



← [Back to blog](#)

# Introduction to Matryoshka Embedding Models

Published February 23, 2024

[Update on GitHub](#)



[tomaarsen](#)  
[Tom Aarsen](#)



[Xenova](#)  
[Joshua](#)





[osanseviero](#)  
[Omar Sanseviero](#)

In this blogpost, we will introduce you to the concept of Matryoshka Embeddings and explain why they are useful. We will discuss how these models are theoretically trained and how you can train them using Sentence Transformers.

Additionally, we will provide practical guidance on how to use Matryoshka Embedding models and share a comparison between a Matryoshka embedding model and a regular embedding model. Finally, we invite you to check out our interactive demo that showcases the power of these models.

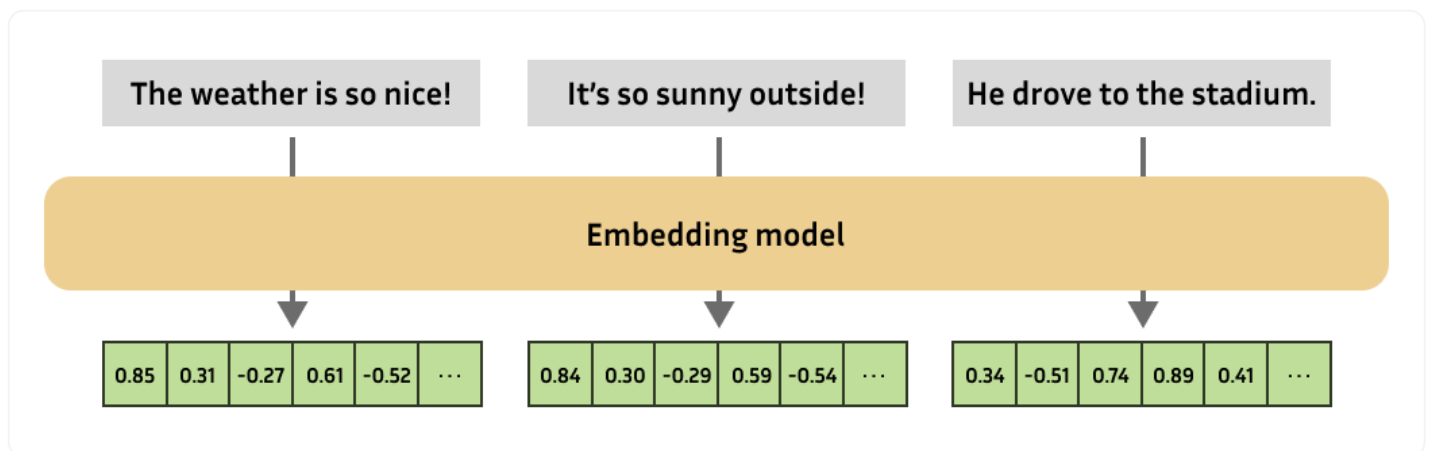
## Table of Contents

- [Understanding Embeddings](#)
-  [Matryoshka Embeddings](#)
-  [Matryoshka Dolls](#)
- [Why would you use !\[\]\(020ca36803168f31a8fb3f576699f65c\_img.jpg\) Matryoshka Embedding models?](#)
- [How are !\[\]\(84f1d587b913ebf353d69e5f3ebedfd2\_img.jpg\) Matryoshka Embedding models trained?](#)
  - [Theoretically](#)

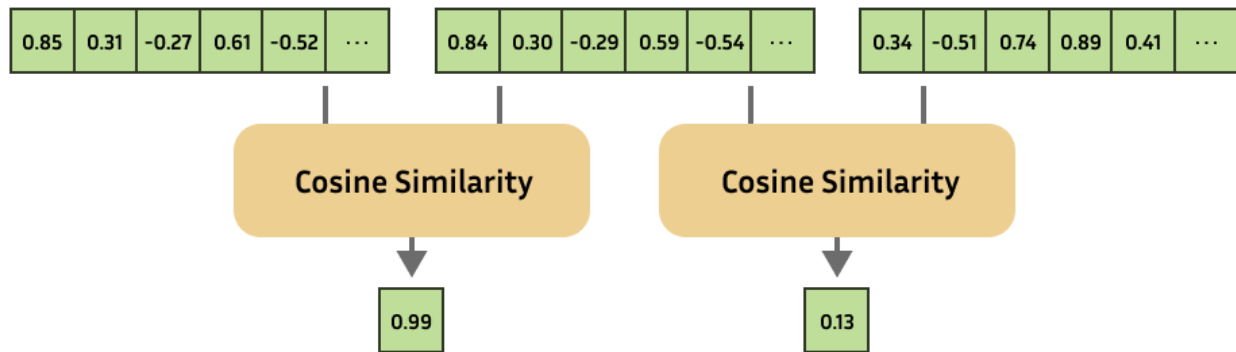
- [In Sentence Transformers](#)
- [How do I use 🧑🔬 Matryoshka Embedding models?](#)
  - [Theoretically](#)
  - [In Sentence Transformers](#)
- [Results](#)
- [Demo](#)
- [References](#)

## 🔗 Understanding Embeddings

Embeddings are one of the most versatile tools in natural language processing, enabling practitioners to solve a large variety of tasks. In essence, an embedding is a numerical representation of a more complex object, like text, images, audio, etc.



The embedding model will always produce embeddings of the same fixed size. You can then compute the similarity of complex objects by computing the similarity of the respective embeddings!

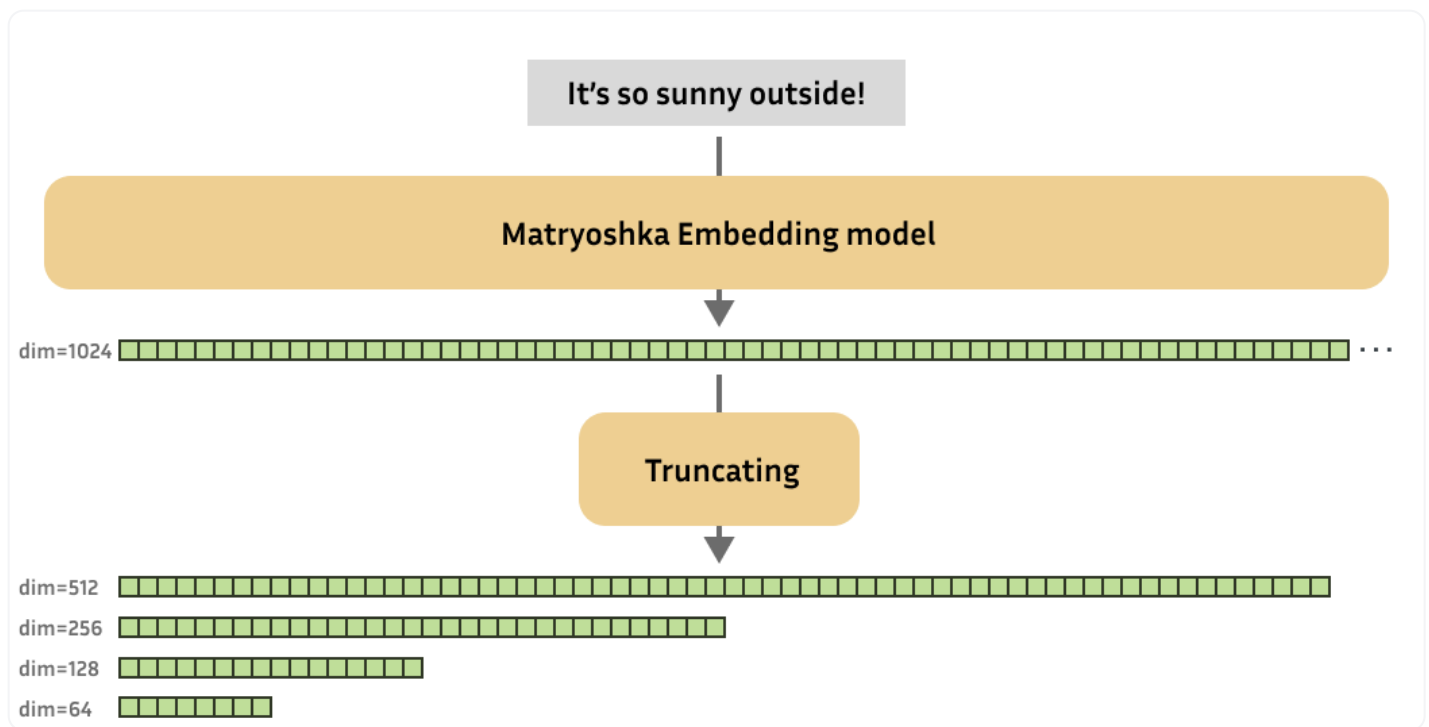


This has an enormous amount of use cases, and serves as the backbone for recommendation systems, retrieval, one-shot or few-shot learning, outlier detection, similarity search, paraphrase detection, clustering, classification, and much more!

## 🔗 🤖 Matryoshka Embeddings

As research progressed, new state-of-the-art (text) embedding models started producing embeddings with increasingly higher output dimensions, i.e., every input text is represented using more values. Although this improves performance, it comes at the cost of efficiency of downstream tasks such as search or classification.

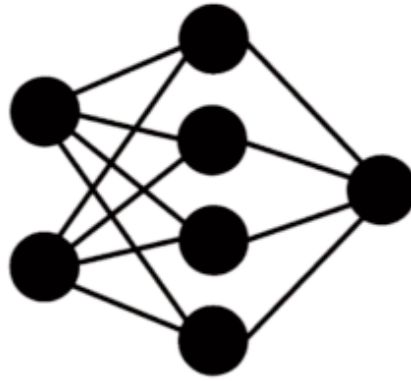
Consequently, [Kusupati et al. \(2022\)](#) were inspired to create embedding models whose embeddings could reasonably be shrunk without suffering too much on performance.



These Matryoshka embedding models are trained such that these small truncated embeddings would still be useful. In short, Matryoshka embedding models can produce useful embeddings of various dimensions.

## 🔗 🇷🇺 Matryoshka Dolls

For those unfamiliar, "Matryoshka dolls", also known as "Russian nesting dolls", are a set of wooden dolls of decreasing size that are placed inside one another. In a similar way, Matryoshka embedding models aim to store more important information in earlier dimensions, and less important information in later dimensions. This characteristic of Matryoshka embedding models allows us to truncate the original (large) embedding produced by the model, while still retaining enough of the information to perform well on downstream tasks.



## 1. Compute Matryoshka Embedding

### 🔗 Why would you use 🇷🇺 Matryoshka Embedding models?

Such variable-size embedding models can be quite valuable to practitioners, for example:

1. **Shortlisting and reranking:** Rather than performing your downstream task (e.g., nearest neighbor search) on the full embeddings, you can shrink the embeddings to a smaller size and very efficiently "shortlist" your embeddings. Afterwards, you can process the remaining embeddings using their full dimensionality.
2. **Trade-offs:** Matryoshka models will allow you to scale your embedding solutions to your desired storage cost, processing speed, and performance.

### 🔗 How are 🇷🇺 Matryoshka Embedding models trained?

#### 🔗 Theoretically

The Matryoshka Representation Learning (MRL) approach can be adopted for almost all embedding model training frameworks. Normally, a training step for an embedding model involves producing embeddings for your training batch (of texts, for example) and then using some loss function to create a loss value that represents the quality of the produced embeddings. The optimizer will adjust the model weights throughout training to reduce the loss value.

For Matryoshka Embedding models, a training step also involves producing embeddings for your training batch, but then you use some loss function to determine not just the quality of your full-size embeddings, but also the quality of your embeddings at various different dimensionalities. For example, output dimensionalities are 768, 512, 256, 128, and 64. The loss values for each dimensionality are added together, resulting in a final loss value. The optimizer will then try and adjust the model weights to lower this loss value.

In practice, this incentivizes the model to frontload the most important information at the start of an embedding, such that it will be retained if the embedding is truncated.

## 🔗 In Sentence Transformers

Sentence Transformers is a commonly used framework to train embedding models, and it recently implemented support for Matryoshka models. Training a Matryoshka embedding model using Sentence Transformers 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 original embedding dimension of 768, it can now be trained on 768, 512, 256, 128 and 64. 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],
    matryoshka_weight=[1, 1, 1, 1, 1],
)

model.fit(
    train_objectives=[(train_dataset, loss)],
    ...,
)

```

Training with MatryoshkaLoss does not incur a notable overhead in training time.

## References:

- [MatryoshkaLoss](#)
- [CoSENTLoss](#)
- [SentenceTransformer](#)
- [SentenceTransformer.fit](#)
- [Matryoshka Embeddings - Training](#)

See the following complete scripts as examples of how to apply the MatryoshkaLoss in practice:

- [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](#) documentation.
- [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](#) documentation.
- [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](#) documentation.

## 🔗 How do I use 🧸 Matryoshka Embedding models?

### 🔗 Theoretically

In practice, getting embeddings from a Matryoshka embedding model works the same way as with a normal embedding model. The only difference is that, after receiving the embeddings, we can optionally truncate them to a smaller dimensionality. Do note that if the embeddings were normalized, then after truncating they will no longer be, so you may want to re-normalize.

After truncating, you can either directly apply them for your use cases, or store them such that they can be used later. After all, smaller embeddings in your vector database should result in considerable speedups!

Keep in mind that although processing smaller embeddings for downstream tasks (retrieval, clustering, etc.) will be faster, getting the smaller embeddings from the model is just as fast as getting the larger ones.

### 🔗 In Sentence Transformers

In Sentence Transformers, you can load a Matryoshka Embedding model like normal, and run inference with it using `SentenceTransformers.encode`. After getting the embeddings, we can truncate them to our desired size, and we can normalize them if we want.

Let's try to use a model that I trained using `matryoshka_nli.py` with `microsoft/mpnet-base`:

```
from sentence_transformers import SentenceTransformer
from sentence_transformers.util import cos_sim

model = SentenceTransformer("tomaarsen/mpnet-base-nli-matryoshka")

matryoshka_dim = 64
embeddings = model.encode([
```



```

        "The weather is so nice!",
        "It's so sunny outside!",
        "He drove to the stadium.",
    ]
)
embeddings = embeddings[..., :matryoshka_dim] # Shrink the embedding dimensions
print(embeddings.shape)
# => (3, 64)

# Similarity of the first sentence to the other two:
similarities = cos_sim(embeddings[0], embeddings[1:])
print(similarities)
# => tensor([[0.8910, 0.1337]])

```

- Link to the model: [tomaarsen/mpnet-base-nli-matryoshka](https://huggingface.co/tomaarsen/mpnet-base-nli-matryoshka)

Feel free to experiment with using different values for `matryoshka_dim` and observe how that affects the similarities. You can do so either by running this code locally, on the cloud such as with [Google Colab](#), or by checking out the [demo](#).

## References:

- [SentenceTransformer](#)
- [SentenceTransformer.encode](#)
- [util.cos\\_sim](#)
- [Matryoshka Embeddings - Inference](#)

► [Click here to see how to use the Nomic v1.5 Matryoshka Model](#)

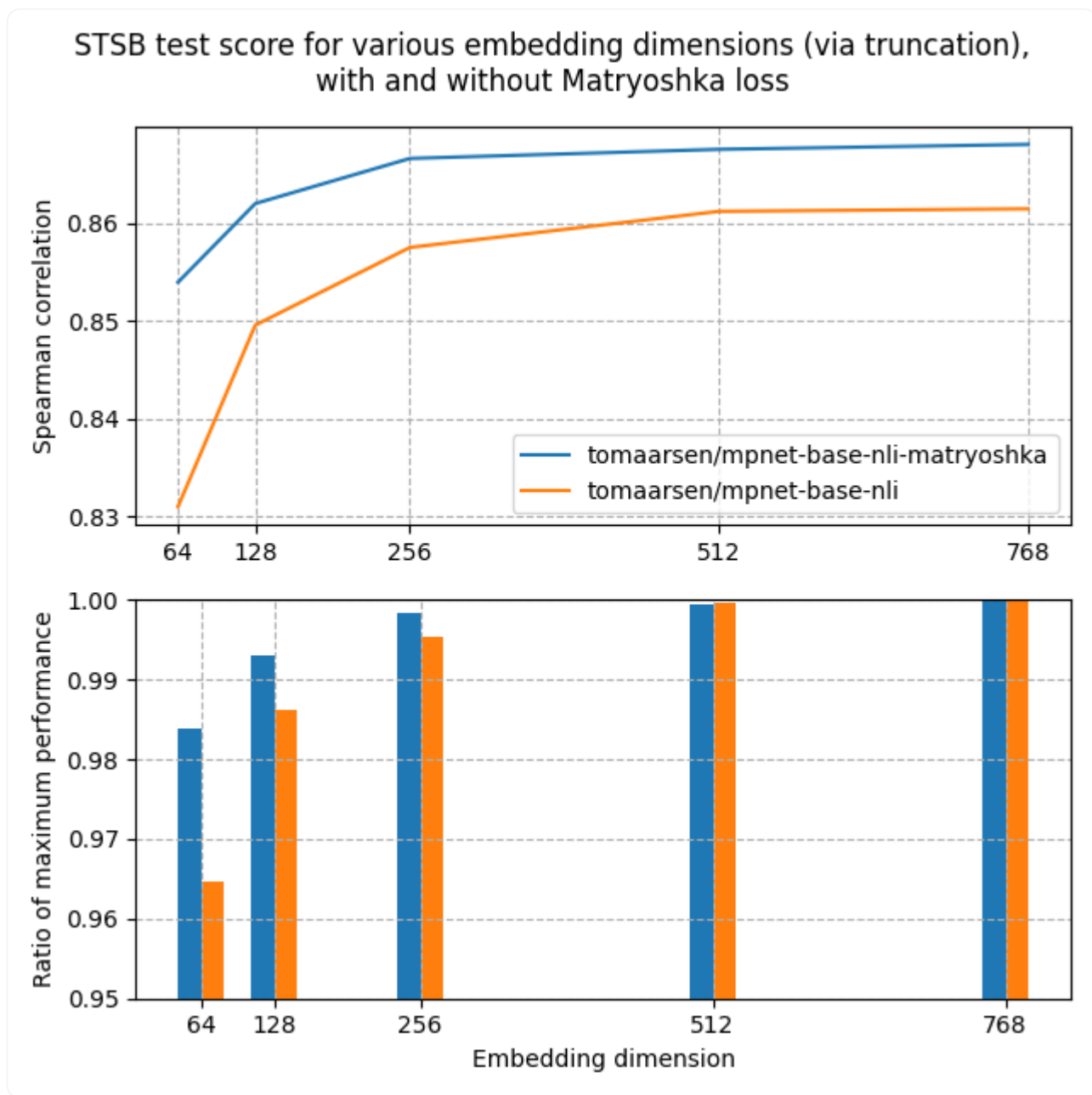
## 🔗 Results

Now that Matryoshka models have been introduced, let'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: Trained by running matryoshka\_nli.py with microsoft/mpnet-base.
- tomaarsen/mpnet-base-nli: Trained by running a modified version of matryoshka\_nli.py where the training loss is only `MultipleNegativesRankingLoss` rather than `MatryoshkaLoss` on top of `MultipleNegativesRankingLoss`. I also use microsoft/mpnet-base as the base model.

Both of these models were trained on the AllNLI dataset, which is a concatenation of the SNLI and MultiNLI datasets. I have evaluated these models on the STSBenchmark test set using multiple different embedding dimensions. The results 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.

## 🔗 Demo

In this demo, you can dynamically shrink the output dimensions of the [nomic-ai/nomic-embed-text-v1.5](#) Matryoshka embedding model and observe how it affects the retrieval performance. All of the embeddings are computed in the browser using 😊 [Transformers.js](#).

## Query

What is a panda?

## Text

A panda's diet consists almost entirely of bamboo.  
The typical life span of a panda is 20 years in the wild.  
I love pandas so much!  
Bamboo is a fast-growing, woody grass.  
Hello world.  
Ailuropoda melanoleuca is a bear species endemic to China.  
I love the color blue.  
A panda is a large black-and-white bear native to China.  
Once upon a time, in a land far, far away...  
My favorite movie is Kung Fu Panda.  
This is an example sentence.

## References

- Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., ... & Farhadi, A. (2022). Matryoshka representation learning. Advances in Neural Information Processing Systems, 35, 30233-30249. <https://arxiv.org/abs/2205.13147>
  - Matryoshka Embeddings — Sentence-Transformers documentation. (n.d.). <https://sbert.net/examples/training/matryoshka/README.html>
  - UKPLab. (n.d.). GitHub. <https://github.com/UKPLab/sentence-transformers>
  - Unboxing Nomic Embed v1.5: Resizable Production Embeddings with Matryoshka Representation Learning. (n.d.). <https://blog.nomic.ai/posts/nomic-embed-matryoshka>
- 

### More articles from our Blog



## Parameter Efficient Finetuning

With PyTorch on Cloud TPUs and GPUs

### Fine-Tuning Gemma Models in Hugging Face

By svaibhav February 23, 2024 guest



# Welcome Gemma

Google's new open LLM

Welcome Gemma - Google's new open LLM



## Company

[TOS](#)

[Privacy](#)

[About](#)

[Jobs](#)

## Website

[Models](#)

[Datasets](#)

[Spaces](#)

[Pricing](#)

[Docs](#)

