = SentenceTransformer("microsoft/mpnet-base")

base_loss = CoSENTLoss(model=model)
loss = MatryoshkaLoss(model=model, loss=base_loss, matryoshka_dims=[768, 512, 256, 128, 64])
```

* **Reference**: [`MatryoshkaLoss`](../../../docs/package_reference/sentence_transformer/losses.html#matryoshkaloss)

Additionally, this can be combined with the `AdaptiveLayerLoss` such that the resulting model can be reduced both in the size of the output dimensions, but also in the number of layers for faster inference. See also the [Adaptive Layers](../adaptive_layer/README.html) for more information on reducing the number of model layers. In Sentence Transformers, the combination of these two losses is called `Matryoshka2dLoss`, and a shorthand is provided for simpler training.

```
    from sentence_transformers import SentenceTransformer
    from sentence_transformers.losses import CoSENTLoss, Matryoshka2dLoss

    model = SentenceTransformer("microsoft/mpnet-base")

    base_loss = CoSENTLoss(model=model)
    loss = Matryoshka2dLoss(model=model, loss=base_loss, matryoshka_dims=[768, 512, 256, 128,
64])
```

* **Reference**: [`Matryoshka2dLoss`](../../../docs/package_reference/sentence_transformer/losses.html#matryoshka2dloss)

Inferenceïƒ•

After a model has been trained using a Matryoshka loss, you can then run inference with it using [`SentenceTransformers.encode`](../../../docs/package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.encode).

```
    from sentence_transformers import SentenceTransformer
    import torch.nn.functional as F
```

```
    matryoshka_dim = 64
    model = SentenceTransformer(
        "nomic-ai/nomic-embed-text-v1.5",
        trust_remote_code=True,
        truncate_dim=matryoshka_dim,
    )

    embeddings = model.encode(
        [
            "search_query: What is TSNE?",
                "search_document: t-distributed stochastic neighbor embedding (t-SNE) is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map.",
            "search_document: Amelia Mary Earhart was an American aviation pioneer and writer.",
        ]
    )
    assert embeddings.shape[-1] == matryoshka_dim

    similarities = model.similarity(embeddings[0], embeddings[1:])
    # => tensor([[0.7839, 0.4933]])
```

As you can see, the similarity between the search query and the correct
document is much higher than that of an unrelated document, despite the very
small matryoshka dimension applied. Feel free to copy this script locally,
modify the `matryoshka_dim`, and observe the difference in similarities.

Note : Despite the embeddings being smaller, training and inference of a Matryoshka model is not faster, not more memory-efficient, and not smaller. Only the processing and storage of the resulting embeddings will be faster and cheaper.

Code Examplesïƒ•

See the following scripts as examples of how to apply the [`MatryoshkaLoss`](../../../docs/package_reference/sentence_transformer/losses.html#matryoshkaloss) in practice:

*

[matryoshka_nli.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/matryoshka_nli.py) : This example uses the MultipleNegativesRankingLoss with MatryoshkaLoss to train a strong embedding model using Natural Language Inference (NLI) data. It is an adaptation of the [NLI](../nli/README.html) documentation.

*

[matryoshka_nli_reduced_dim.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/matryoshka_nli_reduced_dim.py) : This example uses the MultipleNegativesRankingLoss with MatryoshkaLoss to train a strong embedding model with a small maximum output dimension of 256. It trains using Natural Language Inference (NLI) data, and is an adaptation of the [NLI](../nli/README.html) documentation.

*

**[matryoshka_eval_stsb.py](https://github.com/UKPLab/sentence-transformers/tree/master/exampl

es/training/matryoshka/matryoshka_eval_stsb.py)** : This example evaluates the embedding model trained with MatryoshkaLoss in [matryoshka_nli.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/matryoshka_nli.py) on the test set of the STSBenchmark dataset, and compares it to a non-Matryoshka trained model.

*

[matryoshka_sts.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/matryoshka_sts.py) : This example uses the CoSENTLoss with MatryoshkaLoss to train an embedding model on the training set of the STSBenchmark dataset. It is an adaptation of the [STS](../sts/README.html) documentation.

And the following scripts to see how to apply [`Matryoshka2dLoss`](../../../docs/package_reference/sentence_transformer/losses.html#matryoshka2dloss):

*

[2d_matryoshka_nli.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/2d_matryoshka_nli.py) : This example uses the `MultipleNegativesRankingLoss` with `Matryoshka2dLoss` to train a strong embedding model using Natural Language Inference (NLI) data. It is an adaptation of the [NLI](../nli/README.html) documentation.

*

[2d_matryoshka_sts.py](https://github.com/UKPLab/sentence-transformers/tree/master/examples/training/matryoshka/2d_matryoshka_sts.py) : This example uses the `CoSENTLoss` with `Matryoshka2dLoss` to train an embedding model on the training set of the STSBenchmark dataset.

It is an adaptation of the [STS](../sts/README.html) documentation.

[Previous](../ms_marco/README.html "MS MARCO") [Next

](../adaptive_layer/README.html "Adaptive Layers")

* * *