Contrastive Learning in Distilled Models

Valerie Lim vlim31@gatech.edu
Kai Wen Ng kng71@gatech.edu
Kenneth Lim klim83@gatech.edu

Abstract

Natural Language Processing (NLP) models like BERT can provide state of the art word embeddings for downstream NLP tasks. However, these models: (1) yet to perform well on Semantic Textual Similarity (STS), and (2) may be too large to be deployed as lightweight edge applications. We seek to apply a suitable contrastive learning method based on the SimCSE paper, to a model architecture adapted from a knowledge distillation based model, DistilBERT, to address these two issues. Our final lightweight model DistilFACE achieves an average of 72.1 in Spearman's correlation on STS tasks, a 34.2% improvement over BERTbase with default parameters.

1 Introduction

1.1 Problem Statement

Pre-trained word embeddings are a vital part of natural language processing (NLP) systems, providing state of the art results for many NLP tasks as the pre-trained embedding-based models frequently outperform alternatives such as the embedding-layer-based Word2Vec [5]. Accurate word embeddings have huge downstream benefits for tasks related to information retrieval and semantic similarity comparison [19], especially in a typical industrial setting where labelled data is scarce. As a result, in recent years, word embeddings from pre-trained models such as BERT (Bidirectional Encoder Representations from Transformers) have been gaining traction in many NLP works.

However, studies [6] [19] have shown that BERT does not produce good enough embeddings for semantic textual similarity tasks. As a result, recent breakthroughs [6] [13] have found that a contrastive objective is able to address the anisotropy issue of BERT. At a high level, contrastive learning is a technique where given two augmented samples of x, $x+$ (positive sample), $x-$ (negative sample), we'd like for $x+$ to be close to x and for $x-$ to be far away from x. This improves the quality of BERT sentence representation.

Yet, a problem often associated with BERT and the contrastive loss models derived off it is their large size. There is an increasing need to build machine learning systems that can operate on the edge, instead of calling a cloud API, in order to build privacy respecting systems and prevent having users send sensitive information to servers [1]. Thus, there is a growing impetus for lightweight and responsive models. However, most of the pre-trained models [6] [19] with contrastive loss are still relatively large and computationally heavy, making it hard to deploy on the edge. Deployment on the edge would also allow for reduced time and network latency, which could be critical in certain use cases such as when there is a high QOS (Quality of Service) requirement [1]. A common solution for creating lighter models is via knowledge distillation, where a trained, complex teacher model trains a lighter, shallower student model to generalise [15]. Hence, our goal was to address this joint problem of lack of lightweight models that can perform well on semantic textual similarity tasks, by creating an unsupervised pretrained model that combines knowledge distillation and contrastive learning to address their respective problems. To do this, we first adopt the architecture and pre-trained checkpoints of a knowledge distillation based model, DistilBERT, start off with a lightweight model. We then adapt from authors who successfully implemented contrastive learning on sentence embeddings via unsupervised learning [6], by feeding samples, termed as x, into the pre-trained encoder twice: by applying standard dropout twice to obtain two different sets of embeddings termed as $x+$. Then we take other sentences in the same mini-batch as $x-$ and the model predicts the positive sample among the negatives. We subsequently experiment with different pooling layers, automatic mixed precision, quantization and hyperparameter tuning to enhance performance.

1.2 Who Cares? The Implications

Our group believes there is significant value in experimenting with whether a traditional knowledge distillation method like DistilBERT, can be directly adapted to

be combined with an existing contrastive learning method such as SimCSE, without having to create an entirely new model. Such an approach makes such a combination of techniques more accessible. In addition, there are definitely environmental and economical benefits to doing so, given how computational intensive large-scale pretraining of new models are. For instance, 16 TPU chips were required to train BERTbase, with each round of pretraining taking 4 days [4], costing about US$6,912 each time [17]. Furthermore, success in our project will enable the creation of lightweight pre-trained embedding-based models that work well in semantic textual similarity tasks. This has many applications, such as searching and ranking of relevant documents based on similarity [8], and now it could be even further applied in edge scenarios. For example, such functionality can now be enabled at sensitive government, defence or health institutions, where computers are not always allowed to be connected to the cloud freely [15].

2 How Is It Done Today and Limits

2.1 Contrastive Learning Models

SimCSE [6] presented a simple contrastive learning framework that vastly improved the state-of-the-art sentence embeddings. By training unsupervised models using contrastive objective, the resulting quality of sentence embeddings improved by 4.2%. The contrastive objective can be understood as maximising the lower bound of mutual information between different views of the data [7]. Ultimately, the goal of contrastive learning is to learn effective representation via pulling semantically close neighbours together and pushing apart non neighbours, through a score function which measures similarity between these two features. This would be elaborated on the solution section. The input sentences are encoded with a pretrained language model such as DistilBERT , which is elaborated upon in the next section.

2.2 Knowledge Distillation Models

In our present work, contrastive learning is applied on a distilled version of BERT, DistilBERT. Sanh et. al [15] was able to reduce BERT model size by 40% while retaining 97% of language understanding capabilities by leveraging on knowledge distillation. DistilBERT is able to retain a significant extent of model performance despite a huge reduction of memory footprint because of the benefits conferred from knowledge distillation. In knowledge distillation, DistilBERT is known as the student, while BERT is known as the teacher. Both have the same architecture. For instance, given this sample input sentence "the <mask >licked its fur and howled", the teacher model outputs "dog" as the mask, but "wolf" and "fox" also have high scores. The distillation loss between the teacher and the student aims to align the probability distributions, rather than just the hard predictions, which is the output from the teacher model. The probability distribution is a useful addition as the student is able to learn to rank "wolf" and "dog" highly, which is useful info about the world [9].

2.3 Combining Contrastive Learning with Knowledge Distillation

The core key elements of contrastive learning are augmentation techniques to create positive samples, hard negative mining techniques to create negative samples, and sufficiently large batch size [2]. We hypothesise that there is nothing inherent about these properties that would prevent contrastive learning's effective use when applied to DistilBERT instead of BERT. We believe that it would be especially meaningful to pursue this combination of contrastive learning and knowledge distillation techniques as these were the two methods that were highlighted as the most dominant and effective surrogate tasks used in unsupervised learning and self supervised learning in the lectures of the deep learning class at Georgia Tech. [10].

There has been some work done today to combine the two techniques. The Contrastive Distillation on Intermediate Representations (CoDIR) framework [16] uses contrastive learning to distil the teacher's hidden states to the student through three objectives, an original training objective, a distillation objective, and an additional contrastive learning objective where the student is trained to distil knowledge through intermediate layers of the teacher. However, this utilisation of both teacher's output layer and its intermediate layers for student model training is a deviation from traditional knowledge distil-

lation methods, and involves the creation of a new framework.This differs from our intention of checking compatibility of existing methods. This framework is also instead applied directly on RoBERTa, and is more complex to implement. In addition the main objective of applying contrastive learning in CODIR is to compress BERT [16], whereas for us the compression is mainly done through the use of DistilBERT, and application of contrastive learning is mainly for enhancing performance.

2.4 Pooling Methods

The original authors of BERT [4] experimented with different pooling strategies and the top three performers were concat last 4, weighted sum of last four hidden and second-to-last hidden. However, these were on fine-tuned tasks and may not be good benchmarks for semantic quality of sentence representations. Han Xiao [18] claims that the last layer is too close to the target function and is biased towards the pre-training task targets and hence, argues that the second-to-last layer is a better sentence representation. Other studies [11] and [14] showed that taking average of first and last layers leads to better sentence representations. In this study, we also aim to experiment with various pooling methods to contribute more data for the research community on which pooling method works best.

3 Approach

As DistilBERT proved to be promising in reducing memory footprint with minimal loss in language understanding, we decided to start off with an initial evaluation benchmark for BERT and DistilBERT. We evaluated these two models on Semantic Textual Similarity (STS) tasks using Spearman Correlation score as the main evaluation metric. Surprisingly, results showed that pretrained DistilBERT had consistently outperformed pretrained BERT on STS tasks. BERT scored an average of 53.73, while DistilBERT scored an average of 58.23. Hence, we think DistilBERT is a promising candidate and can be successful in achieving our goals.

We adopted a similar approach to unsupervised SimCSE [6] in applying contrastive learning to pre-trained language models. To perform contrastive learning in self-supervised fashion, we take sentences $\{x_i\}_{i=1}^{m}$ and use respective x_i as x_i^+. During forward pass, dropouts from DistilBERT encoder is applied, resulting in embeddings that are similar but not identical for each pair of x_i and x_i^+. We can denote these two embeddings as $h_i^z = f(x_i, z)$ and $h_i^{z'} = f(x_i, z')$ where z and z' are two different random dropout masks applied on x_i. The cosine similarity is calculated for positive pairs x_i and x_i^+, and negative pairs x_i and x_j^+. Finally, our training loss for contrastive learning is computed as:

$$\ell_i = -log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j^+)/\tau}}$$

where τ is a temperature hyperparameter that can be tuned for improving model performance. Here, minimizing the loss function will result in maximizing the similarity between positive pairs (increasing the numerator), and minimising the similarity between negative pairs (decreasing the denominator). Using contrastive loss function, the 66M learned parameters of DistilBERT will be tuned. The overall contrastive learning architecture using DistilBERT is shown in Figure 1 below:

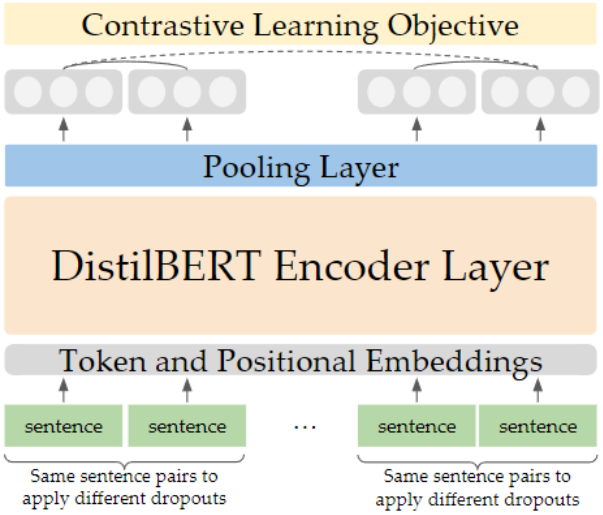

Figure 1: Overall DistilFACE Architecture. Similarity of final embeddings after the pooling layer are measured. Solid lines are positive examples, while dotted lines are negative examples.

3.1 Training Dataset: Wiki 1M

DistilFACE model is trained using Wiki 1M sampled dataset obtained from SimCSE. The Wiki 1M dataset was derived from the original English Wikipedia dataset through random sampling. Each row in the dataset is a sentence extracted from Wikipedia articles. This is chosen as the dataset for training because authors of SimCSE [6] found that such NLI datasets are effective for learning sentence embeddings.

3.2 Evaluation Dataset: STS Task Datasets

In the evaluation phase, we used STS-12, STS-13, STS-14, STS-15 and STS-B as our evaluation set for measuring model performance on semantic tasks. Each row in STS Task datasets contains a pair of sentences. Each pair is manually labelled with a score ranging from 0 to 5 to indicate how similiar or relevant the pair of sentence is to each other. We chose to use this dataset for evaluation since it has been widely used as a benchmark for STS performance.

4 Methodology

Pre-trained DistilBERT model with initial checkpoints from Transformers Hub is downloaded and used as the starting point for applying contrastive learning on DistilBERT.

With reference to common hyper-parameters used by previous works in BERT, DistilBERT, and SimCSE, we started with a default set of hyperparameters in this study shown in table below:

Optimizer	Maximal Length	Batch Size	Learning rate
Adam	32	64	5e-05
Pooling method		Temperature	Max Steps
Average first last		0.05	30,000

Table 1: Default Hyperparameters

A grid search is performed to find the optimal set of hyperparameters, more explanation can be found in the Results & Analysis section. In grid search, we also perform evaluation to understand the effect of hyperparameters on STS task performance.

4.1 Success Metrics

Following the methodology in SimCSE, the SentEval engine by Conneau et. al. [3] was mainly used to evaluate model performance on these evaluation datasets, and Spearman Correlation is used as the evaluation metric. This metric helps us to determine the quality of the sentence embeddings produced for determining semantic similarity.

To ensure our Spearman correlation results are comparable with SimCSE, the final Spearman correlation score for each dataset is calculated by first concatenating all subsets of data within the STS dataset, and then calculating Spearman correlation once, similar to the approach by SimCSE. This is slightly different from the default calculation from SentEval, which takes the average of Spearman correlation scores for each data subset within each STS dataset.

To measure success, after hyperparameter tuning is complete, the best hyperparameters will be used with max steps tuned to adjust for overfitting in the evaluation set. Spearman Correlation scores for the final DistilFACE will be compared with other models such as BERT, DistilBERT and SimCSE.

4.2 Further Enhancements

To make our model even more efficient and require less GPU memory, we also adopted the use of Automatic Mixed Precision (AMP). Instead of always storing weights and biases in 32-bit float format, 16 bit is sometimes used where appropriate. This process of autocasting to lower precision where appropriate reduces wastage of GPU memory. An initial issue faced was that performance was noticeably worse. However, we realized that this was likely due to underflow, as the gradients can become too small to be stored via this format. This can be managed by applying a scaling factor to gradients. The difference made by this scaling factor increased average scores across benchmarks by 8.6%. Overall, utilizing AMP resulted in a speed up of about 1.4x for training with a batch size of 64, to up to 3.2x for batch size of 256, and meant we needed a maximum of 8GB in GPU memory during

our training process, enabling local training in line with our secondary project objectives.

We also managed to further reduce the model's size by 52% through quantization (refer to Appendix Table 9). This involved converting a floating point model's weights and activations to reduced precision integers post-training. However, this reduced model performance by about 6.9% on average. Hence, we decided to stop pursuing this trade-off further, and instead focus on our core metric of model performance.

5 Results & Analysis

We compare the evaluation results of our final DistilFACE model with other models:

Model	STS12	STS13	STS14	STS15	STSB	Avg.
BERT (Avg. first-last pooling)	39.70	59.37	49.68	66.03	53.87	53.73
DistilBERT (Avg. first-last pooling)	47.59	61.87	52.94	69.69	59.05	58.23
SimCSE-BERTbase	68.40	82.41	74.38	80.91	76.85	76.25
DistilFACE (Avg. last-4 pooling)	63.72	75.29	69.49	78.22	73.80	72.10

Table 2: Comparison of Model Performances. We report Spearman Correlation above as $\rho \times 100$

We can see that our application of contrastive learning to DistilBERT has been clearly successful, with DistilFACE performing on average 34.2% and 23.8% higher in Spearman Correlation compared to BERT and DistilBERT respectively. We have also achieved our secondary goal of building a lightweight model, since DistilFACE, which is based on DistilBERT, is significantly smaller and faster than BERT [15] (shown in Appendix Table 8). In fact, DistilFACE and quantized DistilFACE are respectively 1.64 and 3.15 times smaller than the BERT implementation (shown in Appendix Table 9).

We believe that our experiments were successful for two key reasons. The first, is that our hypothesis in section 2.3 that the properties of contrastive learning are compatible with DistilBERT is logically sound. The second, is effective hyperparameter tuning and experimentation with pooling methods enabled good performance. We detail the results and analysis of that process below.

5.1 Hyperparameter tuning

These are the hyperparameters that we tuned: Similarity Temperature, Batch Size, Learning Rate and Pooling Method. Each hyperparameter is tested against their selected set of values, which are adapted from [6]. The optimal hyperparameters and evaluation score by training steps is shown below:

Optimizer	Maximal Length	Batch Size	Learning rate
Adam	32	128	1e-05
Pooling method		Temperature	Max Steps
Average last 4		0.05	20,000

Table 3: Optimal Hyperparameters for Final DistilFACE

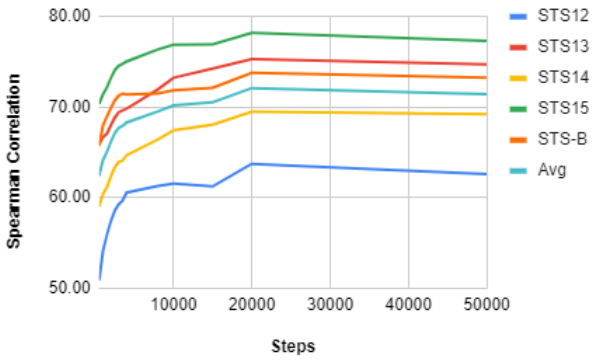

Figure 2: Spearman Corr. by Steps on STS datasets

We tuned the final model based on the max steps to avoid overfitting as well. As shown in Figure 2 above, we can see DistilFACE starts to overfit at 20,000 steps and above. Hence, the best max steps for our final model is at 20,000.

5.2 Challenges Faced

A problem we encountered was during experimentation with hyperparameter tuning frameworks like Ray and Optuna to enable more sophisticated methods. We did not anticipate issues as there are examples of successful applications to BERT models. However, these frameworks were found to be incompatible with our use case. Firstly,

the trainer required labels from the dataset by default, which is not applicable for our unsupervised model. This caused the "compute_ metric" function to be ignored by the tuning framework, which is not ideal given that we require the use of the SentEval toolkit for evaluation. Secondly, our custom model in evaluation mode does not directly output evaluation loss. Instead such metrics are calculated separately outside the model via the SentEval toolkit, which is too complex to be integrated directly.

Hence, we instead adopt the grid-search methodology adopted by the authors in [6]. Each set of hyperparameter values are used in training for 30,000 steps, and then evaluated independently on these 5 datasets described above. We selected this number of steps as we found training loss to be stable by this threshold across the different hyperparameters. This reasonable amount of steps also meant that using grid search proved sufficient, as gains from sophisticated methods like parallelisation via Ray and Optuna would be more limited.

5.3 Learning Rate

Learning rates of 1e-5, 3e-5 and 5e-5 were selected as hyperparameter values as adapted from [6]. Learning rate of 1e-5 resulted in the best average Spearman correlation, as highlighted in the yellow bar in Figure 3 and Table 4 in Appendix B. The figure also shows that smaller the learning rate, the greater the Spearman correlation.

5.4 Similarity Temperature

Temperatures of 0.001, 0.01, 0.05, 0.1, 1 were tested, with 0.05 achieving the best average Spearman correlation scores. (see as Figure 4 and Table 5 in Appendix B). This parameter scales the inner product between two normed feature embeddings[12] when calculating the training objective. Temperature controls how sensitive the objective is to specific embedding locations [20]. As such this value needs to be neither too large nor small, and in this scenario optimally at 0.05, as highlighted in the yellow bar (see Figure 4 in Appendix B).

5.5 Batch Size

Batch sizes of 64, 128, 256 were tested, with size 128 achieving the best average Spearman correlation scores, as highlighted in the yellow bar (see Figure 5 and Table 6 in Appendix B). However, this figure also shows that the model is not too sensitive to batch sizes if a suitable learning rate is used, consistent with the findings from [6]. This further validates that contrastive learning need not require very large batch sizes [6], unlike what was previously thought[2]. This also further supports our overarching hypothesis that contrastive learning is suitable to be applied to more light weight models.

5.6 Pooling Methods

We experimented with various pooling methods taking reduce functions: average and max, as well as the selection of hidden layers: last hidden, second-to-last, all hidden, first and last, last two, last four and concatenation of last four layers (see Figure 6 and Table 7 in Appendix B). Our findings are as follows:

Best Pooling Methods. Concat and average of the last four layers were the best two performing pooling methods, as highlighted in the yellow bar. This is consistent with BERT [13] where concatenation of the last four layers worked best for them.

Next Best Pooling Methods. Authors [11] [14] mentioned that taking the average of first and last leads to better results. We find this consistent with our results with average of first and last to be one of the better performing pooling methods out of all we have experimented.

Never too close. Our results show that the last hidden is the third best performing pooling method, better than the second-to-last layer. This discredits the claim from Han Xiao [18] that the last layer is too close to pre-trained tasks and the second-to-last layer is a better sentence representation. Here, we find this to be false.

6 Conclusion

Overall, we believe our project to be a success using the success metrics in Section 4.1, namely Spearman Correlation on STS tasks. This can be seen from the results and rationale in Section 5. Hence we hope our DistilFACE model can help lay the groundwork for future work involving the building of lightweight models for such tasks using contrastive learning and knowledge distillation.

References

[1] H. G. Abreha, M. Hayajneh, and M. A. Serhani. Federated learning in edge computing: A systematic survey. *Sensors*, 22(2):450, Jan 2022. url: http://dx.doi.org/10.3390/s22020450.

[2] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations, 2020, arXiv:2002.05709.

[3] A. Conneau and D. Kiela. Senteval: An evaluation toolkit for universal sentence representations, 2018, arXiv:1803.05449.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv:1810.04805.

[5] M. Farahmand. Pre-trained word embeddings or embedding layer?, 2019. url: link.

[6] T. Gao, X. Yao, and D. Chen. Simcse: Simple contrastive learning of sentence embeddings, 2021, arXiv:2104.08821.

[7] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization, 2018, arXiv:1808.06670.

[8] A. Kashyap, L. Han, J. Sleeman, T. Satyapanich, S. Gandhi, and T. Finin. Robust semantic text similarity using lsa, machine learning, and linguistic resources. *Lang Resources & Evaluation*, 50, 2016. url: link.

[9] Z. Kira. Cs7643 lecture 12: Masked language models, 2022. url: Lecture 12.

[10] Z. Kira. Cs7643 lecture 18: Unsupervised and self-supervised learning, 2022. url: Lecture 18.

[11] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online, Nov. 2020. Association for Computational Linguistics. url: https://aclanthology.org/2020.emnlp-main.733.

[12] S. Li, J. Xu, and B. Hooi. Probabilistic contrastive loss for self-supervised learning, 2021, arXiv:2112.01642.

[13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019, arXiv:1907.11692.

[14] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019, arXiv:1908.10084.

[15] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019, arXiv:1910.01108.

[16] S. Sun, Z. Gan, Y. Cheng, Y. Fang, S. Wang, and J. Liu. Contrastive distillation on intermediate representations for language model compression, 2020, arXiv:2009.14167.

[17] M. S. Tony Peng. The staggering cost of training sota ai models, 2019. url: link.

[18] H. Xiao. Why not the last hidden layer? why second-to-last?, 2018. url: Why not the last hidden layer.

[19] Y. Yan, R. Li, S. Wang, F. Zhang, W. Wu, and W. Xu. Consert: A contrastive framework for self-supervised sentence representation transfer, 2021, arXiv:2105.11741.

[20] O. Zhang, M. Wu, J. Bayrooti, and N. Goodman. Temperature as uncertainty in contrastive learning, 2021, arXiv:2110.04403.

A Project Code Repository and Results

The Github repository for our final project is at: https://github.com/kennethlimjf/contrastive-learning-in-distilled-models. Results can be found in Google Sheets: STS Benchmark Results on Google Sheet

B Tables and Charts

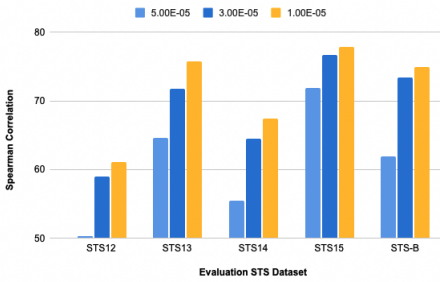

Figure 3: Spearman Corr. by Learning Rate on STS datasets

Learning Rate	STS12	STS13	STS14	STS15	STS-B	Average
5.00E-05	50.23	64.59	55.43	71.85	61.93	60.806
3.00E-05	58.98	71.78	64.52	76.71	73.46	69.09
1.00E-05	61.08	75.8	67.43	77.84	75	71.43

Table 4: Spearman Corr. by Learning Rate on STS datasets

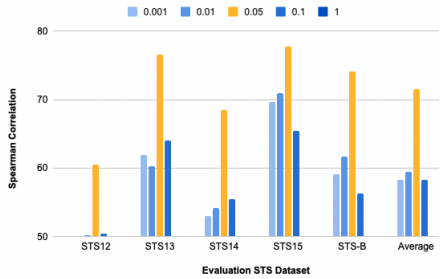

Figure 4: Spearman Corr. by Similarity Temperature on STS datasets

Temperature	STS12	STS13	STS14	STS15	STS-B	Average
0.001	47.6	61.87	52.94	69.69	59.05	58.23
0.01	50.18	60.31	54.21	71	61.69	59.478
0.05	60.56	76.64	68.53	77.71	74.1	71.508
0.1	50.38	64.04	55.4	65.4	56.31	58.306
1	43.05	45.45	39.26	47.86	32.36	41.596

Table 5: Spearman Corr. by Similarity Temperature on STS datasets

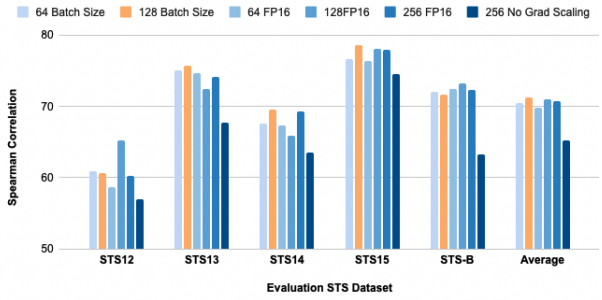

Figure 5: Spearman Corr. by Batch Size on STS datasets

Batch Size	STS12	STS13	STS14	STS15	STS-B	Average
64 Batch Size	60.91	74.97	67.58	76.55	72.05	70.412
128 Batch Size	60.65	75.64	69.51	78.56	71.58	71.188
64 FP16	58.6	74.63	67.26	76.33	72.39	69.842
128 FP16	65.22	72.45	65.91	78.03	73.17	70.956
256 FP16	60.29	74.07	69.24	77.98	72.26	70.768
256 No Grad-Scaling	56.97	67.69	63.57	74.45	63.21	65.178

Table 6: Spearman Corr. by Batch Size on STS datasets

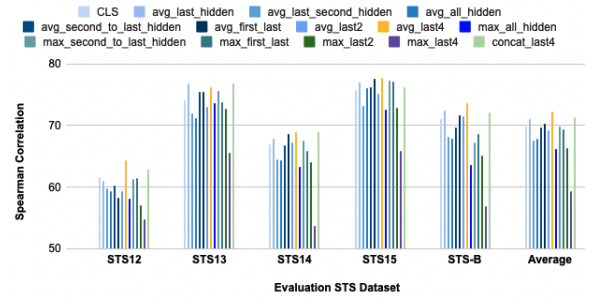

Figure 6: Spearman Corr. by Pooling Methods on STS datasets

Pooling Method	STS12	STS13	STS14	STS15	STS-B	Average
cls	61.62	74.03	66.96	75.8	70.96	69.874
avg_last_hidden	60.93	76.76	67.84	76.95	72.38	70.972
avg_last_second_hidden	59.75	72.01	64.45	73.15	68.16	67.504
avg_all_hidden	59.31	71.25	64.35	76.09	67.84	67.768
avg_second_to_last_hidden	60.28	75.41	66.77	76.17	69.7	69.666
avg_first_last	58.27	75.48	68.61	77.51	71.58	70.29
avg_last2	59.37	73.04	67.18	75.21	71.51	69.262
avg_last4	64.26	76.28	68.95	77.76	73.62	72.174
max_all_hidden	58.14	73.65	63.3	72.56	63.51	66.232
max_second_to_last_hidden	61.35	75.63	67.56	77.22	67.15	69.78
max_first_last	61.47	73.72	65.8	77.15	68.6	69.35
max_last2	57.04	72.71	63.99	72.86	65.02	66.32
max_last4	54.78	65.55	53.74	65.84	56.83	59.35
concat_last4	62.82	76.81	68.86	76.27	72.16	71.38

Table 7: Spearman Corr. by Pooling Methods on STS datasets

Model	# params (millions)	Inference time (seconds)
BERT	110	668
DistilBERT	66	410

Table 8: Difference in number of parameters and inference time between BERT and DistillBERT from [15]. Inference time of a full pass of STS-B on CPU with batch size of 1

Model	Model size (KB)
BERT	435650
DistilFACE	265489
Quantized DistilFACE	138116

Table 9: Comparison of Model Size

Name	Contributions

Kenneth	- Literature review
	- Coding DistilFACE
	- Experiment with
	  - Pooling Methods
	  - Max Steps
	- Final Model tuning
	- Report Writing for Solution and Methodology

Kai Wen	- Literature review
	- Experimenting with:
	  - Hyperparameter tuning framework (eg Ray and Optuna)
	  - Effects of Batch Size
	  - Effects of Temperature
	  - Enhancing efficiency (eg using AWP and grad scaling)
	  - Reducing size (eg quantization)
	- Final editing, ensure alignment with rubrics (eg sections on implications, limits of current practice, problems and mitigation)

Valerie	- Conceptualised project idea - Literature review - Drafted introduction section of report - Ran experiments on different learning rates - Generated charts and table

Table 10: Team member contributions