

# Sentence-T5 (ST5): Scalable Sentence Encoders from Pre-trained Text-to-Text Models

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma,  
Keith B. Hall, Daniel Cer, Yinfei Yang  
Google Research  
Mountain View, CA

## Abstract

We provide the first exploration of sentence embeddings from text-to-text transformers (T5) including the effects of scaling up sentence encoders to 11B parameters. Sentence embeddings are broadly useful for language processing tasks. While T5 achieves impressive performance on language tasks, it is unclear how to produce sentence embeddings from encoder-decoder models. We investigate three methods to construct Sentence-T5 (ST5) models: two utilize only the T5 encoder and one using the full T5 encoder-decoder. We establish a new sentence representation transfer benchmark, SentGLUE, which extends the SentEval toolkit to nine tasks from the GLUE benchmark (Wang et al., 2018). Our encoder-only models outperform the previous best models on both SentEval and SentGLUE transfer tasks, including semantic textual similarity (STS). Scaling up ST5 from millions to billions of parameters shown to consistently improve performance. Finally, our encoder-decoder method achieves a new state-of-the-art on STS when using sentence embeddings.<sup>1</sup>

## 1 Introduction

Sentence embeddings providing compact meaning representations that are broadly useful for a variety of language processing tasks include classification, question-answering, semantic retrieval, bi-text mining, and semantic similarity tasks. We explore sentence embeddings from a new family of pre-trained models: Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020). Unlike encoder-only models, which use a transformer encoder to predict random masked tokens, T5 uses an encoder-decoder architecture and a generative span corruption pre-training task. T5 models can be scaled up to hundreds of billions of parameters

<sup>1</sup>Our models are released at <https://tfhub.dev/google/collections/sentence-t5/1>.

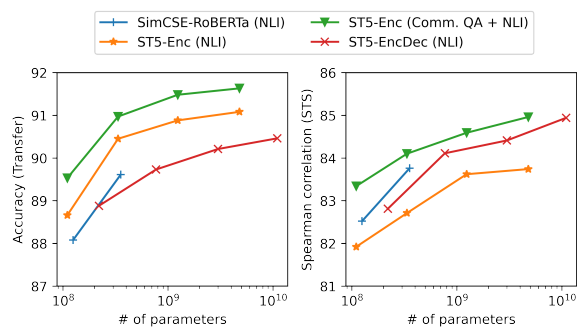


Figure 1: Scaling up our ST5 model size improves performance on SentEval (left) and STS (right).

	Transfer	STS
ST5-EncDec (11B params)	90.46	84.94
ST5-Enc (11B params)	<b>91.63</b>	<b>84.96</b>
SimCSE-RoBERTa (large) (Gao et al., 2021) <sup>2</sup>	90.23	83.76
SBERT (large) (Reimers and Gurevych, 2019)	87.69	76.55
USE (Cer et al., 2018)	85.10	71.22
InferSent (Conneau et al., 2017)	85.59	65.01

Table 1: ST5 versus notable sentence embedding models on SentEval tasks. The reported numbers are the average of transfer tasks (classification accuracy) and STS tasks (spearman correlation).

(Fedus et al., 2021) and have achieved state-of-the-art performance on a broad range of NLP tasks including Generalized Language Understanding Evaluation (GLUE) (Wang et al., 2018) and SuperGLUE (Wang et al., 2019). However, it is difficult to efficiently apply T5 to some tasks such as retrieval or clustering. To score retrieval candidates, T5 would need to perform full inference with cross-attention on each query-candidate pair. In contrast, sentence embeddings allow for efficient retrieval and clustering (Gillick et al., 2018; Reimers and Gurevych, 2019; Yang et al., 2020).

As shown in Figure 2, we explore three ways of turning a pre-trained T5 encoder-decoder model into a sentence embedding model: (i) using the

<sup>2</sup>SimCSE-RoBERTa achieves its best performance on transfer tasks by adding an additional masked language model loss during training, which is not used by ST5 or other models.

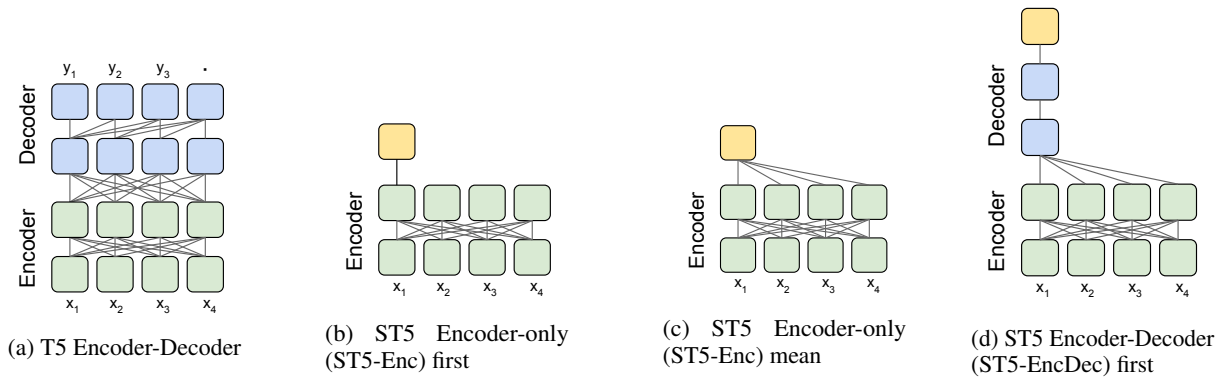


Figure 2: Architecture diagrams for T5 and three ST5 variants to extract sentence representations from T5.

first token representation of the encoder; (ii) averaging all token representations from the encoder; (iii) using the first token representation from the decoder. We evaluate the quality of the resulting sentence embeddings on sentence transfer tasks using the SentEval (Conneau and Kiela, 2018) toolkit and on our extension of SentEval to GLUE benchmark tasks (SentGLUE) in addition to semantic textual similarity (STS) (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017). We contrast raw representations from pre-trained T5 models with those learned through fine-tuning T5 on natural language inference (NLI) using dual encoders and contrastive learning (Conneau et al., 2017; Cer et al., 2018; Gao et al., 2021). We introduce a multi-stage contrastive learning recipe involving fine-tuning first on semi-structured web-mined corpora and then on NLI. Finally, we investigate scaling our T5 sentence embedding model up to 11B parameters. Illustrated in Figure 1, performance on transfer tasks and semantic textual similarity (STS) both improve with increased model capacity.

To our knowledge, we are the first to study using large-scale pre-trained text-to-text models for sentence representation learning and to scale sentence embedding models up to 11 billion parameters. We summarize our contributions as follows: (i) even without fine-tuning, encoder-only ST5 models perform well on sentence transfer tasks, outperforming state-of-the-art fine-tuned models such as SimCSE BERT and SimCSE RoBERTa (Gao et al., 2021); (ii) encoder-decoder sentence embedding models achieve strong performance on STS, establishing a new state-of-the-art on sentence embedding based STS; (iii) contrastive learning is effective for fine-tuning sentence encoders from T5-style pre-trained models, particularly using our proposed two-stage contrastive learning approach;

(iv) training ST5 longer and with more data using a contrastive loss leads to consistent improvement on both sentence transfer and STS tasks; (v) creating a new sentence representation transfer benchmark, SentGLUE, which extends the SentEval sentence evaluation toolkit (Conneau and Kiela, 2018) to nine tasks from the GLUE benchmark (Wang et al., 2018). We contribute baselines on SentGLUE using influential sentence embedding models from prior work and contrast the performance with our proposed ST5 embedding models.

## 2 Related work

Sentence embedding models have been trained using a variety of methods including: supervised natural language inference pairs (NLI) (Conneau et al., 2017; Reimers and Gurevych, 2019, 2020; Gao et al., 2021); conversational input-response and question-answer pairs (Cer et al., 2018; Yang et al., 2020); translation pairs (Yang et al., 2020; Feng et al., 2020); paraphrasing pairs (Wieting et al., 2016) and adjacent sentence pairs (Kiros et al., 2015; Logeswaran and Lee, 2018). Gao et al. (2021) achieved the previous state-of-the-art on STS with BERT and RoBERTa models by combining contrastive learning that constructs positive and negative sentence pairs using NLI data.

In parallel, Text-to-Text transfer transformers (T5) (Raffel et al., 2020), as shown in Figure 2a, are gaining popularity due to their competitive performance, effective scaling to larger model sizes, and ease of use in solving tasks as simple text-to-text mapping problems. However, extracting high quality text embeddings from T5 has not been previously explored. Moreover, while recent work has shown that scaling up models improves sentence embedding performance (Gao et al., 2021), the largest model sizes investigated only include

355 million parameters rather than the 11 billion parameters available in the largest T5 model.

### 3 Sentence-T5 (ST5)

We explore producing sentence embeddings from T5 models, ranging in size from 220 million to 11 billion parameters, both as raw sentence embeddings extracted from pretrained T5 models and using fine-tuning to refine the representations.

#### 3.1 Model Architecture

As shown in Figures 2b to 2d, we explore three strategies to extract T5 sentence representations:

- **Encoder-only first** (ST5-Enc first): the encoder output of the first token.
- **Encoder-only mean** (ST5-Enc mean): the average encoder output across all input tokens.
- **Encoder-Decoder first** (ST5-EncDec first): the first decoder output when the input text is fed into the encoder and the standard “start” symbol is fed as the only decoder input.

The first two are pooling strategies widely used in encoder-only pre-trained models such as BERT. Unlike BERT models, T5 models do not have a ‘CLS’ token at the beginning of each sentence. For T5 encoder-decoder models, we assume the decoder is aware of the semantics of the entire input sentence when generating its first token prediction; and if so, the first decoder output embeddings (i.e. input to the softmax layer) might naturally capture the sentence semantics.

For sentence encoder training, we adopt a *dual encoder* architecture (Gillick et al., 2018; Cer et al., 2018; Reimers and Gurevych, 2019). As shown in Figure 3, this architecture consists of two shared-weight transformer modules that encode the inputs. The transformer module can be either an encoder-only or encoder-decoder architecture. In our experiments, we initialize the transformer modules from a pre-trained T5 model. After each module computes a fixed-length representation for its input sentence, a projection layer and L2 normalization are applied to the resulting embeddings. The projection layer transforms the output to a configurable fixed dimension sentence embedding. The embeddings from paired encoding towers can be scored for similarity tasks using a dot-product<sup>3</sup> or provided to additional layers for classification tasks (e.g., NLI).

<sup>3</sup>Since L2 normalization is applied to the output of each

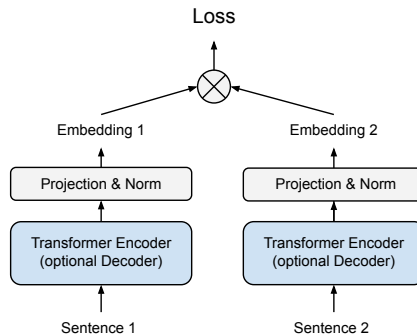


Figure 3: Architecture of the dual encoder model.

#### 3.2 Contrastive Learning

Applying contrastive learning to sentence embeddings improves the uniformity of the embedding space, leading to better performance on downstream tasks such as STS (Gao et al., 2021). We apply contrastive learning to fine-tune the T5 sentence representations.<sup>4</sup>

##### 3.2.1 Contrastive Loss

Using a contrastive loss to train a sentence encoder requires paired examples  $\mathcal{D} = \{(v_i, v_i^+)\}$  as a training set, where  $v_i$  is an input sentence and  $v_i^+$  is a related sentence (e.g., that is semantically nearby). During training,  $v_i^+$  is considered as a positive example for  $v_i$  and all other examples in a batch are considered as negatives. The model should learn to pull the positive pairs closer together while pushing away the in-batch negatives. We operationalize our contrastive loss using an in-batch sampled softmax (Henderson et al., 2017):

$$\mathcal{L} = \frac{e^{\text{sim}(v_i, v_i^+)/\tau}}{\sum_{j \in \mathcal{B}} e^{\text{sim}(v_i, v_j^+)/\tau}}, \quad (1)$$

The similarity scoring function is *sim*.  $\mathcal{B}$  is a mini-batch of examples and  $\tau$  is the softmax temperature. When additional negatives  $v_j^-$  are provided for the input example  $v$ , the loss can be computed as:

$$\mathcal{L} = \frac{e^{\text{sim}(v_i, v_i^+)/\tau}}{\sum_{j \in \mathcal{B}} e^{\text{sim}(v_i, v_j^+)/\tau} + e^{\text{sim}(v_i, v_j^-)/\tau}}. \quad (2)$$

tower, the dot-product between the embeddings will produce their cosine similarity.

<sup>4</sup>In preliminary experiments, we also explored fine-tuning with the classification loss used in InferSent (Conneau et al., 2017) and Sentence-BERT (Reimers and Gurevych, 2019). However, as previously reported in (Gao et al., 2021), our results confirmed that fine-tuning for classification on an NLI dataset is inferior to contrastive learning.

### 3.3 Two-stage Training

We explore two-stage training to refine T5 sentence embeddings: (i) first training on web mined conversational input-response and question-answering pairs; (ii) then, contrastive training on NLI pairs.

## 4 Experimental Setup

### 4.1 Training Corpus

For our fine-tuned sentence embeddings, we follow prior work showing good sentence embeddings can be obtained from supervised training on NLI (Conneau et al., 2017; Reimers and Gurevych, 2019, 2020; Gao et al., 2021) in combination with training to match conversational input-response and question-answer (CQA) pairs (Cer et al., 2018; Yang et al., 2020). We make use of two-stage training using two datasets: one is comprised of 2 Billion conversational input-response and QA (CQA) pairs drawn from web forums such as Reddit and StackExchange; the other consists of NLI pairs from the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) and Multi-Genre Natural Language Inference (MNLI) (Williams et al., 2017) datasets. For the first stage, we fine-tune using the CQA pairs under a dot-product retrieval loss with batch negatives (Cer et al., 2018; Yang et al., 2018, 2020). For the second stage, we use NLI pairs with a contrastive loss (Gao et al., 2021), where the positives are the ‘entailment’ pairs while the negatives are the ‘contradict’ pairs.<sup>5</sup>

### 4.2 Evaluation

We evaluate using SentEval, which includes 7 transfer and 7 STS tasks (Conneau and Kiela, 2018) and using our extension of SentEval to the GLUEBenchmark tasks (SentGLUE). For the transfer tasks, sentence embeddings are evaluated by how well they perform as features for a linear classification model. For STS, embeddings are evaluated by how well their cosine similarities correlate with human annotated similarity scores.<sup>6</sup>

### 4.3 Configurations

Our models are implemented using JAX<sup>7</sup> and trained on Cloud TPU-v3. We initialize the dual

<sup>5</sup>Using only the entailment and contradict pairs results in 275K contrastive NLI pairs being available for training.

<sup>6</sup>Following SimCSE (Gao et al., 2021), we report Spearman’s correlation for the ‘all’ setting for all STS tasks which aggregates the data across different subsets.

<sup>7</sup><https://github.com/google/jax>

encoder modules from public T5 checkpoints.<sup>8</sup> During training, we use Adafactor (Shazeer and Stern, 2018) as the optimizer and set the learning rate to 0.001. Linear decay is applied after 10% of the total number of training steps, reducing the learning rate to 0 by the end of training. To fine-tune on NLI we use a batch size of 512, while for the Community QA (CQA) dataset the batch size is 2048. We use a softmax temperature  $\tau$  of 0.01.

## 5 Experimental Goals

Our experiments aim to answer the following:

- Q1: What is the best way to extract sentence representations from T5?
- Q2: How well do raw T5 sentence embeddings perform on downstream tasks?
- Q3: How much do contrastive sentence embedding tasks (e.g., NLI, QA) improve T5 sentence embeddings.
- Q4: Can we benefit from scaling up T5 model capacity for better sentence representations?

With these goals, we study transfer and STS performance of T5 sentence embeddings using a variety of model and training configurations, comparing ST5 to state-of-the-art methods including SBERT/SRoBERTa (Reimers and Gurevych, 2019) and SimCSE (Gao et al., 2021).

## 6 Results

Table 2 and 3 provide performance on transfer and STS tasks, respectively. We compare ST5 models with two types of baselines: (i) a model that extracts sentence embeddings from a pre-trained BERT model, listed in rows 1–2 of each table; (ii) the current state-of-the-art sentence embedding models fine-tuned from BERT or RoBERTa, listed in rows 6–8 of each table.

### 6.1 Raw T5 Sentence Embeddings

We evaluate T5 sentence embeddings without fine-tuning using the extraction strategies from section 3.1: (i) Encoder-only first token, (ii) Encoder-only mean, and (iii) Encoder-decoder start token.

**Transfer tasks** Results for ST5 models using raw embeddings on transfer tasks are shown in rows 3–5 of Table 2. Unlike BERT, T5’s first token is not reserved as a special placeholder (i.e., CLS)

<sup>8</sup><https://github.com/google-research/text-to-text-transfer-transformer>



Model	Fine-tune data	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg
BERT (CLS-vector)	N/A	78.68	84.85	94.21	88.23	84.13	91.4	71.13	84.66
BERT (mean) ♣	N/A	78.66	86.25	94.37	88.66	84.40	92.80	69.45	84.94
ST5-Enc first	N/A	76.90	86.38	90.93	88.68	80.01	94.40	66.38	83.38
ST5-Enc mean	N/A	<b>86.56</b>	<b>91.31</b>	<b>96.01</b>	<b>90.57</b>	<b>90.77</b>	<b>94.60</b>	<b>72.93</b>	<b>88.96</b>
ST5-EncDec first	N/A	79.96	77.93	91.02	84.66	86.27	84.00	68.00	81.69
SBERT-NLI ♣	NLI	83.64	89.43	94.39	<u>89.86</u>	88.96	<u>89.60</u>	<u>76.00</u>	87.41
SimCSE-BERT ♣	NLI	82.69	89.25	<u>94.81</u>	89.59	87.31	88.40	73.51	86.51
SimCSE-RoBERTa ♣	NLI	<u>84.92</u>	<u>92.00</u>	94.11	89.82	<u>91.27</u>	88.80	75.65	<u>88.08</u>
ST5-Enc mean	NLI	86.17	91.71	94.70	90.90	90.44	90.00	<b>76.70</b>	88.66
ST5-EncDec first	NLI	<b>86.22</b>	91.60	94.05	90.93	90.72	92.60	76.06	88.88
ST5-Enc mean	CQA+NLI	85.75	92.08	94.58	<b>90.95</b>	91.76	<b>96.40</b>	75.19	89.53
ST5-Enc-1.1 mean	CQA+NLI	86.12	<b>92.50</b>	<b>94.73</b>	90.59	<b>92.15</b>	95.80	76.52	<b>89.77</b>

Table 2: Performance on transfer tasks on the SentEval benchmark. All models are using the **Base** architecture. ♣ results are from (Gao et al., 2021). For all tasks, a logistic regression classifier is trained using the sentence embeddings as features and the classification accuracy on test sets are reported.

Model	Fine-tune data	STS12	STS13	STS14	STS15	STS16	STSB	SICK-R	Avg
BERT (CLS-vector)	N/A	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
BERT (mean) ♣	N/A	<b>38.78</b>	<b>57.98</b>	<b>57.98</b>	63.15	61.06	46.35	58.40	54.81
ST5-Enc first	N/A	17.50	6.35	-20.70	2.29	21.87	16.71	28.60	10.37
ST5-Enc mean	N/A	37.78	56.83	49.37	<b>65.48</b>	<b>64.68</b>	<b>57.51</b>	<b>60.11</b>	<b>55.97</b>
ST5-EncDec first	N/A	10.91	29.59	14.90	28.91	30.61	9.45	39.31	23.38
SBERT-NLI ♣	NLI	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SimCSE-BERT ♣	NLI	75.30	84.67	80.19	85.40	80.82	84.25	80.39	81.57
SimCSE-RoBERTa ♣	NLI	<u>76.53</u>	<u>85.21</u>	<u>80.95</u>	<u>86.03</u>	<u>82.57</u>	<u>85.83</u>	<u>80.50</u>	<u>82.52</u>
ST5-Enc mean	NLI	77.37	83.65	80.41	86.04	81.70	84.49	79.79	81.92
ST5-EncDec first	NLI	77.90	85.62	<b>82.24</b>	86.81	82.13	84.98	79.97	82.81
ST5-Enc mean	CQA+NLI	<b>78.05</b>	<b>85.84</b>	82.19	<b>87.46</b>	<b>84.03</b>	<b>86.04</b>	79.75	<b>83.34</b>
ST5-Enc-1.1 mean	CQA+NLI	77.58	85.12	81.46	87.14	82.89	85.82	<b>80.18</b>	82.88

Table 3: Spearman’s correlation coefficient ( $\times 100$ ) on STS tasks on the SentEval benchmark. All models are using the **Base** architecture. ♣ results are from (Gao et al., 2021).

and there are no specific pre-training tasks using the first token’s embeddings. It is unlikely that without additional fine-tuning the first token’s representation would capture the semantics of the whole sentence. Indeed, our experiments show the first token’s representation from encoder or decoder are much worse on all SentEval tasks compared to the mean pooling of the encoder-only model.

Mean pooled T5 encoder embeddings greatly outperform mean pooled BERT embeddings. Moreover, even without fine-tuning, *mean pooled T5 encoder embeddings outperform the prior best model, SimCSE-RoBERTa (Gao et al., 2021), on transfer learning even though SimCSE-RoBERTa benefited from contrastive fine-tuning on NLI.*

The strong performance of ST5 may be due to the fact that T5 is trained on more data than BERT or RoBERTa. Additionally, the original T5 models also include downstream tasks (e.g., GLUE, SuperGLUE) during pre-training, and this multi-task setting may improve transfer performance. However, we note that there are only two SentEval tasks

(SST and MRPC) included in GLUE while the other five tasks are not. As shown in Table 2, we observe significant improvements on the five tasks that are not included.

**STS tasks** As shown in rows 3–5 of Table 3 and similar to prior work involving BERT and RoBERTa (Ethayarajh, 2019; Gao et al., 2021), mean pooling of T5 embeddings performs poorly on STS, achieving an average correlation of 55.97. While slightly better than BERT using mean pooling, this is still worse than sentence embedding models that have been fine-tuned on supervised tasks such as Sentence-BERT and SimCSE.

## 6.2 Fine-Tuning T5 Sentence Embeddings

We next evaluate ST5 models that are fine-tuned on CQA and NLI tasks using our contrastive loss.

**Fine-tuning on NLI** Given that mean pooling performs much better than the first token output representation from encoder only T5, we opt to discard the first token T5 model for our fine-tuning

experiments. The last three rows of Table 2 show that the transfer performance of ST5 models is very consistent across different embedding extracting strategies after fine-tuning. The best fine-tuned model is 0.57 better than the best raw T5 sentence embeddings. In Table 3, we see that fine-tuning on NLI data significantly improves the STS task performance of ST5.

**Fine-tuning on CQA + NLI** To investigate the impact of additional training data on contrastive learning, we experiment with the ST5 models first trained on CQA and then fine-tuned on NLI. As shown in Tables 2 and 3, fine-tuning on an additional dataset brings a large performance boost for both transfer and STS tasks. This suggests that we may be able to improve sentence embedding quality further through the mining of additional semi-structured data for continued contrastive learning.

To exclude the effect of mixing in downstream tasks, we also trained a ST5 variant based on the T5 1.1 model which was only pre-trained on the C4 dataset (Raffel et al., 2020). As shown on the last row of Table 2 and Table 3, it achieves comparable performance to the original T5 model, outperforming on most tasks but under-performing on STS.

### 6.3 Encoder-only vs. Encoder-decoder

In this section, we compare the performance of two architectures: encoder-only and encoder-decoder.

**Better generalizability for T5’s encoder** In Table 2, we saw that the encoder-only Base model performs on-par with the encoder-decoder model on transfer tasks. When we scale the ST5 model up from Base to Large, 3B and 11B, the encoder-only models’ performance on transfer tasks consistently outperforms the encoder-decoder models as shown in Table 5. This shows that building ST5 on top of the T5’s encoder gives strong transfer performance.

Recently, Chung et al. (2021) have shown that larger output embeddings (i.e. larger embedding size) effectively prevent the encoder from over-specializing to the pre-training task, thus making the encoder’s representations more general and more transferable. We hypothesize that the decoder in the encoder-decoder architecture can improve the generalizability of the encoder’s representation in a similar fashion, as the decoder focuses on optimizing for specific tasks.

**Effectiveness of the decoder** In the last two rows of Table 3, we observe that the encoder-

# of params	Base	Large	3B	11B
Model				
ST5-Enc	110M	335M	1.24B	4.8B
ST5-EncDec	220M	770M	3B	11B

Table 4: Number of parameters for different models.

decoder architecture outperforms encoder-only models for all STS tasks. As we scale up the ST5 model, we also observe improvement on STS tasks. As shown in Table 5, the ST5 encoder-decoder Large model outperforms the state-of-the-art model SimCSE-RoBERTa Large, improving the Spearman’s correlation score from 83.76 to 84.11.

One explanation is that the additional parameters from the decoder are helpful for improving performance on textual similarity tasks. Another possibility is that the decoder architecture itself helps to improve the sentence embedding quality. As shown in Figure 2d, the decoder can be considered as an additional attention pooling layer on top of the encoder outputs.

## 7 Scaling up T5

We leverage the existing checkpoints from large T5 models to study the effect of scaling sentence encoders. The parameters of the T5 models are listed in Table 4. Note however that ST5-EncDec doesn’t fully leverage the model parameters; the decoder’s learned self-attention is effectively ignored as only the start token is fed into the decoder.

### 7.1 Effect on Directly Using T5 Embeddings

As shown in Table 5, the performance on the transfer tasks of directly using T5 embeddings consistently improves as T5 scales up. This corroborates that large pre-trained models can improve transfer performance of sentence embeddings.

On the other hand, increasing the model capacity alone is not enough to achieve good performance. Even the embeddings from the T5 11B model still do worse on STS tasks than the fine-tuned models. We believe that the pre-training corruption task of T5 does not require models to avoid anisotropy.<sup>9</sup> This highlights the importance of choosing fine-tuning tasks for sentence embedding models that are aligned to the goal of similarity and/or retrieval performance.

<sup>9</sup>Having sentence embeddings smoothly and uniformly distributed within the learned embedding space, which however can be achieved by using a contrastive loss or regularization.

Model	Fine-tune data	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg
ST5-Enc mean (Large)	N/A	89.13	92.69	97.06	<b>90.70</b>	92.92	93.60	73.74	89.98
ST5-Enc mean (3B)	N/A	90.35	92.77	97.43	90.15	93.85	<b>95.60</b>	72.70	90.41
ST5-Enc mean (11B)	N/A	<b>91.15</b>	<b>93.33</b>	<b>97.55</b>	90.20	<b>94.07</b>	94.40	<b>74.26</b>	<b>90.71</b>
SBERT-NLI Large <sup>♣</sup>	NLI	84.88	90.07	94.52	90.33	90.66	87.40	75.94	87.69
SimCSE-RoBERTa Large <sup>♣</sup>	NLI	<u>88.12</u>	<u>92.37</u>	<u>95.11</u>	<u>90.49</u>	<u>92.75</u>	<u>91.80</u>	<u>76.64</u>	<u>89.61</u>
ST5-Enc mean (Large)	NLI	88.82	93.43	95.73	<b>91.75</b>	93.08	94.00	76.35	90.45
ST5-EncDec first (Large)	NLI	87.63	92.85	94.32	91.37	91.98	93.00	76.99	89.73
ST5-Enc mean (3B)	NLI	89.92	93.27	96.19	91.54	94.18	94.20	76.87	90.88
ST5-EncDec first (3B)	NLI	87.83	92.85	94.75	91.01	93.14	93.60	78.26	90.21
ST5-Enc mean (11B)	NLI	90.13	93.85	96.02	91.39	93.96	95.20	76.99	91.08
ST5-EncDec first (11B)	NLI	90.00	93.94	95.01	91.53	93.85	92.20	76.70	90.46
ST5-Enc mean (Large)	CQA+NLI	88.89	93.46	95.38	91.50	94.23	96.20	77.10	90.97
ST5-Enc mean (3B)	CQA+NLI	89.94	94.09	95.85	91.58	94.84	<b>96.20</b>	77.86	91.48
ST5-Enc mean (11B)	CQA+NLI	<b>90.83</b>	<b>94.44</b>	<b>96.33</b>	91.68	<b>94.84</b>	95.40	<b>77.91</b>	<b>91.63</b>

Model	Fine-tune data	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg
ST5-Enc mean (Large)	N/A	28.01	52.60	41.35	61.28	63.58	56.31	59.48	51.80
ST5-Enc mean (3B)	N/A	24.89	51.49	41.09	61.37	64.51	52.57	59.99	50.85
ST5-Enc mean (11B)	N/A	<b>34.97</b>	<b>60.19</b>	<b>47.59</b>	<b>66.40</b>	<b>70.62</b>	<b>62.83</b>	<b>63.57</b>	<b>58.02</b>
SBERT-NLI Large <sup>♣</sup>	NLI	72.27	78.46	74.90	80.99	76.25	79.23	73.75	76.55
SimCSE-RoBERTa Large <sup>♣</sup>	NLI	<u>77.46</u>	<u>87.27</u>	<u>82.36</u>	<u>86.66</u>	<u>83.93</u>	<u>86.70</u>	<u>81.95</u>	<u>83.76</u>
ST5-Enc mean (Large)	NLI	76.52	85.75	81.01	87.13	83.26	85.45	79.85	82.71
ST5-EncDec first (Large)	NLI	79.15	87.42	83.61	87.64	83.92	86.35	80.64	84.11
ST5-Enc mean (3B)	NLI	77.13	86.73	82.53	87.36	84.51	85.71	<b>81.39</b>	83.62
ST5-EncDec first (3B)	NLI	79.24	87.80	83.95	87.75	84.60	86.62	80.91	84.41
ST5-Enc mean (11B)	NLI	77.42	87.50	82.51	87.47	84.88	85.61	80.77	83.74
ST5-EncDec first (11B)	NLI	<b>80.11</b>	88.78	84.33	88.36	<b>85.55</b>	<b>86.82</b>	80.60	84.94
ST5-Enc mean (Large)	CQA+NLI	79.10	87.32	83.17	88.27	84.36	86.73	79.84	84.11
ST5-Enc mean (3B)	CQA+NLI	79.02	<b>88.80</b>	84.33	<b>88.89</b>	85.31	86.25	79.51	84.59
ST5-Enc mean (11B)	CQA+NLI	80.10	88.75	<b>84.70</b>	88.86	85.17	86.77	80.39	<b>84.96</b>

Table 5: Comparison of model performance on the SentEval benchmark when scaling up model size. <sup>♣</sup> results are from (Gao et al., 2021). The first set of results are for transfer tasks, while the second set are for the similarity task.

## 7.2 Scaling Up Improves Fine-tuning

As shown in Table 5, we find that scaling up model capacity leads to consistently better performance on all downstream tasks. For the ST5 11B model, the encoder-only model achieves an average score of 91.08 for transfer tasks which is better than 90.45 from the ST5 Large model; while the encoder-decoder model pushes the STS score to 84.94 and also outperforms the ST5 Large model. For STS tasks, we observe that the gain from increasing model size from 3B to 11B is smaller than that from Large to 3B. This might be due to the fact that the embedding sizes are fixed for all models in our experiments. One potential exploration is to increase the sentence embedding size for larger models to fully leverage the model capacity.

### 7.2.1 Alignment and Uniformity

We further investigate the quality of the sentence embeddings by measuring aggregate distance metrics in the learned geometric space. In particular, we compute the alignment loss and uniformity loss

as defined in Wang and Isola (2020):

$$\mathcal{L}_{\text{align}} = - \mathbb{E}_{v, v^+ \sim p_{\text{pos}}} \|f(v) - f(v^+)\| \quad (3)$$

$$\mathcal{L}_{\text{uniform}} = \log \mathbb{E}_{v, w \stackrel{\text{i.i.d.}}{\sim} p_{\text{data}}} e^{-2\|f(v) - f(w)\|}, \quad (4)$$

Above,  $p_{\text{pos}}$  is all positive data and  $p_{\text{data}}$  is the data distribution.  $\mathcal{L}_{\text{align}}$  denotes the expected distance between embeddings of the positive pairs, while  $\mathcal{L}_{\text{uniform}}$  indicates how uniformly the embeddings are distributed.

For both losses, lower numbers indicate better performance. Gao et al. (2021) has shown that models with lower numbers for these two aggregate metrics tend to have better performance on downstream tasks. As shown in Figure 4, when models scale up, both the encoder and encoder-decoder models decrease the uniformity loss by a large margin meanwhile only slightly increasing the alignment loss. This indicates that scaling up might help the sentence embeddings to spread out more uniformly in the space while keeping semantically similar pairs clustered together. We leave the further exploration of the connection between

Model	Sent. Embed. Fine-tuning	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE
InferSent (Wang et al., 2018)	NLI	66.71	8.60	83.90	76.50	80.20	81.70	67.80	-	63.50	71.50
SBERT (RoBERTa Base) <sup>♣</sup>	NLI	73.40	21.22	90.83	73.34	74.08	80.75	77.21	78.13	73.92	57.76
SBERT (RoBERTa Large) <sup>♣</sup>	NLI	75.81	20.69	93.00	73.39	76.26	82.26	79.46	80.18	75.80	60.65
SimCSE (RoBERTa Base) <sup>♣</sup>	NLI	77.05	35.87	90.71	76.47	83.93	81.39	70.74	72.05	76.30	60.29
SimCSE (RoBERTa Large) <sup>♣</sup>	NLI	76.23	40.11	93.23	70.23	81.45	84.45	73.44	73.56	75.95	60.65
ST5 Enc (Base)	CQA+NLI	76.89	22.73	91.40	76.88	86.58	84.55	69.73	70.00	79.54	61.73
ST5 Enc (Large)	CQA+NLI	78.52	29.46	93.92	77.26	86.07	85.32	72.20	72.44	79.64	66.43
ST5 Enc (3B)	CQA+NLI	79.06	34.78	94.95	78.71	85.84	85.78	72.38	73.10	79.70	66.06
ST5 Enc (11B)	CQA+NLI	<b>80.07</b>	<b>43.91</b>	<b>95.30</b>	78.46	<b>86.54</b>	<b>86.21</b>	<b>73.46</b>	<b>74.42</b>	<b>80.12</b>	66.06
ST5 Enc 1.1 (Base)	CQA+NLI	76.63	21.59	90.60	76.66	86.34	84.53	70.40	70.76	77.92	61.01
T5 (Base) (Raffel et al., 2020)	-	83.40	53.84	92.68	88.92	88.02	91.56	84.24	84.57	90.48	76.28

Table 6: Performance on transfer tasks on the Dev set of the GLUE benchmark. <sup>♣</sup> denotes that the models are released by HuggingFace. T5 (base) is a cross-attention model and other models are embedding based.

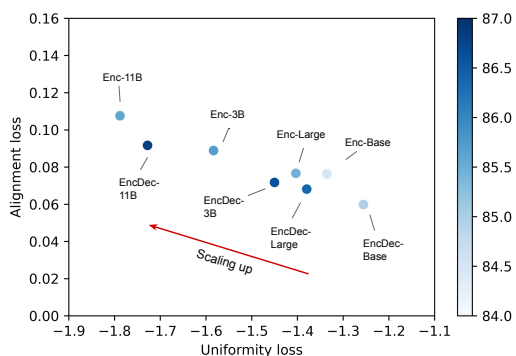


Figure 4: Alignment and uniformity losses for different model sizes. We consider the test split of the STS-B dataset.  $\mathcal{L}_{\text{align}}$  is calculated considering all pairs with score greater than 4.  $\mathcal{L}_{\text{uniform}}$  is computed using all sentences. The colormap denotes the models’ Spearman’s correlation score.

model capacity and the geometry characteristics of resulting sentence embeddings to future work.

## 8 SentGLUE Evaluation

In this section, we introduce a new sentence representation transfer benchmark – SentGLUE – which extends the sentence evaluation toolkit, SentEval, to nine tasks from the GLUE benchmark including: CoLA, SST-2, MRPC, STS-B, QQP, MNLI-m, MNLI-mm, QNLI, RTE <sup>10</sup>. The GLUE benchmark has been widely adopted for assessing language understanding models. GLUE tasks are either single sentence or sentence pair classification (e.g. NLI) or similarity (STS) tasks. The best models on the GLUE leaderboard are fine-tuned cross-attention models like BERT or T5. Such models change all the parameters in the underlying model during fine-tuning and for pairwise tasks they allow for early fusion of input features from both sentences being

<sup>10</sup>We found the WNLI task from the GLUE benchmark is too challenge for existing sentence embedding models, thus we exclude it in the current version.

compared. For SentGLUE, we introduce the constraint that each input needs to be independently encoded into a fixed embedding space representation that can then be feed to additional layers in order to make a prediction. We believe this best adapts the spirit of the original SentEval benchmark for sentence embeddings to the GLUE tasks.

From Table 6, ST5-Enc Base outperforms both SBERT-RoBERTa Base and SimCSE-RoBERTa Base on all SentGLUE tasks except CoLA and MNLI. <sup>11</sup> With the model’s increased capacity, ST5 Enc 11B’s sentence embeddings achieve the best overall performance. Notably, as model size is scaled up, aggregate performance using sentence embeddings approaches that of T5 base. This is remarkable given that T5 base makes use of full cross-attention between sentence pairs and adjusts all of the parameters in the model during fine-tuning.

## 9 Conclusion

We obtaining sentence embeddings from T5, investigating three architectures and two-stage contrastive learning for fine-tuning our representations. We compare the difference between encoder-only and encoder-decoder methods and analyze their performance on downstream tasks. Through extensive experiments on STS, SentEval and GLUE tasks, we show that encoder-only models have strong transfer performance while encoder-decoder models perform better on STS tasks. We demonstrate the effectiveness of scaling up T5 models, greatly improving sentence embedding quality. These findings suggest that future improvements in the scale and quality of pre-trained T5 models may provide further sentence embeddings improvements.

<sup>11</sup>SimCSE and ST5 only use the ‘entailment’ and ‘contradict’ pairs from MNLI datasets; while for SBERT, it also uses the ‘neutral’ pairs. This might explain why SBERT outperforms the others on MNLI.



## Acknowledgments

We thank the anonymous reviewers for their helpful comments. We thank Zora Tung, Daniel Andor, Adam Roberts, Hyung Won Chung, Anselm Levskaya and Livio Baldini Soares for help with the JAX implementation, and Alexis Conneau and Chris Tar for feedback and suggestions.

## References

- Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. [SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability](#). In *SemEval 2015*.
- Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. [SemEval-2014 task 10: Multilingual semantic textual similarity](#). In *SemEval 2014*.
- Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. [SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation](#). In *SemEval-2016*.
- Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. [SemEval-2012 task 6: A pilot on semantic textual similarity](#). In *\*SEM 2012/SemEval 2012*.
- Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. [\\*SEM 2013 shared task: Semantic textual similarity](#). In *\*SEM*.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.
- Daniel Matthew Cer, Mona T. Diab, Eneko Agirre, I. Lopez-Gazpio, and Lucia Specia. 2017. [Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation](#). In *SemEval-2016*.
- Daniel Matthew Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, C. Tar, Yun-Hsuan Sung, B. Strope, and R. Kurzweil. 2018. Universal sentence encoder. In *EMNLP*.
- Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. 2021. [Re-thinking embedding coupling in pre-trained language models](#). In *ICLR*.
- Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *LREC*.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.
- Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *EMNLP*.
- W. Fedus, Barret Zoph, and Noam M. Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv*, abs/2101.03961.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. [Language-agnostic BERT sentence embedding](#). *CoRR*, abs/2007.01852.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *ArXiv*, abs/2104.08821.
- D. Gillick, A. Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *ArXiv*, abs/1811.08008.
- Matthew Henderson, Rami Al-Rfou, B. Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and R. Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *ArXiv*, abs/1705.00652.
- Ryan Kiros, Yukun Zhu, R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In *NIPS*.
- Lajanugen Logeswaran and Honglak Lee. 2018. [An efficient framework for learning sentence representations](#). In *ICLR*.
- Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21/140.
- Nils Reimers and Iryna Gurevych. 2019. Sentencebert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*.
- Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *EMNLP*.
- Noam M. Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Yinfei Yang, Daniel Matthew Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, G. Ábrego, Steve Yuan, C. Tar, Yun-Hsuan Sung, B. Strope, and R. Kurzweil. 2020. Multilingual universal sentence encoder for semantic retrieval. In *ACL*.

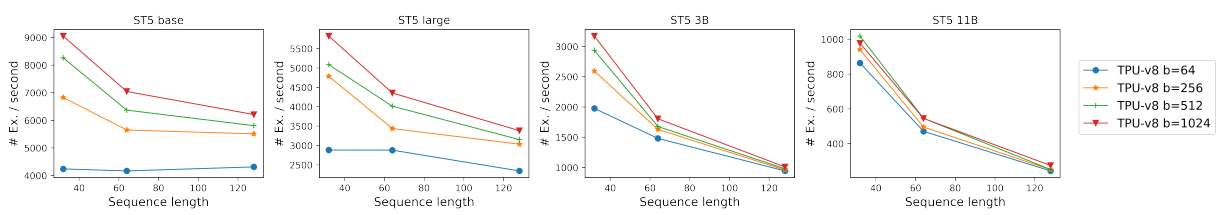
Yinfei Yang, Steve Yuan, Daniel Matthew Cer, Shengyi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, B. Strope, and R. Kurzweil. 2018. Learning semantic textual similarity from conversations. In *Rep4NLP@ACL*.

## A Model Inference

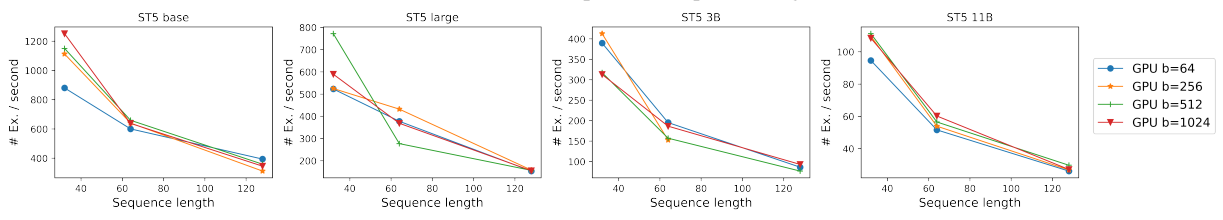
We run ST5 encoder-only on different platforms to investigate the computational cost of inference. Figure 5 summarizes the inference speed for different model sizes, sequence length, batch size and platforms. ST5 achieves the fastest inference speed on Cloud TPU-v3. As we increase the batch size, the inference speed can be further improved. For the 11B model, we are able to achieve a speed of 274 examples per second for sequence length 128 and batch size 1024. This shows the feasibility of deploying such large models on TPU hardware.

We also report the speed on Nvidia Tesla V100 GPU and CPU. The ST5 11B model is able to run on 4 V100 GPUs with sequence length 128 and batch size 1024, achieving an inference speed of 27 examples per second. For CPU, with batch size 512, ST5 11B achieves 0.5 examples per second.

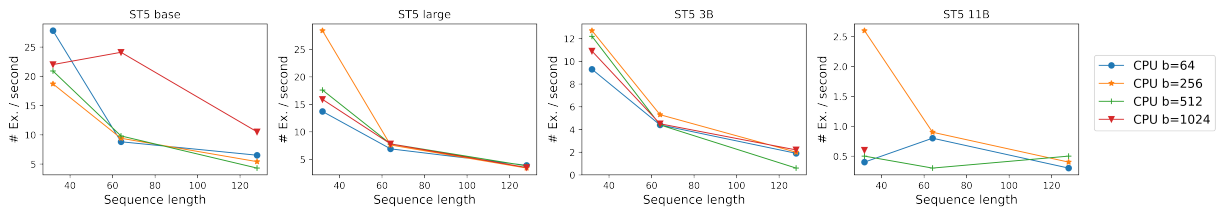
Although the speed on GPU and CPU are considerably slower than on TPU, the sentence embedding models are much faster than cross-attention based models whose computation time increases quadratically with the number of examples (e.g., clustering 1,000 sentences requires inference over 1 million sentence pairs).



(a) TPU inference speed vs. sequence length.



(b) GPU inference speed vs. sequence length.



(c) CPU inference speed vs. sequence length.

Figure 5: Comparison of inference speed for different model sizes on different platforms.