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

Getting Started

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

Sentence Transformer

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

* [Input Sequence Length](../../../examples/applications/computing-embeddings/README.html#id1)
* [Multi-Process / Multi-GPU Encoding](../../../examples/applications/computing-embeddings/README.html#multi-process-multi-gpu-encoding)
* [Semantic Textual Similarity](semantic_textual_similarity.html)
* [Similarity Calculation](semantic_textual_similarity.html#similarity-calculation)
* [Semantic Search](../../../examples/applications/semantic-search/README.html)
* [Background](../../../examples/applications/semantic-search/README.html#background)
* [Symmetric vs. Asymmetric Semantic Search](../../../examples/applications/semantic-search/README.html#symmetric-vs-asymmetric-semantic-search)
* [Manual Implementation](../../../examples/applications/semantic-search/README.html#manual-implementation)
* [Optimized Implementation](../../../examples/applications/semantic-search/README.html#optimized-implementation)
* [Speed Optimization](../../../examples/applications/semantic-search/README.html#speed-optimization)
* [Elasticsearch](../../../examples/applications/semantic-search/README.html#elasticsearch)
* [Approximate Nearest Neighbor](../../../examples/applications/semantic-search/README.html#approximate-nearest-neighbor)
* [Retrieve & Re-Rank](../../../examples/applications/semantic-search/README.html#retrieve-re-rank)
* [Examples](../../../examples/applications/semantic-search/README.html#examples)

* [Retrieve & Re-Rank](../../../examples/applications/retrieve_rerank/README.html)
* [Retrieve & Re-Rank Pipeline](../../../examples/applications/retrieve_rerank/README.html#retrieve-re-rank-pipeline)
* [Retrieval: Bi-Encoder](../../../examples/applications/retrieve_rerank/README.html#retrieval-bi-encoder)
* [Re-Ranker: Cross-Encoder](../../../examples/applications/retrieve_rerank/README.html#re-ranker-cross-encoder)
* [Example Scripts](../../../examples/applications/retrieve_rerank/README.html#example-scripts)
* [Pre-trained Bi-Encoders (Retrieval)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)
* [Pre-trained Cross-Encoders (Re-Ranker)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)
* [Clustering](../../../examples/applications/clustering/README.html)
* [k-Means](../../../examples/applications/clustering/README.html#k-means)
* [Agglomerative Clustering](../../../examples/applications/clustering/README.html#agglomerative-clustering)
* [Fast Clustering](../../../examples/applications/clustering/README.html#fast-clustering)
* [Topic Modeling](../../../examples/applications/clustering/README.html#topic-modeling)
* [Paraphrase Mining](../../../examples/applications/paraphrase-mining/README.html)
* [`paraphrase_mining()`](../../../examples/applications/paraphrase-mining/README.html#sentence_transformers.util.paraphrase_mining)
* [Translated Sentence Mining](../../../examples/applications/parallel-sentence-mining/README.html)

* [Margin Based Mining](../../../examples/applications/parallel-sentence-mining/README.html#margin-based-mining)
    * [Examples](../../../examples/applications/parallel-sentence-mining/README.html#examples)
  * [Image Search](../../../examples/applications/image-search/README.html)
    * [Installation](../../../examples/applications/image-search/README.html#installation)
    * [Usage](../../../examples/applications/image-search/README.html#usage)
    * [Examples](../../../examples/applications/image-search/README.html#examples)
  * [Embedding Quantization](../../../examples/applications/embedding-quantization/README.html)
    * [Binary Quantization](../../../examples/applications/embedding-quantization/README.html#binary-quantization)
    * [Scalar (int8) Quantization](../../../examples/applications/embedding-quantization/README.html#scalar-int8-quantization)
    * [Additional extensions](../../../examples/applications/embedding-quantization/README.html#additional-extensions)
    * [Demo](../../../examples/applications/embedding-quantization/README.html#demo)
    * [Try it yourself](../../../examples/applications/embedding-quantization/README.html#try-it-yourself)
  * Speeding up Inference
    * PyTorch
    * ONNX
    * OpenVINO
    * Benchmarks
  * [Creating Custom Models](custom_models.html)
    * [Structure of Sentence Transformer

Models](custom_models.html#structure-of-sentence-transformer-models)

* [Sentence Transformer Model from a Transformers Model](custom_models.html#sentence-transformer-model-from-a-transformers-model)

* [Pretrained Models](../pretrained_models.html)
  * [Original Models](../pretrained_models.html#original-models)
  * [Semantic Search Models](../pretrained_models.html#semantic-search-models)
    * [Multi-QA Models](../pretrained_models.html#multi-qa-models)
    * [MSMARCO Passage Models](../pretrained_models.html#msmarco-passage-models)
  * [Multilingual Models](../pretrained_models.html#multilingual-models)
    * [Semantic Similarity Models](../pretrained_models.html#semantic-similarity-models)
    * [Bitext Mining](../pretrained_models.html#bitext-mining)
  * [Image & Text-Models](../pretrained_models.html#image-text-models)
  * [INSTRUCTOR models](../pretrained_models.html#instructor-models)
  * [Scientific Similarity Models](../pretrained_models.html#scientific-similarity-models)
* [Training Overview](../training_overview.html)
  * [Why Finetune?](../training_overview.html#why-finetune)
  * [Training Components](../training_overview.html#training-components)
  * [Dataset](../training_overview.html#dataset)
    * [Dataset Format](../training_overview.html#dataset-format)
  * [Loss Function](../training_overview.html#loss-function)
  * [Training Arguments](../training_overview.html#training-arguments)
  * [Evaluator](../training_overview.html#evaluator)
  * [Trainer](../training_overview.html#trainer)
    * [Callbacks](../training_overview.html#callbacks)
  * [Multi-Dataset Training](../training_overview.html#multi-dataset-training)
  * [Deprecated Training](../training_overview.html#deprecated-training)
  * [Best Base Embedding Models](../training_overview.html#best-base-embedding-models)

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

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

* [Citation](../../../examples/training/multilingual/README.html#citation)
* [Model Distillation](../../../examples/training/distillation/README.html)
* [Knowledge Distillation](../../../examples/training/distillation/README.html#knowledge-distillation)
* [Speed - Performance Trade-Off](../../../examples/training/distillation/README.html#speed-performance-trade-off)
* [Dimensionality Reduction](../../../examples/training/distillation/README.html#dimensionality-reduction)
* [Quantization](../../../examples/training/distillation/README.html#quantization)
* [Augmented SBERT](../../../examples/training/data_augmentation/README.html)
* [Motivation](../../../examples/training/data_augmentation/README.html#motivation)
* [Extend to your own datasets](../../../examples/training/data_augmentation/README.html#extend-to-your-own-datasets)
* [Methodology](../../../examples/training/data_augmentation/README.html#methodology)
* [Scenario 1: Limited or small annotated datasets (few labeled sentence-pairs)](../../../examples/training/data_augmentation/README.html#scenario-1-limited-or-small-annotated-datasets-few-labeled-sentence-pairs)
* [Scenario 2: No annotated datasets (Only unlabeled sentence-pairs)](../../../examples/training/data_augmentation/README.html#scenario-2-no-annotated-datasets-only-unlabeled-sentence-pairs)
* [Training](../../../examples/training/data_augmentation/README.html#training)
* [Citation](../../../examples/training/data_augmentation/README.html#citation)
* [Training with Prompts](../../../examples/training/prompts/README.html)
* [What are Prompts?](../../../examples/training/prompts/README.html#what-are-prompts)
* [Why would we train with Prompts?](../../../examples/training/prompts/README.html#why-would-we-train-with-prompts)
* [How do we train with

Prompts?](../../../examples/training/prompts/README.html#how-do-we-train-with-prompts)

* [Training with PEFT Adapters](../../../examples/training/peft/README.html)

* [Compatibility Methods](../../../examples/training/peft/README.html#compatibility-methods)

* [Adding a New Adapter](../../../examples/training/peft/README.html#adding-a-new-adapter)

* [Loading a Pretrained Adapter](../../../examples/training/peft/README.html#loading-a-pretrained-adapter)

* [Training Script](../../../examples/training/peft/README.html#training-script)

* [Unsupervised Learning](../../../examples/unsupervised_learning/README.html)

* [TSDAE](../../../examples/unsupervised_learning/README.html#tsdae)

* [SimCSE](../../../examples/unsupervised_learning/README.html#simcse)

* [CT](../../../examples/unsupervised_learning/README.html#ct)

* [CT (In-Batch Negative Sampling)](../../../examples/unsupervised_learning/README.html#ct-in-batch-negative-sampling)

* [Masked Language Model (MLM)](../../../examples/unsupervised_learning/README.html#masked-language-model-mlm)

* [GenQ](../../../examples/unsupervised_learning/README.html#genq)

* [GPL](../../../examples/unsupervised_learning/README.html#gpl)

* [Performance Comparison](../../../examples/unsupervised_learning/README.html#performance-comparison)

* [Domain Adaptation](../../../examples/domain_adaptation/README.html)

* [Domain Adaptation vs. Unsupervised Learning](../../../examples/domain_adaptation/README.html#domain-adaptation-vs-unsupervised-learning)

* [Adaptive Pre-Training](../../../examples/domain_adaptation/README.html#adaptive-pre-training)

* [GPL: Generative Pseudo-Labeling](../../../examples/domain_adaptation/README.html#gpl-generative-pseudo-labelin

g)

* [Hyperparameter Optimization](../../../examples/training/hpo/README.html)

* [HPO Components](../../../examples/training/hpo/README.html#hpo-components)

* [Putting It All Together](../../../examples/training/hpo/README.html#putting-it-all-together)

* [Example Scripts](../../../examples/training/hpo/README.html#example-scripts)

* [Distributed Training](../training/distributed.html)

* [Comparison](../training/distributed.html#comparison)

* [FSDP](../training/distributed.html#fsdp)

Cross Encoder

* [Usage](../../cross_encoder/usage/usage.html)

* [Retrieve & Re-Rank](../../../examples/applications/retrieve_rerank/README.html)

* [Retrieve & Re-Rank Pipeline](../../../examples/applications/retrieve_rerank/README.html#retrieve-re-rank-pipeline)

* [Retrieval: Bi-Encoder](../../../examples/applications/retrieve_rerank/README.html#retrieval-bi-encoder)

* [Re-Ranker: Cross-Encoder](../../../examples/applications/retrieve_rerank/README.html#re-ranker-cross-encoder)

* [Example Scripts](../../../examples/applications/retrieve_rerank/README.html#example-scripts)

* [Pre-trained Bi-Encoders (Retrieval)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-bi-encoders-retrieval)

* [Pre-trained Cross-Encoders (Re-Ranker)](../../../examples/applications/retrieve_rerank/README.html#pre-trained-cross-encoders-re-ranker)

  * [Pretrained Models](../../cross_encoder/pretrained_models.html)
    * [MS MARCO](../../cross_encoder/pretrained_models.html#ms-marco)
    * [SQuAD (QNLI)](../../cross_encoder/pretrained_models.html#squad-qnli)
    * [STSbenchmark](../../cross_encoder/pretrained_models.html#stsbenchmark)
    * [Quora Duplicate Questions](../../cross_encoder/pretrained_models.html#quora-duplicate-questions)
    * [NLI](../../cross_encoder/pretrained_models.html#nli)
    * [Community Models](../../cross_encoder/pretrained_models.html#community-models)
  * [Training Overview](../../cross_encoder/training_overview.html)
  * [Training Examples](../../cross_encoder/training/examples.html)
    * [MS MARCO](../../../examples/training/ms_marco/cross_encoder_README.html)
    * [Cross-Encoder](../../../examples/training/ms_marco/cross_encoder_README.html#cross-encoder)
    * [Cross-Encoder Knowledge Distillation](../../../examples/training/ms_marco/cross_encoder_README.html#cross-encoder-knowledge-distillation)

Package Reference

  * [Sentence Transformer](../../package_reference/sentence_transformer/index.html)
    * [SentenceTransformer](../../package_reference/sentence_transformer/SentenceTransformer.html)
    * [SentenceTransformer](../../package_reference/sentence_transformer/SentenceTransformer.html#id1)
    * [SentenceTransformerModelCardData](../../package_reference/sentence_transformer/SentenceTran

sformer.html#sentencetransformermodelcarddata)

*

[SimilarityFunction](../../package_reference/sentence_transformer/SentenceTransformer.html#similarityfunction)

* [Trainer](../../package_reference/sentence_transformer/trainer.html)

*

[SentenceTransformerTrainer](../../package_reference/sentence_transformer/trainer.html#sentencetransformertrainer)

* [Training Arguments](../../package_reference/sentence_transformer/training_args.html)

*

[SentenceTransformerTrainingArguments](../../package_reference/sentence_transformer/training_args.html#sentencetransformertrainingarguments)

* [Losses](../../package_reference/sentence_transformer/losses.html)

*

[BatchAllTripletLoss](../../package_reference/sentence_transformer/losses.html#batchalltripletloss)

*

[BatchHardSoftMarginTripletLoss](../../package_reference/sentence_transformer/losses.html#batchhardsoftmargintripletloss)

*

[BatchHardTripletLoss](../../package_reference/sentence_transformer/losses.html#batchhardtripletloss)

*

[BatchSemiHardTripletLoss](../../package_reference/sentence_transformer/losses.html#batchsemihardtripletloss)

* [ContrastiveLoss](../../package_reference/sentence_transformer/losses.html#contrastiveloss)

*

[OnlineContrastiveLoss](../../package_reference/sentence_transformer/losses.html#onlinecontrastiv

eloss)

*

[ContrastiveTensionLoss](../../package_reference/sentence_transformer/losses.html#contrastivetensionloss)

*

[ContrastiveTensionLossInBatchNegatives](../../package_reference/sentence_transformer/losses.html#contrastivetensionlossinbatchnegatives)

    * [CoSENTLoss](../../package_reference/sentence_transformer/losses.html#cosentloss)

    * [AnglELoss](../../package_reference/sentence_transformer/losses.html#angleloss)

*

[CosineSimilarityLoss](../../package_reference/sentence_transformer/losses.html#cosinesimilarityloss)

*

[DenoisingAutoEncoderLoss](../../package_reference/sentence_transformer/losses.html#denoisingautoencoderloss)

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

*

[CachedGISTEmbedLoss](../../package_reference/sentence_transformer/losses.html#cachedgistembedloss)

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

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

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

*

[Matryoshka2dLoss](../../package_reference/sentence_transformer/losses.html#matryoshka2dloss)

*

[AdaptiveLayerLoss](../../package_reference/sentence_transformer/losses.html#adaptivelayerloss)

*

[MegaBatchMarginLoss](../../package_reference/sentence_transformer/losses.html#megabatchmarginloss)
*
[MultipleNegativesRankingLoss](../../package_reference/sentence_transformer/losses.html#multiplenegativesrankingloss)
*
[CachedMultipleNegativesRankingLoss](../../package_reference/sentence_transformer/losses.html#cachedmultiplenegativesrankingloss)
*
[MultipleNegativesSymmetricRankingLoss](../../package_reference/sentence_transformer/losses.html#multiplenegativessymmetricrankingloss)
*
[CachedMultipleNegativesSymmetricRankingLoss](../../package_reference/sentence_transformer/losses.html#cachedmultiplenegativessymmetricrankingloss)
    * [SoftmaxLoss](../../package_reference/sentence_transformer/losses.html#softmaxloss)
    * [TripletLoss](../../package_reference/sentence_transformer/losses.html#tripletloss)
  * [Samplers](../../package_reference/sentence_transformer/sampler.html)
    * [BatchSamplers](../../package_reference/sentence_transformer/sampler.html#batchsamplers)
*
[MultiDatasetBatchSamplers](../../package_reference/sentence_transformer/sampler.html#multidatasetbatchsamplers)
  * [Evaluation](../../package_reference/sentence_transformer/evaluation.html)
*
[BinaryClassificationEvaluator](../../package_reference/sentence_transformer/evaluation.html#binaryclassificationevaluator)
*
[EmbeddingSimilarityEvaluator](../../package_reference/sentence_transformer/evaluation.html#emb

eddingsimilarityevaluator)

*

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

*

[NanoBEIREvaluator](../../package_reference/sentence_transformer/evaluation.html#nanobeirevaluator)

* [MSEEvaluator](../../package_reference/sentence_transformer/evaluation.html#mseevaluator)

*

[ParaphraseMiningEvaluator](../../package_reference/sentence_transformer/evaluation.html#paraphraseminingevaluator)

*

[RerankingEvaluator](../../package_reference/sentence_transformer/evaluation.html#rerankingevaluator)

*

[SentenceEvaluator](../../package_reference/sentence_transformer/evaluation.html#sentenceevaluator)

*

[SequentialEvaluator](../../package_reference/sentence_transformer/evaluation.html#sequentialevaluator)

*

[TranslationEvaluator](../../package_reference/sentence_transformer/evaluation.html#translationevaluator)

*

[TripletEvaluator](../../package_reference/sentence_transformer/evaluation.html#tripletevaluator)

* [Datasets](../../package_reference/sentence_transformer/datasets.html)

*

[ParallelSentencesDataset](../../package_reference/sentence_transformer/datasets.html#parallelsentencesdataset)
*
[SentenceLabelDataset](../../package_reference/sentence_transformer/datasets.html#sentencelabeldataset)
*
[DenoisingAutoEncoderDataset](../../package_reference/sentence_transformer/datasets.html#denoisingautoencoderdataset)
*
[NoDuplicatesDataLoader](../../package_reference/sentence_transformer/datasets.html#noduplicatesdataloader)
    * [Models](../../package_reference/sentence_transformer/models.html)
      * [Main Classes](../../package_reference/sentence_transformer/models.html#main-classes)
      * [Further Classes](../../package_reference/sentence_transformer/models.html#further-classes)
    * [quantization](../../package_reference/sentence_transformer/quantization.html)
*
[`quantize_embeddings()`](../../package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.quantize_embeddings)
*
[`semantic_search_faiss()`](../../package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_faiss)
*
[`semantic_search_usearch()`](../../package_reference/sentence_transformer/quantization.html#sentence_transformers.quantization.semantic_search_usearch)
  * [Cross Encoder](../../package_reference/cross_encoder/index.html)
    * [CrossEncoder](../../package_reference/cross_encoder/cross_encoder.html)
      * [CrossEncoder](../../package_reference/cross_encoder/cross_encoder.html#id1)

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

*
[`mine_hard_negatives()`](../../package_reference/util.html#sentence_transformers.util.mine_hard_negatives)
*
[`normalize_embeddings()`](../../package_reference/util.html#sentence_transformers.util.normalize_embeddings)
*
[`paraphrase_mining()`](../../package_reference/util.html#sentence_transformers.util.paraphrase_mining)
*
[`semantic_search()`](../../package_reference/util.html#sentence_transformers.util.semantic_search)
*
[`truncate_embeddings()`](../../package_reference/util.html#sentence_transformers.util.truncate_embeddings)
* [Model Optimization](../../package_reference/util.html#module-sentence_transformers.backend)
*
[`export_dynamic_quantized_onnx_model()`](../../package_reference/util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model)
*
[`export_optimized_onnx_model()`](../../package_reference/util.html#sentence_transformers.backend.export_optimized_onnx_model)
*
[`export_static_quantized_openvino_model()`](../../package_reference/util.html#sentence_transformers.backend.export_static_quantized_openvino_model)
* [Similarity Metrics](../../package_reference/util.html#module-sentence_transformers.util)
* [`cos_sim()`](../../package_reference/util.html#sentence_transformers.util.cos_sim)
* [`dot_score()`](../../package_reference/util.html#sentence_transformers.util.dot_score)

* [`euclidean_sim()`](../../package_reference/util.html#sentence_transformers.util.euclidean_sim)
* [`manhattan_sim()`](../../package_reference/util.html#sentence_transformers.util.manhattan_sim)
* [`pairwise_cos_sim()`](../../package_reference/util.html#sentence_transformers.util.pairwise_cos_sim)
* [`pairwise_dot_score()`](../../package_reference/util.html#sentence_transformers.util.pairwise_dot_score)
* [`pairwise_euclidean_sim()`](../../package_reference/util.html#sentence_transformers.util.pairwise_euclidean_sim)
* [`pairwise_manhattan_sim()`](../../package_reference/util.html#sentence_transformers.util.pairwise_manhattan_sim)

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

* [](../../../index.html)
* [Usage](usage.html)
* Speeding up Inference
* [Edit on GitHub](https://github.com/UKPLab/sentence-transformers/blob/master/docs/sentence_transformer/usage/efficiency.rst)

* * *

Speeding up Inferenceïƒ•

Sentence Transformers supports 3 backends for computing embeddings, each with its own optimizations for speeding up inference:

PyTorch The default backend for Sentence Transformers. ONNX Flexible and efficient model accelerator. OpenVINO Optimization of models, mainly for Intel Hardware. Benchmarks Benchmarks for the different backends. User Interface GUI to export, optimize, and quantize models.

PyTorchïƒ•

The PyTorch backend is the default backend for Sentence Transformers. If you donâ€™t specify a device, it will use the strongest available option across â€œcudaâ€•, â€œmpsâ€•, and â€œcpuâ€•. Its default usage looks like this:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2")

sentences = ["This is an example sentence", "Each sentence is converted"]
embeddings = model.encode(sentences)
```

If youâ€™re using a GPU, then you can use the following options to speed up your inference:

float16 (fp16)

Float32 (fp32, full precision) is the default floating-point format in `torch`, whereas float16 (fp16, half precision) is a reduced-precision floating-point format that can speed up inference on GPUs at a minimal loss of model accuracy. To use it, you can specify the `torch_dtype` during initialization or call [`model.half()`](https://pytorch.org/docs/stable/generated/torch.Tensor.half.html#torch.Tensor.half "\(in PyTorch v2.5\)") on the initialized model:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2", model_kwargs={"torch_dtype": "float16"})
# or: model.half()

sentences = ["This is an example sentence", "Each sentence is converted"]
embeddings = model.encode(sentences)
```

bfloat16 (bf16)

Bfloat16 (bf16) is similar to fp16, but preserves more of the original

accuracy of fp32. To use it, you can specify the `torch_dtype` during

initialization or call

[`model.bfloat16()`](https://pytorch.org/docs/stable/generated/torch.Tensor.bfloat16.html#torch.Tens

or.bfloat16

"\(in PyTorch v2.5\)") on the initialized model:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2", model_kwargs={"torch_dtype": "bfloat16"})
# or: model.bfloat16()

sentences = ["This is an example sentence", "Each sentence is converted"]
embeddings = model.encode(sentences)
```

ONNXïƒ•

Export, Optimize, and Quantize Hugging Face models

This Hugging Face Space provides a user interface for exporting, optimizing,

and quantizing models for either ONNX or OpenVINO:

*

[sentence-transformers/backend-export](https://huggingface.co/spaces/sentence-transformers/back

end-export)

ONNX can be used to speed up inference by converting the model to ONNX format and using ONNX Runtime to run the model. To use the ONNX backend, you must install Sentence Transformers with the `onnx` or `onnx-gpu` extra for CPU or GPU acceleration, respectively:

```
pip install sentence-transformers[onnx-gpu]
# or
pip install sentence-transformers[onnx]
```

To convert a model to ONNX format, you can use the following code:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2", backend="onnx")

sentences = ["This is an example sentence", "Each sentence is converted"]
embeddings = model.encode(sentences)
```

If the model path or repository already contains a model in ONNX format,

Sentence Transformers will automatically use it. Otherwise, it will convert
the model to ONNX the format.

All keyword arguments passed via `model_kwargs` will be passed on to
[`ORTModel.from_pretrained`](https://huggingface.co/docs/optimum/main/en/onnxruntime/package_reference/modeling_ort#optimum.onnxruntime.ORTModel.from_pretrained
"\(in optimum vmain\)"). Some notable arguments include:

* `provider`: ONNX Runtime provider to use for loading the model, e.g. `"CPUExecutionProvider"` . See <https://onnxruntime.ai/docs/execution-providers/> for possible providers. If not specified, the strongest provider (E.g. `"CUDAExecutionProvider"`) will be used.

* `file_name`: The name of the ONNX file to load. If not specified, will default to `"model.onnx"` or otherwise `"onnx/model.onnx"`. This argument is useful for specifying optimized or quantized models.

* `export`: A boolean flag specifying whether the model will be exported. If not provided, `export` will be set to `True` if the model repository or directory does not already contain an ONNX model.

Tip

Itâ€™s heavily recommended to save the exported model to prevent having to re-
export it every time you run your code. You can do this by calling
[`model.save_pretrained()`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.save_pretrained
"sentence_transformers.SentenceTransformer.save_pretrained") if your model was
local:

```
model = SentenceTransformer("path/to/my/model", backend="onnx")
model.save_pretrained("path/to/my/model")
```

or with
[`model.push_to_hub()`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.push_to_hub "sentence_transformers.SentenceTransformer.push_to_hub") if your model was
from the Hugging Face Hub:

```
model = SentenceTransformer("intfloat/multilingual-e5-small", backend="onnx")
model.push_to_hub("intfloat/multilingual-e5-small", create_pr=True)
```

Optimizing ONNX Modelsïƒ•

Export, Optimize, and Quantize Hugging Face models

This Hugging Face Space provides a user interface for exporting, optimizing,
and quantizing models for either ONNX or OpenVINO:

*

[sentence-transformers/backend-export](https://huggingface.co/spaces/sentence-transformers/backend-export)

ONNX models can be optimized using Optimum, allowing for speedups on CPUs and GPUs alike. To do this, you can use the [`export_optimized_onnx_model()`](../../package_reference/util.html#sentence_transformers.backend.export_optimized_onnx_model "sentence_transformers.backend.export_optimized_onnx_model") function, which saves the optimized in a directory or model repository that you specify. It expects:

* `model`: a Sentence Transformer model loaded with the ONNX backend.

* `optimization_config`: `"O1"`, `"O2"`, `"O3"`, or `"O4"` representing optimization levels from [`AutoOptimizationConfig`](https://huggingface.co/docs/optimum/main/en/onnxruntime/package_reference/configuration#optimum.onnxruntime.AutoOptimizationConfig "\(in optimum vmain\)"), or an [`OptimizationConfig`](https://huggingface.co/docs/optimum/main/en/onnxruntime/package_reference/configuration#optimum.onnxruntime.OptimizationConfig "\(in optimum vmain\)") instance.

* `model_name_or_path`: a path to save the optimized model file, or the repository name if you want to push it to the Hugging Face Hub.

* `push_to_hub`: (Optional) a boolean to push the optimized model to the Hugging Face Hub.

* `create_pr`: (Optional) a boolean to create a pull request when pushing to the Hugging Face Hub. Useful when you donâ€™t have write access to the repository.

* `file_suffix`: (Optional) a string to append to the model name when saving it. If not specified, the optimization level name string will be used, or just `"optimized"` if the optimization config was not just a string optimization level.

See this example for exporting a model with [optimization level 3](https://huggingface.co/docs/optimum/main/en/onnxruntime/usage_guides/optimization "\(in optimum vmain\)") (basic and extended general optimizations, transformers-specific fusions, fast Gelu approximation):

Hugging Face Hub Model

Only optimize once:

```
from sentence_transformers import SentenceTransformer, export_optimized_onnx_model

model = SentenceTransformer("all-MiniLM-L6-v2", backend="onnx")
export_optimized_onnx_model(
    model,
    "O3",
    "sentence-transformers/all-MiniLM-L6-v2",
    push_to_hub=True,
    create_pr=True,
)
```

Before the pull request gets merged:

```
from sentence_transformers import SentenceTransformer

pull_request_nr = 2 # TODO: Update this to the number of your pull request
model = SentenceTransformer(
    "all-MiniLM-L6-v2",
    backend="onnx",
    model_kwargs={"file_name": "onnx/model_O3.onnx"},
    revision=f"refs/pr/{pull_request_nr}"
)
```

Once the pull request gets merged:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer(
    "all-MiniLM-L6-v2",
    backend="onnx",
    model_kwargs={"file_name": "onnx/model_O3.onnx"},
)
```

Local Model

Only optimize once:

```
from sentence_transformers import SentenceTransformer, export_optimized_onnx_model

model = SentenceTransformer("path/to/my/mpnet-legal-finetuned", backend="onnx")
export_optimized_onnx_model(model, "O3", "path/to/my/mpnet-legal-finetuned")
```

After optimizing:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer(
    "path/to/my/mpnet-legal-finetuned",
    backend="onnx",
    model_kwargs={"file_name": "onnx/model_O3.onnx"},
)
```

Quantizing ONNX Modelsïƒ•

Export, Optimize, and Quantize Hugging Face models

This Hugging Face Space provides a user interface for exporting, optimizing,
and quantizing models for either ONNX or OpenVINO:

*

[sentence-transformers/backend-export](https://huggingface.co/spaces/sentence-transformers/backend-export)

ONNX models can be quantized to int8 precision using Optimum, allowing for
faster inference on CPUs. To do this, you can use the
[`export_dynamic_quantized_onnx_model()`](../../package_reference/util.html#sentence_transformers.backend.export_dynamic_quantized_onnx_model
"sentence_transformers.backend.export_dynamic_quantized_onnx_model") function,
which saves the quantized in a directory or model repository that you specify.
Dynamic quantization, unlike static quantization, does not require a
calibration dataset. It expects:

* `model`: a Sentence Transformer model loaded with the ONNX backend.

* `quantization_config`: `"arm64"`, `"avx2"`, `"avx512"`, or `"avx512_vnni"` representing quantization configurations from [`AutoQuantizationConfig`](https://huggingface.co/docs/optimum/main/en/onnxruntime/package_reference/configuration#optimum.onnxruntime.AutoQuantizationConfig "\(in optimum vmain\)"), or an [`QuantizationConfig`](https://huggingface.co/docs/optimum/main/en/onnxruntime/package_reference/configuration#optimum.onnxruntime.QuantizationConfig "\(in optimum vmain\)") instance.

* `model_name_or_path`: a path to save the quantized model file, or the repository name if you want to push it to the Hugging Face Hub.

* `push_to_hub`: (Optional) a boolean to push the quantized model to the Hugging Face Hub.

* `create_pr`: (Optional) a boolean to create a pull request when pushing to the Hugging Face Hub. Useful when you donâ€™t have write access to the repository.

* `file_suffix`: (Optional) a string to append to the model name when saving it. If not specified, `"qint8_quantized"` will be used.

On my CPU, each of the default quantization configurations (`"arm64"`, `"avx2"`, `"avx512"`, `"avx512_vnni"`) resulted in roughly equivalent speedups.

See this example for quantizing a model to `int8` with [avx512_vnni](https://huggingface.co/docs/optimum/main/en/onnxruntime/usage_guides/quantization "\(in optimum vmain\)"):

Hugging Face Hub Model

Only quantize once:

```
from sentence_transformers import SentenceTransformer, export_dynamic_quantized_onnx_model

model = SentenceTransformer("all-MiniLM-L6-v2", backend="onnx")
export_dynamic_quantized_onnx_model(
    model,
    "avx512_vnni",
    "sentence-transformers/all-MiniLM-L6-v2",
    push_to_hub=True,
    create_pr=True,
)
```

Before the pull request gets merged:

```
from sentence_transformers import SentenceTransformer

pull_request_nr = 2 # TODO: Update this to the number of your pull request
model = SentenceTransformer(
    "all-MiniLM-L6-v2",
    backend="onnx",
    model_kwargs={"file_name": "onnx/model_qint8_avx512_vnni.onnx"},
    revision=f"refs/pr/{pull_request_nr}",
)
```

Once the pull request gets merged:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer(
    "all-MiniLM-L6-v2",
    backend="onnx",
    model_kwargs={"file_name": "onnx/model_qint8_avx512_vnni.onnx"},
)
```

Local Model

Only quantize once:

```
from sentence_transformers import SentenceTransformer, export_dynamic_quantized_onnx_model

model = SentenceTransformer("path/to/my/mpnet-legal-finetuned", backend="onnx")
export_dynamic_quantized_onnx_model(model, "O3", "path/to/my/mpnet-legal-finetuned")
```

After quantizing:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer(
    "path/to/my/mpnet-legal-finetuned",
    backend="onnx",
    model_kwargs={"file_name": "onnx/model_qint8_avx512_vnni.onnx"},
)
```

OpenVINOïƒ•

Export, Optimize, and Quantize Hugging Face models

This Hugging Face Space provides a user interface for exporting, optimizing,
and quantizing models for either ONNX or OpenVINO:

*

[sentence-transformers/backend-export](https://huggingface.co/spaces/sentence-transformers/backend-export)

OpenVINO allows for accelerated inference on CPUs by exporting the model to
the OpenVINO format. To use the OpenVINO backend, you must install Sentence
Transformers with the `openvino` extra:

```
pip install sentence-transformers[openvino]
```

To convert a model to OpenVINO format, you can use the following code:

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2", backend="openvino")

sentences = ["This is an example sentence", "Each sentence is converted"]
embeddings = model.encode(sentences)
```

All keyword arguments passed via `model_kwargs` will be passed on to [`OVBaseModel.from_pretrained()`](https://huggingface.co/docs/optimum/intel/openvino/reference#optimum.intel.openvino.modeling_base.OVBaseModel.from_pretrained).
Some notable arguments include:

* `file_name`: The name of the ONNX file to load. If not specified, will default to `"openvino_model.xml"` or otherwise `"openvino/openvino_model.xml"`. This argument is useful for specifying optimized or quantized models.

* `export`: A boolean flag specifying whether the model will be exported. If not provided, `export` will be set to `True` if the model repository or directory does not already contain an OpenVINO model.

Tip

Itâ€™s heavily recommended to save the exported model to prevent having to re-export it every time you run your code. You can do this by calling [`model.save_pretrained()`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.save_pretrained "sentence_transformers.SentenceTransformer.save_pretrained") if your model was local:

```
model = SentenceTransformer("path/to/my/model", backend="openvino")
model.save_pretrained("path/to/my/model")
```

or with [`model.push_to_hub()`](../../package_reference/sentence_transformer/SentenceTransformer.html#sentence_transformers.SentenceTransformer.push_to_hub "sentence_transformers.SentenceTransformer.push_to_hub") if your model was from the Hugging Face Hub:

```
model = SentenceTransformer("intfloat/multilingual-e5-small", backend="openvino")
model.push_to_hub("intfloat/multilingual-e5-small", create_pr=True)
```

Quantizing OpenVINO Modelsïƒ•

Export, Optimize, and Quantize Hugging Face models

This Hugging Face Space provides a user interface for exporting, optimizing,
and quantizing models for either ONNX or OpenVINO:

*

[sentence-transformers/backend-export](https://huggingface.co/spaces/sentence-transformers/backend-export)

OpenVINO models can be quantized to int8 precision using Optimum Intel to
speed up inference. To do this, you can use the
[`export_static_quantized_openvino_model()`](../../package_reference/util.html#sentence_transformers.backend.export_static_quantized_openvino_model
"sentence_transformers.backend.export_static_quantized_openvino_model")
function, which saves the quantized model in a directory or model repository
that you specify. Post-Training Static Quantization expects:

* `model`: a Sentence Transformer model loaded with the OpenVINO backend.

* `quantization_config`: (Optional) The quantization configuration. This parameter accepts either:

`None` for the default 8-bit quantization, a dictionary representing
quantization configurations, or an `OVQuantizationConfig` instance.

* `model_name_or_path`: a path to save the quantized model file, or the repository name if you want to push it to the Hugging Face Hub.

* `dataset_name`: (Optional) The name of the dataset to load for calibration. If not specified, defaults to `sst2` subset from the `glue` dataset.

* `dataset_config_name`: (Optional) The specific configuration of the dataset to load.

* `dataset_split`: (Optional) The split of the dataset to load (e.g., â€˜trainâ€™, â€˜testâ€™).

* `column_name`: (Optional) The column name in the dataset to use for calibration.

* `push_to_hub`: (Optional) a boolean to push the quantized model to the Hugging Face Hub.

* `create_pr`: (Optional) a boolean to create a pull request when pushing to the Hugging Face Hub. Useful when you donâ€™t have write access to the repository.

* `file_suffix`: (Optional) a string to append to the model name when saving it. If not specified, `"qint8_quantized"` will be used.

See this example for quantizing a model to `int8` with [static
quantization](https://huggingface.co/docs/optimum/main/en/intel/openvino/optimization#static-quantization):

Hugging Face Hub Model

Only quantize once:

```
from sentence_transformers import SentenceTransformer, export_static_quantized_openvino_model

model = SentenceTransformer("all-MiniLM-L6-v2", backend="openvino")
export_static_quantized_openvino_model(
    model,
    quantization_config=None,
    model_name_or_path="sentence-transformers/all-MiniLM-L6-v2",
    push_to_hub=True,
    create_pr=True,
)
```

Before the pull request gets merged:

```
from sentence_transformers import SentenceTransformer

pull_request_nr = 2 # TODO: Update this to the number of your pull request
```

```
    model = SentenceTransformer(
        "all-MiniLM-L6-v2",
        backend="openvino",
        model_kwargs={"file_name": "openvino/openvino_model_qint8_quantized.xml"},
        revision=f"refs/pr/{pull_request_nr}"
    )
```

Once the pull request gets merged:

```
    from sentence_transformers import SentenceTransformer

    model = SentenceTransformer(
        "all-MiniLM-L6-v2",
        backend="openvino",
        model_kwargs={"file_name": "openvino/openvino_model_qint8_quantized.xml"},
    )
```

Local Model

Only quantize once:

```
from sentence_transformers import SentenceTransformer, export_static_quantized_openvino_model
    from optimum.intel import OVQuantizationConfig

    model = SentenceTransformer("path/to/my/mpnet-legal-finetuned", backend="openvino")
    quantization_config = OVQuantizationConfig()
    export_static_quantized_openvino_model(model, quantization_config, "path/to/my/mpnet-legal-finetuned")
```

After quantizing:

```
    from sentence_transformers import SentenceTransformer

    model = SentenceTransformer(
        "path/to/my/mpnet-legal-finetuned",
        backend="openvino",
        model_kwargs={"file_name": "openvino/openvino_model_qint8_quantized.xml"},
    )
```

Benchmarksïƒ•

The following images show the benchmark results for the different backends on GPUs and CPUs. The results are averaged across 4 models of various sizes, 3

datasets, and numerous batch sizes.

Expand the benchmark details

Speedup ratio:

* **Hardware:** RTX 3090 GPU, i7-17300K CPU
* **Datasets:** 2000 samples for GPU tests, 1000 samples for CPU tests.
    * [sentence-transformers/stsb](https://huggingface.co/datasets/sentence-transformers/stsb): 38.9 characters on average (SD=13.9)
    * [sentence-transformers/natural-questions](https://huggingface.co/datasets/sentence-transformers/natural-questions): answers only, 619.6 characters on average (SD=345.3)
    * [stanfordnlp/imdb](https://huggingface.co/datasets/stanfordnlp/imdb): texts repeated 4 times, 9589.3 characters on average (SD=633.4)
* **Models:**
    * [sentence-transformers/all-MiniLM-L6-v2](https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2): 22.7M parameters; batch sizes of 16, 32, 64, 128 and 256.
    * [BAAI/bge-base-en-v1.5](https://huggingface.co/BAAI/bge-base-en-v1.5): 109M parameters; batch sizes of 16, 32, 64, and 128.
    * [mixedbread-ai/mxbai-embed-large-v1](https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1): 335M parameters; batch sizes of 8, 16, 32, and 64. Also 128 and 256 for GPU tests.
    * [BAAI/bge-m3](https://huggingface.co/BAAI/bge-m3): 567M parameters; batch sizes of 2, 4. Also 8, 16, and 32 for GPU tests.

Performance ratio: The same models and hardware was used. We compare the

performance against the performance of PyTorch with fp32, i.e. the default backend and precision.

* **Evaluation:**
    * **Semantic Textual Similarity:** Spearman rank correlation based on cosine similarity on the [sentence-transformers/stsb](https://huggingface.co/datasets/sentence-transformers/stsb) test set, computed via the EmbeddingSimilarityEvaluator.
    * **Information Retrieval:** NDCG@10 based on cosine similarity on the entire [NanoBEIR](https://huggingface.co/collections/zeta-alpha-ai/nanobeir-66e1a0af21dfd93e620cd9f6) collection of datasets, computed via the InformationRetrievalEvaluator.

* **Backends:**
    * `torch-fp32`: PyTorch with float32 precision (default).
    * `torch-fp16`: PyTorch with float16 precision, via `model_kwargs={"torch_dtype": "float16"}`.
    * `torch-bf16`: PyTorch with bfloat16 precision, via `model_kwargs={"torch_dtype": "bfloat16"}`.
    * `onnx`: ONNX with float32 precision, via `backend="onnx"`.
    * `onnx-O1`: ONNX with float32 precision and O1 optimization, via `export_optimized_onnx_model(..., "O1", ...)` and `backend="onnx"`.
    * `onnx-O2`: ONNX with float32 precision and O2 optimization, via `export_optimized_onnx_model(..., "O2", ...)` and `backend="onnx"`.
    * `onnx-O3`: ONNX with float32 precision and O3 optimization, via `export_optimized_onnx_model(..., "O3", ...)` and `backend="onnx"`.
    * `onnx-O4`: ONNX with float16 precision and O4 optimization, via `export_optimized_onnx_model(..., "O4", ...)` and `backend="onnx"`.
    * `onnx-qint8`: ONNX quantized to int8 with "avx512_vnni", via `export_dynamic_quantized_onnx_model(..., "avx512_vnni", ...)` and `backend="onnx"`. The different quantization configurations resulted in roughly equivalent speedups.

* `openvino`: OpenVINO, via `backend="openvino"`.

* `openvino-qint8`: OpenVINO quantized to int8 via `export_static_quantized_openvino_model(..., OVQuantizationConfig(), ...)` and `backend="openvino"`.

Note that the aggressive averaging across models, datasets, and batch sizes prevents some more intricate patterns from being visible. For example, for GPUs, if we only consider the stsb dataset with the shortest texts, ONNX becomes better: 1.46x for ONNX, and ONNX-O4 reaches 1.83x whereas fp16 and bf16 reach 1.54x and 1.53x respectively. So, for shorter texts we recommend ONNX on GPU.

For CPU, ONNX is also stronger for the stsb dataset with the shortest texts: 1.39x for ONNX, outperforming 1.29x for OpenVINO. ONNX with int8 quantization is even stronger with a 3.08x speedup. For longer texts, ONNX and OpenVINO can even perform slightly worse than PyTorch, so we recommend testing the different backends with your specific model and data to find the best one for your use case.

[![Benchmark for GPUs](../../../_images/backends_benchmark_gpu.png)](../../../_images/backends_benchmark_gpu.png) [![Benchmark for CPUs](../../../_images/backends_benchmark_cpu.png)](../../../_images/backends_benchmark_cpu.png)

Recommendationsïƒ•

Based on the benchmarks, this flowchart should help you decide which backend to use for your model:

```
        %%{init: {
    "theme": "neutral",
    "flowchart": {
       "curve": "bumpY"
    }
}}%%
graph TD
A(What is your hardware?) -->|GPU| B(Is your text usually smaller than 500 characters?)
A -->|CPU| C(Is a 0.4% accuracy loss acceptable?)
B -->|yes| D[onnx-O4]
B -->|no| F[float16]
C -->|yes| G[openvino-qint8]
C -->|no| H(Do you have an Intel CPU?)
H -->|yes| I[openvino]
H -->|no| J[onnx]
click D "#optimizing-onnx-models"
click F "#pytorch"
click G "#quantizing-openvino-models"
click I "#openvino"
click J "#onnx"
```

Note

Your milage may vary, and you should always test the different backends with

your specific model and data to find the best one for your use case.

User Interfaceïƒ•

This Hugging Face Space provides a user interface for exporting, optimizing,

and quantizing models for either ONNX or OpenVINO:

*

[sentence-transformers/backend-export](https://huggingface.co/spaces/sentence-transformers/backend-export)

[Previous](../../../examples/applications/embedding-quantization/README.html "Embedding Quantization") [Next](custom_models.html "Creating Custom Models")

* * *